Path computation algorithms in IP networks : reliable hot-potato routing & deployable multi-constrained tunnels

Jean-Romain Luttringer

UNIVERSITÉ DE STRASBOURG

ÉCOLE DOCTORALE MATHÉMATIQUES, SCIENCES DE L'INFORMATION ET DE L'INGÉNIEUR
Laboratoire ICube — UMR7357

THÈSE présentée par :

Jean-Romain LUTTRINGER

soutenue le : **28 novembre 2022**

pour obtenir le grade de : **Docteur de l'université de Strasbourg**
Discipline/ Spécialité : **Informatique**

Calcul de Chemins pour Réseaux IP

Routage de la Patate Chaude et Froide lors de Pannes
&
Chemins Multi-contraints pour Segment Routing

THÈSE dirigée par :
Mme. PELSSER Cristel — Professeure, Université de Strasbourg

THÈSE co-encadrée par :
M. MERINDOL Pascal — Maître de conférences, Université de Strasbourg
M. BRAMAS Quentin — Maître de conférences, Université de Strasbourg

PRESIDENTE DU JURY :
Mme. TEXIER Géraldine — Professeure, IMT Atlantique

RAPPORTEURS :
M. KUIPERS Fernando A. — Professeur, TU Delft
M. LEDUC Guy — Professeur, Université de Liège

AUTRES MEMBRES DU JURY :
M. CLAUSS Philippe — Professeur, Université de Strasbourg

Jean-Romain LUTTRINGER

Calcul de Chemins pour Réseaux IP : Routage de la patate Chaude et Froide lors de Pannes & Chemins Multi-contraints pour Segment Routing

Résumé

Les travaux présentés dans cette thèse se décomposent en deux parties centrées autour du routage.

Nous nous intéressons d'abord aux calculs de chemins multicritères, notamment utiles pour router du trafic exigeant une latence faible. Le problème NP-Difficile étudié, appelé DCLC, devient radicalement plus complexe lorsque l'on considère les contraintes opérationnelles rajoutées par la technologie utilisée pour déployer ces chemins, Segment Routing. Nous proposons différents méthodes et algorithmes afin de résoudre DCLC dans un tel contexte opérationnel, et montrons l'efficacité de nos solutions via une évaluation sur des réseaux large-échelle.

Nous nous concentrons ensuite sur les effets néfastes induits par les interactions inter-protocolaires. Les interactions entre BGP (le protocole de routage utilisé dans l'Internet) et l'IGP (utilisé au sein d'un réseau) provoquent un temps de convergence long lors de changements topologiques. Nous retravaillons ces interactions et proposons OPTIC, ramenant ce temps de convergence à une durée marginale. Nous montrons la faisabilité d'OPTIC via évaluation théorique basée sur des données réelles.

Mot-clés : Routage, Calcul de chemins, DCLC, Segment Routing, BGP, IGP, Qualité de Service, Ingénierie de Trafic

Résumé en anglais

The work presented in this thesis is divided into two parts centered around routing.

First, we focus on multi-criteria path computations, which are particularly useful for routing traffic requiring low latency. The NP-hard problem studied, called DCLC, becomes radically more complex when we consider the operational constraints added by the technology used to deploy these paths, Segment Routing. We propose different methods and algorithms to solve DCLC in such an operational context, and show the efficiency of our solutions via an evaluation on large-scale networks.

We then focus on the adverse effects induced by inter-protocol interactions. Interactions between BGP (the routing protocol used in the Internet) and the IGP (used within a network) cause long convergence times during topological changes. We rework these interactions and propose OPTIC, reducing this convergence time to a marginal duration. We show the feasibility of OPTIC via theoretical evaluation based on real data.

Keywords: Routing, Path Computation, DCLC, Segment Routing, BGP, IGP, Quality of Service, Traffic Engineering

Acknowledgments

As the very last word that I will add to this manuscript, these acknowledgments mark the end of this three-year journey. As this journey was above all else a collective ride (and effort), I cannot imagine any more fitting way to figuratively and literally turn this page. As everyone reading this is probably well aware, putting gratitude into words is not easy, but I will try and do my best. The words I will probably struggle to scribble on this page will, at best, express but a fraction of my appreciation for the support I got from every single one of you [1].

I thank Cristel Pelsser, my advisor during this Ph. D. I greatly appreciated your guidance (scientific or not) during these three years. Your precise feedback always helped moved my work forward, and always sparked new possibilities to continue to extend our work. I also always appreciated your involvement when trying to stay on top of the latest contributions in our field, something that I was pretty terrible at, and would have been even more terrible at if it wasn't for you. Despite your busy schedule, you always made time when necessary to discuss our next possible steps when things were blurry. Perhaps even more importantly, I loved that you were always in a good mood, smiling and kind; a mindset I hope to be able to achieve going forward. Thanks for everything, from your kindness to the Belgian beers your brought (and without forgetting the chocolate box you gave me after my defence). I wish you the best in Belgium!

I also want to thank Quentin Bramas. You've been a huge help during these three years. Your good spirit, curiosity, and relaxed attitude were wonderful additions to the team. The fact that every challenge, be it little, small, mathematical, technical, sensical, or non-sensical always sparks your interest was (and still is) always impressive to me, and a huge source of inspiration and motivation. I cannot thank you enough for your implication and help during this Ph. D, and the positive atmosphere you brought with you. Obviously, this paragraph cannot end before also thanking you for the numerous gaming sessions, which helped me stay mostly sane during these past three years. By the way, you will be glad to know that I checked and that you *indeed* won one of our 15 Starcraft games. However, as very few people will read this paragraph, no one will ever know. I also want to thank you for always having a jar of pickles in your office, it really helped me when I was craving something sour.

I want to deeply thank Pascal Mérindol, for which these acknowledgments must be extended to cover not only 3 years, but 6! There are far too many things I am grateful for to be able to put everything into words. Back in 2017, I could not have imagined the amazing journey that my mail shyly asking for an internship would spark. Thank you for welcoming me without hesitation within your ongoing research projects and for all the times you vouched (and continue vouching) for me. The work in this thesis, as well as myself, would not be where it is now without you. The endless discussions with you, be they scientific, pedagogical (or simply gossip) will probably be the best memories of these six years. During these discussions, I have learned perhaps one of the most important lessons of this Ph. D; the joy of what I now believe research should be: discussing and getting lost within each trivially important topic. Perhaps even more importantly, it is thanks to you that I discovered the joy that teaching can bring. Working with you to create and discuss class materials sparked an entirely new passion that I plan to nurture and follow. For that, I can't be grateful enough.

[1] Readers be warned that these acknowledgments have been written in the same spirit and fashion as some of the work found within this thesis, *i.e.*, in a panic, a few hours before the deadline.

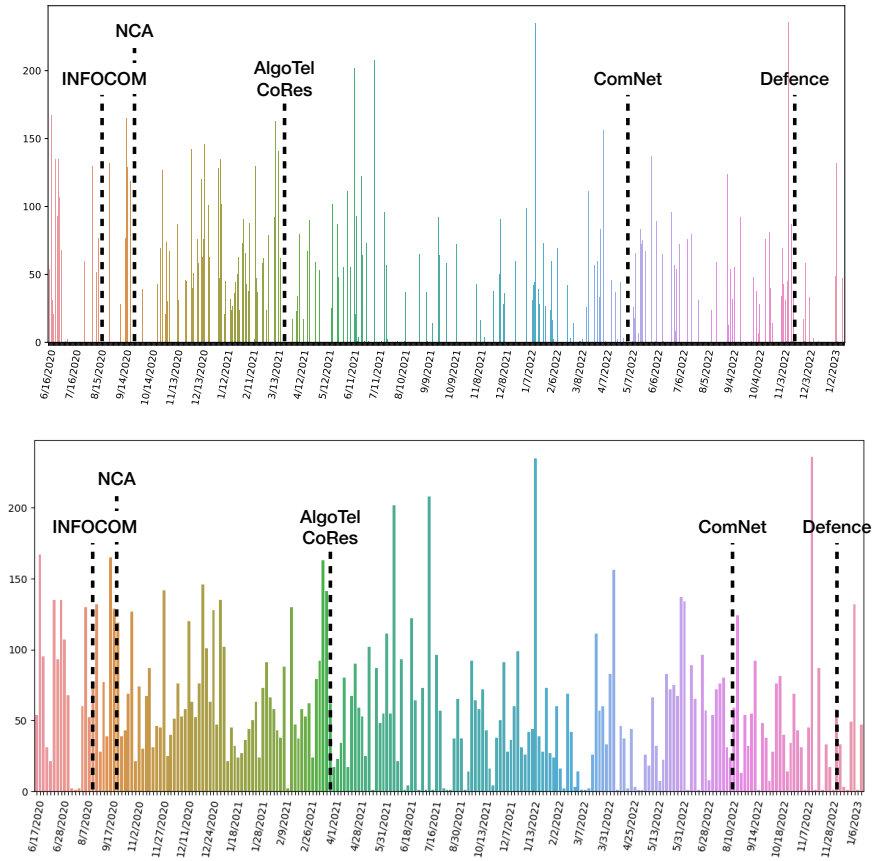


FIGURE 1: Number of minutes Pascal spent on phone calls with me per day. Top figures shows days with no phone calls, while bottom figure ignore them. Correlated with publications.

Your scientific and pedagogical skills, enthusiasm, passion, optimism, and friendliness set an ideal that I will continue to strive for going forward. Simply put, I couldn't have hoped for a better mentor.

More importantly, as my defence is now done, I can finally reveal *publicly* a fraction of the mentoring work you put into this thesis. I hope that you will enjoy, in Fig 1, a plot of all of our ≈ 450 phone calls, starting in June 2020, for a grand total of 219 hours.

Overall, a huge thanks to my advisors for their guidance, their unfailing good moods, their patience and their kindness. The words you said to me during my defence still resonate with me. You have made this Ph.D a truly awesome and amazing experience I will remember and cherish going forward.

I want to thank the jury of my Ph. D. defense, *i.e.*, Géraldine Texier, Guy Leduc, Fernando A. Kuipers, and Philippe Clauss. Thanks for reading this fairly lengthy manuscript. I hope this experience was as painless as possible! I also want to thank you for the interesting discussions during the defence which sparked several challenging ideas. Overall, thank you for acknowledging my work. A special thanks to Géraldine for accepting to be the president of the Jury. Hearing that I was now "Docteur de l'Université de Strasbourg" was probably the part of this journey that felt the most surreal and that I will remember for a long time.

I also want to thank the Network Research Team of the ICube laboratory. I could hardly have imagined a better environment for my Ph. D. The discussions and debates we had during coffee breaks and lunches will be some of my most cherished moments of this Ph. D. Thanks to Thomas Noël, for the various discussions, administrative help, and telling me that I was admitted as a Ph. D student (I still have that mail). Thanks to Stéphane Cateloin, once again for the discussions we had and your help. In particular, know that you saved my TD when you took the time to help me retrieve these markers. Thanks to Guillaume Schreiner, for always being willing (and patient enough) to share your technical expertise with me. I swear I'll start working on the Tofino you installed soon. Thanks to Anissa Lamani for always taking the time to listen and answers my non-sensical questions, ranging from robots to IP networks. I admire the effort you put into everything you do (I personally wouldn't have gone to the University on a Friday evening to solve this VM issue). Thanks to Pierre David for the numerous pedagogical discussions we had, and of course the chocolate bars in the coffee room. Your dedication to teaching will always remain inspiring to me. Thanks to Julien Montavont, for the various pop-culture-related (and sometimes pedagogical) discussions that we had for way too long in the coffee room. Thanks to Fabrice Theoleyre, both for your valuable scientific feedback, but also for your help and insight regarding the future of my career (and, even more importantly, for the new chairs in the Ph. D. room). I will however admit that I probably only remember about 25% of all the acronyms you taught me.

A huge thank to the Ph. D. students, *les compagnons de galère*, to cite Dr. Abegg. Although quite a few of you left a fairly long time before I defended, you created a very friendly and welcoming environment, which we tried our best to maintain for future generations. Thanks to Andréas Guillot, for his help when discovering this brand-new ecosystem and for the numerous climbing sessions. Thanks to Loïc Miller for the very interesting (and always very scientific) discussions throughout this Ph. D., and for always being present to talk. Finalizing each exam subject of Inter-Domain Routing at 2 am on a Sunday is a memory that will remain engrained in my brain for a long time (for better or for worse). Speaking of Mini-Internet, thanks to Thomas Holterbach for your help deploying it here and for sharing your feedback on OPTIC, as well as for taking the time to share your experience on the dreaded programmable networks. Thanks to Rodrigo Teles Hermeto for always being nice and welcoming and listening to the probably useless feedback of an inexperienced first-year. Thanks to Renato Juacaba Neto for always creating a light and chill atmosphere within the Ph. D. room. How you manage to always stay calm and in a good mood will remain a mystery to me (and sorry I couldn't help you with this graph problem). Thanks to Sebastian Lucas Sampayo, or, as some of my students called him, "the Brazilian guy". The advice you gave, scientific or not, was always on point. Chatting with you was so enjoyable that I *almost* didn't mind waiting for you to finish your meals at the RU. To continue on the Brazilian side (;D), thanks to Julian Martin Del Fiore. Your enthusiasm and happiness were key ingredients of this amazing environment. I hope that the Spanish sun made up for some of the cold winters you had to suffer. The only thing I do *not* thank you for is for having made acknowledgments that are too good, and which now leave me in a very delicate situation, if you catch my drift. A huge thanks to Amine Falek, for the very lengthy and all-over-the-place discussions we had. I hope that we'll get

to at least partially some of the future works we listed two years ago. Also, if you're reading this, answer your messages. Special thanks to Jean-Philippe Abegg and Thomas Alfroy, my "desk-mates" (if that's a thing). The very serious and scientific discussions we had, ranging from video games to anime, as well as our hours-long pauses trying to guess obscure words were a true glimpse of what research should be. On a more serious note, thanks Thomas for being so motivated and involved regarding both OPTIC and BEST2COP. You were a big and appreciated help on these projects. JP, I hope you started working on B^2GP, because I didn't. Finally, thanks to the youngest members of the ICube Ph. D. cult; Amaury Bruniaux and Farzad Veisi. Discussing with you was always very enjoyable (and thanks Farzad for always sharing your snacks with other Ph. D. students). I wish you all the best for the future.

I want to thank several friends whose sheer presence helped during this Ph. D. Thanks to Julien Luthringer, Pierre Mura, and Thomas Muller, my friends since kindergarten (approximately, I don't want to get too technical here). Despite my lack of communication during the past years and the geographical distance, it was always comforting to know that I could count on you if need be. I hope that soon enough, the stars will align so that we can finally drink a beer together again (or perhaps wine is now more fitting). Thanks to Arthur Jud, who is probably and understandably too busy to read this. The few times we managed to meet were always very appreciated breaks during this 3 years-long rush. Thanks to Cyrille Muller and Titouan Rabu. I'm runningly out of "jokish" ways to talk about my bursts of months-long disappearances, so know that your infallible support and friendship were (and are) always deeply appreciated, and were always in a corner of my mind during these past years. As my friends, you probably know that writing such acknowledgments is not an exercise I'm super comfortable at, so I'll end it with a final Thank You.

Merci à ma famille pour leur soutien inconditionnel et incommensurable, non seulement pendant ces trois dernières années, mais également durant les 24 qui les précédaient. Vous m'avez inculqué les valeurs qui m'ont permis d'aller aussi loin, cette curiosité et cette envie d'apprendre qui m'ont poussée à aller dans cette voie dans laquelle je m'épanouis à présent. Votre soutien et vos encouragements constants durant ce périple académique m'ont permis d'avancer même dans les moments de doute. Pour votre soutien, votre réconfort, votre confiance et vos encouragements, du fond du coeur, merci.

Enfin, un énorme merci à ma conjointe, Tatiana Mutalapova. Il est difficile de mettre des mots sur l'impact que ta présence à mes côtés a eu pendant ce doctorat. Bien que tu refuseras probablement de l'admettre, je te dois une grande partie des victoires de cette thèse, que j'ai eu la force d'aller chercher grâce à ta gentillesse et ta patience inébranlable. Malgré le fait que ces trois premières années n'étaient pas (toujours) des plus agréables, tu es toujours restée souriante, pour les hauts comme les bas. Ta patience, ton soutien à chaque deadline compliquées, tes paroles réconfortantes lors des moments difficiles ainsi que ta compersion lors de chaque petite victoire étaient des rayons de soleil constants. Ces trois années auraient été bien plus difficiles sans toi. Merci.

As these acknowledgements come to an end, I wish the few of you courageous enough to continue reading this manuscript a pleasant reading. But before that, let me conclude with perhaps the only suitable word to end this section : a final genuine, wholehearted

Thank you

Résumé

Les réseaux informatiques constituent l'épine dorsale de la plupart des communications modernes. En tant que tels, ils sont censés acheminer, ou router, les données d'une source vers toute destination accessible. Le terme "routage" désigne la manière dont les données sont transférées sur les réseaux et dont les chemins sous-jacents empruntés par ces données sont calculés. La manière dont le routage doit être effectué est spécifiée par les *protocoles de routage*. Par conséquent, les communications au sein d'Internet dépendent fortement des performances, du comportement et de l'interaction des protocoles de routage en jeu.

Internet étant un ensemble hétérogène de réseaux indépendants (appelés Systèmes Autonomes, ou *Autonomous System* (AS)), plusieurs protocoles de routage sont nécessaires. Alors que *Border Gateway Protocol* (BGP) détermine par quels ASes le trafic doit passer, les *Internal Gateway Protocols* (IGPs) détermine le chemin que le trafic doit suivre au sein de chaque ASes.

À l'origine, le routage était effectué selon la philosophie du *best-effort*, sans garantie sur la qualité de la communication. Au fil du temps, le *routage qualitatif* est devenue de plus en plus important. En effet, même les communications non critiques sont censées bénéficier d'une connectivité constante. En outre, les flux plus cruciaux et premium peuvent avoir des exigences strictes, par exemple en ce qui concerne la latence des communications. Ces nouveaux défis, centrés autour de la *résilience* et de la *qualité de service*, sont au cœur de cette thèse.

Tout d'abord, nous examinons comment faire en sorte que le trafic transitant par un AS ne souffre pas d'événements internes à ce dernier, comme une panne. Cette question est particulièrement difficile car le principal protocole en jeu, BGP, est intrinsèquement très lent puisqu'il doit maintenir les informations d'accessibilité de toutes les destinations au sein d'Internet. Néanmoins, nous proposons des algorithmes et des structures de données qui garantissent que de tels événements deviennent transparents pour le trafic de transit, qui bénéficie alors immédiatement de la nouvelle route optimale vers sa destination. Nous montrons que le coût de management de notre solution est gérable, et drastiquement réduit par rapport à BGP pour la majorité des ASes au sein d'Internet.

Ensuite, nous nous intéresserons au calcul de chemins qui respectent une contrainte supérieure sur la latence, tout en considérant le coût IGP, une métrique fixée par l'opérateur sur chaque lien, représentative de la bande passante ou de l'intention de conception des opérateurs. Si le calcul de ces chemins est complexe en soi (un problème NP-Hard en théorie), nous considérons une contrainte supplémentaire en nous assurant que ces chemins peuvent effectivement être déployés (*i.e.*, utilisés par le trafic concerné) avec les technologies actuellement déployées (*Segment Routing* (SR) en particulier), une contrainte opérationnelle souvent négligée. Malgré les particularités de cette nouvelle contrainte, qui nécessite une attention particulière pour être considérée correctement, nous proposons deux méthodes efficaces à cet effet et les évaluons dans divers scénarios. En particulier, nous montrons que le calcul de tels chemins peut être effectué en moins d'une seconde dans des réseaux de 100 000 noeuds.

Abstract

Computer networks are the backbone of most modern communication. As such, they are expected to *route* data from a source to any reachable destination. *Routing* refers to how data is transferred across networks, and how the underlying paths should are computed. The way routing should be performed is specified by *routing protocols*. Consequently, communications within the Internet are heavily reliant on the performance, behavior, and interaction of the routing protocols at play.

The Internet being a heterogenous collection of independent networks (called *Autonomous Systems* (ASes)), several routing protocols are required. While *Border Gateway Protocol* (BGP) dictates through which ASes the traffic should go, *Internal Gateway Protocols* (IGPs) dictate which path the traffic should follow within each ASes.

Originally, routing was performed following the *best-effort* philosophy, with no guarantee on the quality of the communication. However, as time went on, offering *qualitative routing* become increasingly important. Indeed, even non-critical communications are expected to benefit from constant connectivity. Furthermore, more crucial, premium flows may have strict requirements, for example regarding their experience latency. These news challenges, centered around *resiliency* and *quality of service*, are at the core of this thesis.

First, we consider how to ensure that traffic transiting through an AS does not suffer from internal events within the latter such as failure. This issue is particularly challenging since the main protocol in play, BGP, is intrinsically very slow as it must maintain the reachability information of all destinations within the Internet. Nevertheless, we propose algorithms and data structures that ensure that such events become transparent to transiting traffic, which immediately benefits from the optimal new route to its destination. We show that the management cost of our solution is manageable and drastically reduced compared to BGP for the majority of the ASes within the Internet.

Second, we will delve into the computation of paths that respect an upper constraint on the latency, while still considering the IGP cost, a metric set by the operator on each link, representative of the bandwidth or the operators' design intent. While the computation of these paths is complex in itself (an NP-Hard problem in theory), we consider an additional constraint by ensuring that these paths can actually be deployed (*i.e.*, used by the relevant traffic) with the currently deployed technologies (*Segment Routing* (SR) in particular), an operational constraint often overlooked. Despite the peculiarities of this new constraint, which requires specific care to be considered properly, we propose two efficient methods to this effect and evaluate them in various scenarios. In particular, we show that computing such paths can be done in less than 1 second within networks of 100 000 nodes.

List of publications during the Ph.D.

Journals

- Jean-Romain Luttringer, Thomas Alfroy, Pascal Mérindol, Quentin Bramas, François Clad, Cristel Pelsser. *Deploying near-optimal delay-constrained paths with Segment Routing in massive-scale networks*, in Computer Networks, 2022 [Luttringer *et al.* 2022].

Conferences

- Jean-Romain Luttringer, Quentin Bramas, Cristel Pelsser and Pascal Mérindol, *A Fast-Convergence Routing of the Hot-Potato* in IEEE INFOCOM 2021 - IEEE Conference on Computer Communications, 2021 [Luttringer *et al.* 2021d].

- Jean-Romain Luttringer, Thomas Alfroy, Pascal Mérindol, Quentin Bramas, François Clad and Cristel Pelsser, *Computing Delay-Constrained Least-Cost Paths for Segment Routing is Easier Than You Think*" in IEEE 19th International Symposium on Network Computing and Applications (NCA), 2020 [Luttringer *et al.* 2020b].

National Conferences

- Jean-Romain Luttringer, Thomas Alfroy, Pascal Mérindol, François Clad, Cristel Pelsser, *Le Problème à trois Contraintes : Calcul et Déploiement de Segments de Routage* in 23èmes Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications (AlgoTel), 2021 [Luttringer *et al.* 2021b].
  (Best Paper Award)

- Jean-Romain Luttringer, Quentin Bramas, Cristel Pelsser, Pascal Mérindol, *L'Art d'Anticiper les Changements IGP pour Acheminer Optimalement la Patate en Transit* in 6ème Rencontres Francophones sur la Conception de Protocoles, l'Évaluation de Performance et l'Expérimentation des Réseaux de Communication (CoRes), 2021 [Luttringer *et al.* 2021c].

Other publications

Journal

- J. Luttringer, Y. Vanaubel, P. Mérindol, J. Pansiot and B. Donnet, *Let There Be Light: Revealing Hidden MPLS Tunnels With TNT*, in IEEE Transactions on Network and Service Management (TNSM), June 2020 [Luttringer *et al.* 2020c].

Conference

- Y. Vanaubel, J. -R. Luttringer, P. Mérindol, J. -J. Pansiot and B. Donnet, *TNT, Watch me Explode: A Light in the Dark for Revealing MPLS Tunnels*, 2019 Network Traffic Measurement and Analysis Conference (TMA), 2019. [Vanaubel *et al.* 2019].

Software developed during the Ph.D.

The software and tools we have developed collaboratively during this Ph.D. are available publicly at the following link.

- BEST2COP and LCA (with SAMCRA)
  This repository contains *Best Exact Segment Track for 2-Constrained Optimal Paths* (BEST2COP), our algorithm computing *Delay-Constrained, Least-Cost* (DCLC) paths deployable with *Segment Routing* (SR), as well as *Live Conversion Algorithm* (LCA), our algorithm encoding path to segment lists on the fly in a fashion allowing to solve DCLC for SR. In particular, we fitted LCA onto the *Self-Adaptive Multiple Constraints Routing Algorithm* (SAMCRA) algorithm, which we re-implemented.
  https://github.com/talfroy/BEST2COP

- OPTIC-P4
  This repository contains the data-plane implementation of *Optimal Protection Technique for Inter-intra-domain Convergence* (OPTIC) in P4. Although this implementation is merely proof a concept, the latter is usable with a custom, simplified python control-plane available in the repository.
  https://icube-forge.unistra.fr/jr.luttringer/optic-p4

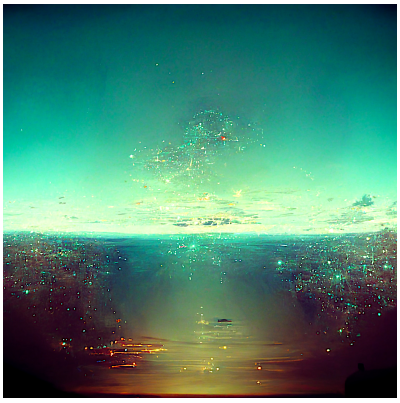
- YARGG
  This repository contains *Yet Another Realistic Graph Generator* (YARGG), a large-scale multi-metric topology generator which creates topologies following realistic structures based on geographical data and common network design guidelines. The precise topologies used to evaluate BEST2COP can be found online [Luttringer *et al.* 2021a].
  https://github.com/JroLuttringer/YARGG

- Evaluation tools
  These repositories contain the tools used to evaluate BEST2COP, LCA and OPTIC. These tools allow to reproduce the experimentation performed to evaluate these contributions, and the associated figures (or perform complementary evaluations).
  Simtool (evaluating BEST2COP and LCA): https://git.unistra.fr/bramas/simtool/
  OPTIC tool (evaluating OPTIC): https://zenodo.org/record/3972128#.Yxu_Ui8RrFY

Contents

Chapter 1

Introduction



Drawing of the Midjourney AI when prompted *"What does the Internet look like ?"*

Over the past decades, the challenges that computer networks had to tackle evolved substantially. At first, these challenges mainly revolved around connectivity. Enabling communication among computers within a network is indeed a considerable task already, in particular given the dynamic nature of these entities which may undergo failures or other topological changes. As computer networks started to become more and more prevalent, it became clear that connectivity should not only be ensured among devices within a given network but also *across* independent networks, which would in turn allow global connectivity across all devices. This collection of interconnected independent networks (also called domains or *Autonomous Systems* (ASes)) would later form the modern-day Internet.

To achieve constant connectivity, *routing protocols* were designed, such as *Open Shortest Path First* (OSPF) [Moy 1998], *Intermediate System to Intermediate System* (IS-IS) [rfc 1990], or BGP [Rekhter *et al.* 2006]. Routing protocols dictate how *routing* should be performed, *i.e.*, how topological information should be exchanged across devices in charge of forwarding data, and how the best forwarding paths should be computed.

Routing protocols referred to as IGPs specify how *intra-domain* routing should be performed. Usually, intra-domain routing relies on paths that minimize the IGP cost, an additive metric set on each link by the operator of the network according to its design wishes and the paths it wants the traffic to take.

While routing across networks (*i.e.*, *inter-domain* routing) is also specified by routing protocols, these protocols strongly differ from IGPs, as they evolve in a drastically different context. Indeed, each AS is governed by distinct administrative entities, with different (or even conflicting) views regarding what the best paths should be, depending on political or economical relationships between each domain. Consequently, inter-domain routing is mainly governed by a different routing protocol, BGP, designed to work in such an environment.

Communications across the Internet thus depend on the interaction between different routing protocols: while the domains the traffic goes through are decided by BGP, its path within each domain is decided by the IGP deployed therein.

As stable and global connectivity was achieved, the Internet started to become the backbone of more and more communications, ranging from trivial, unimportant flows to

critical and real-time ones with strict requirements regarding the characteristics of the communications, such as the experienced latency. Consequently, challenges evolved towards *qualitative routing*, *i.e.*, ensuring an overall better experience and service quality within computer networks. In this thesis, we will further investigate two main types of qualitative routing challenges: resiliency and quality of service.

Resiliency became a fundamental challenge of computer networks. Indeed, the growth in popularity of real-time services over the Internet means that service availability must now be almost permanent, despite the numerous unplanned changes that computer networks undergo. Such resiliency is not trivial to achieve, as topological changes force routing devices to *re-converge*, *i.e.*, recompute the best forwarding paths. This reconvergence puts the network into a transient, inconsistent state during which connectivity may be lost or the routing performed may be suboptimal. Computer networks must thus be resilient, and adapt quickly to such events.

Numerous improvements have been proposed and implemented to improve the convergence time of routing protocols. Improvements have been made to IGPs to better deal with events occurring within one's domain, or to BGP, to offer better resiliency to remote events occurring in distant domains. The challenge is arguably even more critical when considering BGP, as the scale at which the protocol operates leads to long delays, both regarding the notification of the event and the re-computation of the best paths [Labovitz *et al.* 2000a].

This challenge becomes even more interesting when considering that both types of events and protocols are entangled. More precisely, BGP has to re-convergence even after intra-domain events. Indeed, when several inter-domain routes seem equally attractive, the one enabling traffic to exit the domain as quickly as possible (following the best intra-domain paths) is selected. Because of this *hot-potato routing*, inter-domain routes must be re-evaluated even upon (frequent) intra-domain events, an intrinsically long process that may lead to suboptimal routing or long-lasting breaches of service.

Moreover, even perfectly resilient networks may not offer sufficient performance or *Quality of Service* (QoS) for some kind of traffic. Indeed, computer networks usually provide *best-effort* delivery, meaning that the network does not provide any guarantee regarding the actual quality of the forwarding paths taken, be it their latency or reliability. While best-effort delivery is sufficient for most of the traffic, some modern flows demand very strict requirements regarding their forwarding paths. For example, financial trading flows exchange time-sensitive data which involves large monetary stakes [Giacalone *et al.* 2015], compelling clients to pay a considerable amount for guaranteed low latencies. 5G slicing may also require the ability to bound the end-to-end latency of some flows [Programme 2020]. Although such flows rarely have to traverse the Internet, they may rely on *Virtual Private Network* (VPN) to communicate. *Internet Service Providers* (ISPs) providing such VPN services are thus faced with *Service Level Agreements* (SLAs) even more stringent than usual, in particular regarding the end-to-end latency of the forwarding paths.

Ensuring that the relevant traffic does indeed benefit from forwarding paths that provide such strict SLAs is a complex task, both from a computational and technical standpoint. These forwarding paths must indeed respect the required latency, but should ideally also aim to minimize the IGP costs, as the latter reflect the operators' intent and design. Finding paths that minimize the IGP cost while respecting an upper constraint on the delay requires

solving DCLC, an NP-Hard problem.

This problem becomes even more interesting when considering the technical side of the issue. Indeed, as the computed paths may deviate from the standard, best-effort paths (still used by the majority of the traffic), additional technologies are required to ensure that the relevant flows do indeed follow the DCLC paths, *i.e.*, to *deploy* such paths. Currently, SR (an implementation of the source routing paradigm) is the most popular technology providing such capabilities. Within networks deploying SR (sometimes referred to as SR domains), specific paths can be specified in the form of detours (called segments) added to the packet itself. These segments are forwarding instructions, which are read and executed by the routers along the forwarding paths. However, the number of segments is limited by the underlying hardware. Consequently, paths must not only respect the required latency but also be deployable in fewer segments than the upper limit of the considered hardware. This additional metric exhibits a peculiar behavior compared to standard additive metrics. In particular, it cannot be represented as an additional static weight on the graph and breaks the fundamental sub-path optimality property, upon which most path computation algorithms rely. Additional methods are thus required for this metric to be considered properly, further increasing the practical difficulty of an already complex theoretical problem.

1.1 Contributions

In this thesis, I will present several contributions, revolving around both the resiliency- and QoS-centric challenges just mentioned. While bound by the common theme of qualitative routing, both challenges (and so, their associated contributions) are separate and independent.

First, we provide several schemes enabling the efficient computation of DCLC paths deployable within SR domains. To devise such schemes, we first propose to rely on the inaccuracy of the latency measurements to reduce the theoretical complexity of DCLC while remaining practically exact, or at least within a bounded distance from the optimal solution.

To correctly consider the number of segments despite the metric peculiarities, we propose a structure, the SR Graph, which enables us to consider this metric properly by encompassing all three relevant metrics (number of segments, delay, and cost) and transforming the number of segments into a more manageable metric (the hop count). We propose two ways to rely on this construct to compute deployable DCLC paths.

The first one consists in directly exploring the SR Graph to easily keep track of all three metrics. Indeed, within this augmented graph, the number of segments becomes a standard metric, allowing the use of traditionnal algorithms and methods. While the SR Graph is particularly dense, we show that this method is viable by designing an algorithm, *Best Exact Segment Track for 2-Constrained Optimal Paths* (BEST2COP) which solves DCLC for SR domains by exploring the SR Graph. By leveraging the characteristics of the SR Graph and multi-threading, we show that BEST2COP can compute deployable DCLC paths efficiently even in large-scale networks. In particular, we also extend BEST2COP to leverage the structure of large-scale operators' networks. We design a large-scale multi-metric topology generator, *Yet Another Realistic Graph Generator* (YARGG), to evaluate our algorithm, and

show that this extension of BEST2COP is able to solve DCLC for SR in about 1s within networks of 100 000 nodes.

As exploring the SR Graph may be costly (because of its density) for some algorithm, we propose an alternative method to solve DCLC-SR with traditionnal multi-metric shortest paths algorithms. By relying on the information contained within the SR Graph, it is possible for such an algorithm to solve DCLC-SR while considering the original graph. It is however necessary to modify the algorithm by (i) converting paths to segment lists, to keep track of the number of segments, and (ii) modify the way distances are compared, to consider the effect of this new metric on the subpaths optimality property. We thus devise a conversion algorithm, *Live Conversion Algorithm* (LCA), enabling multi-metric paths computation algorithms to keep track of the number of segments necessary to encode paths on the fly, while exploring the original (sparser) graph of the network. As this new metric behaves differently from standard metrics, we formally describe and prove the necessary modifications that should be made so that path computation algorithms correctly handle these peculiarities when relying on LCA, and correctly retrieve all relevant paths. We implement these modifications on a state-of-the-art multi-metric algorithm and evaluate the latter in various scenarios.

Second, to ensure better network resiliency upon intra-domain events, we propose OPTIC, which aims at making intra-domain events transparent for the transit traffic going through the domain despite the ill effects of hot-potato routing. OPTIC achieves these results through adequate data structures and algorithms. By leveraging the way inter-domain routes are selected, OPTIC can efficiently pre-compute backup sets of routes *guaranteed* to contain the new best route towards a remote destination upon *any* possible internal event. Upon an internal event, the new optimal route can be found efficiently within these sets instead of relying on the slow BGP convergence. Furthermore, remote destinations sharing the same sets of backup routes share the same entry in memory, allowing for efficient grouped updates. We show through a theoretical evaluation that the number of distinct sets (and so, of entries to manage) is greatly inferior to the number of vanilla BGP entries for the majority of ASes in the Internet and devise efficient algorithms to update these backup sets when required (although, for most events, these sets should remain stable). As OPTIC was designed with programmable hardware in mind, we discuss in detail how our solution fits within these modern architectures through a clever hardware-software co-design and the benefits that ensue.

1.2 Overview

This thesis is organized into six chapters. In Chapter 2, we start by providing the necessary background to discuss our contributions and the associated related work. We first introduce fundamental graph theory concepts used throughout this thesis. We then discuss path computation, a central aspect of this thesis, and, more generally, routing. We present the most well-known mono-metric shortest paths computation algorithms, and present the general concepts required to understand how multiple metrics affect the computation of shortest paths. We then describe how these algorithms fit within routing protocols to ensure reachability within and across computer networks, before discussing *Traffic-Engineering* (TE) technologies, with a focus on SR.

In Chapter 3, we detail the related work around our contribution. We start by discussing resiliency and reviewing solutions aiming to improve the convergence time of protocols or provide fast re-routing capabilities, be it in an intra-domain or inter-domain context. We then review the work and propositions centered around SR, focused on TE propositions and the translation of forwarding paths to segment lists. Last but not least, we review constrained paths computation algorithms, visiting heuristics, approximations, exact schemes, and other approaches.

In Chapter 4, we present our contributions related to the computation of deployable multi-metric paths for SR domains. We describe how we deal with the NP-Hardness of DCLC in a way fitted for operator networks. We describe in detail our construct, the SR graph, and both algorithms that rely upon it, BEST2COP and LCA. We then evaluate their performance, before concluding this chapter and discussing the possible short- to mid-term perspectives.

In Chapter 5, we present OPTIC. We explain how OPTIC constructs and maintains backup sets of routes, enabling BGP to react almost instantaneously upon intra-domain events. We then present our theoretical evaluation of OPTIC, exhibiting its manageable operational cost, before discussing the possible perspectives.

Finally, we conclude this thesis in Chapter 6 by summarizing our results and discussing long-term perspectives. Note that the code of our contributions and the code enabling us to easily reproduce our experiments is made available online (the links being listed at the beginning of this document).

Have a pleasant reading

Chapter 2

Background

Contents

In this chapter, we review the background necessary to describe my contributions (and their surrounding context) in the upcoming chapters. We start by reminding some graph theory definitions in Section 2.1. We review the basis of mono-metric and multi-metric path computation in Section 2.2. We then explain the inner workings of both inter and intra-domain routing protocols in Section 2.3, and how path computation comes into play within the latter. Finally, we review the latest Traffic-Engineering deployments technologies in Section 2.4, and how the latter enable subtler routing features within today's networks.

NOTATION	DEFINITION
$G = (E, V, w)$	A MultiGraph G with a set of vertices V, a set of edges E, and a weight function $w : E \to \mathbb{N}^m$
m	Number of components of $w(u, v)$
(u, v)	An edge in E between node u and node v
$(u, v)_i$	A specific (u, v) edge within MultiGraph G.
$E(u, v)$	The set of all edges between node u and node v
$w(u, v)$	The weight vector of edge (u, v)
$w_i(u, v)$	The i^{th} component of $w(u, v)$
$p(s, t)$	A path between node s and t, defined as a list of edges (may be referenced as p if context allows).
$d(p)$	The distance vector of path p. (may be referenced as d if context allows).
$d_i(p)$	The i^{th} component of $d(p)$, $i \leq m$.
$P(s, t)$	All paths between node s and t
$p^*(s, t)$	A shortest path between node s and t (mono-criterion)
$P^*(s, t)$	All shortest path between node s and t (mono-criterion)
$d^*(s, t)$	The best distance between node s and t (mono-criterion)
$p_i^*(s, t)$	A shortest path between nodes s and t when considering the i^{th} criterion.
$P_i^*(s, t)$	All shortest path between node s and t when considering the i^{th} criterion.
$d_i^*(s, t)$	The best distance between node s and t when considering the i^{th} criterion.
$\mathcal{P}$	A Pareto-front (or Pareto-set)

FIGURE 2.1: Notations introduced within this section

2.1 Graph Theory Concepts

Graphs are an intuitive way to abstract networks, including computer networks. Thus, network-related problems are often studied using graph theory, which offers a powerful formal framework. As routing problems are no exception, we will start by describing formally some graph theory notions that we will be using throughout this thesis. A summary of the notations used within this chapter can be found in Table 2.1

A Graph $G = (V, E)$ consists of $|V|$ vertices (or nodes) $v \in V$ and $|E|$ edges $(u, v) \in E \subseteq V \times V$. Edges can thus be seen as a set of binary relations between nodes. For example, when modeling cities as nodes, an edge between two nodes u and v generally represents the existence of a road between the pair of cities represented by u and v. In the case of wired computer networks, nodes usually represent routers (or hosts) while edges represent the existence of a fiber-optic link or copper cable between said routers.

The relation represented by each edge may be asymmetrical. For example, one may consider a one-way road between two cities. In these cases, the graph is said to be *directed*, and an edge (u, v) is different from the edge (v, u). The relation may however also be symmetrical. In this case, the graph is said to be *undirected*, and (u, v) and (v, u) represent the same edge. Two nodes u and v are said to be *adjacent*, or *neighbors*, if there is an edge between them (*i.e.*, $(u, v) \in E$). Node u may also be referred to as the *predecessor* or *successor* of v (the two terms being equivalent in undirected graphs). If two given vertices may be connected by more than one edge, G is called a *multigraph*. We then use $E(u, v)$ to denote the set of all edges between the nodes u and v. When necessary, we denote specific links (*e.g.*, the i^{th} edge) between u and v as $(u, v)_i$.

Computer networks are often modeled as undirected graphs, as the existence of a physical link generally enables two-way communication. Thus, in this thesis, we will represent networks as undirected graphs for the sake of simplicity. The contributions we will present in the following chapters do not, however, require the graph to be undirected and may be used on directed graphs. Note that computer networks often exhibit several links between two given nodes, mainly for scalability and resiliency purposes. Thus, in the following chapters, we will consider multigraphs.

While edges represent the same relation, they are often not equivalent. For example, links between routers may have different bandwidths, loss rates, or latencies. Thus, to better encompass the underlying network, edges are often *weighted*, or *labeled*. A weighted graph $G = (V, E, w)$ respects the previous definitions, but also possesses a weight function w which associates with each edge a given weight vector, as an edge may be characterized by several distinct values (*e.g.*, its latency, distance, or cost). The weight function may thus be defined as $w : E \to \mathbb{N}^m$, where m is the number of components of the weight vector [1]. We denote the components of the weight vector $w_i, i = 1 \dots m$. If $m = 1$, G is said to be a *monocriterion* (or monometric) graph. Otherwise, G is said to be *multicriteria* (or multimetric).

[1] Although some graphs may exhibit negative weights, we will only consider non-negative weight valuations within this thesis.

A *path* p is an ordered list of edges that joins a sequence of vertices. When necessary, we denote $p(s,t)$ a path which starts at s and ends at t with $s,t \in V$. When $s = t$, the path is called a *cycle*. We denote $P(s,t)$ the set of all paths between s and t. We may sometimes use P when specifying end nodes isn't necessary. A path is said to be *simple* if no vertice appears more than once within the sequence, *i.e.*, if it contains no loop. The *distance* $d(p)$ of a path $p = (v_1, v_2), (v_2, v_3), \ldots, (v_{k-1}, v_k)$ is the multidimensional sum vector of the weight vector of each edge in p. More formally, $d(p) = (d_1, \ldots, d_m)$, with $d_i = \sum(w_i(v_1, v_2), \ldots, w_i(v_{k-1}, v_k))$ and $i = 1 \ldots m$.

It is worth noting that this definition of the distance of a path only holds when the *metric* M_i associated with the weight components w_i is an *additive* metric (*e.g.*, representing the latency). Some metrics, such as the loss probability or bandwidth, exhibit *multiplicative* or *concave* behavior [Wang 1999]. Multiplicative metrics usually consider probabilistic properties, *e.g.*, the loss probability over a link. Concave metrics usually consider the minimum weight value along the path and can express the behavior of link characteristics such as the bandwidth, for which a hard limit on a link affects the entire path. In this thesis, we will however only consider additive metrics, as some of the most challenging and relevant problems arise when considering such types of metrics [2].

While using graphs to model computer networks seems very intuitive, it is nevertheless a powerful abstraction method. Several relevant routing problems may be formally studied within this formal framework, relying on the notions and theorems made available. One such problem is the notorious *Shortest Path Problem* (SPP).

2.2 Shortest Path Computation

While computing multicriteria paths is an interesting topic with numerous applications, interesting use-cases already arise when considering a single criterion (*i.e.*, when $m = 1$). For example, most standard routing protocols rely solely on a single additive metric to compute forwarding paths between routing. In addition, most multicriteria path computation algorithms rely, at their core, on classic monocriteria path algorithms. Thus, in this section, we will first discuss the SPP in a mono-metric context, before discussing its multi-metric variant.

2.2.1 Mono-metric Shortest Path Computation

In this subsection, we will present the most well-known and practically relevant mono-metric path computation problem, and discuss the two main solutions to these problems, Dijkstra's algorithm, and the *Bellman-Ford-Moore* (BFM) algorithm.

As aforementioned, monocriteria path computation is a central topic in routing. Indeed, standard intra-domain routing protocols only consider a single additive metric, called the IGP cost, or simply cost. This cost is usually representative of the bandwidth of the link, but can in practice be set at an arbitrary value by the network's operator, *e.g.*, to reflect design choices or operational costs. Data packets then follow the paths whose distance between the given source and destination is minimal.

[2] Some authors note that multiplicative metrics may be transformed into additive ones by taking their logarithm [Goel *et al.* 2001].

Since this metric is not necessarily correlated to the links' physical properties, the computed paths do not aim to meet specific quality of service requirements. They are said to provide *best effort* delivery. Despite their lack of guarantees, these paths are often sufficient for ordinary traffic and are thus used for the vast majority of the flows transiting through a network. In addition, because they rely on a single additive metric, these paths can be computed very efficiently even in fairly large networks, by solving the well-known *single-source shortest path* problem.

The SPP is a prominent problem in graph theory [Madkour *et al.* 2017, Schrijver 2012]. As its name suggests, solving SPP consists in finding a shortest path, *i.e.*, a path with minimal distance, between two nodes or sets of nodes in a given graph. The SPP usually considers an additive metric. The most relevant variants of the SPP problem within computer networks are the *all-to-all* and *one-to-all*.

Solving the one-to-all SPP consists in finding the paths with minimal distance from one source node s to all other nodes t in the graph. The problem is defined more formally in definition 2.2.1 :

Definition 2.2.1 (SPP)**.** *Given a weighted graph G, solving SPP consist in finding a path $p \in P(s,t)$ such that $d(p) = min\{d(q) \ : \ q \in P(s,t)\}, \forall t \in V$.* [a]

[a] In the algebraic routing context, this problem is studied by considering a (min, +) weight algebra (minimizing a sum function) [Sobrinho 2002].

We will denote $d^*(s,t)$ the shortest distance between s and t. While the shortest distance is unique (recall that $m = 1$), there may be several shortest paths, which we will denote $P^*(s,t)$. The all-to-all SPP (also referred to as *All Pair Shortest Path* (APSP)) consists in finding a shortest path between all pairs of nodes in the graph. Note that APSP may be solved by solving the one-to-all SPP for each node as a source.

Notice that while negative weights do not necessarily impede the computation of shortest paths, the latter could lead to the presence of *negative cycles* within the graph, *i.e.*, a cycle whose distance is negative. Such cycles render the notion of shortest paths meaningless (or rather, shortest paths are not well-defined), as a shorter path may always be found by traversing the negative cycle a sufficient amount of times [3].

To compute shortest paths, most standard algorithms solving SPP (which we will call *Shortest Path Algorithmss* (SPAs)) maintain the current best distances known towards all given nodes within the network. The best current distance $dist[u]$ is initialized at ∞ for each node u, except the source, for which it is set at 0. The graph is then explored, and the best distances are updated through a process called *edge relaxation* [Cormen *et al.* 2009].

The relaxation process is a simple one, but it is the most important aspect of SPAs. Relaxing an edge (u,v) consists in checking whether the latter improves the current best distance to v, *i.e.*, if $dist[v] > dist[u] + w(u,v)$. If so, the current best distance to v is updated accordingly. The distance to u is said to have been *extended* by the edge (u,v).

[3] Interestingly, computing the shortest path that avoids negative cycles is an NP-Complete problem, as solving the latter also solves the well-known *longest path problem* [Chekuri 2021].

In order to find a shortest path, edges must have been relaxed in a specific sequence [4]. Thus, the order in which edges are relaxed heavily impacts the overall complexity of the underlying algorithm, as the latter could be relaxing edges in a pathological ordering before eventually stumbling upon the correct sequence. For example, a poorly designed SPAs (*e.g.*, relaxing edges randomly) may first discover the highest possible distance, and infinitesimally improve it throughout its execution, which may lead it to consider all existing paths.

The efficiency of SPAs thus mainly resides in the *order* in which edges are relaxed. SPAs try to prevent the exploration of edges that are not relevant to the shortest path computation through clever relaxation orderings. Most of the time, SPAs rely on the *sub-path optimality* property to that effect [5]. The sub-path optimality property refers to the fact that subpaths of shortest paths are shortest paths. Property 2.2.1 defines this property more formally.

Property 2.2.1 (Subpath optimality)**.** *Let $p(v_1, v_k) = (v_1, v_2), \ldots, (v_{k-1}, v_k)$ be a shortest path from v_1 to v_k. Then $p(v_i, v_j) = (v_i, v_{i+1}), \ldots, (v_{j-1}, v_j)$, with $1 \leq i \leq j \leq k$ is a shortest path from v_i to v_j.*

Consequently, there is no need to try to extend paths that are not shortest paths themselves. This property may be leveraged, through an adequate graph exploration, to reduce the number of relaxation operations to be performed. Both of the most well-known SPAs, the *Dijkstra* and the *Bellman-Ford-Moore* algorithms, use these concepts and properties to compute shortest paths efficiently.

2.2.1.1 Bellman-Ford-Moore

The BFM algorithm [Bellman 1958, Ford 1956, Moore 1959], often called simply Bellman-Ford and published around 1958, is one of the most well-known SPA and is at the core of our contribution BEST2COP detailed in Chapter 4. BFM works by relaxing, during each of its iterations, all edges in the graph (in arbitrary order), discovering shortest paths by increasing number of hops (or edges). By relaxing all edges, BFM will end up extending, during iteration k, the shortest paths of $k-1$ edges found at the previous iteration, thus leveraging the subpath optimality property. Notice that since a path of $k+1$ edges may have a shorter distance than a path of k edges, the distance (or *label*) associated with a node may be improved (or *corrected*) at any point during the remaining iterations. Thus, BFM is said to be a *label-correcting* algorithm.

An interesting property of the exploration scheme of BFM is that the distance of a shortest path composed of k edges will be found by the end of the k^{th} iteration [Thulasiraman *et al.* 2016]. We will refer to this property as the Iteration-Hop relation or I-HOP relation. This relation has been exploited several times over the years [Guerin *et al.* 1997, Cavendish & Gerla 1998]. Considering this, the algorithm stops (at worst) at the $|V| - 1^{th}$ iteration, as paths composed of more than $|V| - 1$ edges necessarily contain cycles and thus cannot be shortest paths when considering strictly

[4] More precisely, to find a shortest path $p = e_1, \ldots, e_k$ with $e_i \in E$, the edge relaxation sequence must include $e_1, \ldots, e_k$, although not necessarily consecutively [Cormen *et al.* 2009].

[5] Also referred to as *isonoticity* when considering routing algebras, or substructure optimality.

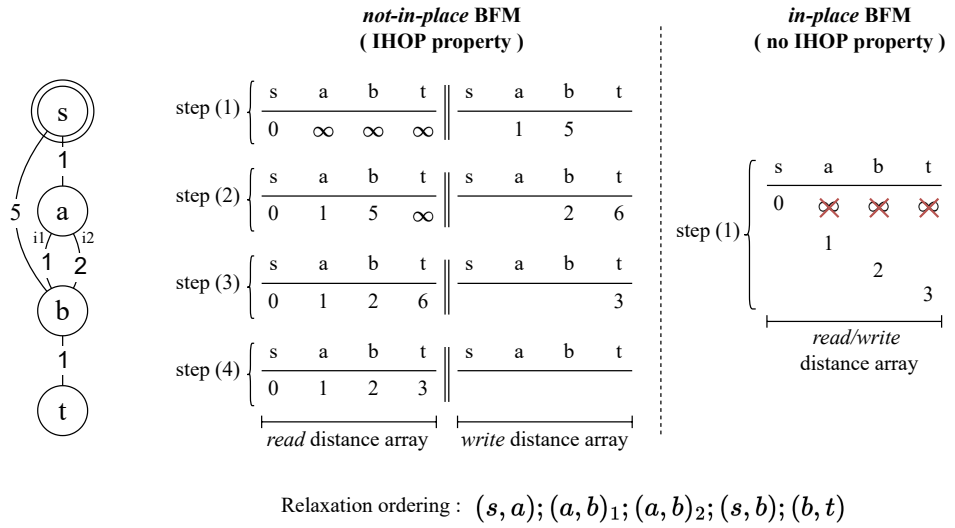


FIGURE 2.2: Illustration of the in-place and not-in-place variant of BFM. Here, the edge relaxation ordering allows the in-place variant to find the shortest paths in a single iteration, while the not-in-place variant always requires a number of steps equal to the diameter of the graph.

positive weights [6]. Thus, the overall worst-case complexity of the BFM algorithm is $O(|V| \times |E|)$. The algorithm may be terminated before the $|V|^{th}$ iteration if no new best paths were found.

Note that the I-HOP relation only holds when considering a *not-in-place* implementation of BFM, but not the *in-place* one. The difference between both approaches is illustrated in Fig. 2.2. The order in which edges are relaxed is shown at the bottom of the figure. In the *not-in-place* variant, distances discovered at the current iteration are placed within a temporary array, and not considered until the following iteration. For example, during step (1) of Fig. 2.2, although the edges $(a,b)_{1,2}$ are relaxed after the edge (s,a), the newly learned distance to a is not considered until the *next* iteration when following the not-in-place implementation method. This may lead to extending paths that have already been rendered obsolete by distances discovered during the current iteration. For example, the path to b of distance 5 is extended to t, although a better distance to b (2) has already been found. However, the I-HOP property holds: the path to b of distance 5 discovered during the first iteration is indeed the shortest path of 1 hop at most. Similarly, the path to t of distance 6 is the shortest path of 2 hops at most. Oftentimes, the I-HOP property is not important when computing paths. In addition, using the not-in-place variant always requires a number of iterations equal to the diameter of the graph (*i.e.*, the length of the longest shortest path). Thus, the *in-place* implementation, more efficient, is often preferred.

When considering the in-place implementation, the new and improved distances are directly placed within the distance array currently being considered. They may thus be extended during the same iteration. Note that the behavior of BFM then becomes a bit more erratic and the I-HOP relation, in particular, is lost. In Fig 2.2, the path of distance

[6] Note that BFM can use this property to detect negative cycle, by checking whether distances are still being improved past the $|V|^{th}$ iteration. As such, it is one of the few algorithms that can handle the presence of negative cycles in the graph.

6 to t, for example, is never explored. As one can observe in Fig 2.2, there is no relation between the number of hops in the path and the iteration at which it was found. However, if the relaxation order is favorable, this implementation method may drastically reduce the overall number of iterations required. In the example considered, the relaxation order allows BFM to find all shortest paths in a single iteration when using the in-place variant.

Indeed, it is worth noting that while BFM requires to consider all edges at each iteration, the order in which said edges must be considered is arbitrary. This led to two distinct interesting schools of thought regarding the optimization of BFM. The first one aimed at designing a subtler relaxation order to reduce the maximum number of iterations, which was reduced to $|V|/2$ with an average of $|V|/3$ [Bannister & Eppstein 2012, Cormen *et al.* 2009] (note that these optimizations require the in-place variant). The other one leveraged the fact that no order was required by relaxing edges concurrently using multi-threaded architectures (although care must still be taken to ensure that no data-race occurs) [Papaefthymiou & Rodrigue 1997].

Finally, a fairly well-known optimization of BFM resides in the observation that there is no point in even considering an edge (u, v) if the distance towards u was not improved during the past iteration. Thus, one may maintain, during the iteration, the list of nodes for which a better distance was found, and only consider these nodes at the following iteration. For example, in Fig 2.2, when considering the not-in-place variant, only considering edges originating from b and t during step (3) is sufficient as no better distance to other nodes has been found.

Even more drastic, each node for which a new distance has been discovered may be inserted in a FIFO queue and considered one by one until the queue is empty. This optimization, proposed (originally) by Moore, is sometimes also referred to as the Shortest Path Faster Algorithm [Moore 1959]. However, the use of a priority queue makes the leveraging of multiple threads less natural and more complex to implement.

We will see that our contribution, BEST2COP, remains close to BFM while still using a similar optimization technique as the Shortest Path Faster Algorithm to prevent (some) useless relaxations. We thus also benefit from the highly parallelizable nature of BFM. In addition, we will see that minor modifications allow us to easily prevent data-races when relaxing edges concurrently.

It should be noted that we consider here (and in our contribution BEST2COP) a somewhat *centralized* variant of BFM, where the node running the algorithm possesses a full, static view of the network as a weighted graph. However, BFM may also be implemented in a distributed fashion [Bertsekas & Gallager 1992]. When using distributed-BFM, each node s maintains for each t a vector $(d(s,t), u)$ which contains the currently best-known distance to t and the neighbor u used to reach it. This vector is sent to the node's neighbors, which in turn may update their best known distances accordingly, by adopting or not the received distance. As computed distances are disseminated through the network, the algorithm constructs the shortest paths.

Although distributed BFM does possess some advantages compared to the standard variant (in particular, in networks where the capacity of the nodes is limited), distributed BFM exhibits several well-known issues preventing its use in more complex, large scale networks. For example, a pathological ordering in the message exchange may lead to a path exploration that does not leverage subpath optimality and result in an exponential message complexity

(when the message exchange is asynchronous) [Lambert *et al.* 2009, Awerbuch *et al.* 1994]. This effect is usually simply referred to as *path exploration* within the literature.

2.2.1.2 Dijkstra

The Dijkstra algorithm [Dijkstra 1959], published in 1959, is probably the most well-known SPA and the one used in most cases. Although Dijkstra's algorithm relies on the same core concepts as BFM, its relaxation order is different and overall more effective than BFM, as it better harnesses the subpath optimality property, by ensuring to only extend paths that are already known to be shortest. However, it can be noted that Dijkstra's algorithm does not support negative weights, even without the presence of negative cycles.

Dijkstra's algorithm maintains a priority queue composed of the nodes that have been reached, ranked according to their distance from the root node (initially, only the source node is put within the queue, with a distance of 0). The algorithm then selects the node with the lowest distance and relaxes all the (outgoing) edges of this node, updating the queue if the current known distances are improved upon. The algorithm then repeats this process until the queue is empty.

An interesting property to note is that once a vertex u is extracted from the queue, it is *settled*, *i.e.*, its current distance cannot be improved and is thus the best possible distance to reach u. Indeed, as all other nodes exhibit a higher distance, the only way to improve the current best distance to u later on is through an edge exhibiting a negative cost, which is forbidden by hypothesis. Thus, conversely to BFM, Dijkstra's algorithm is often referred to as a *label-setting* algorithm.

Because Dijkstra's algorithm always chooses the node with the lowest distance, it is often said to be a *greedy* algorithm. Its greedy nature allows it to reach a better overall complexity. However, this complexity is less straightforward than BFM, as it heavily depends on the data structures used. Agnostically of implementation choices, all $|V|$ nodes in the graph will be extracted (and inserted) from the queue. Each time an edge is considered, a distance may be relaxed. Thus, if we denote x, i, and r the cost of the extraction, insertion (within the queue) and relaxation respectively, the overall complexity of Dijkstra's algorithm becomes $\mathcal{O}((x+i) \times |V| + r \times |E|)$.

Choosing adequate data structures is fairly complex, as the resulting performance also depends on the structure of the graph and the implementation overhead [Chen *et al.* 2007]. The most commonly used structure is a binary heap, which results in a complexity of $\mathcal{O}((|E|+|V|)log(|V|))$. This complexity can be improved to $\mathcal{O}(|E|+|V|log(|V|))$ by using a *Fibonacci heap* or similar intricate structures [Fredman & Tarjan 1987, Takaoka 2003]. Note that in practical cases, the binary heap may outperform the Fibonacci heap [Larkin *et al.* 2014].

Dijkstra's algorithm has been extended, most notably through bidirectional search variants [Dantzig & Thapa 2003, Goldberg *et al.* 2006] (when only a single target is considered) or through goal-oriented variants such as the A^* algorithm [Hart *et al.* 1968]. It may be noted that Dijkstra's greedy nature does not enable to parallelize it as naturally as BFM, although some parallelization methods do exist [Crauser *et al.* 1998a].

2.2.1.3 Retrieving and representing shortest paths

One may have noticed that both BFM and Dijkstra's algorithms mostly consider and manipulate *distances* rather than actual paths: as presented, both algorithms return the optimal distances to all nodes within the graph, but not the associated paths. Actively listing all shortest paths may not be tractable in practice. Indeed, while the minimal distance between two nodes is unique by definition, several shortest paths may possess this distance. In fact, all $(|V|-2)!$ paths between two nodes (approximately) may be shortest paths. As such, we will now often refer to the object manipulated by path computation algorithms as distances rather than paths, to make this distinction clear.

To retrieve computed paths, SPAs usually remember, when relaxing an edge (u, v), that u is the predecessor of v in the shortest path. Thanks to the subpaths optimality property, this information is enough to compute the path from a target t to the source s through a recursive backtracking reconstruction. Note that when several shortest paths exist, several predecessors may be remembered[7]. The lists of the predecessors of all nodes thus form either a *shortest path tree* or a shortest path *Directed Acyclic Graph* (DAG), within which any path from s to a given node u is a path with minimal distance.

By walking through the shortest path DAG, it is thus theoretically possible to retrieve and make use of all shortest paths between two pairs of nodes. In computer networks, in particular, utilizing several if not all shortest paths (a feature known as *Equal Cost Multi Paths* (ECMP) [Hopps 2000]) is crucial to allow for better resiliency, scalability, and load balancing.

Dijkstra's algorithm serves as a basis in several routing protocols [Moy 1998], where it is performed by each node to compute the shortest paths to its peers. However, despite the seemingly greater efficiency of Dijkstra's algorithm, we have seen that BFM possesses an overall simpler structure and interesting properties. Consequently, both algorithms are often used as a basis for many other path computation algorithms which iterates upon them, including multi-criteria path computation algorithms.

2.2.2 Multi-metric Path Computation

Path computation on multi-criteria graphs is a generalization of the mono-criterion variant presented in the previous section. Computing multicriteria paths also possess numerous practical applications. Indeed, we have seen that several metrics may be relevant when computing paths within computer networks, *e.g.*, latency, bandwidth, jitter (delay variation), or loss rate. These problems may however require vastly different approaches compared to their mono-criterion counterparts. In this section, we will present the new challenges brought by multi-criteria paths computation and the associated concepts, as well as quickly present the various categories of algorithms used to tackle such problems.

2.2.2.1 General concepts

Network metrics may follow different characteristics. They may be concave, additive, or multiplicative. The relevant problems that consider a concave metric along a multiplicative or additive one (*e.g.*, the bandwidth and the delay) possess a computational complexity

[7] Although several shortest paths towards v may thus be remembered, they are not all extended separately. Solely their unique, common distance is considered and extended.

close to standard mono-criterion path algorithms. For example, guaranteeing a minimal bandwidth may be performed by simply pruning inadequate links. Finding paths of low delay with maximal minimal link bandwidth (*i.e.*, the least detrimental bottleneck) may also be done efficiently through shortest-widest or widest-shortest path computation [Wang 1999].

However, it may be interesting to consider several additive and/or multiplicative metrics. For example, some premium flow may require paths that offer strong latency guarantees while minimizing an additive metric such as the IGP cost.

Perhaps surprisingly, considering such a set of metrics drastically impacts the computational complexity of the path computation problems.

Let us, for example, consider one of the most traditional multicriteria path computation problem, *Multi Constrained Path* (MCP). MCP consists in finding a path p that respects a set of m constraints. Paths that satisfy all given constraints are called *feasible*.

Definition 2.2.2 (Multi Constrained Path (MCP))**.** *MCP consists in finding, given a graph $G = (V, E, w)$ and a set of constraints $c_i (1 < i \leq m)$, a path p that satisfies all the constraints, i.e., such that $d_i(p) \leq c_i$ with $1 < i \leq m$.*

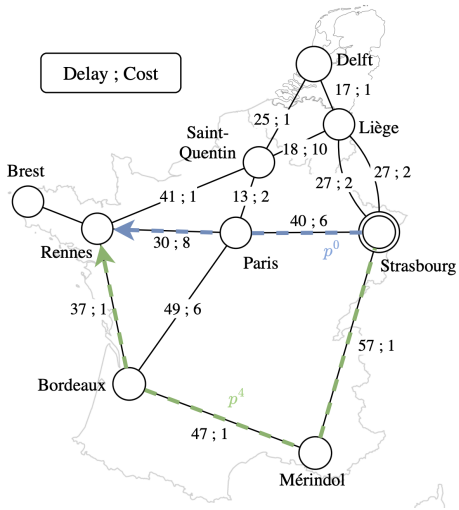
While MCP is a fairly natural problem when considering multicriteria networks, it is known to be NP-Complete if at least two of the metrics being considered are additive and/or multiplicative. This proof was published by (amidst others) Wang and Crowcroft and consists in a reduction from the PARTITION problem [Wang 1999].

This drastic increase in computational complexity can be explained by the fact that, conversely to monocriterion path computation, there is no natural way to define a total ordering among a set of multicriterion paths. Consequently, the notion of optimality becomes blurry, and choosing which path to extend becomes less straightforward.

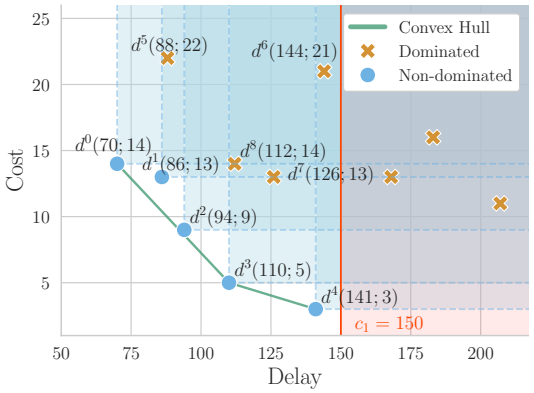
For example, let us consider two paths $p_1(s, v)$ and $p_2(s, v)$ with weight vectors $(1, 10)$ and $(10, 1)$ respectively. Only extending p_1 may lead to violate a constraint c_2 on the second metric, if said constraint is close to 10. The same reasoning holds for p_2 and a constraint c_1. Since there is no way to (non-arbitrarily) order these weight vectors, both can be considered optimal and have thus to be extended. Such paths (or rather, their distance vectors) may be referred to as *pareto-optimal*. Pareto-optimal distance vectors are thus non-comparable. The number of non-comparable distances may grow exponentially with respect to the size of the network, resulting in the NP-Completeness of the problem.

This behavior can also be observed in *Multi Constrained Optimal Path* (MCOP), a generalization of MCP. Solving MCOP consists in finding a path satisfying $m - 1$ constraints and, among the feasible paths, finding the one optimizing the remaining criterion.

Definition 2.2.3 (Multi Constrained Optimal Path (MCOP))**.** *MCOP consists in finding, given a graph G, a criterion to optimize j, and a set of constraints $c_i, 1 < i \leq m, i \neq j$, a path p which satisfies the set of constraints, while optimizing d_j, i.e., $d_j(p) = d_j^*$ and $d_i(p) \leq c_i, 1 < i \leq m$, with d_i^* the optimal distance for the i^{th} criterion.*



(A) Example (hypothetical) network spanning over France, Belgium, and the Netherlands. Edges are labeled with their delay and cost. The cost is tuned arbitrarily. The delay is representative of the geographical (and here, euclidian) distance between cities.



(B) The distances of (nearly) all simple paths from Strasbourg to Rennes plotted in the delay-cost space. Some distances have been removed for the sake of readability. Distances in the red zone violate the delay constraints. Distances within at least one blue zone are dominated. Other distances (shown in blue) are non-dominated solutions.

FIGURE 2.3: Example illustrating the notion of Pareto front and dominance.

One of the most practically relevant MCOP problems with regards to computer networks occurs when $m = 2$, and is usually referred to as DCLC, where the delay must respect a given constraint, and the cost must be optimized. This problem is also sometimes called the Restricted Shortest Path problem or the Constrained Shortest Path problem in the literature. DCLC is said to be intractable (*i.e.*, requires an exponential number of operations with respect to the size of the graph) and proved to be NP-Complete [Wang & Crowcroft 1996, Hansen 1980] [8], but has direct practical applications, as network operators may want to guarantee a strict upper constraint on the latency of a path while optimizing the IGP cost to respect their design choices.

Even though MCOP (and, subsequently, DCLC) does possess an optimization objective, paths reaching a given node v may still be non-comparable. Figure 2.3 illustrates this issue. For the sake of simplicity, we will consider $m = 2$ in the following explanations. We will denote w_1 the delay, and w_2 the cost.

[8] The history behind the proof of NP-Completeness of DCLC deserves a little aside. Garey and Johnson, often cited regarding this matter, attribute the proof to Meggido in 1977 through a reduction from PARTITION, but the reference is listed as a private communication [Garey & Johnson 1979]. The proof is also often attributed to Hansen, in 1980. However, Hansen discussed the *intractability* of a parent problem of DCLC (although the argument could be adapted to DCLC). Hansen warned that it did not constitute a proof of NP-Completeness [Hansen 1980]. Finally, Serafini (in 1986) proved the NP-Completeness of the associated *decision problem*, *i.e.*, deciding *if* a path respecting two given constraints exists, through a reduction from KNAPSACK [Serafini 1987]. The reduction from the decision problem to DCLC being straightforward, this proved that DCLC is NP-Hard. Wang and Crowcroft (in 1999) would then prove that the decision problem associated to MCP (with $m \geq 2$) is NP-Complete [Wang & Crowcroft 1996] for additive and multiplicative metrics through a reduction from PARTITION. It should however be noted that a proof of intractability is stronger than NP-Completeness, as NP-Complete problems could be solved in polynomial time if $P = NP$ was to be proven true.

Figure 2.3 shows a hypothetical network comprised of 9 nodes, spanning multiple countries. Each link is characterized by a weight vector $(delay; cost)$. The cost is tuned arbitrarily while the delay is representative of the actual distance between the corresponding cities. Let us consider the DCLC problem, from Strasbourg to Brest with $c_1 = 150$. Several simple paths to Rennes may be extended to reach Brest, which offer different compromises between the delay and the cost. An intuitive way to visualize their distance is in the *delay-cost space*, shown in Figure 2.3b.

Distance d^0, for example, offers the best delay, but at a high cost. Conversely, d^4 offers the best cost, but with a high delay, which nears the constraint. Finally, d^1, d^2, and d^3 offer alternative compromises. Although d^4 is the distance whose associated path solves this instance of DCLC (with $c_1 = 150$) when considering Rennes as the destination, it does not necessarily solve the same instance of DCLC when considering Brest as the destination[9]. This depends on the delay necessary to reach Brest from Rennes. Distance d^4, which seems like the best candidate as of now, may thus very well exceed the constraint c_1 if the edge $(Rennes, Brest)$ is discovered to have a delay $w_1(Rennes, Brest) \geq 10$. When exploring the graph, the weights of such further edges yet to be discovered is unknown. Thus, solely extending the DCLC distances of nodes along the way may not be sufficient, as it could lead to a path exceeding the constraint for the destination node. To ensure that a solution is found, additional distances should be explored. Here, distance d^3 should also be discovered and maintained, in case d^4 ends up violating the constraint. Following the same reasoning, distances d^2, d^1, and d^0 should also be discovered and maintained, as any one of them could become the DCLC distance to Brest (or any further node) depending on the weights of the remaining links. The number of distances to keep track of and extend may grow quickly, and is the reason being the intractability of DCLC.

Notice, however, that some distances may be pruned from the exploration. Since only strictly positive weights are considered, there is no point in extending distances that already violate a constraint. Furthermore, all orange distances visible in Fig. 2.3b are worse on *all* criteria than at least one of the distances shown in blue. They are said to be *dominated*. For example, d^7 possesses a worst delay and cost than d^2 to reach the same node (Rennes). Since individual criteria normally still follow the substructure optimality principle, d^7 cannot become a better distance than d^2 after being extended [10]. The distance d^7 is said to be dominated by d^2. Fig 2.3b represents this dominance relationship. All distances within a blue area are dominated by the blue distance from which the blue area originates. This dominance relationship, central to multicriteria path computation, is formalized in Definition 2.2.4.

Definition 2.2.4 (Dominance)**.** *A distance d dominates a distance d' if $d_i \leq d'_i, \forall i$. A distance d for which no distance d' exist such that d' dominates d is said to be* non-dominated*,* pareto-optimal*, or* efficient*. A path p is dominated (resp. non-dominated) if its distance $d(p)$ is dominated (resp. non-dominated).*

[9] This can be seen as the loss of subpaths optimality (or isotonicity) when considering MCOP. Indeed, extending the DCLC paths to Rennes does not necessarily enable finding the DCLC path to further node when considering the same constraint. Consequently, a MCOP path for a given set of constraints is not necessarily composed of MCOP paths with the same constraints.

[10] More intuitively, as both distances can be extended by the same edges, there is no possible for the dominated path to catch up with the non-dominated one.

The set of non-dominated (also called *efficient*) distances, shown in blue in Fig. 2.3b, forms the *Pareto front* of the solution [Deb 2005]. We denote the *Pareto front* by $\mathcal{P}$. Distances lying in the green area form the *convex hull* of the Pareto front, *i.e.*, the smallest convex set of distances (or here, points) enclosing the Pareto front. Some path computation algorithms distinguish between distances lying on this convex set or not, as we will see in Section 3.3. To solve DCLC, (or MCOP/MCP), it is necessary to remember and extend *all* non-dominated distances that form the Pareto front, *i.e.*, all distances shown in blue in Fig 2.3b. As such, while solving DCLC requires returning a single path per destination, the entirety of the Pareto-front must still be explored. Dominated distances can however be pruned from the exploration.

At worst, all distances are non-dominated, leading to an exponentially large Pareto-front with $|\mathcal{P}| \approx (|V| - 2)!$, making the problem intractable. In addition, maintaining the actual Pareto front (*i.e.*, extracting it from all the dominated and non-dominated distances discovered) may be costly (at worst quadratic with respect to the number of distances), although various techniques exist exhibiting different trade-offs. In particular, when $m = 2$, sorting distances according to one metric (*e.g.*, by increasing delay) allows extracting Pareto-front in linear complexity with respect to the size of the Pareto-front [Kuipers & Van Mieghem 2005].

It should be noted that, while theoretically possible, Pareto fronts of exponential sizes have a low chance of occurring in practice within computer networks. Indeed, it has been observed that the number of non-dominated distances is relatively small in practice [Müller-Hannemann & Weihe 2006, Van Mieghem & Kuipers 2003]. The exponential nature of this problem is in reality very reliant on both the structure of the graph and the weights of each link. For example, strongly correlated weights lead to a small Pareto-front, as a distance with low delay will also possess a low cost and likely dominate all other distances [Mote *et al.* 1991].

The cardinal of $\mathcal{P}$ can also be reduced depending on the granularity of the metrics. Intuitively, continuous metrics offer a wider spectrum for distances to fall into. Conversely, discrete metrics reduce the number of possible distances, and thus the maximum number of dominated distances. This is particularly important in computer networks, as weights are encoded within a router's memory within a fixed number of bits, holding a fixed number of values (typically, 2^{16} [Moy 1998]). In this case, one can bound the number of non-dominated distances to $|\mathcal{P}| \leq \frac{\prod_{i=1}^{m} L_i}{\max_{1 \leq i \leq m} L_i}$, with L_i the number of values of the i^{th} metric (or its associated constraint c_i) [Van Mieghem & Kuipers 2003].

Thanks to both aforementioned effects, the number of non-dominated distances is expected to be fairly low when considering realistic computer networks, for which weights are often correlated and discrete. In particular, Müller-Hannemann and Weihe found that, for certain categories of graphs with characteristics that could be found within operators networks, the growth of the Pareto-front is polynomial with respect to the size of the graph. Furthermore, their experimental evaluations have shown that the Pareto front is in practice often small (whatever the graph), and that an explicit search of the Pareto front may be viable even though the worst-case complexity remains exponential [Müller-Hannemann & Weihe 2006].

Because of the practical relevance of DCLC (and related multicriteria problems), several

solutions have been designed, some dating as far back as 1980 [Hansen 1980]. The strange duality of DCLC (an intractable problem rarely exhibiting exponential complexity in practice) lead to three main axes of research : *Heuristics*, *Approximation schemes*, and *Exact methods* [Garroppo *et al.* 2010].

While a panel of these solutions will be reviewed more extensively in Chapter 3, we will here present some of the general paradigms the latter adopt when solving multicriteria problems.

2.2.2.2 Heuristics

Heuristics aim to avoid the exponential nature of this problem. Their main goal is to try to solve the problem in a reasonable amount of time. While the results are usually shown to be close to the optimal in most cases, they do not provide any guarantee regarding the distance to the optimal solution[11]. Their main advantage usually resides in their very low execution time.

Many heuristics, for example, try to establish a global order between multicriteria distances to reduce the problem back to its monocriterion variant. This may be achieved by replacing the weights vectors by a linear combination of its components, *i.e.*, $w = \alpha_1 w_1 + \alpha_2 w_2 + \cdots + \alpha_m w_m$. The main drawback of this method is finding appropriate α values. In addition, the resulting w may not hold any semantic meaning. While it may serve as an indicator to guide the search, the loss of information makes it impossible to assess the multicriteria quality of the underlying path solely through the resulting mixed metric [Wang 1999]. Other algorithms may choose to only explore part of the Pareto front or rely on duality theory [De Neve & Van Mieghem 2000, Juttner *et al.* 2001a].

Heuristics may return distances that either do not optimize the objective or that do not satisfy the constraints, depending on their designs. As such, while they do offer an efficient way to find a solution to multicriteria path problems, their lack of guarantees regarding how far the solution returned is to the optimal value may make them unfit for intricate, strict traffic engineering purposes.

2.2.2.3 Exact methods

Exact methods aim to provide the optimal solution to the MCOP problem. As a result, they are expected to be computationally expensive, as they have to explore the entirety of the Pareto front. However, it should be reminded that even such schemes may prove to be fairly efficient in practice on realistic instances. Exact schemes often originate from an extension of Dijkstra's algorithm or BFM, and thus usually fall within the label-setting or label-correcting families.

Multicriteria label-setting algorithms use a distance-based priority queue, *i.e.*, where each individual non-dominated distance is added to the queue. These distances are sorted in a way that ensures that an extracted (and so, extended) distance is necessarily non-dominated with respect to the following ones in the queue. By ensuring that the distances extended are necessarily non-dominated, the label-setting property is enforced.

[11] Heuristics are sometimes defined as including such type of controlled approximation. In this thesis, we consider heuristics and approximation mutually exclusive

Multicriteria label-correcting algorithms do not ensure that elected labels are settled. They may for example work in a BFM-like fashion, or implement a simpler priority queue than their label-setting peers, for example following a simple FIFO ordering. Compared to label setting algorithms, this may lead to explore more (and, more importantly, dominated) distances. However, the simplicity of the queue drastically reduces the associated management overhead. As such, both design choices offer interesting performances [Hribar *et al.* 1995, Raith & Ehrgott 2009].

Other exact methods exist, such as *ranking*, which rely on k-shortest paths algorithms [Martins & Santos 1999], and the *two-phases* method [Mote *et al.* 1991], which discovers distances through different means, depending on whether the distances lie on the convex hull of the Pareto front or not.

Using exact methods allows enforcing strong SLAs. However, although they may be efficient in most realistic use cases, they are not protected from peculiar or pathological patterns arising within graph structures or weighting schemes, which may lead to a drastic increase in computation time. Indeed, some constructions exhibit a Pareto front of exponential size with respect to the number of nodes, even with a bounded degree and only two criteria [Hansen 1980, Breugem *et al.* 2017].

2.2.2.4 Approximation methods

Approximation methods offer an interesting compromise between exact methods and heuristics. Approximation methods do not guarantee to return the optimal solution but can bound the distance between the returned solution and the optimal one. More precisely, $(1+\varepsilon)$-approximations guarantee that the solution returned is optimal by a factor of $(1+\varepsilon)$, usually with regards to the optimization objective.

An approximation scheme taking ε as input and finding a $(1+\varepsilon)$-solution in polynomial time with respect to the size of the input is referred to as a *Polynomial Time Approximation Scheme* (PTAS). However, PTASes allow the complexity to grow exponentially with respect to $\frac{1}{\varepsilon}$, meaning that the time complexity grows rapidly as one aims to get closer to the optimal solution. More restrictive, *Efficient Polynomial Time Approximation Schemes* (EPTASes) only allows $\frac{1}{\varepsilon}$ to appear as an exponent of a constant within the time complexity. Finally, *Fully Polynomial Time Approximation Schemes* (FPTASes) require the time complexity to be polynomial in both the input size *and* in $\frac{1}{\varepsilon}$.

While not all NP-Hard problems admit FPTASes, it has been shown that DCLC and MCOP do [Hansen 1980, Tsaggouris & Zaroliagis 2009, Papadimitriou & Yannakakis 2000]. More precisely, *strongly* NP-Hard problems cannot be approximated efficiently, while *weakly* NP-Hard problems [12] can admit FPTASes. As the size of the Pareto front, which dictates the worst-case complexity of MCOP, can be bounded by the magnitude of the data[13], both DCLC and MCOP are weakly NP-Hard. This result has then been leveraged to show that

[12] Weakly NP-Hard problems admit algorithms whose time complexity is polynomial in the magnitude of the input data.

[13] Recall that the size of the Pareto front $|\mathcal{P}|$ is bounded in the number of values the distances can take, which can be expressed as the largest numbers within the input, once distances are scaled to integers. Exploring said Pareto front can thus be done in a time complexity depending on the magnitude of said data, *i.e.*, in pseudo-polynomial time

FPTASes can be designed [14].

The ability to solve MCOP through FPTASes generated a vast interest within the research community regarding the design of such solutions, both in operational research and networking. Indeed, FPTASes allow enforcing strict SLAs (with a bounded error margin), while offering polynomial time complexity. Consequently, numerous approximation schemes for MCOP exist. The latter will be reviewed more thoroughly in Section 3. In particular, our contribution, BEST2COP, falls within the FPTAS category. However, we do not approximate the optimization objective by a factor $1 + \varepsilon$, but rather the constraint. The design choices of BEST2COP are explained more thoroughly in the dedicated chapter, Chapter 4.

In this section, we have discussed the computation of shortest paths within weighted graphs. Although such computations can be performed very efficiently when considering a single metric, the computational complexity becomes exponential with respect to the size of the graph when considering multiple additive and/or multiplicative metrics. Indeed, considering multiple metrics breaks the global ordering between distances and requires maintaining and extending a potentially exponential number of distances when exploring the graph. These problems are however practically relevant and have generated vast research effort. In particular, *Shortest Path Computations* (SPCs) lie at the core of computer networks, which rely on the latter to route data packets.

Routing, be it best-effort or not, is indeed centered around the computation of shortest paths, either mono or multicritera. Nevertheless, such computations, by themselves, are not sufficient. First, the information required may not be known beforehand by routing devices. Second, computed paths must be encoded and handled in a way suited and optimized to route the traffic. Consequently, shortest-path algorithms are used in conjunction to *routing protocols*.

2.3 Routing & Routing Protocols

Routing consists in selecting the (best) paths onto which traffic must be forwarded for a given destination. It is a central and crucial aspect of computer networks. Within computer networks, routing is performed by routers, dedicated devices in charge of both routing and forwarding data. In this section, we will describe the general architecture of routing devices. We will detail how routers construct, compute and install forwarding paths to forward data efficiently. We will see that routers rely on routing protocols to exchange and compute the necessary routing information. In particular, we will see that routers may depend on *several* routing protocols, and discuss the inner working and specificities of the routing protocols relevant to our contributions.

Routers enable communication to remote devices by computing paths to the IP prefix that contains the destination IP address. More precisely, routers store relevant path information in their *Routing Information Bases* (RIBs), also called routing tables. A RIB is a list of entries, matching any given destination prefix to its associated path information. Originally,

[14] In short, these algorithms often rely on ways to reduce $|\mathcal{P}|$ by making the metric coarser through standard techniques [Sahni 1977], until the problem is solvable in polynomial time.

NOTATION	DEFINITION
$IH(D)$	IGP Next-Hops for destination D
$L(D)$	Outgoing interfaces for a given destination D
$G(D)$	Gateway (exit point) for an external destination D
$\mathcal{A}_i$	An *Open Shortest Path First* (OSPF) area.
AX_i	i^{th} *Area Border Router* (ABR) between the backbone and area X
R	*Border Gateway Protocol* (BGP) route. Individual attributes may be expressed as a vector $\alpha + \beta =$(LOCAL-PREF, AS-PATH, MED, IGP)
$\beta(R)$	Inter-domain attributes of a BGP route R
$\alpha(R)$	Intra-domain attributes of a BGP route R
$\prec, =, \succ$	Comparison operators between BGP routes.

FIGURE 2.4: Notations introduced within Section 2.3

a router only has knowledge of networks (or prefixes) it is directly connected to, *i.e.*, its direct links and loopback addresses. Information regarding remote prefixes must however be learned.

This information can be fed to the router manually. In this case, the routing is said to be *static*. While static routing may be viable in very small networks, many networks are too large for static routing to be an option. In addition, most computer networks are highly dynamic and undergo frequent unplanned changes [Merindol *et al.* 2018], meaning that the RIBs would have to be constantly adjusted.

Consequently, *dynamic* routing is usually preferred. Dynamic routing allows routing devices to construct and update their RIBs automatically and compute the best paths to each prefix through path computation algorithms, such as the one mentioned in Section 2.2.1 or Section 2.2.2. The resulting paths are then used as forwarding paths for the traffic.

Ensuring permanently consistent dynamic routing is challenging. First, as we have seen in previous sections, path computation algorithms rely on previously acquired information (*e.g.*, a static view of a weighted graph). In practice (and, in particular, in computer networks), such information is not always known beforehand by routers. Furthermore, computer networks are (traditionally) distributed. As such, in legacy networks, there is no central entity aware of the entire topology or controlling all routing devices.

Second, forwarding is usually performed in a *hop-by-hop* fashion, meaning that each router along the path taken by a packet forwards it to the next router along the best path, called the *Next-Hop* (NH), according to its own, local knowledge and computations[15]. Even if a router may have enough information to compute the entire path to a destination, only the NH (or NHs, if ECMP is supported) towards this destination is maintained within the routing table. Thus, inconsistent routing information or even inconsistent routing decisions among routers may lead to anomalies such as forwarding loops.

Consequently, *routing protocols* have been designed to specify how routing should be

[15]Or, said otherwise, each router forward the traffic to a given destination as if the traffic originated from itself.

performed, from the way routers should exchange the topological information enabling them to compute forwarding paths, to the way this computation should be performed. Routing protocols ensure consistency among routing devices and that the RIB remains complete and up-to-date automatically even upon network events (*e.g.*, failures, maintenance, or reconfigurations).

There exist a vast array of routing protocols that follow different paradigms, both regarding topology distribution and path selection. These protocols usually consider different contexts and thus answer different challenges, for example in terms of scalability. A single router may thus rely on several routing protocols, with each routing protocol maitaining different information and populating different *local* RIBs. Note that several local RIBs may have routes to the same destination. Paths from different RIBs are discriminated through the *administrative distance*, a priority assigned to each routing protocol. For each destination prefix, the path with the best administrative distance is put within the *global* RIB.

The set of functions in charge of routing, *i.e.*, sharing the required topological information, computing the best paths, and populating the RIBs, is called the *control-plane* of the router. As the control-plane performs varied complex tasks, it is often software-based. It is however not in charge of the actual forwarding of the data.

Indeed, although the global RIB does technically possess enough information to forward traffic, it is not used directly. Rather, the minimal information required to forward data is extracted from the RIB and put within *Forwarding Information Base* (FIB) (or forwarding table), which is optimized for fast lookup.

The FIB usually only contain, for each destination prefix, the best interface (or interfaces, if techniques such as ECMP are supported) onto which the traffic should be forwarded. When traffic enters the routers, the FIB finds the most precise prefix covering the destination prefix by performing a longest prefix match, either performed through adequate data-structures or specialized hardware components (*e.g.*, *Ternary Content-Addressable Memory* (TCAM)). Once found, the packet is switched to the associated outgoing interface (as indicated in the FIB) through the switching fabric, which allows interfaces to communicate. Usually, there is one FIB per *line-card*, a physical entity managing one or several interfaces on a routing device. By being physically close to the interfaces, the FIB allows to very quickly transfer incoming data to the outgoing interface according to the information stored therein.

The part of the routers' architecture in charge of forwarding data through the mechanisms just described is referred to as the *data-plane*. The data-plane performs relatively simple tasks but must perform them fast enough to not degrade the overall throughput. Consequently, the data-plane traditionally relies on dedicated hardware, *e.g.*, *Application-Specific Integrated Circuits* (ASICs), which allows it to forward packets at *line-rate* (or wire-speed), *i.e.*, without slowing the traffic down.

It is worth noting that the past few years have witnessed the rise of efficient programmable data-planes, which allow performing subtler operations within the data-plane while still forwarding packets at line-rate. The data-plane may be programmed through specific programming languages such as P4 [Bosshart *et al.* 2014]. By mitigating the ossification of wired networks, this technology quickly gained traction both within the industry and academia and has for example been used to perform line-rate in-band telemetry or fast re-route directly in the data-plane [Kfoury *et al.* 2021, Hauser *et al.* 2021]

Once the topological information has been exchanged across all routers and that the best paths have been computed and installed within the FIB, the protocol is said to have *converged*. The convergence speed of a protocol is a critical aspect of the latter, as anomalies or non-optimal routing may occur while the protocol converges.

This convergence speed highly varies depending on the protocol. Not only does each protocol require different kinds of computation, but they may also operate at vastly different scales depending on the destinations they maintain.

In particular, a router within a given *domain*, or AS (*i.e.*, a network controlled by a sole, unique administrative entity), should, thanks to routing protocols, gain reachability knowledge of any *internal* or *intra-domain* destinations (*e.g.*, to reach other routers within its own AS). However, a router should also be able to reach *external*, or *inter-domain* destination (*e.g.*, remote prefixes within the Internet).

Maintaining a full RIB containing all external prefixes throughout the Internet raises fundamentally different challenges compared to maintaining a full RIB of purely internal prefixes, most notably regarding (but not limited to) scalability. As such, inter- and intra-domain connectivity is usually handled by two distinct protocols designed to tackle these specific challenges.

The contributions in this thesis rely on different aspects of several protocols. BEST2COP is focused on intra-domain path computation, and thus relies on and leverages intra-domain protocols and their specificities. OPTIC, on the other hand, lies at the frontier of intra- and inter-domain routing and requires understanding both types of protocols as well as the interaction that may occur between them. Consequently, in the remainder of this section, we will review the standard paradigm used in intra- and inter-domain routing, as well as the standard routing protocols used in these contexts.

2.3.1 Internal destination reachability

The reachability of internal destinations is handled by an *Internal Gateway Protocol* (IGP). An IGP ensures that all routers within a given domain (or AS) possess enough information to route packets to any intra-domain destination following the optimal paths.

Once the IGP has converged, each router should know the best next-hops to any internal destination prefix and the associated outgoing interfaces. As such, the routing enabled by IGPs may be seen as a composition of functions $Ports(D) = Int \circ IH \circ LPM(D)$. $LPM(D)$ returns the most specific prefix (found via longest prefix match) known that covers the targeted destination D. IH returns the internal next-hop used to reach a given prefix (or a set of next-hops if several shortest paths exist). Finally, Int returns, for a given set of internal next-hops, the associated outgoing interfaces. Int may also return a set of interfaces per next-hop, *e.g.*, in the presence of parallel links[16].

Recall that this composition of functions is not computed each time a packet is forwarded. Rather, it is resolved for each prefix beforehand. Solely the result is pushed within the FIB. Thus, in steady state, the FIB only has to compute the longest prefix match $LPM(D)$ for the packet's destination, before matching it directly to the outgoing

[16] In practice, a single interface is chosen when forwarding traffic through the use of a hash function applied on some fields of the packets header.

interfaces.

As aforementioned, the path a packet will take is not set by the source (in usual IP routing) but constructed through the independent decision of each intermediary hop according to their FIBs. Intuitively, this may be cause for concern as conflicting decisions may lead to anomalies even if the topological information is consistent. However, when considering intra-domain mono-criterion best paths, this is not an issue as the subpath optimality principle holds. Thus, the routing decision taken by the next hop should match the one taken by the current node if the information is consistent across routers.

To enable this routing, an IGP must thus first ensure that all routers are aware of all internal prefixes before computing the optimal internal paths towards the latter. As mentioned in Section 2.2.1, optimal paths within the context of intra-domain routing refer to paths minimizing the IGP cost, an additive metric with (strictly) positive weights assigned to each link. The IGP costs are usually set at the inversed capacity of each link but may be tuned arbitrarily by the operators to reflect design choices or operational costs.

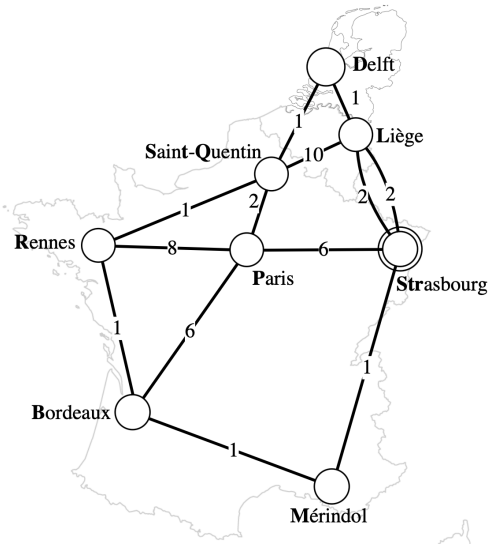
The most natural way to fulfill these requirements is to ensure that all routers within the network possess a full, up-to-date view of the entire network in the form of a weighted graph, onto which to perform shortest path computations, *e.g.*, through Dijkstra's algorithm. This is the approach taken by *link-state routing protocols*.

2.3.1.1 Link-State Routing Protocols

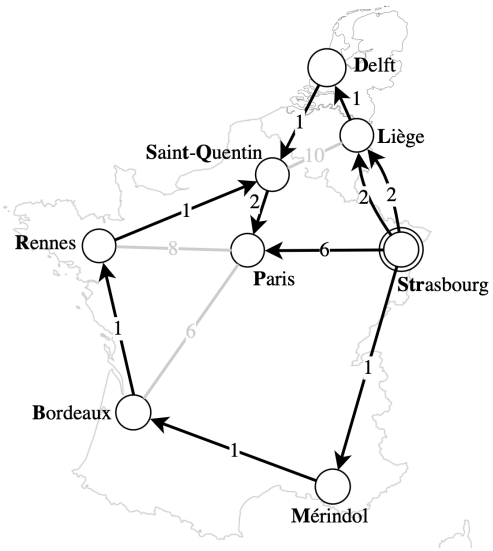
When relying on a link-state routing protocol, each participating node within the domain exchanges local connectivity information (*i.e.*, the networks to which it is connected, including its directly connected links, and the associated costs) throughout the network. The cost may be any additive metric. As each node exchanges its local links and the respective costs, all nodes within the network construct a full map of the network in the form of a weighted graph, also called a *connectivity map*.

This map is used by routers to compute the best paths (minimizing the cost) to all other internal destinations, *e.g.*, through the algorithms presented in Section 2.2.1. Thus, all routers possess the shortest path tree (or DAG, if a multi-paths feature is supported and enabled), containing the best paths to all other destinations within the domain. Upon topological changes (appearance/disappearance of a link, or weight change), the modification is flooded throughout the network so that each router may update its connectivity map.

Fig. 2.5 showcases the behavior of a link-state IGP, considering the network introduced in the previous section. Each link is labeled solely with its IGP weight (which were left unchanged from Section 2.2.2). We consider the point of view of the router within Strasbourg. Once all topological information is exchanged, each router possesses a connectivity map of the network, *i.e.*, a graph representation of the network as shown in Fig. 2.5a. A shortest path algorithm is then run by the router, which results in the shortest path DAG shown in Fig 2.5b (we consider that the protocol supports the use of multiple paths per destination). One may see that several shortest paths (of cost 6) exist between Strasbourg and Paris. One such path goes through the parallel links towards Liège, while others go through the directly connected link or via Mérindol. Similarly, two shortest paths of cost 4 exist from Strasbourg



(A) Network shown in Fig. 2.3. The delays have been removed from the labels for the sake of readability. Solely the IGP costs are shown. The node located in Brest has been removed.



(B) Shortest path DAG (rooted at Strasbourg) as would be computed by a ECMP-compliant link-state IGP.

FIGURE 2.5: Figure illustrating the principle of link-state IGP. Through topological information exchange, a link-state IGP constructs the connectivity map of the network in the form of a graph, as shown in Fig. 2.5a. The shortest path are then computed through Dijkstra, which results in the shortest path DAG shown in Fig. 2.5b.

TABLE 2.1: IGP RIB and FIB resulting from the network shown in Fig. 2.5.

(A) IGP RIB

Dest. D	$d^*(D)$	Next-hops $IH(D)$	Interfaces $L(IH(D))$
Bordeaux	2	Mérindol	(Str, M)
Delft	3	Liège	$(Str, L)_1$ $(Str, L)_2$
		...	
Mérindol	1	Mérindol	(Str, M)
Paris	6	Paris,Liège,Mérindol	(Str, P) $(Str, L)_1$ $(Str, L)_2$ (Str, M)

(B) FIB

Dest. D	Interfaces $L(IH(D))$
Bordeaux	(Str, M)
Delft	$(Str, L)_1$ $(Str, L)_2$
	...
Mérindol	(Str, M)
Paris	(Str, P) $(Str, L)_1$ $(Str, L)_2$ (Str, M)

to Saint-Quentin, through Mérindol or Liège.

Extracts of the resulting RIB and FIB are shown in Table 2.1. The RIB, shown in Fig 2.1a, contains all best-path related information, *i.e.*, the best distance d^* to all destination D, the associated next-hops $IH(D)$ and the respective interfaces $Int(IH(D))$. The interfaces are here denoted by the corresponding link (the name of each city has been reduced to the letters shown in bold in Fig. 2.5b). In particular, one can see the four interfaces corresponding to the shortest paths to Paris, going towards Liège, Mérindol, and Paris itself, which is directly connected. The correspondence between a destination and the outgoing interfaces is pushed in the FIB (shown in Fig. 2.1b) used for forwarding.

Link-state protocols have been used as far back as 1979, when the link-state protocol *Shortest Path First* (SPF) was deployed within ARPANET. Interestingly, SPF used the measured delay of the links as a cost [McQuillan *et al.* 1979]. This metric would later be replaced by the IGP cost, which considers the bandwidth by default but (more importantly) offers operators more control over their network. The IGP cost is the default metric used in both *Open Shortest Path First* (OSPF) and IS-IS.

Open Shortest Path First

Generalities OSPF [Moy 1998] and IS-IS [rfc 1990] are the most well-known link-state routing protocols. In this thesis, we will consider OSPF, although our contributions may be used with both protocols as they share numerous common concepts, albeit named differently. The first version of OSPF was first standardized in 1989 (similar to IS-IS), about a decade after SPF was deployed in ARPANET. It was quickly replaced by the second version of OSPF a few years later. OSPF was tested in several networks in the 1990s, including the NASA Science Internet network [rfc 1991]. The success of these test deployments showed that OSPF, and link-state protocols in general, were promising. It thus quickly became the recommended IGP for the Internet [Gross 1992]. Although it has undergone several revisions, the second version of OSPF is still the one used today.

OSPF enable routers to establish sessions with their neighbors and advertise the list of networks to which they are directly connected (*i.e.*, directly connected links and loopbacks) through a specific type of message, called *Link-State Advertisement* (LSA), which are propagated throughout the network. The LSA not only contains the network prefix, but also the associated IGP cost (*i.e.*, the weight of the link) as tuned by the network operator. The received LSA are put within the *Link-State Database* (LSDB), which should be identical among all routers if the network is stable and connected, and which forms the connectivity map. Dijkstra's algorithm is then used to compute the shortest path DAG.

Dijkstra's algorithm used by OSPF is indeed extended to leverage the existence of multiple shortest paths through ECMP as described in Section 2.2.1. By allowing to route traffic over multiple paths sharing the same optimal distance, ECMP leads to better resiliency and network performance, by natively offering alternative paths and by enabling to *load-balance* traffic across the multiple best paths.

It should be noted that although the shortest paths considered through ECMP share the best IGP distance, they may differ on other metrics, such as the latency. Consequently, leveraging multipath routing may prove challenging when aiming to provide QoS, as it is possible that only some paths among a set of ECMP paths actually provide the necessary

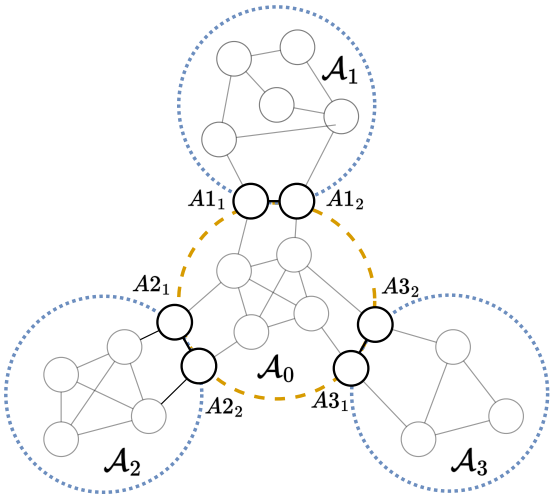


FIGURE 2.6: Illustration of a multi-area network structure. Stub areas (in blue dotted lines) are connected through the backbone area (in orange dashed lines) through Area Border Routers. In practice, the backbone area would span a large geographical area (*e.g.*,, a country). The routers located within a given city in the backbone would serve as ABRs for a stub area that would cover said city.

guarantees. Thus, when computing and deploying multi-criteria paths, one has to be aware of this path diversity that may be natively *enforced* by intermediary hops. Ideally, this path diversity must however still be leveraged when possible.

Although OSPF relies on the (efficient) Dijkstra's algorithm for path computation, its computation time may still be too high when the network grows in size. Indeed, networks are growing rapidly (already reaching several thousand nodes in some deployment [Matsushima *et al.* 2022]). More importantly, maintaining a complete connectivity map of such large-scale networks through message exchange may not be viable, in particular if changes are frequent. Consequently, using OSPF, networks may be divided into *areas* [Moy 1998].

Multi-area topologies Areas were designed to allow OSPF to scale by allowing to partition the network into several opaque subnetworks called areas. Routers maintain a full connectivity map of their own area only, and only share a *summary* of their area with other areas. *Area Border Routers* (ABRs), *i.e.*, routers at the border of two areas, are in charge of exchanging the corresponding summaries between the two areas they are connected to. Usually, this division is both physical and logical, which allows reducing the size of the instance onto which Dijkstra's algorithm must be performed, limits the flooding of updates as both can be limited to a single area, and reduce the overall memory consumption.

A simplified example of a multi-area OSPF network is illustrated in Fig 2.6. OSPF areas, denoted $\mathcal{A}_x$ are centered around the *backbone area*, or area 0 ($\mathcal{A}_0$, in orange dashed line). Non-backbone areas are referred to as *stub* areas [17]. In Fig 2.6, three stub areas are shown in blue dotted lines ($\mathcal{A}_1$,$\mathcal{A}_2$,$\mathcal{A}_3$). ABRs possess an interface in both the backbone area $\mathcal{A}_0$

[17] In practice, several other types of areas exist. However, we here consider solely stub and backbone areas for the sake of simplicity.

and a stub area $\mathcal{A}_x, x \neq 0$. An ABR linking $\mathcal{A}_0$ and $\mathcal{A}_x$ is denoted Ax_i, with i the id of the ABR. Indeed, there are usually two ABRs at such a junction, to prevent single points of failure.

As they are at the intersection of two areas, these routers are in charge of sending reachability information from one area to the other. For scalability purposes, ABRs do not share the whole connectivity map of one area to the other, but rather a *distance summary*. More precisely, an ABR Ax may only send to the routers within $\mathcal{A}_x$ the *distances* towards routers within $\mathcal{A}_y, y \neq x$, and vice versa. In practice, these distance summaries are sent through a specific type of LSA which may only be originated by ABRs. As such, non-ABRs routers are aware of the distance to reach routers within other areas, but not of the complete topology of said areas.

This partitioning reduces the size of the graph onto Dijkstra's algorithm must be run. Indeed, a router only needs to compute the shortest path to intra-area routers (including the ABR), as the distances from the ABR to networks in remote areas are already known through the distance summary. This design also limits message exchange, as update messages remain within the area they originate from. Note that an ABR Ax does however maintain the topological information of both the backbone area $\mathcal{A}_0$ and the stub area $\mathcal{A}_x$.

Choosing to rely on OSPF areas may have a significant impact on the overall physical structure of the networks. We will see in Chapter 4 that the presence of such obvious separators within the network (*i.e.*, the ABRs) may also be leveraged when computing multi-criteria paths to reach interesting performance even on large-scale networks.

Despite the ability to divide the network into distinct, smaller instances, the convergence time of OSPF may still be too high for modern-day requirements upon unplanned events. Indeed, a router must first detect the change, send LSAs to its peers, re-compute the shortest paths, and (eventually) update its FIB accordingly. This operation may take several hundreds of milliseconds [Francois *et al.* 2005a, Filsfils *et al.* 2017], during which some destinations may become unreachable. Several techniques have been designed to reduce this unreachability period [Chiesa *et al.* 2021, Raj & Ibe 2007]. These *Fast ReRoute* (FRR) schemes usually rely on pre-computations to enable a quick failover once a failure has been detected while maintaining a coherent (*i.e.*, loop-free) routing of the data packets. Most notable schemes of this kind will be reviewed in Section 3.1.

Although the link-state paradigm is probably the most popular one when performing intra-domain routing, it is not the only one. The first routing protocols actually followed a different paradigm, called *distance-vector* routing. Although these intra-domain protocols were admittedly abandoned due to their technical limitations, the distance-vector routing paradigm is still used (in some form) within some central modern routing protocols.

2.3.1.2 Distance-vector Routing Protocols

Implemented as early as 1970 [6net 2008], distance-vector protocols were one of the first routing protocols used. In fact, ARPANET relied on a distance-vector protocol before deploying the link-state protocol SPF [McQuillan & Walden 1977]. Conversely to link-state protocols, distance-vector protocols do not construct a full graph of the network. Rather, each router simply knows *how far* a destination is.

Distance-vector protocols usually rely on the distributed Bellman-Ford algorithm mentioned in Section 2.2.1 and thus follow the same general principle. Routers advertise their best distances towards each known destination (*i.e.*, their *distance-vectors*). The distance may be the number of hops or any given cost. For example, ARPANET used an estimate of the delay as the distance. Upon receiving a distance-vector for a given destination, a router updates its RIB accordingly. If the new distance is better, it updates its FIB and sets the next-hop to the originator of the distance vector. If the new distance is higher, the latter could still need to be considered, as it may be more up-to-date than the current known distance. In this case, the router typically waits during a *hold-down* period before adopting the new distance, to ensure that other updates reflect this increased distance as well [Malkin 1998]. An updated distance is sent to the router's neighbors. All routers thus iteratively improve (or, at least, update) their best distances. As mentioned in Section 2.2.1, the number of messages required to compute the shortest distances may be exponential with respect to the size of the network when considering asynchronous communications, because of path exploration (such a message ordering is however unlikely to occur in practice).

The most well-known implementation of the distance-vector paradigm is the *Routing Information Protocol* (RIP), which was standardized around 1988 [Malkin 1998] but had been already deployed before the standard was published [Doyle & Carroll 2005]. RIP is the successor of *Gateway Information Protocol* (GWINFO), originally designed by Xerox's Palo Alto Research Center. The latter was then adapted for use in the *Berkeley Standard Distribution* (BSD) distribution of the UNIX operating system in 1982. The popularity of BSD echoed on RIP, which became the standard of routing protocols for TCP/IP at the time [Kozierok 2005] (even though SPF was already deployed in ARPANET).

However, RIP suffered from several technical issues. First, RIP was designed to consider the number of hops as a metric. Second, RIP was slow to converge on large networks, which exacerbated the path exploration effect. To mitigate the increase of convergence time, the maximum number of hops RIB supports was set to 16, which also prevented its usage on larger networks. Finally, as routers only exchange distances, they are not aware of the underlying path behind the distances received. Upon a failure, it is then possible for two distinct routers to consider each other as the best next-hop to a given destination and iteratively increase their distance towards said destination as they *chase* the induced loop. This issue was however eventually mitigated through the *split-horizon* mechanism, which forbids a router to send a newly learned distance through the interface onto which it was received.

For all these reasons, it seemed clear that RIP and distance-vectors protocol were not suited for large-scale dynamic routing. This sparked the standardization effort for a standard link-state protocol, following and building upon the experience learned through the deployment of SPF within ARPANET. This effort lead to OSPF which was then quickly recommended as the standard IGP for the Internet [Gross 1992].

Due to the way paths are built, distance-vector protocols are sometimes referred to as *incremental* protocols within the literature [Lambert *et al.* 2009]. Indeed, paths are built incrementally by each router depending on the information received by its neighbor. Despite their limitations, incremental protocols do have some benefits. For example, there are considered to be less CPU and memory intensive.

Perhaps more interestingly, incremental protocols can also be seen as following a *more distributed* paradigm. The complete topological information is indeed unknown to routers, which base their routing decision solely according to the reachability information their neighbors were inclined to share, which could deliberately filter information. In fact, this type of filtering was at the core of the split-horizon technique.

In the intra-domain context, this is hardly relevant, as all nodes share the same global objective and should not (most of the time) benefit from hiding information from their neighbors. However, should nodes behave as independent entities with different preferences, the latter could easily share different distances to different neighbors, or even not share any distances at all following an egocentric behavior.

This feature proves quite useful when performing *inter-domain* routing. Within the Internet, there is no global objective anymore: ASes have different preferences, according to economical or political considerations. For these reasons, a variant of the distance-vector paradigm is used to exchange routes between domains when performing inter-domain routing.

2.3.2 External destinations reachability through BGP

An IGP only ensures connectivity *within* an AS, or domain. However, it does not enable *inter-domain* connectivity. The latter is ensured with the support of an *Exterior Gateway Protocols* (EGPs) [18].

Originally (around 1980), the Internet had a different structure, quite similar to multi-area OSPF topologies. ARPANET acted as a backbone between other stub networks. Routers within the ARPANET, called core routers, exchanged information with non-core routers (in other networks) through EGP [rfc 1982b], a distance-vector protocol. The information was then shared among the core routers within the ARPANET through the *Gateway to Gateway Protocol* (GGP) [rfc 1982a], a distance-vector protocol similar to RIP.

Interestingly, the original EGP RFC predicted that over time, the Internet would consist of co-equal *autonomous systems*, and relying on a strictly centralized core would thus become inappropriate. This new Internet architecture eventually came into being. GGP was abandoned and EGP was generalized to allow the exchange of routing information between any two ASes. However, EGP required the underlying graph to be a tree, a restriction that would soon become too strict as the Internet grew.

Consequently, a new protocol was designed to achieve inter-AS connectivity which is, as of now, the de-facto EGP used throughout the Internet: the *Border Gateway Protocol* (BGP) [Rekhter *et al.* 2006].

2.3.2.1 BGP's design peculiarities

Originally designed on cafeteria napkins during an *Internet Engineering Task Force* (IETF) meeting [19] around 1989 [Rekhter *et al.* 2006, Jabloner 2016], BGP grew to become the backbone of the modern day Internet. In short, BGP enables ASes to exchange reachability information (or simply *routes*) towards local or remote destination prefixes to neighboring ASes in a fashion akin to distance-vector protocols.

[18] Note that the term EGP is both the name of a family of protocols and the name of one of the protocols within this family. While the naming is confusing, the distinction should be clear from context in the text.

[19] The IETF is the organization in charge of standardizing the various protocols and concepts that comprise the Internet.

At the time BGP was designed, the main concerns were scalability and flexibility [Gross & Rekhter 1995, Rekhter 1991]. Scalability seems like a natural requirement, as BGP had to be able to manage connectivity information towards *all* destinations within the Internet. The flexibility requirement, on the other hand, is because BGP operates in a drastically different ecosystem compared to IGPs. Conversely to IGPs, BGP deals with different autonomous entity with different economical incentives and preferences. Thus, BGP had to enable operators to express heterogeneous political or economical preferences, which may drastically differ across ASes.

Indeed, at the AS scale, the Internet may be seen as a vast hierarchical graph. There are 15 *Tier-1* ASes at the top of the hierarchy, which are all interconnected to one another through peering agreements: being of similar sizes, the latter agree to exchange traffic without paying any fees. A Tier-1 AS may thus reach any remote network within the Internet through this peering clique. These (usually) cost-free relations are referred to as *peer-to-peer* relations. *Tier-2* networks may buy transit service to one or several Tier-1 networks to reach remote networks within the Internet, establishing a *customer-to-provider* relationship with the latter. Tier-2 networks may themselves provide transit services to Tier-3 (or lower tier) ASes. ASes at the bottom of the hierarchy, which do not have any clients, are called *Stub* ASes.

ASes may receive several routes towards the same remote prefix from several neighboring ASes. Different ASes may prefer different routes, depending on their own economical relationships with the AS advertising them. Typically, an AS will prefer to use a route advertised by a client or a peer rather than a provider, as going through their networks is free.

To answer both these requirements (scalability and flexibility), it seemed more natural to design BGP as a vector-based protocol [20]. However, distance-vectors did not seem like a viable choice, given their technical limitations mentioned in Section 2.3.1.2, such as fairly long-lasting transient routing loops during convergence. BGP was thus conceived as neither a link-state protocol nor a distance-vector one, but rather as a *path-vector* routing protocol, an extension of the distance-vector routing paradigm. Furthermore, BGP was also designed in a way that allows operators to apply their own policies and preferences within the protocol. As such, BGP is often said to be *policy-based*.

At high-level, BGP operates similarly to other vector-based protocols. BGP speakers originate reachability information called *routes* over BGP sessions to other BGP routers, which may be within the same AS or within a neighboring AS. If two interconnected BGP routers lie in two different ASes, they are referred to as *border routers*. A route that is elected as the best route by a router may be re-advertised iteratively to subsequent BGP routers, allowing the reachability information to propagate within the Internet.

[20] As mentioned previously, vector-based protocols were seen as less memory and CPU intensive. In addition, link-state routing seems like a less natural option for inter-domain routing. Indeed, the definition of a global routing metric (as traditionally done in link-state protocols) hardly makes sense here, as the attractiveness of a route may change across ASes. However, one could argue that the choice of paradigm behind BGP is mainly historical. Indeed, BGP's ancestors, GGP and EGP, were designed during the peak popularity of RIP, and thus conceived as distance-vector protocols. This design choice probably echoed on BGP. One could theorize that should BGP have been designed from scratch a decade later, it may have been a link-state protocol.

However, BGP exhibits key differences from other protocols. Being a path-vector protocols, BGP maintains the ASes a route goes through. This list of ASes is attached to the route and referred to as the AS-PATH. The AS-PATH allows to easily prevent route advertisements from looping by discarding the ones whose AS-PATH contains a repetition.

Furthermore, conversely to other protocols, *all* received routes are stored within the BGP RIB (if there is enough memory) to allow for faster re-convergence, as back-up routes may then already be present within the RIB. Still, only the best selected route should (by default) be pushed within the global RIB (and FIB) and, most importantly, advertised to other BGP routers. If the preferred route change, a withdrawal message is sent to get rid of (or *withdraw*) the obsolete route.

This route selection and advertisement mechanisms are where BGP differs most from standard protocols, due to its policy-centric approach. Indeed, BGP offers various ways for an operator to alter the outcome of the best path selection process through the modification of the *attributes*, and how routes are exchanged through the use of *filters*.

Routes exchanged through BGP are not characterized by a scalar distance, but rather by a vector of *attributes* used to select the routes by ranking them lexicographically. This ranking process is referred to as the BGP *decision process*, which will be further detailed in Section 2.3.2.2. For example, the length of the AS-PATH attached to the route is one of these attributes and thus acts as a form of inter-domain hop-count.

However, not all attributes follow standard arithmetic metrics. Some of them (in particular, the LOCAL-PREF, which has the highest priority) may be set arbitrarily when the route is imported, to change the outcome of the decision process and prefer certain routes over others, *e.g.*, by preferring cheaper paths advertised by customers or peers over the ones advertised by providers.

Moreover, once a BGP router elects its best route to a given destination, the route is advertised or not, depending on the export policy implemented by the operator. Indeed, operators can configure routers to explicitly filter the routes they receive and the ones they send. For example, a BGP speaker can choose not to advertise routes towards remote prefixes to neighbors that do not pay them for transit services towards said prefix (*i.e.*, to not offer *free lunch* to these neighbors). Thus, interestingly (and conversely to IGPs), reachability may not be assured even if the graph is connected.

Fig 2.7 shows a simplified illustration of how BGP operates. This inter-domain gadget shown is composed of four ASes. The border routers of each AS are respectively denoted $i_{a,b,c}$. AS 1 advertises a route to its own local prefix D with the associated attributes and AS-PATH to its provider AS 2 via a BGP session between their border routers. The route is selected by 2_a and re-advertised to the other BGP routers within AS 2, which in turn share it with the remote ASes 3 and 4 after having updating the AS-PATH and other attributes accordingly. Notice that AS 3 has no interest in advertising this route to AS 4. Indeed, AS 4 does not pay AS 3 for transit services (in fact, quite the contrary). Thus, AS 3 does not want AS 4 to use it as a transit AS to the prefix D (*i.e.*, to provide *free lunch*). Consequently, AS 3 chooses not to advertise the route received to AS 4 thanks to export filters.

As routers can choose to elect and/or advertise whatever paths they desire regard-

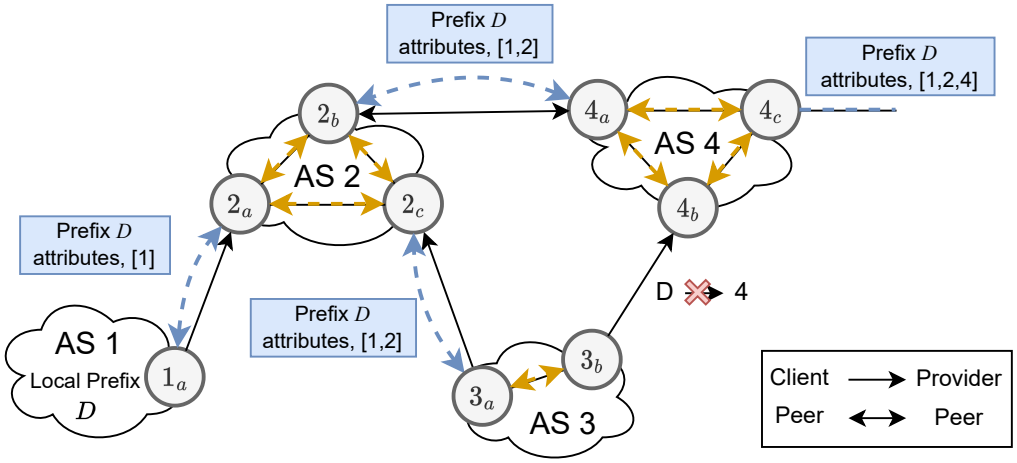


FIGURE 2.7: Simplified illustration of how BGP operates. AS 1 shares its route to its local prefix D to the neighboring AS 2, which in turn advertise the route to ASes 3 and 4.

less of the preferences of their neighbors or any global objective, the subpath optimality property does not hold anymore. This drastic difference compared to IGPs gives rise to various issues. Indeed, such increased expressiveness may cause data-plane loops, generate non-deterministic outcomes, or event prevent BGP from converging at all [Griffin & Wilfong 2002b, Griffin & Wilfong 2002a], *i.e.*, generate control-plane loops.

Guidelines have however been proposed to ensure that the protocol converges across ASes. As these guidelines do not require coordination between ASes and make economic sense, they tend to be widely adopted [Gao & Rexford 2001], although more complex policies are sometimes deployed [Gill *et al.* 2014].

In practice, BGP routers may establish two kinds of BGP sessions. The exchange of routes between BGP routers lying in different ASes is done through *External BGP* (eBGP). As such, eBGP is the submodule of BGP allowing the dissemination of routes within the Internet. The advertisement of routes between routers within the same AS is done through iBGP, which thus allows the dissemination of reachability information within the AS. Using Fig 2.7 as an example, lines shown in blue represent eBGP sessions, while lines shown in orange represent iBGP sessions.

While the dynamics of eBGP at the AS-scale are interesting objects of study in themselves, our contribution, OPTIC, is focused on the way BGP operates within an AS and its interactions with the underlying IGP. Consequently, we will now focus on the intra-AS behavior of BGP, more precisely, on the way routes are elected and exchanged within iBGP.

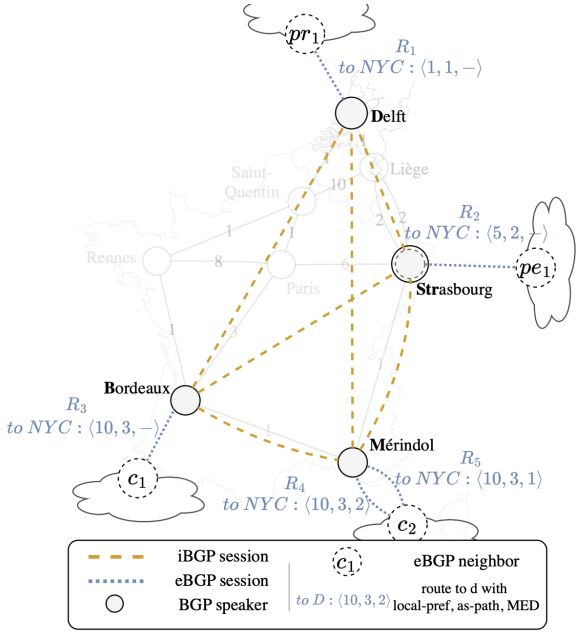


FIGURE 2.8: Illustration of the exchange of BGP routes through eBGP and iBGP, whose session are shown in blue dotted line and orange dashed line respectively. The physical topology is the same one used throughout this chapter, greyed out to focus on the BGP sessions. Notice that not all routers within the topology are BGP speakers. Clients, peers and providers are denoted c, pe and pr respectively.

Dest. D	Interfaces $L(IH(G(D)))$
Bordeaux	(Str, M)
Delft	$(Str, L)_1$ $(Str, L)_2$
.	
.	
.	
St-Quentin	$(Str, L)_1$ $(Str, L)_2$
NYC	(Str, M)

TABLE 2.2: FIB of the Strasbourg Node shown throughout this section, now with BGP information.

TABLE 2.3: BGP RIB resulting from the network and route advertisements shown in Fig. 2.8. The information at the right of the vertical bar is computed through recursive look-ups before pushing the information within the FIB.

Dest. D	$R(D)$ (LP, AS, MED, IGP)	BGP NH $G(D)$	IGP NH $IH(G(D))$	Interface $L(IH(G(D)))$
NYC	$(10, 1, -, 3)$	Delft	Liège	$(Str, L)_1$ $(Str, L)_2$
NYC	$(5, 2, -, 0)$	self	self	(Str, pe_1)
NYC	$(1, 3, 1, 1)$	Mérindol	Mérindol	(Str, M)
NYC	$(1, 3, 2, 1)$	Mérindol	Mérindol	(Str, M)
NYC	$(1, 3, -, 2)$	Bordeaux	Mérindol	(Str, M)

BGP RIB — rec. look-up — Best sent to FIB

2.3.2.2 The internal behavior of BGP

As mentioned, initially, ASes learn reachability information towards remote prefixes via eBGP sessions with the border routers of a neighboring AS. A route learned by a border

router can then be shared with the other BGP speakers of the AS through iBGP sessions.

Despite spanning a single AS, iBGP is a complex sub-protocol. While the latter offers the same tools as eBGP (*e.g.*, route filtering), it also brings forward well-known additional challenges, most notably regarding the proper dissemination of routes within the AS, and detrimental interactions with the underlying IGP [Griffin & Wilfong 2002b, Teixeira *et al.* 2004].

Sharing routes through iBGP Among other concepts, the way eBGP and iBGP interact is shown in more detail in Fig 2.8. The border routers of an AS (also called *Autonomous System Border Routers* (ASBRs)) connect to other ASes (which may be customers, peers, or providers) through eBGP sessions, shown in dotted blue lines. Here, routers learn routes denoted R_n to the remote destination prefix NYC through these sessions. This information is then shared to the other BGP speakers of the AS, through the iBGP sessions shown in dashed orange lines. Notice that not all intermediary routers are required to be BGP speakers, as border routers may rely on tunneling mechanisms to alleviate the load on core routers [21].

While eBGP sessions are usually set up between directly connected routers, iBGP may be set up across distant routers, and do not necessarily follow the physical topology. The way iBGP sessions are organized is referred to as the iBGP *topology*. By default, iBGP messages are not transitive : a message received over an iBGP session is not forwarded across another iBGP session. This prevents control messages from looping [22]. Hence, to ensure that routes are disseminated to all other BGP speakers within the AS, the iBGP topology is supposed to be a full-mesh between all said BGP speakers.

A full-mesh of iBGP sessions presents some advantages. As a BGP speaker receives the best route of all its iBGP peers, it may have access to several routes per external destinations, which may offer faster re-convergence or at least backup routes. It is also conceptually simple and should exhibit predictable behavior, as iBGP routers should have access to all required information.

However, such a naive iBGP topology design does not scale well and is quite verbose, which, in turn, negatively impacts network resources. Alternative solutions to the iBGP full-mesh were thus proposed, the most popular one being route reflection [Chen *et al.* 2006].

Route reflection allows setting up some BGP speakers as *route reflectors*, which are allowed to share (or *reflect*) routes learned through iBGP to their *clients*, a subset of the other BGP speakers within the AS. A route reflector can thus be seen as an iBGP router which centralizes BGP routes, before selecting the best one through the BGP decision process and advertising it to its clients. Several route reflectors may be set up within an AS, following more or less intricate iBGP topologies. These topologies may even exhibit a hierarchy of route reflectors [Chen *et al.* 2006]. Within such topologies, the propagation of the routes

[21] By default, all routers within an AS are supposed to be BGP speakers, to know the next-hop associated with the remote destination prefix. However, this induces a heavy load on all routing devices. Consequently, packets may be *tunneled* by the BGP-aware border router at the edge of the network. Core routers then forward traffic solely according to the tunnel information, and do not require actual reachability knowledge of the remote prefix. Several technologies may be used to deploy BGP-free cores while allowing the traffic to follow the optimal IGP paths, *e.g.*, IP-in-IP, *Multi-Protocol Label Switching* (MPLS) [Viswanathan *et al.* 2001] or SR [Filsfils *et al.* 2017].

[22] Recall that the messages here remain within the same AS, so the AS-PATH can here not be used to that effect

TABLE 2.4: An example of the set of attributes taken into consideration by BGP.

	Step	*Criterion*	
β	1	Highest `local-pref` LP	(economical relationships)
	2	Shortest `as-path`	
	3	Lowest `origin`	IGP over EGP
	4	Lowest MED	(cold-potato routing)
α	5	eBGP over iBGP	
	6	Lowest IGP distance	(hot-potato routing)
	7	Lowest `router-id` rid	(arbitrary tie-break)

follows specific rules: the route advertisement may go up several stages within the route reflection hierarchy, then one hop across (staying at the same level), before going back down to the clients.

When relying on route reflection, clients are (by default) only aware of their own routes, and the route advertised by their route reflector. In addition, the route chosen by the route reflector is not necessarily the route its client would have elected, had it access to the same information as its route reflector.

The fact that different BGP speakers may elect different routes is due to the way the BGP decision process is designed. While the latter considers preferences that should not vary within the AS (*e.g.*, economical relationships), it is also deeply intertwined with the IGP, resulting in heterogeneous choices among routers depending on their position within the internal topology.

The BGP decision process BGP speakers may know several routes towards the same prefixes received through different eBGP or iBGP sessions. For example, within Fig. 2.8, the Strasbourg router learns several routes R_n to the remote prefixes NYC, either through Delft, Mérindol, Bordeaux, or a directly connected external peer.

BGP speakers must then decide the best exit point (also called *gateway* or BGP NH) within the AS to reach a given external destination. A BGP speaker may consider itself as the best exit point to reach a remote prefix, but may very well prefer to use another gateway within the AS depending on preferences that have been manually set up.

To select the best route and associated gateway, routers rely on the BGP decision process, which consists of ranking routes lexicographically according to their attributes. The attributes of a BGP routes are given in Table. 2.4 in decreasing importance.

- As aforementioned, the LOCAL-PREF (LP) is traditionally used to express economical preferences (*i.e.*, prefers routes learned from clients over peers over providers).
- The AS-PATH (AS) length refers to the number of ASes the route goes through.
- The *Multi-Exit Discriminator* (MED) is a peculiar attribute, which should only be used to discriminate routes originating from the *same* AS, *i.e.*, to discriminate between different

entry points into the neighboring AS. The MED is usually set by the neighboring AS to indicate a preference regarding how traffic should enter its domain. This attribute is said to enforce *cold-potato* routing. As it breaks the total ordering between routes, the MED is well-known to be the cause of convergence issues within BGP [Griffin & Wilfong 2002a].

These attributes (LOCAL-PREF, AS PATH LENGTH, MED), which we will sometimes abbreviate (LP, AS, MED) are inter-domain related and should, by default, be equal among all BGP speakers within an AS for a given route. We group these attributes and refer to them as β, or $\beta(R)$ when considering a specific route R.

Several routes may still be contending at this point of the decision process. These routes, that share the same β attributes, are sometimes referred to as *AS-dominant routes* [Vissicchio *et al.* 2012], as all routers should prefer these routes over others[23]. To discriminate the remaining routes, other attributes come into play.

Essentially, among all AS-dominant routes, a router will prefer the route which allows the traffic to exit the AS as soon as possible, according to the IGP weights (in other words, a router will prefer sending the traffic towards the closest gateway in term of IGP distance). This is referred to as *hot-potato routing*, conversely to *cold-potato routing*, where routers may not eject traffic out of the network as fast as possible to respect the neighboring AS' desire.

The remaining attributes thus enforce this hot-potato routing principle, before reaching an arbitrary tie-break.

- Routes learned over eBGP are preferred to the ones learned through iBGP (as the IGP distance to the gateway is here 0)

- A router will first prefer the closest gateway in terms of IGP distance (Line 6 in Table 2.4).

These attributes may change across routers within the same AS for a given route, depending on their location within the internal topology. We group these attributes and refer to them as α, or $\alpha(R)$ when considering a specific route R.

The route returned at the end of the decision process is the route chosen for the given prefix. In the following, we state that $R_x \prec R_y$ if R_x is better than R_y according to the decision process. The chosen route is then advertised to eBGP and iBGP peers, if not filtered out. Finally, it is pushed within the global RIB before being possibly pushed within the FIB. Note that, by default, *only* the best route is advertised. If the best route changes, the new one is advertised and the previous one is *withdrawn*.

Before installing the selected route within the FIB, the outgoing interface must be resolved. Indeed, the next-hop attached to a BGP route is the *gateway* (or BGP NH) that has advertised it. The best *internal* next-hop used to reach the chosen gateway and the associated interface must be computed. This step is referred to as *recursive look-up*.

There is thus a significant interplay between BGP and the IGP. First, the IGP distances between BGP speakers influence the outcome of the decision process. Second, the path used to reach the selected BGP next-hop is decided by the IGP.

[23] The term *quasi-evalent* can also be found in the literature [Buob *et al.* 2007].

The routing performed through BGP can then be seen as an extension to the composition of functions $Ports(D) = Int \circ IH \circ LMP(D)$, defined previously, to $Ports(D) = Int \circ IH \circ G \circ LPM(D)$ when external destinations are considered. G returns, for a given external prefix, the best exit point (or gateway) as elected through the decision process [24]. The Int and IH functions have the same definition as previously and are used to perform the recursive look-up mentioned previously.

Once again, the entirety of the composition of functions is not computed each time a packet is forwarded. Only the resolved result is pushed to the FIB for all known prefixes, allowing the FIB to only compute the longest prefix match for the destination before checking the associated outgoing interface. Note that when relying on traditional router architectures, once a route is pushed within the FIB, only the outgoing interface remains: information regarding the BGP next-hop and IGP next-hop of the route is lost.

The decision process and recursive look-up are illustrated with Table 2.2 and Table 2.3, which show the BGP RIB and the FIB associated with the network shown in Fig. 2.8. Routes exchanged are denoted by their attributes as a vector (LOCAL-PREF, AS-PATH LENGTH, MED). All routes R_n to NYC shown are learned by Strasbourg (either through iBGP or eBGP) which adds them all to its RIB. Note that, in addition to the attributes, only the associated gateway (or BGP next-hop) of the route is maintained.

The best route is then selected. Due to the LOCAL-PREF, customers, reachable through Bordeaux and Mérindol, are preferred. In other words, we have $\beta(R_{3,4,5}) < \beta(R_2) < \beta(R_1)$. However, the routes R_3, R_4, R_5 announced by Bordeaux and Mérindol share the same β (MED excluded, as the latter should not be considered between different ASes). Thus, the internal distance from Strasbourg to these gateways is considered (*i.e.*, the α attributes). Because of hot-potato routing, Mérindol is preferred. Finally, the MED can be considered. Here, the MED dictates that $R_5 \prec R_4$, resulting in the election of route $R_5 = (1, 3, 1)$. Once the best route is selected, it is advertised to relevant BGP peers. The interface used to reach the corresponding gateway is then resolved and pushed in the FIB.

We have seen that iBGP is quite intricate. The design of scalable iBGP topologies is challenging. The way routes are selected is more complex and less intuitive than within IGPs, in particular when considering the MED. In addition, its interactions with the underlying IGP may cause detrimental effects, *e.g.*, frequent re-convergence or sub-optimal routing. The iBGP sub-protocol thus leaves room for several kinds of issues to arise, which we will now further discuss.

2.3.2.3 The issues of iBGP

The iBGP protocol may suffer from various issues. In this section, we will talk in further detail about the issues directly related to our contributions: the lack of route diversity within one's AS, and the high convergence time.

Insufficient route diversity While an AS may learn several routes towards a remote prefix, routers within the AS may only be aware of a few routes. According to a study by

[24] While using several gateways for a given destination is possible [Balon & Leduc 2007, Cisco-Systems 2022], the BGP decision process ensures, by default, that a single gateway is elected.

Uhlig and Tandel modeling a Tier-1 AS, the majority of routers only know a single route besides their own [Uhlig & Tandel 2006]. Indeed, routes learned by border routers are not necessary all disseminated throughout the AS.

The fact that BGP speakers only advertise their preferred route is one of the main causes of reduced route diversity. For example, while a border router may be connected to several remote border routers and learn different routes from each one of them, it will only advertise one (the best) route. Furthermore, by default, routes learned via eBGP may not even be advertised if a better one has already been learned through iBGP.

This issue is drastically exacerbated by route reflection. Just like other BGP speakers, a route reflector will elect its best route according to the BGP decision process, and only advertise said route to its clients. When relying on a hierarchy of route reflectors, even said route reflector may thus only possess a partial knowledge of BGP routes, depending on the choice of its peers. Repeated decision processes thus stop the propagation of most BGP routes, allowing only the current most popular one to spread.

This lack of route diversity gives rise to several issues. First, it may impact the *correctness* of the protocol [Vissicchio *et al.* 2012, Griffin & Wilfong 2002b, Griffin & Wilfong 2002a]. In other words, forwarding anomalies may appear (*e.g.*, suboptimal routing or data-plane loops), or the protocol may even fail to converge [25].

Suboptimal routing may occur when using route reflection, as the choice performed by the route reflector does not necessarily match its clients'. Indeed, recall that the IGP distance to the advertising gateway is an attribute within the decision process. Due to this hot-potato routing, the route chosen by the decision process depends on the position of the router within the IGP topology. Thus, the route chosen by a route reflector for its clients may not be the one the clients would have chosen if they had access to all BGP routes [26].

The fact that iBGP may diverge is perhaps unsurprising, as it offers the same type of expressiveness as eBGP. However, iBGP may diverge even without relying on complex filter policies, but solely because of lack of route visibility coupled to hot-potato routing and/or the MED attribute.

As aforementioned, the MED is a peculiar attribute, and source of several problems within BGP. For example, due to the way routes are compared (in a pairwise fashion), the lack of total ordering may result in a non-deterministic route election [27]. This can however be mitigated through the use of the `bgp deterministic med` option [28].

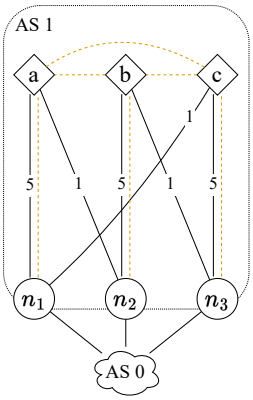
More importantly, the MED may cause routing oscillations and prevent BGP from con-

[25] Note that these problems are in fact due to both iBGP and eBGP. In fact, iBGP includes all the problems of eBGP and more.

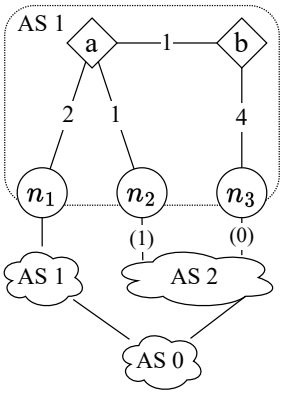
[26] It should be noted that Optimal Route Reflection has been proposed, which required the route reflector to compute the best route from the point of view of each of its clients, or a set of them [Raszuk *et al.* 2021]. However, this solution is either too expensive (when considering each client individually) or does not solve the issue (when considering sets of clients).

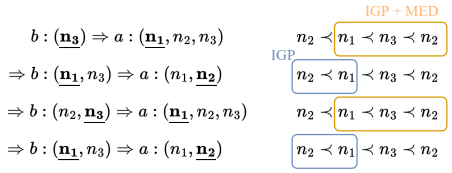
[27] More precisely, depending on the order in which the routes are compared, the outcome of the route selection may change, depending on whether routes originated from the same AS are compared to one another through the MED, or eliminated prior by comparisons with other routes.

[28] `deterministic med` consists in first electing the best route sent by each neighboring AS, considering the MED, and then selecting the best route among this set.



(A) Gadget as proposed by Griffin and Wilfong [Griffin & Wilfong 2002b] illustrated routing oscillation within iBGP induced by route reflection.





(B) MED-EVIL gadget extracted as proposed by Griffin and Wilfong [Griffin & Wilfong 2002a], showcasing how the MED may cause iBGP oscillation. The right side of the figure details an oscillation scenario occuring without Add-Path. The route known by a given router are shown in parentheses. The route elected is shown underlined, and the routes advertised are shown in bold.

FIGURE 2.9: Gadgets illustrating iBGP oscillations induced by hot-potato routing, the MED, and lack of visibility. Route reflectors are shown as diamonds, while other BGP speakers (which here are gateways) are shown as circles. The boundaries of the local AS, AS 1, are shown in dotted lines. IGP costs are shown on each link, while the MED is shown in parentheses. The iBGP topology is shown in orange dashed lines when it differs from the physical topology.

verging. By only being used to compare route originating from the *same* AS the MED breaks the total ordering of BGP routes. This may create a preference cycle between routes for a given router, giving rise to several issues, including routing oscillations that may span several ASes (as exhibited in the BAKED POTATO gadget of Griffin and Wilfong [Griffin & Wilfong 2002a]).

An example of internal MED-oscillation can be seen in Figure 2.9b, which shows a gadget extracted from the paper by Griffin and Wilfond [Griffin & Wilfong 2002a]. We consider routes to AS 0. The iBGP topology follows the physical one. Note that a is always aware of the route advertised by n_1 and n_2 as it shares a direct iBGP session with the latter. Router a prefers (and advertises) n_1 when all routes are known (as n_3 is preferred over n_2 due to the MED, and n_1 preferred to n_3 because of hot-potato routing). However, it prefers n_2 when only n_1 and n_2 are known. Thus, we have $n_1 \prec n_3 \prec n_2 \prec n_1$. Once router a elects its preferred route, only the latter is advertised (and previously advertised routes to AS 0 are withdrawn). The lack of visibility induced by this behavior can make router a chase the preference cycle induced by the MED. This behavior of both a and b is showcased more precisely on the right side of the figure.

It should be noted that MED-related oscillation can be prevent by the `always-compare` BGP option [29], but this option does not respect the MED semantics. Note that the `deterministic-med` option, however, does not prevent these oscillations.

While the peculiar semantics of the MED seem likely to generate issues, iBGP oscillations may also occur without the use of the MED. For example, Fig 2.9a shows an example

[29] The `always-compare` option allows to consider the MED even when comparing routes originating from different ASes

of oscillation induced by route reflection and hot-potato routing, once again extracted from a paper by Griffin and Wilfond [Griffin & Wilfong 2002b]. On this figure, three routes reflectors a, b and c learn routes towards AS 0 through their respective client gateways n_1, n_2, n_3. The routes are exchanged through the iBGP sessions shown in orange. Note that the latter does not follow the physical topology. Intuitively, this gadget is designed so that each route reflector having n_x as a client prefers the gateway $n_{(x+1)\%3}$. However, once a router chooses this preferred exit point, it withdraws the route advertised by its client and thus deprives another router of its preferred route. This effect causes iBGP to diverge. Here, oscillations are mainly caused by the fact that routers deprive their peers of their preferred route because of their own local preferences (*i.e.*, α attributes). Said otherwise, the root cause of these oscillations is the lack of congruence between the IGP and the iBGP topologies, as route-reflectors prefer the routes advertised by the clients of other route-reflectors (due to the IGP topology), rather than their own, direct iBGP clients.

In addition to correctness issues, lack of route diversity prevents iBGP load balancing [Cisco-Systems 2022] and increases convergence time and connectivity loss in the event of a failure or topological change. Indeed, because of reduced route visibility, routers are unlikely to possess the new post-convergence path *a priori*. Re-convergence to the new optimal path thus necessarily requires message exchange, in order for the new post-convergence route to be advertised. More importantly, routers may not even have a non-optimal route to fall back to should their preferred route become unavailable, preventing the use of FRR mechanisms. The impact of lack of route diversity on BGP's convergence is non-negligible. Pei and Van der Merwe have estimated that "route invisibility" is a significant factor in BGP convergence delay [Pei & Van der Merwe 2006].

This second ill-effect of reduced route diversity is particularly problematic, as iBGP's convergence time can be quite high *even* when considering enough visibility.

High convergence time Vanilla BGP may exhibit a very high convergence time [Filsfils *et al.* 2011, Labovitz *et al.* 2000a]. While we have seen that OSPF convergence time may already be too long for modern-day requirements, the convergence time of BGP is orders of magnitude higher.

The convergence time of eBGP is perhaps unsurprising, if only because of the sheer size of the Internet. The scale at which eBGP operates may indeed lead to slow message exchange and exacerbated path exploration [Labovitz *et al.* 2000b, Bremler-Barr *et al.* 2009] [30]. This convergence time is also increased by timers which aim to mitigate routing instabilities [Rekhter *et al.* 2006]. In addition, as BGP is not link-state, the latter cannot infer the total impact of a failure. Rather, it must wait to receive and process the updated information for each remote prefix.

However, and perhaps more surprisingly, even iBGP may be slow to converge [Filsfils *et al.* 2011, Teixeira *et al.* 2004]. Indeed, recall that network events induce the lexicographical re-ordering of BGP routes, to select the new best route. This re-ordering

[30] BGP being a form of vector-based protocol, it is also subject to path exploration, as defined in Section 2.2.1

may not seem particularly tedious when considering a single prefix, *e.g.*, if a gateway withdrew a route to a prefix. Indeed, in this case, a BGP speaker must simply re-rank and re-install within its FIB the routes for the (unique) impacted remote prefix. Thus, only a few route entries have to be considered. However, this process may also be triggered by *internal* events, which potentially impact *all* prefixes.

Indeed, because of the interplay of BGP and the IGP during the decision process (*i.e.*, hot potato routing), even an *internal* change may alter the order of BGP routes. Thus, while a BGP update affects a single prefix, an internal event may affect several BGP routes (*i.e.*, all routes that were selected due to the hot-potato routing rule). While it may be unlikely that the ranking of *all* routes is affected, it has been shown that the majority of the routes may be impacted [Filsfils *et al.* 2011, Teixeira *et al.* 2004].

Once the impact of the IGP event has been noticed by BGP [31], the latter must re-rank the routes and update the FIB, a computationally expensive process. The RIB, containing the routes to rank, contains around 2 700 000 entries, while the FIB contains around 920 000 entries, which must afterward be updated one by one on legacy architectures. By itself, this last process was measured to take several minutes (and averaged at around 500μs per route) [Holterbach 2021, Filsfils *et al.* 2011].

Recall that such internal events are frequent. Even on fairly small networks and with conservative assumptions (*e.g.*, removing link flapping), there may be more than dozens of internal events on a bad day [Merindol *et al.* 2018]. Thus, the BGP re-convergence process may be triggered just as often, which may result in several connectivity losses towards remote prefixes.

Several works aimed at tackling the issues aforementioned. Some solutions focus solely on guaranteeing correctness (*e.g.*, by ensuring that all routers have access to their preferred route). Others aim to increase route diversity, allowing to improve the convergence of BGP and, in some cases, also guaranteeing correctness. The increased path diversity can be leveraged to design FRR mechanisms. These works will be reviewed in Section 3.1. In particular, our contribution, OPTIC, leverages increased path-diversity through adequate data-structure to improve the convergence time of BGP upon internal events.

This section presented the general concept of routing and routing protocols. The concepts we have discussed are summarized in Fig. 2.10. We have seen that routing protocols specify how topological information is exchanged and how forwarding paths should be computed. We have seen that OSPF, which ensures intra-domain connectivity, operates differently from BGP, which ensures inter-domain connectivity.

Dealing with drastically different contexts, these protocols follow different approaches both in terms of information exchange and path selection and populate different RIBs.

OSPF draws a complete map of the network onto which it performs shortest path computation. To improve convergence time and reduce the number of messages to be exchanged, OSPF allows separating the network into distinct subnetworks called areas.

BGP allows ASes to exchange routes across eBGP session, which are then disseminated within the AS through iBGP sessions. Routes are selected through a policy-based flexible

[31] In practice, BGP notices such events either through a periodic check, or through a direct notification from the IGP control-plane.

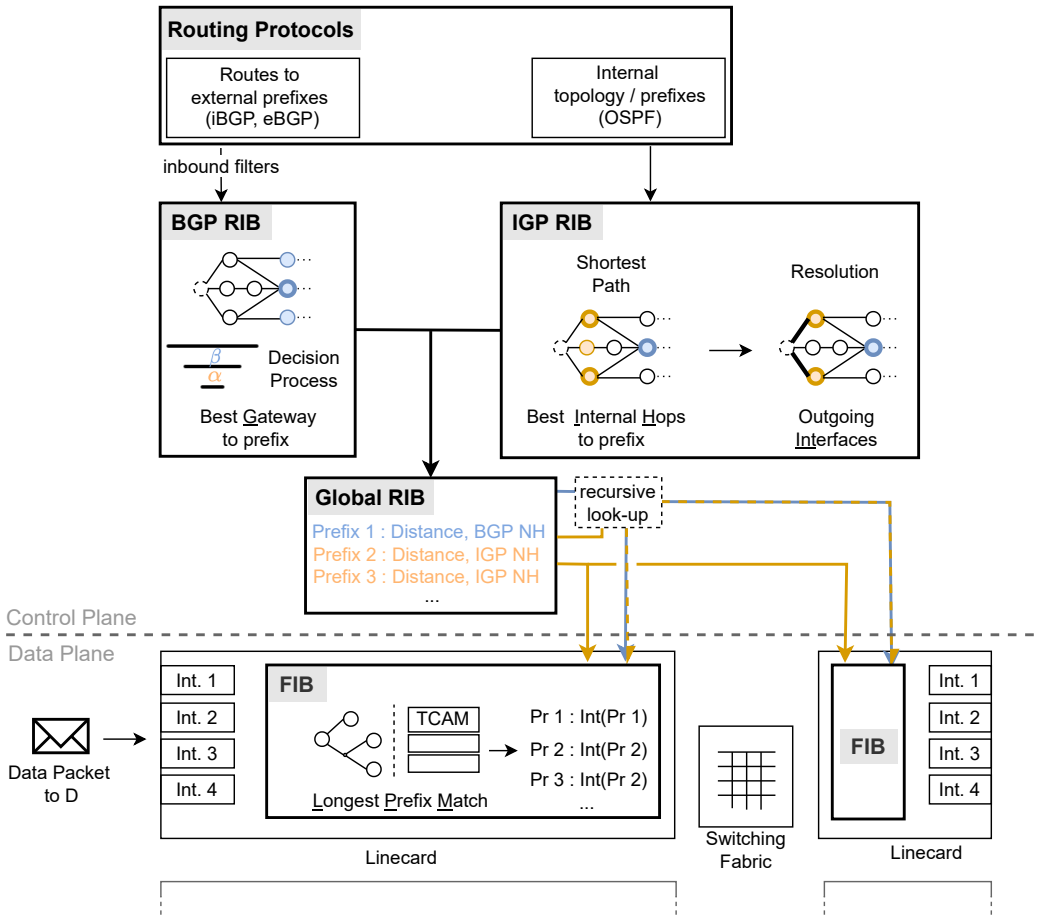


FIGURE 2.10: Figure summarizing the routing related concepts seen throughout this Section. Static routing and tunneling mechanisms are here ignored for the sake of readability.

NOTATION	DEFINITION
$(Node, u, v)$	Node segment read by u instructing to forward data to v
(Adj, u, v, x)	Adjacency segment enforcing the $(u, v)_x$ link. x may be omitted.
$S_1 \mid S_2 \mid \ldots \mid S_n$	A segment list composed of n segments.

FIGURE 2.11: Notations introduced within Section 2.4

decision process considering the routes attributes. The decision process creates an entanglement between BGP and the IGP, as the IGP distance to the advertising gateway is considered when ranking routes (hot-potato routing). Intricate iBGP topologies may lead to a lack of route visibility, which in turn also exacerbates BGP convergence time. This convergence is by itself quite long and occurs after each internal event due to hot-potato routing.

The routing of transiting traffic going through an AS thus results from an interaction between BGP and the underlying IGP. Not only does the BGP decision process considers the IGP distance, but the IGP is relied upon to resolve the chosen gateway before pushing the routing information within the FIB.

All protocols described until now follow the best-effort routing paradigm. Using such forwarding paths does however exhibit limits, which may not be tolerated by some clients. In some cases, one may thus require to forward traffic along different, specific paths. Performing such fine-grained routing is usually referred to as TE.

While there has been research about performing end-to-end TE across multiple ASes [Pelsser 2006], this task is quite arduous as cooperation between different ASes is not always possible. In this thesis, we focus on *intra-domain* TE [32]. In particular, our contributions described in Chapter 4 aim to provide DCLC intra-domain path while considering technical constraints induced by the current technologies. Consequently, we will now review the goals and ways to deploy TE within one's network.

2.4 Traffic-Engineering Deployment Technologies

There are several limitations to using solely shortest IGP paths within one's domain. First and foremost, best-effort forwarding paths do not aim to provide any strict guarantees. However, some types of traffic may require specific paths which enforce such guarantees, especially regarding their latency. Examples include high-stake communications such as financial trade flows, or 5G slicing [Programme 2020], which also requires the ability to guarantee end-to-end delay for the services it aims to provide. Metrics such as the delay may thus become as critical as the IGP cost (which should still be considered, as representative of the bandwidth and operational costs). Such specific flows should then be routed along multi-criterion paths that may deviate from the standard best-effort ones. This is the problem we consider in our contributions.

Second, while networks are usually designed to have enough capacity to route traffic through IGP paths without congestion, unexpectedly high traffic volumes or multiple independent failures may still result in congestion. Such congestion is not always tolerated.

[32] Intra-domain TE is still useful and widely deployed. In particular, intra-domain TE may still allow providing strong, meaningful end-to-end guarantees, for example when the AS is used as a VPN provider.

While IGP costs are oftentimes tuned to balance the traffic load, such configuration is quite complex [Hou *et al.* 2018, Fortz & Thorup 2000], and changing them dynamically without creating instabilities is an arduous task [Leduc *et al.* 2006, Clad 2014, Balon & Leduc 2008]. In addition, tuning IGP cost to this effect may prevent operators to use the latter for other means, *e.g.*, to express general design choices.

For both reasons evoked above, solely relying on shortest IGP paths may not be suitable. Tuning IGP costs may not always be viable or doable, especially if several distinct objectives must be expressed and fulfilled. It is thus often more appropriate to separate connectivity tasks and overall network design (handled by the IGP) from the more intricate aforementioned TE tasks, which require finer routing capabilities and thus additional technologies, which will be reviewed in this section.

2.4.1 Legacy Technologies and Models

The idea of differentiated service is not new. In 1994, the IntServ [Braden *et al.* 1994] model was imagined to allow flows to be routed over paths whose resources have been reserved beforehand, preventing for example unexpected delays. The resources could be reserved through an additional protocol, *Resource Reservation Protocol* (RSVP) [Braden *et al.* 1997]. For various reasons, IntServ had scalability issues, which lead to the conception of DiffServ. The DiffServ [Baker *et al.* 1998] model, standardized in 1998, allows differentiated handling of data packets through different *per hop behaviors*, depending on information added within the packet at the edge of the network. Being hop-by-hop, DiffServ was more in accordance with the overall routing philosophy used throughout the Internet. However, the latter could not enforce guarantees as strict as IntServ.

In practice, one of the most popular tools to perform traffic-engineering is MPLS [Viswanathan *et al.* 2001, Leduc *et al.* 2006] [33]. MPLS was designed during the 1990s and deployed at the very end of the same decade. The general idea behind MPLS is to establish (or deploy) *tunnels* over which data packets are forwarded. More precisely, within the data-plane, a header containing a *label* is added on top of data packets when the packet enters the network (*i.e.*, at the edge). This label indicates the tunnels (and so, path) the packet should take. Routers along the path forward the packet *solely* according to the value of said label, allowing them to forward data packets even if the ultimate IP destination is unknown to them. Conversely to standard hop-by-hop routing, MPLS can thus be seen as a way to implement *source-routing* : the path of the packet is decided upstream by the source (or the router closest to it) which prepends the associated label.

Routers must however synchronize the mapping of tunnels and label values to set up tunnels properly. This task is performed by the control-plane of MPLS through an additional protocol. The control-plane associated with MPLS is known to be quite complex and convoluted. The two main protocols used at this effect are *Label Distribution Protocol* (LDP) and *RSVP - Traffic Engineering* (RSVP-TE).

The LDP protocol allows deploying best-effort tunnels which follow the standard IGP paths. The main purpose of LDP is to allow core routers not to maintain a full RIB, as the

[33] Note that the description of MPLS and the associated protocols given in this section is drastically simplified. For a more in-depth, digestible explanation of these protocols, we refer the interested reader to [Luttringer *et al.* 2020c].

latter may forward packets according only to the label pre-prepended by the source. Thus, only the edge of the network has to maintain a full RIB with all destinations. LDP is for example often used to deploy BGP-free cores.

RSVP-TE allows deploying more intricate tunnels. RSVP-TE is an extension of RSVP. RSVP was indeed seen as suitable to distribute labels within the network: while reserving the required resources along the path, the latter could also be used to distribute the associated labels. As RSVP, RSVP-TE can be used to deploy specific paths with reserved resources which can be *explicitly* configured by the operator to perform fine-grained source routing.

Nevertheless, RSVP-TE suffered from significant drawbacks. Regarding bandwidth optimization, RSVP-TE required *all* traffic to go through tunnels to correctly assess the remaining resources. Thus, tunnels were set up between all pairs of nodes within the network. In addition, RSVP-TE did not allow to natively use ECMP paths. Thus, several tunnels (one for each ECMP path) had to be created between each source and destination, requiring setting up more than $|V|^2$ tunnels. Regarding the deployment of paths respecting latency constraints, the protocol suffered from similar scalability issues. Consequently, while RSVP-TE has been used for other purposes such as Fast Re-Route, the latter was rarely deployed for TE purposes [Filsfils *et al.* 2017].

However, during the last decade, a new technology was designed, allowing to easily perform large-scale, flexible fine-grained TE.

2.4.2 Segment Routing

SR [Filsfils *et al.* 2017, Filsfils *et al.* 2015] is a recent technology that enables network operators to perform various TE tasks, such as bandwidth optimization or explicit routing, while remaining more scalable and easier to configure than its predecessors.

SR is an implementation of the source routing paradigm: the source (or the node closest to it) computes and chooses how the incoming traffic should be steered through the network. Conversely to hop-by-hop routing, the path is thus completely decided by the upstream router and not by a sequence of independent routing decisions.

SR is, as of now, one of the most widely deployed and popular TE technology. In addition, source-routing is particularly suited to deploy multi-criterion paths [34]. As such, SR is at the core of our contributions.

SR implements source-routing by adding instructions within the packet header itself as a list of *segments*. Downstream routers solely consider the instructions within the packet to take forwarding decisions. Segments may be forwarding instructions but can be more general. Segments may for example be used to instruct to forward a packet to a given destination, but also to instruct to deliver the packet to a given application at a given node. SR can thus be used for a variety of purposes, including network programmability and service chaining [Halpern & Pignataro 2015, Filsfils 2020]. Furthermore, ressources specification (such as bandwidth) can be attached to each segment, in order to indicate the priority of the traffic being steered and perform resource reservation [Dong *et al.* 2022]. In this thesis, we will

[34] Recall that relying on hop-by-hop distributed routing may be challenging, as the subpath optimality does not hold per se when considering multi-criterion paths

consider SR to build multi-criterion forwarding paths, and thus mainly focus on forwarding instructions.

Regarding forwarding instructions, the segments used are usually IGP segments. IGP segments are routing instructions linked to the information relative to the IGP. Numerous type of IGP segments exist. In this thesis, we will focus on *prefix segment* (in particular, *node segments*) and *adjacency segments*, as they provide a sufficient framework to deploy multi-criteria paths.

2.4.2.1 Building paths using segments

An IGP prefix segment can be used to instruct routers to forward the packet to a specific IGP prefix, using the shortest paths to said prefix. As IGP segments are constructed upon IGP information, the shortest paths used are the shortest ECMP-aware IGP paths.

Node Segments are a particular type of prefix segments that identify a node within the network (typically, through the loopback interface of the corresponding node). In other words, using a node segment allows instructing a router to forward the packet to a specific node within the network via the shortest paths, in an ECMP-aware fashion. We denote a node segment s instructing to forward a packet to v and read by node u as $s = (Node, u, v)$.

Adjacency segments allow instructing routers to forward the packet through a specific interface. It can thus be used to enforce the use of a specific link along the path. We denote an adjacency segment s enforcing a link $(u, v)_x$ as $s = (Adj, u, v, x)$ where x denotes the specific interface when necessary.

These segments, pre-pended (or pushed) to the packet by the source, act as building blocks to construct the desired forwarding paths. They can thus be combined in a *segment list*, which may encode any desired path.

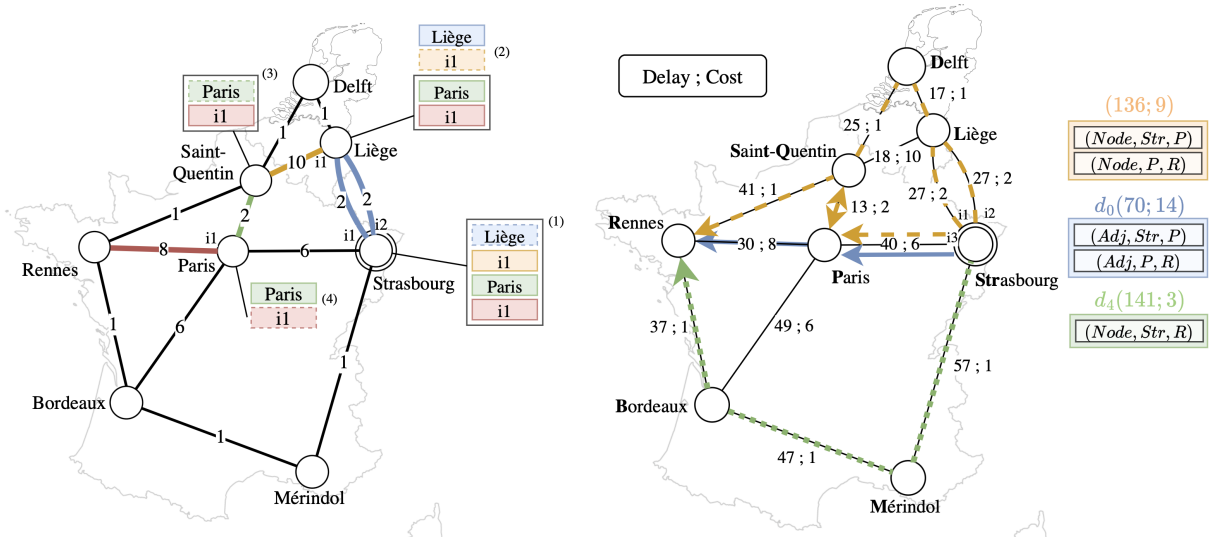
Note that SR does not require specifying the entirety of the forwarding paths. Rather, only the necessary deviations from the shortest paths have to be specified. SR is said to implement *loose* source routing. A segment list thus describes forwarding paths as a concatenation of links and shortest paths. We denote a segment list S comprised of n segments as $S = S_1 \mid S_2 \mid \ldots \mid S_n$.

2.4.2.2 The data-plane and control-plane of Segment Routing

Routers along the path forward a packet according to its topmost segment, which is called the *active* segment. If the active segment is completed, *e.g.*, the packet has reached the intermediate destination specified by the latter, it is removed from the list, and the following segment within the list becomes active.

These mechanisms can be deployed by relying on the MPLS or IPv6 data-plane [Bashandy *et al.* 2019, Filsfils *et al.* 2021a]. In this thesis, we will focus on SR-MPLS, although most of the discussions, as well as our contribution, remain valid for both approaches.

By encoding segments as labels, the standard MPLS data-plane provides all the necessary operations to manage the segment lists. However, SR-MPLS does not rely on the MPLS control-plane to distribute labels or (segments). Rather, the IGP segments are directly advertised through the IGP itself, simplifying the process.



(A) Encoding of an arbitrary path using SR. The segment list necessary to encode the path shown in blue is shown in black boxes. Its evolution is shown at each intermediate hop. Segments removed are shown outside the black box. The segment being interpreted (*i.e.*, the active segment) is shown in dashed lines.

(B) Encoding of multi-criterion paths (as illustrated in Fig 2.3) through SR. A path of a given color can be encoded by the segment list shown in the respective colored box. Paths bear the same names as they did in Fig 2.3. The effect of ECMP must not be neglected in order to respect strict constraints.

FIGURE 2.12: Figures illustrating the translation of paths to segment lists.

The advertised segments may be *global* or *local*. Global segments are supported by all SR-aware nodes within the network and have a global meaning. Conversely, local segments have local meaning. More precisely, local segments advertised by different nodes may share the same value but correspond to different instructions. To ensure that a local segment is interpreted properly, the packet must therefore first be steered to the node that advertised said segments (*e.g.*, through the use of a node segment). Note that all local segments must still be advertised throughout the network, in order to be known and used by the source to construct the path. The main point of local segments is the ability to re-use the same label value, as the number of global segments one may define is limited to around 9000 [Filsfils *et al.* 2017] [35].

When using SR-MPLS, node segments are by default global. The instruction "*forward packet to v through ECMP paths*" associated to these labels is thus understood in the same fashion by all SR-aware nodes. In other words, $(Node, u, v)$ share the same meaning for any node u [36]. However, adjacency segments, which could be translated as "*forward packet over interface X*" have local meaning: one must ensure the packet is first steered to the correct node.

Fig 2.12a illustrates an example of SR within the topology used throughout this chapter. Let us consider the segment list shown on the right side of the figure, which encodes the

[35] With SRv6 (using IPv6 instead of MPLS), the possible number of values for segments is larger, allowing to consider adjacency segments as global segments. However, such technologies are often used with segment compression techniques [Tulumello *et al.* 2020], which once again limit the total number of segment values and require to use local segments.

[36] Given this information, our choice of notation may seem unnecessarily heavy. However, we require this level of precision for further proofs and formalization found within Chapter 4 of this document.

path shown on the topology.

The first active segment is a node segment instructing the packet to reach Liège first. As it is a node segment, the packet is forwarded to Liège following the ECMP paths, *i.e.*, the two parallel links. Arriving at Liège, the segment is completed and removed. The next segment is an adjacency segment that requires forwarding the packet through the $i1$ interface. The instruction is executed, and the segment can be removed immediately as it only has local meaning. Note that using a node segment to reach Saint-Quentin would have made the packet go through Delft as the shortest IGP path is not the direct link. Arriving at Saint-Quentin, the packet is forwarded towards Paris through the best IGP path, which here is the direct link (hence a node segment is sufficient). Paris removes the completed segment, and executes the last instruction, sending the packet to Rennes through interface $i1$ and removing the last segment (which has now been completed).

Fig 2.12b illustrates how SR may be used to encode several paths whose distances were seen in Fig 2.3. Note that this time, we use the formal notations defined within this section. The distance d^4, for example, was the shortest IGP distance to Rennes. As such, a single node segment is enough to encode a path exhibiting these distances.

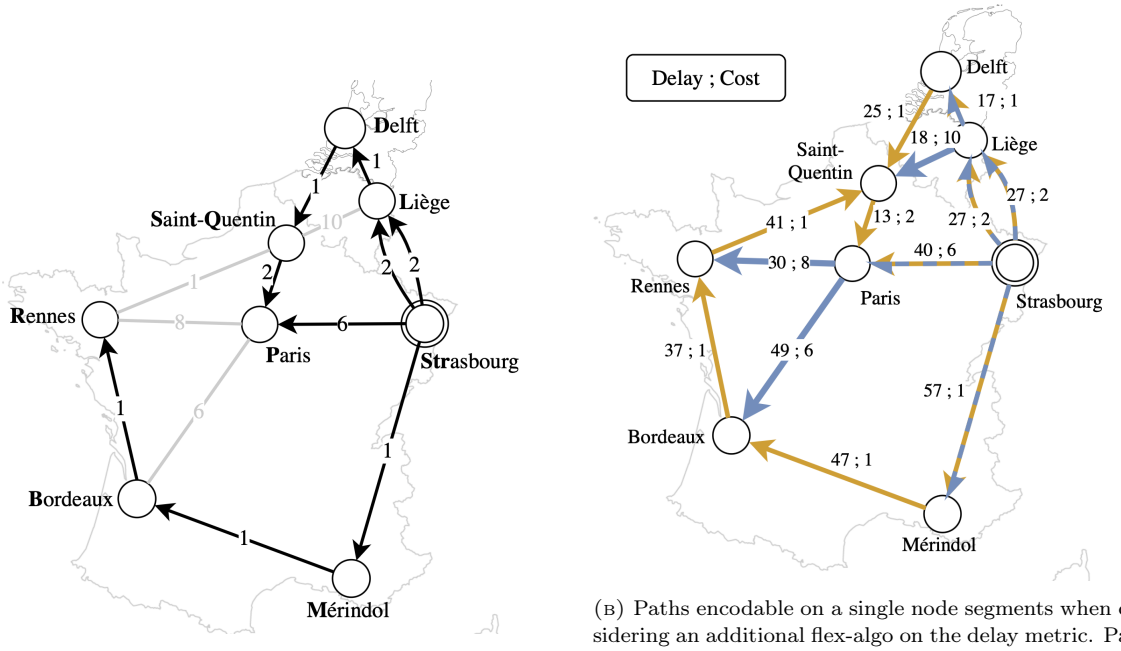
The distance d^0 was particularly interesting, as it was the distance minimizing the latency by going through Paris. Note, however, that solely relying on node segments to steer the packet through Paris (see the orange segment list in Fig. 2.12b) is not sufficient to guarantee that the distance d^0 will be respected. Indeed, packets may take any ECMP paths from Strasbourg to Paris (shown in orange in Fig 2.12b). Thus, packets may *either* take the direct link or go through Liège. Interestingly, once the packet reaches Paris, the use of a node segment to steer the packet to Rennes will result in Paris sending the packet to Saint-Quentin, which could cause a forwarding loop. Indeed, one of the possible paths used to reach Paris is Strasbourg;Liège;Saint-Quentin (one of the two ECMP paths available). By sending the packet to Rennes through its shortest IGP path, Paris sends the traffic back to Saint-Quentin. Consequently, when using two node segments, the traffic may follow any of the paths shown in dashed orange lines, and the *only* latency that may be guaranteed is the *worst latency among all these possible paths*, here 136.

To enforce the path with the best delay (70), the link between Strasbourg and Paris as well as the link between Paris and Rennes must be explicitly enforced through the use of adjacency segments. Therefore, the required segment list is $(Adj, Str, P) \mid (Adj, P, R)$.

2.4.2.3 Considerations on the number of segments

In practice, the insertion of segments in data packets necessarily implies an extra overhead. Thus, the number of segments one may prepend to a packet at line-rate is limited. This limitation is highly dependent on the underlying hardware. High-end routers may allow segment lists of up to $\approx$ 10 segments. Less performant equipment, however, may allow only between 3 and 5 segments [Guedrez *et al.* 2016a]. This limit, referred to as *Maximum Segment Depth* (MSD) must thus be taken into consideration when encoding paths.

Indeed, although most standard applications relying on SR (*e.g.*, Fast ReRoute mechanisms) do not require a lot of segments [Filsfils 2020, Aubry 2020], the deployment of more intricate multi-criterion paths may require a higher number of segments. Such technical con-



(A) Paths encodable in an single IGP node segments (*i.e.*, paths on the shortest path DAG as computed by the IGP).

(B) Paths encodable on a single node segments when considering an additional flex-algo on the delay metric. Paths resulting from the IGP are shown in orange, while the ones considering the delay are shown in blue.

FIGURE 2.13: Comparison of path encodable in a single segment with and without the use of a custom flex-algo.

straint, often ignored in path computation algorithms, does have an impact on how paths should be computed and on the complexity of related algorithms. In Chapter 4, we will detail two of our contributions which allow considering the number of segments necessary to encode multi-criterion paths, and thus ensure that the paths computed may actually be deployed in practice.

It should however be noted that several mechanisms exist to mitigate the MSD limit.

Binding Segments (BSIDs) [Filsfils *et al.* 2018], in particular, were designed to such effect. BSIDs are (usually) local labels bound to a segment list S. When the active segment is a BSID, it is removed and replaced by the associated segment list S. This mechanism, when configured correctly, allows to both reduce the segment list size and the churn. The latter may also allow for network opacity by exchanging only BSIDs with another area or domain rather than explicit segment lists. However, they require additional states to maintain and induce additional complexity regarding the configuration of SR routers.

Flexible Algorithm [Psenak *et al.* 2020], or Flex-Algo, may also be used to reduce the number of segments necessary by enriching the segment set available through the use of multi-topologies. As explained earlier in this section, prefix segments are usually built upon the IGP costs. Node segments thus allow routing traffic to a given node through the path minimizing the IGP costs. In other words, paths whose edges follow the shortest path IGP DAG, as shown in Fig 2.13a, can be encoded in a single segment. Flex-Algo allows considering other metrics (*e.g.*, the delay), even with topological constraints (*e.g.*, excluding sets of links or nodes). Node segments may thus enable to steer packets either following the shortest path according to the IGP cost or according to the delay, depending on the algorithm associated with said node segments. Consequently, when using an additional Flex-Algo considering

the delay, paths whose edges follow the shortest path DAG delay-wise, as illustrated in blue in Fig 2.13b, may also be encoded in a single segment. Delay-specific or IGP-specific segments may be used in conjunction. However, similarly to BSID, Flex-Algo adds numerous additional states within routers. In any case, note that a DCLC path may still require several segments, even when using a Flex-Algo on the delay metric in addition to the standard IGP segments.

We will not delve further on these technologies in the remainder of this thesis. However, our contribution, BEST2COP, may benefit from (and is compatible with) both technologies.

In this section, we discussed technologies allowing to deploy non-best-effort paths. SR is an implementation of loose source routing which allows deploying paths by pre-pending routing instructions within the packet. For various reasons, such as scalability and simplicity of configuration, SR is, as of now, the most popular technology to perform TE. However, algorithms relying on SR must consider the technical constraints of the latter, in particular regarding the maximum number of segments that may be prepended to each packet.

2.5 Conclusion

In this chapter, we have reviewed the necessary background for the remainder of this thesis. We have seen that graphs offer an ideal formal framework to study network-related problems. Although computer networks exhibit several relevant metrics, best-effort paths, used by most of the traffic, usually follow paths that minimize a single metric, the IGP cost, representative of the bandwidth and the operational costs. We have reviewed the most relevant algorithm used to compute such paths, Dijkstra's algorithm and BFM. We have seen that computing multi-criterion path is more complex and often results in NP-Complete problems, leading to a large array of attempts to solve the problem.

We have seen that within networks, the information required to compute such paths is exchanged through routing protocols, whose paradigms may differ. These routing protocols also dictate how the traffic is forwarded. Most of the time, routing is performed in a hop-by-hop fashion, enabled by the subpath optimality property. Inter-domain routing (in particular, BGP) relies however on different path computation techniques, which allows greater flexibility to account for the economical relationships between ASes. While both intra and inter-domain protocols may take a long time to converge, the convergence time is orders of magnitude higher with BGP due to the scale at which the protocol operator, and its sensibility to intra-domain events.

Finally, we have discussed that best-effort paths may not be sufficient for some specific premium flows, which require more elaborated paths. While several technologies enable to deploy of these paths, SR is currently the most popular TE technology. However, the latter exhibits some technical limitations (in particular, the number of segments that one may prepend). The limitation on the number of segments should be considered when computing paths, to ensure that the latter are deployable, and thus adds a constraint to the shortest path problem.

The concepts described in this Chapter will enable us to review the relevant literature in the following chapter in order to better position the contributions of this thesis, which will be explained afterward in Chapter 4 and 5.

Chapter 3

Related Work

Contents

In the previous chapter, we have detailed the concepts necessary to understand and describe not only our contributions but also the associated related work. Although both contributions in this thesis are related to routing, they tackle different objectives and lie within different contexts.

OPTIC aims at mitigating the effect of hot-potato routing. As we've seen, even an internal event may re-trigger the slow BGP decision process. OPTIC aims at better organizing the BGP routes to enable a fast re-route of the transiting BGP traffic through the new optimal route after an internal event. As such, OPTIC closely relates to schemes improving the convergence of protocols, and to *Fast ReRoute* (FRR) schemes. FRR schemes are a key feature of computer networks, as most routing protocols may take too much time to converge after a network event. These schemes will be reviewed first within this chapter.

On the other hand, BEST2COP aims at computing DCLC paths deployable with SR. As such, the latter must solve a multi-criteria path computation problem, but also consider the associated segment lists. We will thus first review SR-TE contributions. While most SR-TE contributions focus on different challenges than BEST2COP, they sometimes share similar

intent. Existing works also often rely on or propose schemes to translate paths into segment lists that share similar insights to the ones we propose.

Since most SR-TE solutions do not aim to solve DCLC, we will review the less specific literature focusing on the computation of multi-criteria paths. The computation of multi-criterion paths is a fairly old and rich subject, with numerous interesting contributions both in the field of operational research and computer networks. The third and final part of this chapter will then review some of these path computations algorithms and help position BEST2COP within this vast ecosystem.

3.1 Resiliency

The convergence time of routing protocols is a critical aspect of computer networks. The faster a protocol can re-converge, the faster traffic can benefit from coherent optimal routing. As we have seen, the actual convergence time may be quite long, ranging from up to a few seconds for intra-domain protocols to several minutes for inter-domain protocols.

Several causes impact the overall convergence time of a protocol. Not only must the failure be detected (which may take a long time for remote routers), but paths must be recomputed, and the FIB updated.

The challenges involved are thus varied and stimulating, as they often require graph theory notions while also considering the inner operations of routing devices. Thus, a lot of research effort went into trying to solve the issues caused by long convergence times.

Low-level improvements usually benefit all protocols. For example, routers may detect connected failures directly through their linecard, *e.g.*, by detecting loss of signal, in less than 50ms [Katz & Ward 2010] (instead of relying on periodic control messages). The FIB being (traditionally) protocol-agnostic, improving its update time also positively impacts all routing protocols. This update time, which was originally measured in milliseconds in the very early 2000s, was reduced to several microseconds over the course of several years [Filsfils *et al.* 2017][1]. Mitigating the FIB update time by prioritizing the updates of the most popular prefixes is also a common technique applied for different protocols [Francois 2007, Brenes *et al.* 2020].

However, we have seen that protocols often rely on drastically different paradigms and computation methods. Thus, most convergence improvements and *Fast ReRoute* (FRR) schemes are designed for a specific protocol (be it intra-domain or inter-domain), to encompass its specificities.

Some try to improve the actual convergence time of said protocols (*e.g.*, through new ways to detect failures or improved path computation), while others try to mitigate the detrimental effects that may occur during the convergence time (*e.g.*, by offering emergency fail-over paths while the protocol re-converges).

Our contribution, OPTIC, aims at improving the convergence of BGP upon internal events. While it is specific to BGP, it thus lies at the border of intra- and inter-domain protocols. Numerous other schemes exist. While some (albeit few) also focus on iBGP and

[1] In 2005, the update time of one FIB entry was measured to take around 140 microseconds [Francois 2007]. Interestingly, these measurements seem to have remained fairly stable over the years [Boucadair 2005, Filsfils *et al.* 2011, Holterbach 2021].

aim to solve the same problem as OPTIC, others focus on purely intra- or inter-domain protocols and may interact with or be used in conjunction with our solution. Thus, in this section, we will review intra- and inter-domain FRR and re-convergence schemes, before reviewing in detail the solutions more closely related to OPTIC.

As it is not my goal to list all contributions here, we refer the interested reader to the extensive survey by Chiesa et al. [Chiesa *et al.* 2020] which describes several fast re-route schemes and offers interesting historical insight. The thesis of Pierre François reviews in-depth the various improvements that enabled faster re-convergence [Francois 2007].

3.1.1 Intra-Domain Fast Reroute

Several works focused on improving the convergence of intra-domain routing protocols. For example, some proposed ways to ensure that no loop arises during the re-convergence of these protocols, through ordered FIB updates or careful incremental IGP cost modifications [Clad *et al.* 2014, Francois & Bonaventure 2007]. These works however focused on coherent convergence upon non-urgent or planned changes.

Other works focus on the convergence speed of intra-domain routing protocols. The cumulation of these improvements had a drastic impact on said convergence time, which was reduced from up to ten seconds to less than one second in the past decades [Filsfils *et al.* 2017, Francois *et al.* 2005b].

Some contributions enable to reduce the path exploration of vector-based protocols (both path-vector and distance-vector). For example, the DUAL protocol is an enhanced distance-vector protocol imposing conditions regarding the update of a router's distance-vector to limit path exploration [D'Angelo *et al.* 2014]. Somewhat similarly, *Metrics and Routing Policies Compliant* (mRPC) delays route updates (or FIB updates) according to path metric and associated policies, so that routers learn the best route first [Lambert *et al.* 2009].

The re-computation of shortest paths has also been improved, mainly through the use of incremental shortest path algorithms [McQuillan *et al.* 1979] which allow routers to only recompute obsolete parts of the shortest path DAG upon changes.

In addition, intra-domain *Fast ReRoute* (FRR) schemes have also been designed to further reduce the unreachability period during convergence time. FRR schemes exist for numerous types of intra-domain protocols.

For example, MPLS FRR allows protecting tunnels deployed through MPLS from disruptions, for example by pre-computing backup tunnels to protect primary tunnels [Taillon *et al.* 2020].

IP FRR, on the other hand, offers fast rerouting capabilities compatible with the standard prevailing paradigm of hop-by-hop IP routing. IP FRR also often relies on path protection, *e.g.*, by pre-computing backup next-hops ensuring a coherent routing for a given destination upon failure. The destinations for which such next-hops have been computed are said to be *protected*. The backup next-hops are installed within the data-plane in the FIB, and used when a failure is detected. Note that while ECMP may by itself suffice to protect against some failures, additional mechanisms are required to ensure that a destination is protected for any topology or configuration.

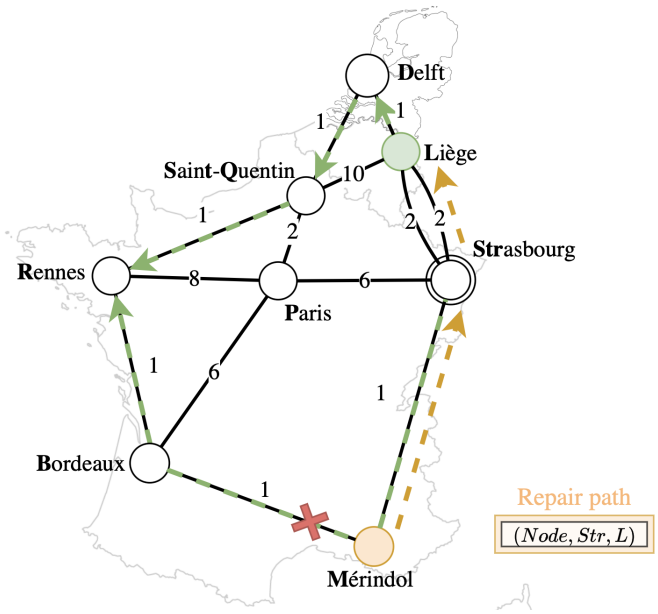


FIGURE 3.1: Example of LFA FRR through *Topology Independent LFA* (TI-LFA). Mérindol does not have any direct LFA, and thus uses a remote node as a repair node through the use of SR.

Loop Free Alternates (LFAs) are the most well-known intra-domain FRR [Atlas & Zinin 2008]. LFAs leverage the knowledge that comes with link-state protocols. Upon a failure, LFAs allow routing traffic to a pre-computed router whose (pre-convergence) shortest path [2] does not go through the failed component. Since the protection is local, this is equivalent to ensuring that the packet does not loop back to the local router. Traffic is then routed to this router, called the *repair node*, which forwards it towards the destination using its standard IGP shortest paths. The repair node may be directly connected or remote, depending on the topology. Remote repair nodes may be reached safely by setting up tunnels to the latter [Bryant *et al.* 2015], which results in additional control-plane overhead.

The first versions of LFA suffered from limited failure coverage on certain topologies [Francois *et al.* 2013]. In addition, the backup path taken from the local router to the destination may not be the post-convergence path. Traffic would then be re-routed first to the backup path, then to the post-convergence one, which may result in desequencing.

The most recent LFA method, TI-LFA, implements the LFA principle through the use of the SR technology [Litkowski *et al.* 2022]. Thanks to SR, TI-LFA has complete control over the backup path for no additional control-plane overhead (if all routers implement SR). Thus, not only does it offer complete coverage against any kind of failure, but the backup path used to the destination can be the post-convergence one (from the point of view of the local router). An example of LFA FRR can be seen in Fig 3.1. When trying to reach Rennes, the traffic from Strasbourg goes through Mérindol. Mérindol detects a directly connected failure on his link to Bordeaux. Note that no direct neighbor can be used as an LFA: using its best path, Strasbourg would forward the packets through the failed component. Thus, Mérindol must forward the packet to Liège, where it can be released and forwarded through the best IGP paths to Rennes without going through the failed link.

[2] Recall that said router is not necessarily yet aware of the failure

This backup path can be deployed through the use of a single node segment.

Other notable FRR solutions exist, which take different interesting stances. For example, overlay-based FRR pre-computes backup overlays (*i.e.*, virtual topologies) that are unaffected by a given set of failures, and route packets on the latter according to the state of the network [Kvalbein *et al.* 2009, Theiss & Lysne 2003]. *Failure Insensitive Routing* (FIR) relies on the assumption that most failures are transient. Thus, only routers directly connected to the failure recomputes their shortest path, but remote routers do not. Rather, upon receiving traffic through an unusual interface for a given destination, the latter infers the location of the failure and route around it. The outgoing interface thus both depends on the destination *and* the ingoing interface [Nelakuditi *et al.* 2003]. As FIR does not fit standard hardware, it was never deployed (although new programmable data-planes seem like a good fit for the latter).

Some solutions are purely data-plane related, such as *Data Drive Connectivity* (DDC) [Liu *et al.* 2013] (which implements the link-reversal algorithm [Gafni & Bertsekas 1981]), and may be implemented *fully* within programmable data-planes, *e.g.*, with P4. On a similar note, Chiesa et al. proposed a P4 primitive, PURR, allowing to implement the data-plane of most FRR schemes very efficiently, notably through the use of optimized TCAM entries to find the first available outgoing port [Chiesa *et al.* 2019]. Interestingly, this idea of leveraging TCAM can be also found in Plinko [Stephens *et al.* 2013], a FRR scheme consisting in rerouting packets to backup routes upon local failures, *"bouncing"* the packet around until it reaches its destination.

Intra-domain fast-reroute may have a positive impact even on BGP traffic. Indeed, the latter may enable to quickly restore the connectivity towards the BGP gateway used by transit traffic. Note however that said gateway may not be optimal anymore due to hot-potato routing.

In any case, these schemes do not manage the failure of BGP gateways and do not improve the convergence time of BGP. Thus, while intra-domain FRR may be deployed on top of BGP-related schemes (including our own), these BGP-related schemes remain necessary to handle convergence challenges specific to this protocol.

3.1.2 Inter-Domain Fast Reroute

BGP's convergence time relies on the same key factors as IGPs, *i.e.*, the detection of the failure, information dissemination, the computation of the new paths, and the update of the FIB. However, the time taken by each of these tasks is exacerbated by the scale at which BGP operates. Failure notifications may have to travel across the Internet and undergo timers [Villamizar *et al.* 1998]. Path computation has to consider nearly three million entries. Finally, nearly a million entries must be distributed to each FIB. Overall, the convergence of BGP may be measured in minutes [Labovitz *et al.* 2000a]. Consequently, several schemes have been designed to improve the convergence time of BGP or provide FRR capabilities.

3.1.2.1 Reducing convergence time at the inter-domain scale

One of the philosophies followed to improve the convergence time at the inter-domain scale is to build upon the current protocols and architectures. These solutions can be deployed quite easily, as they do not aim to completely re-design the Internet, which, despite its highly dynamic nature, is fairly immutable from the protocol perspective.

For example, an option to reduce BGP's convergence time is to limit path exploration. Indeed, as a vector-based protocol, it has been shown that BGP may require to process up to $(n-1)!$ updates, with n the number of nodes [Labovitz *et al.* 2000a]. Schemes such as mRPC (presented above) which aimed at reducing path exploration for incremental protocols can thus also be used with BGP to this effect. Since pathological cases of path exploration only occur when considering asynchronous communications, Bremler-Barr et al. [Bremler-Barr *et al.* 2009] proposed to enforce a relaxed form of ordering in the message exchange through the use of timers considering the length of the route to advertise (through the AS-PATH). EPIC [3] is an enhanced path-vector protocol aiming specifically at reducing path exploration through the use of additional information carried within the messages.

Other solutions proposed to enhance BGP's convergence time relied on adding information regarding the origin of the event generating the update messages [Luo *et al.* 2002, Pei *et al.* 2005], allowing ASes to immediately deduce other routes invalidated by the update and removing the latter from its RIB.

Finally, other works focus on better tuning the BGP parameters to enhance its convergence time. In most of the cases, this improvement concerns the several timers in place to prevent BGP from overreacting to transient changes and noise [Pelsser *et al.* 2011, Sahoo *et al.* 2006]. For example, Bremler-Barr et al. propose to let withdrawals propagate with no delay throughout the Internet (while route advertisements still go through timers) and show that this scheme reduces both the convergence time and message complexity of BGP [Bremler-Barr *et al.* 2003].

Another philosophy is to consider that enhancements patching the current Internet, while easier to deploy, do not offer enough flexibility to tackle its main issues. Consequently, the following solutions propose a more drastic redesign of today's architecture and protocols. While these solutions may indeed propose an improved basis onto which the Internet could be (re)built, actually deploying them is far more challenging than the solutions previously described.

For example, *Scalability, Control, and Isolation On Next-Generation Networks* (SCION) proposes to redesign the Internet architecture to improve security and scalability [Barrera *et al.* 2017]. SCION relies on logical grouping of ASes into trusted domains, among which a few *core* ASes are in charge of connecting to other core ASes within other domains. SCION also rely on a drastically different control-plane, with finer path information and which leverage the proposed architecture to improve scalability and path selection.

Hybrid Link-state Path-vector (HLP) proposes to leverage the hierarchical nature of the Internet [Subramanian *et al.* 2005a]. Seeing Tier-1 ASes as the root of a given hierarchy, HLP uses a link-state approach within a hierarchy, and a path-vector approach between the root of each hierarchy. Leaving aside some peculiarities, this approach is reminiscent of the approach taken by multi-area OSPF topologies (with Tier-1 ASes here assuming a role

[3] Funnily enough, EPIC was also the first name of our contribution, OPTIC.

similar to ABRs in the backbone area).

Because of the scale of the Internet, there seems to be a limit to how much BGP's convergence time can be reduced. For example, even detecting the remote failure may take a long time, as updates have to travel across several ASes. Similarly to intra-domain protocols, FRR schemes have thus been designed to mitigate the effect of the convergence time of BGP upon remote events outside the local domain.

3.1.2.2 Providing inter-domain FRR

Among inter-domain FRR proposals, R-BGP [Kushman *et al.* 2007] aims to provide ASes with backup routes for remote failures. Before the failure, R-BGP allows an AS to announce a route for a given destination to the downstream AS currently used to reach said destination. Ideally, the path advertised to the downstream neighbor is as disjoint as possible from the current path being used. Upon failure, ASes should thus have a disjoint path available, advertised by the upstream AS. The authors however note that the interaction between R-BGP and the BGP convergence may create loops. These forwarding loops may be prevented by exchanging root-cause information within BGP messages.

SWIFT [Holterbach *et al.* 2017] specializes in providing *predictive* FRR for remote failures. In short, Swift uses the initial bursts of BGP control messages received after a remote failure to infer the location of the latter, through an intricate inference algorithm. Thus, routers do not have to wait for all updates before re-routing traffic. Traffic is re-routed through backup routes which have been pre-computed continuously for all routes and relevant possible inter-AS link failures (*i.e.*, lying on the AS-path of the original route). Note that the associated time and memory complexity may thus be quite high.

Blink [Holterbach *et al.* 2019] shares the same general goal as SWIFT. However, the latter tries to provide even faster failure detection by completely bypassing control-plane messages. As control-plane messages take a while to be disseminated throughout the internet, Blink tries to predict a failure purely through data-plane information. More precisely, Blink detect failures by monitoring TCP flows, which exhibit predictable behavior upon remote disruptions. Once the failure is detected, routers may switch to backup paths which were pre-configured by network operators. Interestingly, Blink was implemented fully in P4, and thus offloads these tasks to the data-plane of the router. However, this forces Blink to remain quite simple and thus limited.

Blink and SWIFT have been combined into Snap [Holterbach 2021]. Like Blink, Snap monitors data-plane signals. However, the monitoring is performed in software to provide the smart inference capabilities of SWIFT. The sampling of the flow is however still performed through P4. Interestingly, this change of design seems to be quite common. With the arrival of programmable data-planes, several solutions aim at being fully implemented in P4 even if their design did not really fit P4 capabilities. However, other solutions tend to aim for a *co-design* where intricate tasks are delegated to the control-plane, argumenting that communicating with the latter should not be too expensive. These two points of view lead to interesting (and sometimes contradictory) arguments within the P4 literature.
Similarly to the intra-domain schemes described in Section 3.1.1, these inter-domain schemes aim to tackle different objectives as OPTIC. The latter do not consider the behavior of BGP within a given AS, but rather examine and improve the behavior of inter-domain routing at

a larger scale. OPTIC, on the other hand, aims to improve the BGP convergence within the AS, and reduce the ill-effects of the interaction between BGP and the IGP. Nevertheless, both types of schemes may be deployed concurrently.

3.1.3 Intra/Inter-Domain Convergence

As mentioned in Section 2.3.2.2, several BGP-related issues may arise even within a single AS. The number of entries BGP has to manage leads to time-consuming re-ranking and FIB updates. Because of hot-potato routing, this process is triggered at each IGP event. Finally, reduced route visibility and complex iBGP topologies may prevent the protocol to converge and routers from being aware of backup routes in the event of a failure. Consequently, many solutions have been proposed to tackle these issues.

3.1.3.1 Guaranteeing correctness

Some work focus on guaranteeing correctness, *i.e.*, ensuring that iBGP converges and that the ensuing forwarding does not exhibit anomalies. An iBGP topology that allows iBGP to converge is said to ensure *signaling correctness*. If no forwarding anomalies arise, *forwarding correctness* is ensured.

Guidelines were proposed to ensure both signaling and forwarding correctness [Griffin & Wilfong 2002b]. However, these guidelines may be too strict to be applied easily within an operator's network. Other works only focus on signaling correctness. Buob et al. define *fm-optimality* to model the problem that may arise when iBGP routers disagree on the route that should be used. They propose a way to check if an iBGP topology is fm-optimal, *i.e.*, that a valid signaling path (where no intermediary routers disagree on the preferred exit point) exists between all routers and exit points [Buob *et al.* 2007]. In particular, the authors show that standard routing issues (oscillation, non-optimality, and non-determinism) cannot occur *"if each BGP speaker learns its best possible exit point"*. They subsequently proposed a way to design iBGP topologies achieving such results [Buob *et al.* 2008]. A few years later, the authors proposed another interesting way to guarantee fm-optimality, by redesigning the route distribution protocol itself. This new protocol, iBGP2, ensures that all routers have access to their preferred exit point while remaining scalable [Buob *et al.* 2016]. In short, a BGP speaker v advertises an exit point n to its neighbor only if v lies on the shortest path between u and n. If n is the preferred exit point for u, then it is necessarily the preferred exit point for v, meaning that BGP speakers will indeed be advertised (at least) their preferred exit point. This approach is somewhat similar to the one proposed by Gvozdiev et al. through the *Simple Ordered Update Protocol* (SOUP) protocol, which ensures that a BGP speaker only selects a route if the associated IGP next-hop has chosen the same route. SOUP was also optimized to *Link-Ordered Update Protocol* (LOUP) to speed up the convergence [Gvozdiev *et al.* 2013].

Vissicchio et al. showed however the limitations of some of these works, by noting that an iBGP topology may allow the protocol to converge correctly (*i.e.*, ensuring signaling correctness), but may *still* have some route dissemination issues (*i.e.*, some routers may not be aware of routes to remote prefixes) caused by iBGP route propagation rules [Vissicchio *et al.* 2012]. These dissemination issues may cause forwarding anomalies by themselves. This new requirement for forwarding correctness, called *dissemination correctness*, is then studied.

On a side note, some works also propose incremental re-configuration of BGP to migrate between two network states while preventing traffic disruptions [Vissicchio *et al.* 2013]

Most of these works do not however focus on improving the actual convergence process of BGP.

3.1.3.2 Increasing route diversity

Works that aim at increasing the route diversity not only bring correctness guarantees (as each router may learn its best possible exit point) but also benefit the BGP convergence process by limiting the required message exchanges upon an event (as more information is already locally available).

Improving route diversity may be done through clever iBGP topology design. For example, Pelsser et al. propose an algorithm that augments an iBGP topology with additional iBGP sessions to ensure that any BGP speaker is aware of at least two routes to remote destinations [Pelsser *et al.* 2008]. However, Vissicchio et al. showed that adding such iBGP sessions, while increasing the visibility of the local router, may lead to a loss of visibility to the other routers within the AS due to iBGP routes propagation routes [Vissicchio *et al.* 2012].

Most schemes improving route diversity thus rather rely on slight extensions to the protocol itself, allowing BGP speakers to share more than a single route to their peers.

BGP `best-external` [Marques *et al.* 2012], for example, allow routers to advertise their best route *and* its best external route, *i.e.*, the best route learned by an eBGP peer.

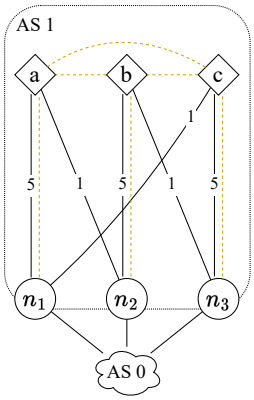
More flexible, Add-Path [Walton *et al.* 2016] allows routers to exchange a set of routes, depending on different paradigms. As OPTIC shares similar concepts as Add-Path and fits very well upon the latter, we will now describe Add-Path in further detail.

Several Add-path *modes* exist, which allow routers to exchange specific sets of routes ensuring different guarantees. Add-Path can for example allow routers to exchange *all* known routes, which solves any internal correctness issues and allows routers to converge immediately to the new post-convergence paths. However, this mode does not scale well for obvious reasons.

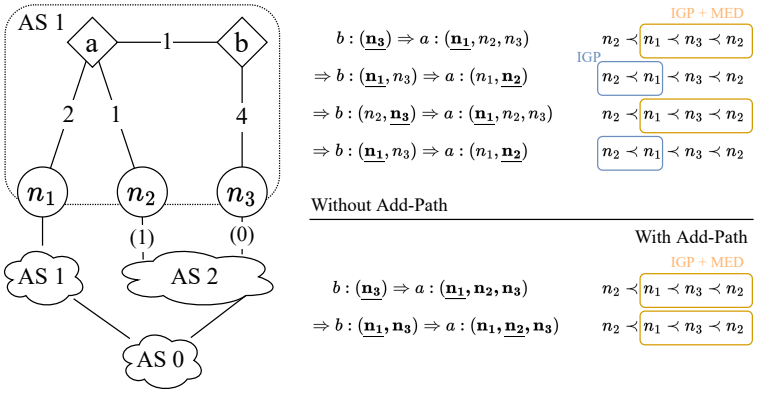
More intricate modes are available, allowing to prevent MED or hot-potato routing oscillations. Indeed, recall from Section 2.3.2.2 that both kinds of oscillations occur because of information hiding. Hot-potato routing anomalies arise since routers may hide the preferred route of their peers because of local preferences, and MED-induced oscillation arose when a router was not aware of the best MED for a given destination.

MED-oscillations can be prevented using the *group-best* mode. Here, each router shares its best route for all neighboring ASes, ensuring that all routers have access to the route with the best MED. Thus, as shown in Fig. 3.2b, upon electing route n_1, b *still* advertises n_3 as both routes originate from different ASes. This effectively prevents the oscillation described previously as all routers now have access to the route with the best MED.

The *AS-wide* mode allow to prevent both MED and hot-potato oscillations. Advertising an AS-wide set consists in advertising all routes with the same (best) attributes up to (and



(A) Gadget as proposed by Griffin and Wilfong [Griffin & Wilfong 2002b] illustrated routing oscillation within iBGP induced by route reflection.



(B) MED-EVIL gadget extracted as proposed by Griffin and Wilfong [Griffin & Wilfong 2002a].

FIGURE 3.2: Reminder of the gadgets shown in Section 2.3.2.2

including, when routes originate from the same AS) the MED, *i.e.*, all routes towards a given destination sharing the same best β attributes (or, said otherwise, all AS-dominant routes) [4]. By doing so, no routers may deprive a neighbor of its best route because of purely local attributes *and* routes with the best MED attribute are known to all BGP speakers, preventing both MED and hot-potato iBGP oscillations and enabling optimal routing. For example, considering Fig 3.2a, routers will keep advertising the route of their client gateway n_x even once having elected $n_{x+1\%3}$ as their best route. Consequently, all routers are consistently aware of the BGP routes going through n_1, n_2 and n_3, preventing oscillations.

The AS-wide possesses an interesting property when considering internal events. As β-attributes are not related to the IGP, they remain stable upon IGP events. Thus, the best route after an IGP event lies within the routes with the best β-attributes pre-event. However, this does not imply that an AS-wide set necessarily contains the post-convergence route upon an internal event. For example, only a single route may possess the best β-attributes, and thus be alone within the AS-wide set.

The double AS-wide mode of Add-Path was thus proposed, which consists in sharing the AS-wide set, and the next best AS-wide set (*i.e.*, all routes with the best and second best β attributes.). Intuitively, this seems to provide enough information for each router to possess the post-convergence path, as stated in the RFC: *"the post-convergence paths will be known by each BGP node in an AS supporting this mode"*.

However, knowledge of the post-convergence path is not guaranteed in all cases. First, all routes within the double AS-wide set may be advertised by the same gateway, whose failure would render the entire set obsolete. This first point is however arguable, as it depends on the

[4] There are actually two definitions of AS-wide sets. Besides the one described here, the latter are also sometimes defined as all routes with the best β attributes *for each neighboring AS* [Van den Schrieck *et al.* 2010]. We here consider the definition given in the main body of this manuscript, as it is the one given by the Add-Path RFC which is more recent [Uttaro *et al.* 2016a].

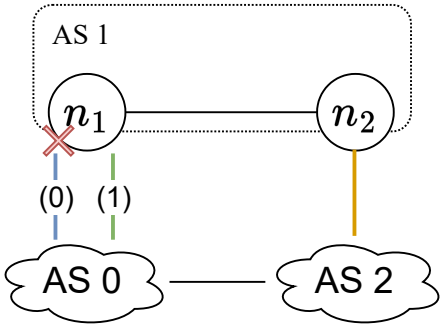


FIGURE 3.3: DOOM-MED gadget : an example of limitations of the double AS-wide set. Router n_2 only has known of the route using the blue entry point. Having a worse MED, the route using the green entry point is not advertised, following the definitions of AS-wide sets given in the RFC.

way Add-Path is actually implemented, and whether the *next-hop-self* option is configured [5]. Second, only the route with the best MED per neighboring AS is advertised. However, the best route after an internal may very well be the route with the second best MED.

Such a scenario is illustrated by the DOOM-MED gadget illustrated in Fig. 3.3. We consider routes towards AS 0. Following the AS-wide set definition (which should provide the best FRR capabilities), n_1 only advertises the route using the blue entry point to n_2. If the outgoing blue interface of n_1 fails, n_2 does not know the post-convergence route, *i.e.*, using the green entry point.

Consequently, the only Add-Path mode currently ensuring that routers know the post-convergence route is the Add-Path All mode which is not scalable enough.

3.1.3.3 Improving convergence time

Few works focus solely on directly improving BGP's convergence time or mitigating its effects. Increased route diversity has a positive impact, but it is, in a way, indirect. While fallback routes (or even, post-convergence ones) may be readily available, the actual decision process must still be run, and the FIB must still be updated for these routes to become active. Recall that the BGP decision process has to be run even after an internal event, due to hot-potato routing. Thus, increased visibility may not result in a sufficient improvement to the overall convergence time, given the frequency of such events.

Among the solutions aiming to improve the behavior of BGP upon internal events, one may cite the *next-hop tracking* feature, implemented on nearly all routers. Essentially, this allows the IGP to notify the BGP control plane upon an IGP event. Historically, BGP simply performed a periodic scan every 60s to check whether BGP routes had been impacted or not. Next-hop tracking thus also enables BGP to benefit from the improvement made to reduce the failure detection time within IGPs.

Besides this feature, the main solution directly tackling this issue is *Prefix Independent*

[5] Next-hop-self allows an ASBRs to advertise a route to a remote prefixe with itself as the next-hop (rather than the ASBR of the remote AS). If next-hop-self is set, several routes of an AS-wide set may originate from the same next-hop (*i.e.*, a local ASBR connected to multiple remote ASBRs). If it is not set, each route should, by default, originate from a different next-hop, as remote ASBRs should advertise only one route.

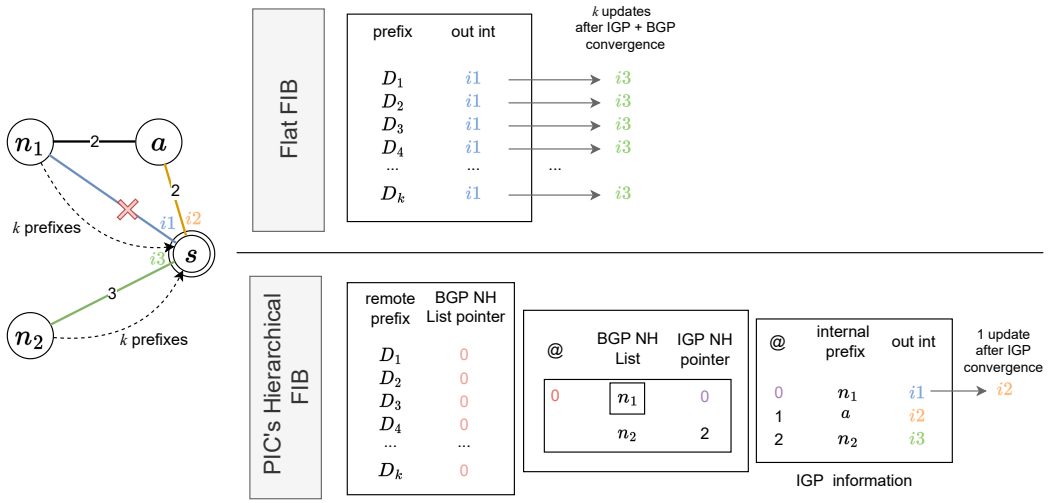


FIGURE 3.4: An example of a flat FIB architecture compared to a hierarchical one. We consider the point of view of s. Routers n_1 and n_2 advertise k prefixes to s through iBGP sessions. Unlabelled edges have an IGP cost of 1. We assume that n_1 is prefered to n_2 because of hot-potato routing.

Convergence (PIC) [Filsfils *et al.* 2011]. Despite its name, I believe that PIC is best described as an FRR mechanism. PIC builds upon increased route diversity (offered by the aforementioned solutions) to drastically reduce the traffic loss induced by BGP's convergence time. However, it does not guarantee that the new path taken by the traffic is the post-convergence one.

To mitigate the ill-effects of BGP's convergence, PIC relies mainly on (i) an improved, hierarchical FIB design and (ii) maintaining a set of BGP NH per prefix. The way PIC re-designs the FIB is shown in Fig. 3.4.

The historical FIB architecture, known as *flat* FIBs, drastically slowed the update time after an internal event. As described in Section 2.3.2.2, detailed NH information is lost once an entry is pushed within the FIB. More precisely, once a BGP route (and the associated BGP NH) is elected, the internal NH (IGP NH) and the associated outgoing interface are resolved through a recursive lookup. Solely the direct correspondence between the remote prefix and the outgoing interface is maintained.

If this correspondance changes, flat FIBs require updating each one of the entries, resulting in a high update time. Fig. 3.4 illustrated a standard flat FIB architecture. Router s is advertising the same k prefixes by n_1 and n_2 through iBGP sessions. For the sake of the argument, we assume that n_1 and n_2 advertised routes with the same β attribute, and that n_1 was thus elected as BGP NH due to hot-potato routing. The associated interface to reach these remote prefixes, $i1$, is pushed within the FIB. If the link between n_1 and s fails, BGP must re-converge, and all k entries must be updated one by one. During this process (*i.e.*, the BGP reconvergence *and* the update of the FIB), connectivity may be lost.

A *Hierarchical FIB* (HFIB) maintains the relation between BGP NH and IGP NH. Destination prefixes point towards their BGP NH, which themselves point towards their IGP NHs. Thus, modifying the IGP information, lying at the end of the relation chain, impacts all the prefixes pointing towards it. Consequently, if the IGP information used to

reach a BGP NH is modified (*e.g.*,, because a new path should be used due to an internal event), the update may benefit and reroute several (or even all) prefixes at once.

An example of an HFIB is shown in Fig. 3.4. Upon the failure of the link between s and n_1, n_1 becomes unreachable. The IGP will converge to find that the new best route to n_1, used as a gateway for all prefixes, is now through $i2$. This unique update, performed after the IGP, impacts all the k destinations that were pointing to this entry. Effectively, this quickly restores the connectivity to the n_1 gateway for all prefixes and prevents long-lasting connectivity. HFIBs have been implemented and deployed in actual hardware. Furthermore, Chang et al. proposed a way to provide an HFIB through the use of an additional SDN switch connected to the router [Chang *et al.* 2015].

An HFIB allows to drastically reduce the connectivity loss time upon an internal event. When relying on a flat FIB, the number of entries to consider impacts the time during which a connectivity loss occurs. When considering 350 000 entries, this time may reach 30s. Conversely, when relying on an HFIB, connectivity loss lasted 134ms regardless of the number of entries. These times have been confirmed by other studies [Holterbach 2021].

However, an HFIB by itself does not protect against the failure of the gateway itself. If there is no gateway towards which to restore the connectivity, one must rely on the BGP convergence to provide a new, available gateway and update the BGP NHs for all prefixes.

To better manage these events, PIC does not only maintain the best BGP NH to a given remote prefix, but the *two* best BGP NH [6]. These NHs are grouped together in a list. Prefixes sharing the same list of two BGP NHs point towards the same entry in memory. This is illustrated by Fig. 3.4, where all prefixes point towards a list composed of n_1 and n_2. Thus, if n_1 fails, the backup BGP NH, n_2, is immediately activated. All prefixes pointing towards the list benefit from this "grouped" update. Note that, when considering B border routers, there may be up to $\binom{B}{2}$ lists (*i.e.*, entries). Thus, updating each list may be far more efficient than updating all prefix entries individually.

The authors show that when relying on these mechanisms, the failure of a gateway only induces a connectivity loss of 80ms when 250 000 routes failed. Conversely, solely relying on the BGP convergence may lead to a connectivity loss lasting several minutes.

Nevertheless, PIC suffers from some limitations, both considering core and edge failures. First, only maintaining two gateways per prefixes may not be sufficient to protect against any failure. For example, if the network is not bi-connected, a single event may render both gateways unreachable. Thus, even if PIC possesses total route visibility (*e.g.*, through Add-Path All), it does not leverage the latter to its full extent. Second, the HFIB possesses the same drawbacks as most FRR schemes: the path through which the traffic is rerouted is not guaranteed to be optimal. Consider Fig. 3.4: while the HFIB restores connectivity quickly to n_1, the internal event modified the ranking of the routes, due to hot-potato routing. Consequently, the optimal gateway is now n_2. The traffic will benefit from the new optimal path only once the traditional BGP convergence has taken place. Furthermore, as traffic goes through a transient fallback path before settling to the post-convergence one,

[6] This is obtained by re-running the BGP decision process after having removed all routes advertised by the first elected BGP NH

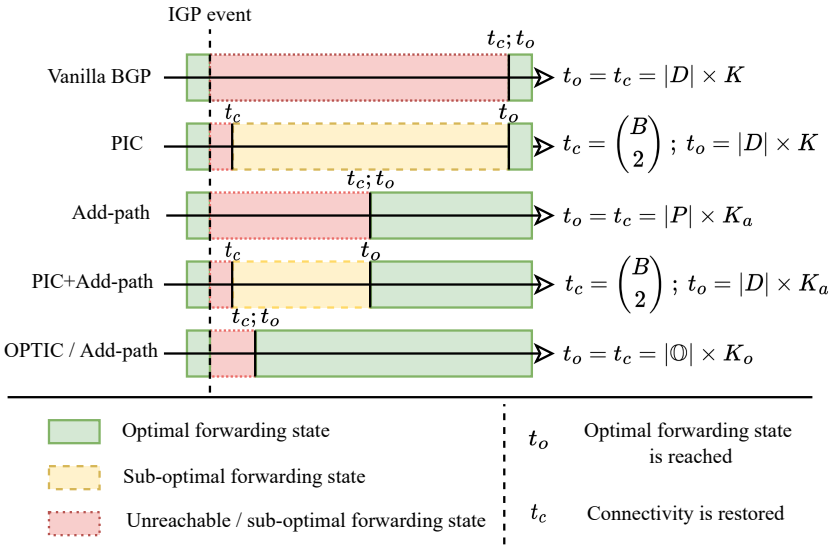


FIGURE 3.5: Connectivity and optimal forwarding state restoration timelines according to different technologies after internal events, depending on the number of prefixes $|D|$ and the number of BGP entries (K_o when using OPTIC and K_a when using Add-path).

the latter may suffer from de-sequencing.

While OPTIC tackles the same problem as PIC, we follow a different approach by *replacing* the traditional BGP convergence. Fig. 3.5 is a pedagogical illustration that does not provide a comprehensive comparison but shows typical cases to position OPTIC's objectives compared to current solutions.

Since BGP routers only exchange their best route towards a given prefix, finding the new optimal forwarding state with vanilla BGP often requires message exchange if the route becomes unusable. In any case, the router is required to perform a lexicographical comparison on all known routes (K) for each prefix (D).

PIC is designed to restore connectivity quickly by going through each of its sets of two gateways and falling back to the backup route of the set, or by benefiting from the IGP convergence through its hierarchical FIB. However, afterward, finding the new optimal gateway may still require message exchanges and a lexicographical comparison for all prefixes. In worst cases, the set of two gateways is not sufficient to protect the prefix (both gateways are unreachable after the event). In such cases, the connectivity cannot be restored immediately: t_c can be as long as t_o.

Add-path allows exchanging subsets of routes through iBGP. With the double AS-wide option in particular, the subsets of routes are likely to contain the new optimal gateway after any IGP event. Through adequate configuration, a BGP router can locally find the new optimal forwarding state by running the BGP decision process on the subset of routes sent through Add-path (K_a) for all prefixes $|D|$. This is however not fully guaranteed, depending on network connectivity and the effect of cold-potato routing (as explained with the DOOM-MED gadget). To ensure the protection of the prefixes upon any failure, all routes should be exchanged which scales poorly.

By combining PIC and Add-path, one can benefit from the enhanced connectivity restoration time of PIC and the advantages of Add-path. However, PIC and Add-path are not designed as a single entity and their union does not allow reaching the full potential of the available gateways [7]. While PIC can restore connectivity quickly by walking through its $\binom{B}{2}$ sets, the time taken to restore the optimal forwarding state is ultimately the same as the one of Add-path alone.

OPTIC is designed to fully harness the potential of increased iBGP route visibility. Through adequate data-structures, OPTIC efficiently pre-computes sets of gateways *guaranteed* to possess the optimal path whatever the network configuration and internal event, taking into account *both* hot and cold potato routing. OPTIC guarantees a fast switch to the new optimal path (thus, $t_c = t_o$) after a single walk-through of said pre-computed sets of gateways. The number ($|\mathbb{O}|$) and size (K_o) of these sets are both limited, as will be shown in our evaluation. Similarly to PIC, the sets may be shared in memory by several prefixes. Restoring connectivity optimally does thus not require working at the prefix granularity but at the set granularity instead. Once each set has been walked through, connectivity is restored optimally for all prefixes. In some degraded cases, the sets of gateways may need to be re-computed to handle any *future* IGP events (while the transit traffic already benefits from the new current optimal route). With OPTIC, this process does not rely on the slow lexicographical full BGP comparison anymore but rather on efficient updates of gateway structures.

Note while PIC is an FRR mechanism that does not guarantee the optimality of the new route the packet will follow, OPTIC goes one step beyond, and guarantees that the new route is the one that would be obtained by relying on the BGP convergence.

3.2 Segment Routing

With its ability to deploy specific paths and even actions to be taken along the latter, *Segment Routing* (SR) quickly gained a lot of traction. SR has been used for a variety of purposes. We have seen in the previous section that SR can be used within FRR schemes. It has also been used to improve network resiliency by steering and duplicating traffic over disjoint paths [Aubry *et al.* 2018] or even to monitor networks [Aubry *et al.* 2016a].

However, one of the main reasons SR is deployed is to perform TE [Adams 2020]. Indeed, because of the complexity of previous protocols, TE is performed by tuning the IGP costs [Filsfils *et al.* 2017, Balon & Leduc 2008]. The expressiveness offered by the IGP was thus sacrificed to the benefit of TE, which should have been ideally performed through other, more adapted protocols [8]. While specific TE needs and explicit routing may have been deployed through RSVP-TE, the protocol is quite cumbersome, and even sometimes abandoned [Filsfils *et al.* 2017]. SR thus re-invigorated the flame of TE through an operator-friendly, scalable and lightweight implementation of source-routing.

Seeing the potential of this technology, operators started to issue more intricate demands and requirements regarding TE capabilities [Filsfils 2019], such as the ability to deploy DCLC

[7] In addition, recall that Add-Path may not provide enough path diversity or remain scalable.

[8] In addition, it should be noted that using IGP costs to that effect also impacts BGP through hot-potato routing, although solutions exist to tune IGP cost for TE while considering its effects on BGP [Balon & Leduc 2008]

paths. This, coupled with the opportunities (and technical challenges) brought by SR, sparked the interest of the research community and the industry alike.

A very complete and interesting meta-analysis on SR-related research and standardization efforts has been done by Ventre et al. [Ventre *et al.* 2020]. Out of nearly a hundred SR-related contributions, using the source-routing capability of SR to perform TE is the most popular subject with 22 references (the second and third most popular subject being centralized control and resiliency with 16 and 9 references respectively).

Our contributions are no exception. Thus, in the following section, we will first detail several SR-TE contributions, and position our contributions among the related work.

We propose several contributions related to SR. First, a construct which efficiently encompass cost, delay, and number of segments necessary to encode paths: the SR Graph. Second, we propose two ways to utilize this construct. BEST2COP explores the SR Graph to compute DCLC path for SR, while LCA allows to use the information within the SR Graph to compute segment lists encoding paths on the fly.

We will thus first focus on SR contributions which compute TE paths, and position BEST2COP with respect to these contributions. Then we will focus on contributions tackling the path encoding problem, *i.e.*, the translation of paths to segment lists, and position the SR Graph and LCA with respect to these schemes.

3.2.1 Traffic-Engineering

SR has been widely used to perform TE with various objectives. An extensive survey has recently been written by Wu and Cui [Wu & Cui 2022]. Most of these works aim to perform bandwidth optimization, *i.e.*, steer numerous flows along paths that better utilize the network resources compared to shortest IGP paths. They usually rely on well-known optimization techniques extended to support SR. Note that all of the SR-related contributions mentioned here (including our own) consider a centralized computation element that possesses all required information.

Jadin et al. [Jadin *et al.* 2019] aim to minimize the *Maximum Link Utilization* (MLU) (a popular objective) using column generation, a linear programming technique. While most related works only allow up to 2 or 3 segments of specific types, their work considers an arbitrary number of segments and allows adjacency segments.

Perhaps surprisingly, Brundiers et al. [Brundiers *et al.* 2021] have shown that when aiming to minimize the MLU, allowing the creation and deployment of permanent forwarding loops with SR may allow achieving better solutions.

Gang et al. aim to maximize the overall throughput by re-directing flows on non-congested links, considering a partial deployment of SR [Gang *et al.* 2018].

Gay et al. focus on quick re-optimization of the network load after unexpected network events [Gay *et al.* 2017], as other SR-TE schemes may be too slow to react. Segment lists are constructed and improved iteratively through local search and guiding heuristics in less than a second.

Lee et al. [Lee & Sheu 2016] aim to increase the overall throughput through SR. They consider the betweenness of the nodes to predict the load of each link and translate this as an additive metric. Interestingly, and similar to our contribution, they use

BFM to compute forwarding paths. The author state that forwarding paths with fewer hops require fewer segments to encode, and thus use the I-HOP property to compute hop-constrained paths which then result in lower SR-related overhead. This statement is however not completely true, as the number of segments necessary to encode a path is related to its deviations from the shortest IGP paths and not necessarily to its number of hops.

Closer to the objectives of BEST2COP, Davoli et al. propose a TE scheme that assigns paths to flows according to bandwidth requirements, while trying to minimize the average latency through previously known techniques. A heuristic then re-assigns flows to further enhance the average delay. The computed paths are translated to segment lists and deployed [Davoli *et al.* 2015]. Similar to other works [Dugeon *et al.* 2017, Hou *et al.* 2018], while the delay is considered, the IGP cost is not considered simultaneously, resulting in vastly different problems and objectives compared to our contribution.

Hartert et al. [Hartert *et al.* 2015] propose DEFO, a flexible framework that computes segment lists for numerous varied objectives that may be expressed as high-level goals. For example, one may ask to route flows over paths under a certain delay while minimizing the MLU and using a given number of detours. The paths are then computed by using the middle-point routing model. This model considers paths as a sequence of so-called middle points, which are linked by partial forwarding graphs representing all shortest paths between two middle-points. Intuitively, a partial graph between two middle points u and v contains all paths encoded by the node segment $(Node, u, v)$. These Middle-points are then iteratively and incrementally set to values (nodes) to build paths that respect the constraints and optimize the objectives, thanks to well-known optimization frameworks (in particular, local-search and constraint programming). Solutions improve as the algorithm runs. Paths computations last during a period specified by the user (*e.g.*, 1 minute).

While DEFO proposes an interesting generic framework to perform TE for different general objectives in an operator-friendly fashion, the problem we aim to tackle (DCLC) is not expressible in their framework.

Similarly to our contribution, BEST2COP, all aforementioned SR-TE contributions aim to compute specific paths that enhance the routing performed through vanilla IGPs. However, they solve drastically different problems compared to the ones we focus on.

Indeed (and interestingly), most SR-TE contributions rarely consider explicitly the delay but rather focus on the bandwidth and load distribution when considering numerous flows. While the delay may be considered (at least as a secondary or indirect objective), the IGP cost (which may behave differently from the bandwidth) is rarely considered directly. The IGP is indeed often seen as enabling basic connectivity tasks. The associated costs are thus ignored when performing TE.

Our path computation algorithm, BEST2COP, aims at considering two additive metrics simultaneously. It can thus be used, amidst other possible use-cases, to compute SR-compliant paths respecting a constraint on the delay and minimizing the IGP cost (although any other pair of additive metrics may be considered). As we have seen in Section 2.2.1, solving this issue requires path computation algorithms tackling very different challenges than the schemes seen above.

This use-case is also practically relevant. The delay is indeed becoming more and more critical with modern applications. In addition, the IGP cost remains an important metric

within a network as it is one of the main tools an operator possesses to express general design choices. Operators have consequently explicitly expressed their desire for the ability to compute delay-constrained paths of minimal IGP cost with SR [Filsfils 2019]. To the best of our knowledge, no SR contribution currently aims at tackling our specific problem, *i.e.*, computing DCLC paths deployable with SR in an efficient manner.

Note that the DCLC paths we aim to compute are likely to be used by small premium flows which are far less numerous than their best effort counterparts, and which are less likely to affect the overall load distribution of the network. We thus consider a non-congested network onto which some premium flows must be routed through DCLC paths. Consequently, we do not require the traffic matrices of the network, nor do we consider the effect of numerous DCLC demands on said network.

This fundamental difference between other TE schemes also includes the evaluation method. Most of the solutions described previously are evaluated on fairly small networks, sometimes composed of less than 100 nodes. Their performance is rather examined with respect to the number of flows (or demands) to steer according to realistic traffic matrices. As explained, in our case, the number of demands is not as critical. Rather, we examine the performance of our contribution of large to very large scale networks of up to 100 000 nodes when considering multiple areas, and several thousand nodes when considering a flat OSPF deployment (*i.e.*, with a single area). We consider an arbitrary number of segments (up to MSD) and consider both main types of routing segments (node and adjacency).

3.2.2 Path Encoding

Most SR-related solutions require, at some point, a way to translate paths to segment lists. Some of them directly work with segments. For example, DEFO, presented above, directly constructs segment lists by concatenating ECMP DAGs through linear programming, which in practice may be encoded by node segments. Others first compute forwarding paths before translating them into segment lists.

Converting a path to segment lists (or, at least, computing the number of required segments) is challenging in itself, and requires specific algorithms. For reasons that we will detail later on, the number of segments may behave differently for any given path, and thus cannot be expressed as an additional weight on the original graph, bur rather requires specific schemes to be considered. As such, some works focus solely on this matter, known as the *path encoding* problem in the literature. While these solutions may differ in some aspects, they usually share a similar core concept. The path to encode, taken as input, is compared to the shortest IGP paths to find out the deviations (if any). These deviations are translated to either nodes or adjacency segments.

As we aim to compute SR-compliant DCLC forwarding paths (encodable in less than MSD segments), we also propose ways to translate paths to segment lists. We propose two different approaches. The SR graph is a graph transformation that transforms the number of segment into a standard metric: the hop-count. Standard multi-metric path computation algorithms may thus explore the SR graph to consider the segments as another standard constraint. Since directly exploring the SR graph may be costly, we propose LCA, which relies on the SR graph indirectly to translate multi-criterion forwarding paths as input into a minimal segment lists, following an encoding paradigm leveraging the presence of multiple paths with equal distances. LCA allows algorithms to explore the original graph of the

network rather than the SR graph but requires some modification to the underlying algorithm due to the peculiarities of this new metric.

In this subsection, we will review existing path encoding schemes and position both of our approaches.

Guedrez et al. propose an algorithm that translates a given path into a minimal segment list [Guedrez *et al.* 2016b], by first decomposing the given path into a series of shortest IGP subpaths through sequential Dijkstra's algorithms. A shortest subpath from u to v is encoded by a single node segment $(Node, u, v)$. A subpath composed of only two nodes (*i.e.*, an edge) is encoded by an adjacency segment (Adj, u, v). The behavior of the algorithm described to decompose the path into subpaths is however unclear when considering the effect of ECMP or when considering multi-graphs. In addition, note that some subpaths composed of two nodes u, v could be encoded by a node segment without issue, if the direct link (u, v) is the shortest path between u and v. Always using adjacency segments in this case may prevent the use of parallel ECMP links between two neighboring nodes [9]. Another algorithm is proposed which supposes the advertisement of adjacency segments with global meaning.

Davoli et al., whose contribution was already mentioned in the previous subsection, rely on a scheme to translate forwarding paths to SR list. Given a path p from s to d, the algorithm considers the set of ECMP paths from s to d. If only one shortest path exists which equals the path to encode, a single node segment is used. Otherwise, the algorithm considers the direct links (s, d) to check if the path may be encoded through a single adjacency segment. If this is not possible, the procedure is repeated but consider the predecessor v of d within p, before re-running the procedure considering the path from v to d.

Aubry discusses the path encoding problem through a greedy algorithm [Aubry 2020]. The algorithm proposed follows a given path p edge by edge on the shortest path DAG rooted at the source node s as far as possible. If an edge (u, v) is not within the current shortest path DAG, or if several shortest paths exist to v, a node segment $(Node, s, u)$ and, if needed, an adjacency segment (Adj, u, v) are added. Adding such segments allows to now continue to follow path p on the shortest path DAG rooted at u or v respectively. The resulting segment list is proven to be minimal by showing that any minimal segment list encoding p can be transformed to the output of their algorithm.

An illustration of the idea behind Aubry's proposition can be seen in Fig 3.6. The (arbitrary) path to encode, from Strasbourg to Bordeaux, is shown in the top figure. We first consider the shortest path DAG rooted at the source, Strasbourg. The first edge to encode, $(Str, L)_{i1}$ lies within the DAG and is thus included within the node segment $(Node, Str, L)$. However, to enforce this specific edge and prevent the traffic from using $(Str, L)_{i2}$, the adjacency segment $(Adj, Str, L, i1)$ must be used. Once this adjacency segment is interpreted, traffic is routed along the shortest path DAG rooted in Liège, which is now considered as the new source. The next edge, (L, StQ), does not appear on any shortest path DAG, and thus can only be encoded through an adjacency segment (Adj, L, StQ). The shortest path DAG rooted in Saint-Quentin is then considered. As all remaining edges to encode lie within this DAG, and ECMP does not enforce unwanted path diversity in this case, the remainder of the path can be encoded through a single node

[9] Although it should be noted that parallel links are likely to be aggregated and seen as a single link by SR.

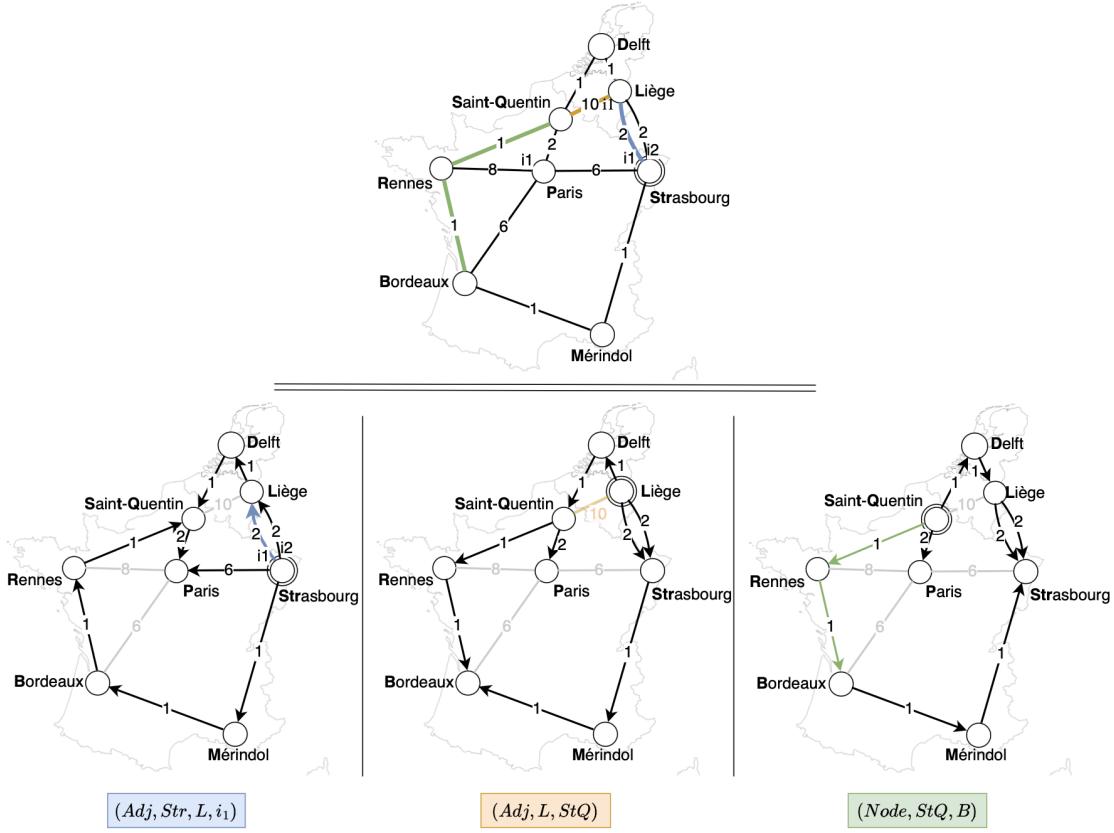


FIGURE 3.6: Illustration of the general principle used by path encoding algorithms. As long as edges remain on the shortest path DAG, a single node segment is required. An adjacency segment is required to encode a link that does not appear in any shortest path DAG, or to circumvent to avoid ECMP paths.

segment $(Node, StQ, B)$.

While the general design of Aubry's algorithm is close to ours, it only considers a single metric. LCA considers both costs and delays when translating paths. Considering two metrics induces further considerations and requires some modifications when computing the shortest path DAG used to guide the conversion.

In addition, the algorithms mentioned above perform what we refer to as *strict* encoding, meaning that the segment list encodes exactly and only the path provided as input. While this approach is valid and interesting, LCA performs a *looser* form of encoding. While the segment lists *contain* the input path, it may include other paths that share the same cost and delay. This method is not only more suited for the problem we aim to solve, but also enable further load-balacing. We prove that LCA returns the minimal segment list respecting such characteristics.

Furthermore, we consider that, for the number of segments to be taken into account properly, this conversion should be performed during the exploration of the paths (if possible and adequate). Indeed, this allows knowing if the paths are worthy of being explored, or if they violate the MSD constraint. Otherwise, if paths are translated to segment list only at the end of the execution, one may discover that no computed path is deployable. While the algorithms described above could be used to translate paths on the fly, they usually tend to consider the translation of end paths only. Conversely, we optimize our conversion scheme to compute segment lists on the fly (slightly updating them each time a new edge is considered), and more importantly, we discuss how this new metric should be taken into account when considering it during the exploration.

Indeed, correctly considering the number of segments transforms the problem into a multi-criterion one (*e.g.*, if one originally aimed at computing shortest [10] or multi-criterion paths). Thus, only MCOP algorithms are, by default, fitted to consider this additional metric.

Even so, the number of segments is a peculiar metric and requires revisiting the definition of dominance, which we also specify and detail. Indeed, most MCOP algorithms consider metrics that can be expressed as weights on an edge. However, this is not the case for the number of segments. Whether extending a distance by an edge (u, v) requires an additional segment depends on the *path* taken to reach u. Furthermore, the number of segments is non-strictly monotonous. While a path $p(s, u) \oplus (u, v)$ may not require fewer segments than $p(s, u)$, it may also not require more segments. For example, let us consider $d^1 = (5, 1, 1)$ and $d^2 = (5, 10, 10)$, with $(\#segments, delay, cost)$. Although d^2 seems dominated, we will show that it may end up requiring fewer segments than d^1. Thus, both paths should be explored.

We propose an implementation of LCA which can be performed *live* (*i.e.*, during the path exploration) for a very low computational cost, relying on information that is already available to the router. Furthermore, we formalize and detail the new conditions that should be followed in order to properly consider this new metric and its peculiarities.

We implement LCA and these modifications within an existing multi-criterion path computation algorithm (*Self-Adaptive Multiple Constraints Routing Algorithm* (SAMCRA)/*Tunable Accuracy Multiple Constraints Routing algorithm* (TAMCRA)), and show

[10] Indeed, one should maintain, per node, the non-dominated paths considering both cost and number of segments

that it remains competitive.

Lazzeri et al. propose a construct and scheme close to the SR graph [Lazzeri *et al.* 2015]. Given a multi-criterion path with both cost and delay, a new graph $G' = (V, E)$ is created, where V contains the nodes of the computed path. An edge (u, v) represents all ECMP paths $P^*(u, v)$ between u and v. The edges are characterized by three metrics: the IGP cost, the delay (set to the worst delay among all ECMP paths represented by said edge), and the number of underlying ECMP paths. Consequently, an edge (u, v) represents the paths and distances encoded by a node segment $(Node, u, v)$. The physical links between the nodes are then added with their respective metrics. An edge (u, v) representing a physical link may be encoded by an adjacency segment (Adj, u, v). The path computation is then re-run on this new graph, in which an edge is equal to a segment. Paths with more than MSD edges are not explored further.

The segment lists found are then discriminated according to their number of segments (the lower the better) and, during a second step, according to the number of underlying ECMP paths (the higher the better). An interesting feature of their computation method is that it may perform strict encoding, by only allowing edges/segments encoding a single path, or looser encoding by allowing any edge to be considered. They evaluated their solution regarding the number of segments required. The authors state that an APSP must first be computed, but do not discuss how the latter should be modified to encompass the latency and ECMP information required.

The structure proposed by Lazzeri et al. is close to our SR graph. However, it must be re-run *each time* a path must be translated, and the actual path computation is not discussed. Similarly to previous arguments, if the conversion needs to be performed during the exploration, using this scheme each time a distance is extended becomes costly.

As mentioned, our SR graph does not aim at translating one path. It is a graph transformation encompassing the whole network. When choosing to explore the SR graph directly, the number of segments necessary to encode a path is equal to the number of edges of said path. As the number of segments thus becomes a standard, strictly monotonous metric (the hop-count), any MCOP algorithm able to handle 3 metrics can explore this structure to compute DCLC paths for SR. However, the SR graph is a complete graph. Directly exploring the latter may be detrimental to most algorithms due to its high density. While we devise an algorithm able to leverage such structure to offer competitive performance, BEST2COP, we also show that using the SR graph indirectly through LCA may be preferable for some algorithms.

Furthermore, we detail how the prior APSP required should be augmented to contain the necessary multi-criteria information, and formally describe the resulting structure. We also remove unnecessary edges if they're dominated to reduce the size of the graph.

In this section, we reviewed the SR contribution of the existing literature close to our contribution BEST2COP. First, regarding the translation of forwarding paths to segment lists, solutions close to LCA do not consider multiple metrics and do not follow the same translation goals. Solutions similar to the SR graph consider multiple metrics, but are presented independently of the path computation scheme and are required to be re-run for each path to encode. Furthermore, they do not leverage the dominance property and do not

describe the required pre-computations. Second, we have seen that while SR is often used to perform TE, the usual goals differ from what BEST2COP solves.

While SR-TE contributions do not aim to solve DCLC, there exist numerous algorithm tackling this problem within the literature. These algorithms are often found in the field of operational research and thus do not consider the specificity of computer networks. However, even relatively recent algorithms which do consider computer networks often do not take into account the underlying deployment technologies and their constraints. Thus, most MCOP computation schemes is not found in SR-related literature.

Consequently, to better understand how BEST2COP relates to other MCOP computation schemes, we will review the corresponding literature in the following section.

3.3 Constrained Paths Computation

Constrained path computation is a popular and extensively studied research topic. The resulting problems and applications are indeed quite appealing. Being NP-Hard [11], designing efficient heuristics or approximation schemes for the problem offers both interesting theoretical and practical challenges. Moreover, their tendency to behave polynomially on realistic instances also motivated research efforts toward the design of exact methods efficient in practice.

As such, multi-constrained path computation has been studied by several research communities, either from a more theoretical standpoint (usually focused on exact schemes or approximation) or focused on practical applications (usually focused on exact schemes or heuristics). The resulting literature is consequently not only vast but also very interesting and comprised of numerous algorithms adopting sometimes drastically different approaches.

Again, I will not review the entirety of the literature as it is far too vast. However, several interesting surveys have been written over the years [Skriver 2000, Kuipers *et al.* 2002, Garroppo *et al.* 2010, Guck *et al.* 2018], showing the evolution of MCOP and MCP algorithms. In particular, the recent survey of Guck et al. not only describes several algorithms but also implements and evaluates the latter on different types of topologies. The results can be examined through an interactive webpage and offer insights into the behavior of the algorithms presented. While multi-constrained path computation is still an active subject, most solutions now build upon older results and algorithms, dating at least as far back as the seventies.

In this section, we will review the most well-known constrained path computation algorithms. Notice that several types of problems exists : DCLC, MCOP, MCP to name a few. While we will focus on solutions tackling MCOP and DCLC, we will also review some MCP solutions, as the latter propose similar and interesting concepts that may be leveraged to solve MCOP [12].

[11] We focus here strictly on additive metrics

[12] In fact, some solutions presented as MCP schemes may even be used as-is to solve MCOP

3.3.1 Heuristics

Because DCLC is NP-Hard [Wang & Crowcroft 1996], several polynomial-time heuristics have been designed, which aim to limit the worst-case computing time to the detriment of theoretical guarantees regarding the quality of the computed path. Such schemes usually focus on practical applications where instances may either be very large and/or where computing time is a critical element (*e.g.*, itinerary planning or routing). Note that in some cases, even if a feasible path exists (*i.e.*, a path respecting the constraint), heuristics may still fail to return it.

Before dwelling further into heuristics, let me first introduce some concepts used by most heuristics (and some other schemes as well). Several heuristics subsequently run Dijkstra's algorithm on a single metric prior to their execution. For DCLC, this may be used to compute the *Least-Delay* (LD) and *Least-Cost* (LC) paths, to check if a solution exists before running the algorithm : if the LD path is unfeasible no solution exists. If the LC path is feasible, it is the optimal solution.

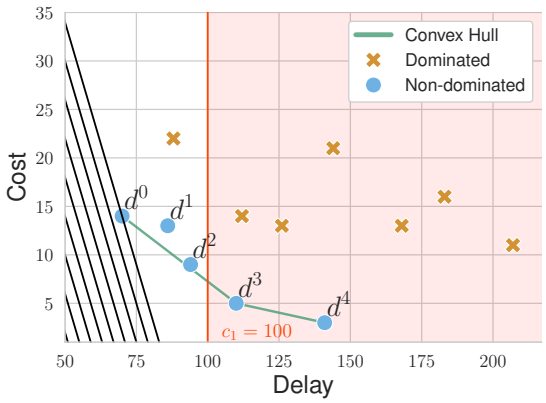
Furthermore, this knowledge is sometimes used to predict if the distance being explored will lead to constraint violations. More precisely, let us consider the delay $d_1(s,u)$ from source node s to node u. Let us consider $p = p_1^*(u,d)$ [13], *i.e.*, a LD path from u to the destination d. If $d_1(s,v) + d_1(p)$ violates the delay constraint c_1, there is no point in relaxing the outgoing edges of u. We will refer to such checks as *projected-distance tests*, or *projected-delay/cost tests* when considering a particular metric.

Some heuristics aim to compute paths in a distributed fashion. Most notably, *Delay Constrained Unicast Routing* (DCUR) [Reeves & Salama 2000], *Distributed Delay Constrained Routing* (DCR) [Zhou 1998] and the algorithm proposed by Ishida et al. [Ishida *et al.* 1998] aim to compute DCLC paths in a distributed environment. Aside from some subtleties, these algorithms rely on the same core concept: nodes either extend the LC or LD path depending on the projected-delay tests. Loops are corrected through backtracking. Sriram et al. [Sriram *et al.* 1998] proposed an enhancement by extending the set of paths a node can choose from, according to one or several heuristics. While this may increase the occurrences of loops, *Selection Function Based DCLC* (SF-DCLC) [Liu *et al.* 2005] improves upon this idea through a selection function proven to avoid loops and lead to a feasible solution.

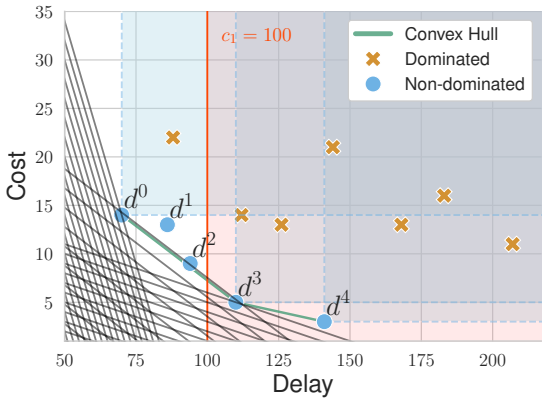
While computing DCLC paths in a distributed fashion is an interesting challenge, in this work we will consider that these paths are usually deployed through source-routing (*Segment Routing* (SR) in particular), which is naturally suited for a centralized computation of these paths.

Other heuristics follow the same principle, *i.e.*, alternating between LD and LC paths, but in a centralized fashion. Fallback routing [Lee *et al.* 1995], for example, simply consists in computing the shortest path for each given metric (e.g., the LD and LC path for DCLC): if one of them is feasible (*i.e.*, respects the constraint), it is returned; otherwise, nothing is returned. Although Fallback is very simple, it may lead to surprisingly good results on instances where costs and delay are positively correlated.

[13] As previously in this manuscript, we will denote d_1 the delay and d_2 the cost when dealing with these two metrics.



(A) Search direction induced when running a mono-metric SPA algorithm considering $w_1 + w_2$ as a metric.



(B) Distances lying within the triangular shaped part of the feasible space defined by two distances on the convex-hull (and the area they dominated) can not be found by running a SPA consider a linear composite metric.

FIGURE 3.7: Figures illustrating how linear composite metrics allow to search the solution space.

A similar approach is taken by *Dual Extended Bellman-Ford* (DEB) [Cheng & Ansari 2003]. DEB runs a shortest path algorithm on the delay metric to find a feasible path if any, and on the cost metric to try and find the optimal solution. It then returns the former or the latter, depending on the results. However, instead of relying on a standard shortest path algorithm, DEB uses a modification of BFM to compute the shortest paths for all possible numbers of hops, increasing the chance of finding feasible paths with a larger exploration.

Delay-Constrained Bellman-Ford (DCBF) [Jia & Varaiya 2006] starts by computing the shortest delay tree, to perform the projected-delay test later on. BFM is then used to compute the shortest paths, except that the relaxation procedure is modified: an edge is relaxed if it allows reaching a lower cost *and* if it passes the projected delay test. However, always greedily choosing paths of lower costs may result in settling on a non-optimal solution.

While some of these solutions also follow the BFM exploration scheme, they do not offer guarantees, do not consider the Pareto front, and overall greatly differ from the design of our own algorithm.

The algorithms presented until now consider the original cost or delay to guide the search. However, many heuristics prefer to rely on *aggregated* (or composite) weights. Jaffe's algorithm [Jaffe 1984] is a notorious example of such a scheme. The weight vector of each link is replaced by a scalar consisting in a linear combination of each weight's components (*i.e.*, $w(u,v) = \alpha_1 w_1 + \cdots + \alpha_m w_m$, α being coefficients). The shortest path considering this new metric, which now encompasses both cost and delay (or any other metric), is then computed.

While Jaffe's algorithm was originally designed for MCP and not MCOP (and, in addition, considered only two metrics), the idea of composite metrics is used throughout the literature and so worthy of discussion.

The way composite metrics operate can be visualized in the delay-cost space, already used

in Section 2.2.2, and represented again in Fig 5.5. In this space, the least-cost (resp. least-delay) distance can easily be deducted by "scanning" the space vertically (resp. horizontally). The first distance encountered is the least-cost (rest. least-delay) distance. These are the distances that would be found by running Dijkstra's algorithm or BFM on the corresponding metric. When considering a linearly aggregated metric, however, the scanning direction is neither vertical nor horizontal, but rather more or less oblique, depending on the coefficients (*i.e.*, on the importance attached to each component).

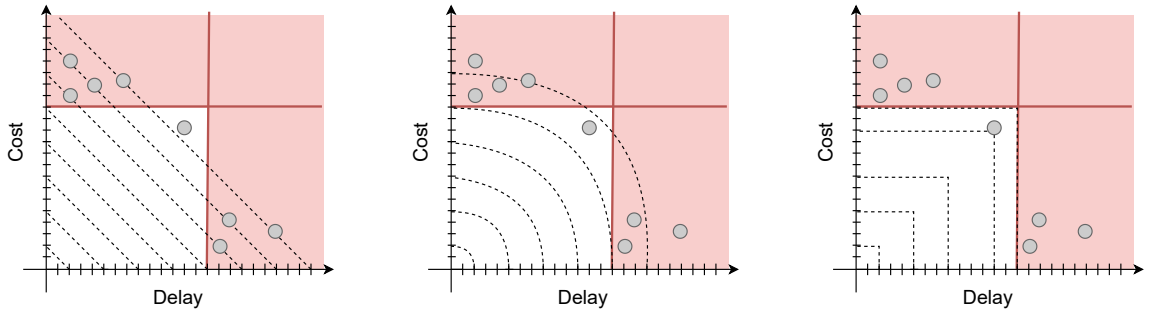
This concept is illustrated in Fig 3.7a. The black lines show the "scanning" direction when considering a linear composite metric in which both cost and delay are of equal importance, *i.e.*, $w = w_1 + w_2$. In this case, the lowest distance is d_0. Note that the lowest distance with respect to this composite metric is not necessarily the optimal solution (or even a feasible one). This significant loss of information is at the core of the performance of the associated schemes (which can thus fallback to using standard, efficient SPAs), but also one of their main limits.

Several techniques can be used to work around or mitigate this effect. One possible approach consists in only using composite metrics as a way to guide the search, but still consider metrics individually during edge relaxation. Another approach is to not stop at the shortest path, but pursue the exploration, *e.g.*, by computing k (or all) shortest paths with respect to the aggregated metric (*i.e.*, by letting the scanning line pursue its course until k paths have been found). Both of these approaches may be costly (as the number of paths may be large) and thus seems to be somewhat contradictory to the use of linear aggregated metrics. They are however sometimes used by exact schemes, which will be reviewed in the next section. Finally, one may also dynamically adapt the coefficients within the composite metric to scan the solution space at just the right angle.

The latter technique is used by *Lagrangian-based* approaches but does not allow capturing all paths within the Pareto front. In short, Lagrangian-based approaches aim to iteratively adjust the coefficient of the aggregated cost to orient the scanning line towards the optimal solution. When considering DCLC, this process is usually bootstrapped by considering the least-cost and least-delay distance as the points defining the new scanning line. These anchor points are then changed iteratively depending on the newly found distances through SPA runs. It can be proven that a finite number of SPA runs is sufficient [Juttner *et al.* 2001b].

Notice, however, that only paths within the convex-hull of the Pareto front (also called *supported* paths) may be obtained. For example, the distances d^1 and d^2 cannot be obtained by running a mono-metric SPA with a linear metric. Intuitively, any linear scanning line would first meet either d^0 or d^3, as shown in Fig. 3.7b. Thus, *unsupported* non-dominated paths, lying within the triangular-shaped feasible spaces defined by the supported path, cannot be obtained in this fashion. In this context of Lagrange relaxation, these paths are said to lie within the *duality gap* [Guck *et al.* 2018].

This procedure, often referred to as *LAgrange Relaxation-based Aggregated Cost* (LARAC), has been proposed multiple times in the literature by different authors [Aneja & Nair 1978, Handler & Zang 1980, Juttner *et al.* 2001b]. Interestingly, the first authors didn't realize that some paths may lie within the duality gap, and proposed the algorithm as an exact scheme. The basic idea behind the algorithm was enhanced by several propositions, *e.g.*, by performing k-shortest path computation at each iteration to speed up the search [Jia & Varaiya 2006], *closing the gap* to the optimal



(A) A linear composite metric is only able to scan at most half of the feasible space before considering unfeasible distances as well.

(B) A non-linear metric allows scanning more of the feasible space before starting to consider unfeasible distances.

(C) Non-linear metrics may even be able to scan the entirety of the feasible space before considering unfeasible distances.

FIGURE 3.8: Figures illustrating how non-linear distance allow to search the feasible space.

solution through an additional k-shortest path computation (using the final aggregated metric) [Handler & Zang 1980] [14] or improved stop conditions [Juttner *et al.* 2001b].

In addition to missing unsupported paths, working with linear composite metrics has another shortcoming: feasible and non-feasible distances may possess the same cost. Therefore, minimizing such metric not only offers no guarantee with regards to the optimality of the path but, depending on the metric, may also offer no guarantee regarding its feasibility. This can be seen in Fig 3.7b. Depending on the metric used, d_0 and d_3 may possess the same aggregated cost (they lie on the same line) even though d_3 is not feasible. Depending on the implementation of the underlying algorithm relying on this aggregated metric, d_3 may even be the optimal path returned.

To circumvent the limitations of purely linearly aggregated costs, some solutions instead rely on *non-linear* aggregated costs that allow to better search the feasible space. Several MCOP schemes use non-linear aggregated costs. However, the benefit of using such composite metrics is better explained when considering MCP. For MCP (with $m = 2$), the feasible space is defined by a square, as shown in Fig 3.8. Using linear composite metrics, half of the feasible space can be explored before the SPA may return unfeasible distances as well (at best). When relying on non-linear composite metrics, the scanning line can take non-linear shapes and better fit the solution space, as shown in Fig 3.8b and 3.8c, which allows the underlying algorithm to scan a more important part (or the entirety) of the feasible space before also considering unfeasible distances.

Note that finding the distance minimizing the non-linear metric used in Fig 3.8c solves MCP. Indeed, if the distance minimizing this metric is not feasible, then no solution exists. Otherwise, it is a solution to MCP.

However, minimizing such a metric can not be achieved through standard SPAs. Interestingly, when considering non-linear composite metrics, the subpath optimality property does not hold anymore. Thus, standard SPA cannot be used as-is to find the optimal path. Consequently, schemes relying on non-linear aggregated metrics must rely on heuristics (such as maintaining not only the optimal but several distances per node, in the hopes that the one ending up minimizing the non-linear metric is among them) or other ways to use

[14] Note that in this case, the proposed algorithm is an exact method.

such a metric.

Heuristic for Multi-Constrained Optimal Path (H_MCOP) [Korkmaz & Krunz 2001] offers an interesting way to use non-linear metrics. Depending on projected-distance tests, the algorithm either focuses on optimality (by minimizing the cost), or on feasibility (by trying to minimize a non-linear composite metric, which should lead to a feasible path). More precisely, the shortest paths from every node u to the destination d are first computed, according to a *linear* aggregated metric. These paths, which we will call *foreseen paths*, are used in the second step. In the second step, the graph is explored starting at the source. To choose whether an edge (u, v) should be relaxed, H_MCOP relies on the foreseen paths. If the foreseen paths going to the target t through v seem feasible (recall that since the foreseen paths are computed through a linearly aggregated metric, the feasibility is not guaranteed), H_MCOP chooses to focus on the optimization objective and chooses the path with the lowest cost. If no path seems feasible, the algorithm chooses the path which minimizes the composite metric instead. Recall that distances minimizing this metric are necessarily feasible, thus, this is a way to *re-orient* the exploration towards feasible distances [15].

H_MCOP has been slightly modified by Feng et al. to *Heuristic for Delay-Cost Constraint* (H_DCC) [Feng *et al.* 2002], which solves MCP for $m = 2$ specifically, considering a cost and a delay. H_DCC is used as a subroutine in *Non-Linear Relaxation for DCLC* (NR-DCLC), a heuristic aiming to solve DCLC. Basically, H_DCC is run iteratively, using the cost of the previously found path as the new cost constraint. The algorithm thus finds paths satisfying increasingly stringent cost constraints until the path of optimal cost is found. The optimality of NR-DCLC depends on the underlying MCP algorithm, which may not return a feasible path even if one exists.

Delay-Cost Constrained Routing (DCCR) [Guo & Matta 1999] converts the DCLC problem into MCP by setting a cost constraint equal to the cost of the LD path. The feasible space is then explored by trying to minimize a non-linear aggregated metric that prioritizes the cost of the paths (as the cost should be optimized). Since the distance minimizing such a non-linear metric cannot be found by solely extending the best distance at each node, DCCR maintains and extends up to k paths per node. Since this cost constraint may be too loose, the authors propose a variant, named *Search Space Reduction + DCCR* (SSR+DCCR), which instead uses the cost of the path returned by a Lagrangian-based scheme run prior as a bound to further reduce the search space.

Finally, TAMCRA [De Neve & Van Mieghem 2000] is an MCP scheme which may also be used to solve MCOP. TAMCRA is a Dijkstra-like algorithm that relies on a priority queue to explore the graph. TAMCRA orients the search according to a non-linear aggregated metric, referred to as LENGTH and defined as $\max_{0 \leq i \leq m} \left(\frac{w_i(p)}{c_i} \right)$, which, in short, represents how close a path is to violating the constraints. The scanning line resulting from such a function is the one shown in Fig 3.8c, which perfectly fits the feasible space. To try to find the distance minimizing this non-linear metric, TAMCRA maintains and extends up to k paths per node (recall that this does not guarantee to find the path minimizing the metric). However, to reduce the number of paths to extend without compromising the quality of the solution, TAMCRA only maintains non-dominated paths within said queue, and as such, is one of the few heuristics leveraging the concept of dominance. Finally, the metric used

[15] This way of exploring the graph is somewhat similar to the exploration method of SF-DCLC

by TAMCRA has another advantage. When extracting the minimum LENGTH from the queue, the associated distance is necessarily non-dominated, and may not be dominated by distances lying further down the queue. Thus, TAMCRA does not explore dominated distances, which improves its performance in practice [16].

The algorithms described offer interesting takes and approaches on the DCLC or MCOP problem. However (and conversely to our algorithm), they lack guarantees on the quality of the path returned. While it is true that the returned path may oftentimes be close (if not equal) to the optimal solution, this lack of concrete guarantees makes heuristics risky to deploy within computer networks, where strict and precise SLAs that were agreed upon by the client and the provider should be respected and enforced.

On the opposite, other solutions tackle the NP-Hardness of the multi-constrained path computation problem head-on, and aim to solve the problem *exactly*.

3.3.2 Exact Methods

A lot of exact methods to solve DCLC and MCOP have been designed, dated at least as far back as 1974. These schemes are often classified as either *ranking*, *two-phases*, or *labelling schemes*.

Out of these schemes, labeling ones follow the most traditional approach. The latter are usually designed by extending common monometric SPAs. Indeed, recall that usual SPAs cannot be used as-is to solve MCOP exactly. As mentioned in Section 2.2.2, extending a single distance per node is not sufficient to guarantee an exact solution to MCOP. Several distances (or *labels*), offering different compromises among metrics, must be extended.

To reduce the number of distances to consider, exact labeling methods usually heavily rely on the dominance relation. Indeed, as dominated distances are worse on all metrics than an other existing distance, finding dominated distances is irrelevant. However, a non-dominated path is necessarily composed of non-dominated subpaths [Hansen 1980]. Thus, during the exploration, dominated distances can be ignored without compromising the quality of the solution and the exactitude of the algorithm.

A standard multicriteria labeling scheme design (which our contribution, BEST2COP follows), is thus to extend a mono-criterion SPA by allowing the latter to maintain and extend all non-dominated distances per node.

This design raises challenges akin to the one faced by mono-metric SPAs, *i.e.*, finding the most efficient order into which these distances should be explored. Similarly to mono-criterion SPA, these algorithms can be classified as:

- label-setting: a distance being extended is necessarily non-dominated, and may not be dominated by a future distance.

- label-correcting: explored distances may be discovered to be dominated later on.

One of the oldest exact labeling schemes for solving DCLC was a label-correcting scheme

[16] We will see later on that other exploration orderings share the same property, which is often leveraged when designing exact schemes.

proposed by Vincke in 1974 [Vincke 1974][17]. Another label-correcting algorithm was proposed by Brumbaugh-Smith and Shier [Brumbaugh-Smith & Shier 1989]. The algorithm puts nodes within a queue (which may follow any given strategy, although several possibilities are investigated). At each iteration, a node u is selected. Its distances are extended by each edge (u, v) which results in new, potentially non-dominated paths to v. The currently known and newly discovered distances to v are merged into a new set of non-dominated distances. Brumbaugh-Smith and Shier discuss several ways to merge the current and new distances, as well as ways to choose which node to extract from the queue. Being a label-correcting algorithm, the latter methods that were investigated were simple schemes, such as *First In First Out* (FIFO) or *Last In First Out* (LIFO) orderings. Since the merging of two Pareto fronts may be fairly expensive, this algorithm was improved upon by Skriver and Andersen [Skriver & Andersen 2000], which essentially consists of additional checks allowing to reduce the number of distances to be merged by relying on previously computed information.

Corlay and Moon [Corley & Moon 1985] also propose a label-correcting algorithm that follows a different design approach by generalizing the BFM algorithm. At each iteration, the algorithm considers each node v. New distances to v are found by extending the currently known distances for all nodes u for which edges (u, v) exist. The new distances found to v are merged with the ones that were already known. From this set, the non-dominated distances are extracted to be considered at the next iterations. Following the description given by Corlay and Moon, the algorithm seems to follow the not-in-place BFM paradigm, meaning that non-dominated distances of k hops are found during the k^{th} iterations. Similarly to BFM, the algorithm can detect negative cycles. This algorithm is perhaps the scheme closest to BEST2COP. However, they do not consider deployment constraints or parallelization. The algorithm is purely theoretical and not implemented in practice. Furthermore, and conversely to BEST2COP, they do not leverage network characteristics (neither their structure nor the peculiarities of their metrics) to reduce the complexity of the algorithm. Finally, they do not fall within the approximation category.

These algorithms give rise to an interesting observation regarding the granularity at which multi-metric path computations should operate. When considering mono-metric SPCs, there is a single best distance per node. It thus seems natural to work at the node granularity, *i.e.*, adding and extracting nodes from the priority queue.

However, when dealing with multi-metric SPCs, there may be several non-dominated distances per node [18]. Some of them may be promising, and others uninteresting. Selecting nodes may thus lead to extending several uninteresting distances and impact the performance of the algorithm. Consequently, working at the distance granularity [19] is often preferred and more efficient [Guerriero & Musmanno 2001, Paixão & Santos 2007] (although recent experiments seem to show that the actual difference is more nuanced [Bökler & Mutzel 2017]).

Schemes working at the distance granularity are often not label-correcting schemes, but rather label-setting ones. Indeed, through an adequate exploration of the distances, one

[17] According to the literature, Vincke's algorithm extended BFM. However, the article is sadly unavailable online.

[18] Recall that we consider the distance of a path as the complete vector comprised of all the considered metrics. When considering DCLC, a distance is thus equal to a vector $(\text{delay}, \text{cost})$

[19] Recall that a distance implicitly corresponds to a path.

can ensure that extended distances are necessarily non-dominated. Thus, conversely to label-correcting schemes, label-setting schemes do not explore paths that may be rendered obsolete later on. However, this relies on more intricate ordering compared to label-correcting schemes. Consequently, no type of algorithm dominates the other, and both offer interesting performances depending on the problem instance [Guerriero & Musmanno 2001].

Label-setting algorithms usually rely on a distance-based *Priority Queue* (PQ) (*i.e.*, distances rather than nodes are stored within it) following a total ordering that reflects part of the dominance relationship (*i.e.*, a path on top of the PQ is necessarily non-dominated). Several such orderings exist, such as ordering the paths lexicographically, according to the sum of their weights, or by considering the lowest maximum weight component [Martins *et al.* 2007] (note that the TAMCRA heuristic followed a similar composite metric to the same effect).

In the literature, the first label-setting algorithm is usually attributed to Hansen [Hansen 1980] in 1980[20]. This algorithm was further improved by Martins [Martins 1984b]. The algorithm operates similarly to Dijkstra's but uses a distance-based PQs, in which distances are ordered lexicographically, leveraging the relationship between dominance and lexicographical order aforementioned. Among other contributions, this lexicographical ordering is also leveraged by Martins and Santos [Martins & Santos 1999]. Iori et al. follow a similar design, but rely on a linear aggregated sum to orient the search, and show that this approach may lead to better results than a lexicographical ordering [Iori *et al.* 2010].

Constrained Bellman-Ford (CBF) (cited as a private communication [Widyono 1994]) maintains an ordered list of distances for each node with increasing cost and decreasing delay. Since the i^{th} distance thus has a better delay but worse cost than the $i-1^{th}$ distance, distances within this list are all non-dominated. Distances are discovered by increasing delay thanks to a delay-based PQ. Since newly discovered distances necessarily have a worse delay than previous ones, CBF simply checks if it improves the current best cost known. Since CBF lists all the non-dominated paths to all nodes within the network, it may be used to solve DCLC for several destinations with independent constraints.

A* Prune [Liu & Ramakrishnan 2001] follows an approach similar to A* [E. Hart *et al.* 1968]. It assumes that a *guess function* is available for each metric. Dijkstra's algorithm is run on each metric m_i from the source node s to all nodes and from the destination d to all nodes. These pre-computations are used to perform projected-distance checks when exploring a path. Paths to explore are put within a PQ and ranked according to the sum of the weight components of their *projected* distances. Thanks to the aforementioned pre-computation, A* Prune prunes paths that lead to constraint violations.

Finally SAMCRA [Van Mieghem & Kuipers 2003] is the exact version of TAMCRA. SAMCRA relies on the same non-linear composite metric (referred to as LENGTH) to guide its search and stores all non-dominated distances within its PQ. As TAMCRA, the composite metric allows to perfectly match the dominated distances and naturally ignore the latter. Conversely to TAMCRA however, SAMCRA dynamically adjusts the number of distances to maintain at a given node to ensure that the algorithm remains exact. SAM-

[20] Although its algorithm does not seem to be label-setting in all scenarios. Hansen's algorithm focuses on DCLC and ranks distances within a PQ according to their cost. When considering distances of equal costs, the tie seems to be broken arbitrarily, which would not lead to a label-setting design.

CRA was improved upon to include a look-ahead procedure [VanMieghem & Kuipers 2004] (similarly to a projected cost/delay test, the remaining LD and LC path is considered when computing the LENGTH of the path) and to perform the path search in both direction [Kuipers & Van Mieghem 2003].

While labeling schemes are the closest to our contribution, two other schemes are quite interesting and worthy of mention, *ranking* and *two-phases* methods.

Ranking methods usually discover paths by increasing costs through a monometric k-shortest path subroutine until the entire Pareto front is discovered. Their performance thus heavily depends on the underlying k-shortest path algorithm used.

Climaco and Martins [Namorado Climaco & Queirós Vieira Martins 1982] rely on computing the k-shortest paths cost-wise, until reaching the cost of the LD path (essentially, finding the two extrema of the Pareto front and gradually filling in between). They rely on a deletion-based k-shortest path [21] proposed by Martins and published afterwards [Martins 1984a]. This approach may however discover a lot of dominated solutions during the exploration.

Martin et al. also proposed a ranking algorithm [Martins *et al.* 2007], which uses a deviation-based k-shortest path algorithm initially proposed by Martins et al. [De Queirós Vieira Martins *et al.* 1999]. To avoid exploring many dominated distances, this algorithm explores distances in lexicographical order.

The *two-phases* approach shares a few concepts with the Lagrangian-based heuristics. In the first phase, all paths lying on the convex-hull of the Pareto front, called *supported paths* are computed. These paths may be computed efficiently through the use of simplex-like algorithms [Mote *et al.* 1991], or, as explained in the previous section, through consecutive SPA runs following different linear composite metrics [Raith & Ehrgott 2009, Kergosien *et al.* 2022] [22]. In the second phase, the remaining non-dominated paths are discovered. Several ways to implement the second phase have been proposed, including labelling [Mote *et al.* 1991, Raith & Ehrgott 2009, Kergosien *et al.* 2022] (such as the ones proposed by Martins), branch-and-bound [Ulungu & Teghem 1995], or k-shortest path schemes [Handler & Zang 1980]. During the second phase, the search space is reduced to the feasible spaces that may still contain a non-dominated path, as they may be deducted from the results of the first phase. More precisely, these spaces are the triangular, non-dominated parts of the feasible space defined by the distances on the convex-hull, as shown in Fig 3.7b (*i.e.*, the duality gaps).

While the two-phases approach is generally seen as less competitive [Skriver & Andersen 2000], it is an interesting paradigm that may be fit for multi-threaded architectures. Indeed, both the first and second phases could be implemented in a parallel fashion. This was investigated fairly recently by Medrano and Church [Medrano & Church 2015].

[21] Once the best path is computed, a new graph is designed which contains all paths within the original graph, except the best path that was computed.

[22] These schemes usually build upon the results of Cohon et al. [Cohon *et al.* 1979] which essentially proposed the idea of approximating the Pareto-front by finding unsupported solutions through linear aggregated metrics. However, the context considered was not path computation, but water storage.

Being exact schemes, all algorithms mentioned in this section possess a high worst-case complexity. Thus, while they always find the optimal path to deploy (if it exists), they may suffer from pathological topologies or weighting schemes that generate an exponential number of non-dominated distances. Conversely, while BEST2COP is close to some exact schemes presented here (*e.g.*, Corlay and Moon), we leverage the nature and limitation of the metrics of computer networks to prevent pathological cases.

While high, the complexity of such exact schemes is often not exponential. Indeed, recall that the size $|\mathcal{P}|$ of the Pareto front, which such algorithms have to explore, may be bounded by the granularity of the metrics and the maximum values of the weights. Thus, their complexity is *pseudo-polynomial* in practice, *i.e.*, polynomial in the magnitude of the data involved (*e.g.*, the number of distinct distances a path may take). Being solvable exactly by pseudo-polynomial algorithm, MCOP is said to be weakly NP-Hard. This can be leveraged to design approximation schemes.

Approximation schemes do not guarantee finding the optimal path but can enforce strong guarantees regarding the paths they find. Thus, conversely to heuristics, the latter may still be used to define strict SLAs and, while benefiting from a reduced theoretical complexity.

3.3.3 Approximation Schemes

Approximation schemes often rely on a loose dominance relation. Such algorithms do not return a precise view of the Pareto front, but rather a coarse approximation of the latter. Usually, such algorithms only ensure that a path considered non-dominated is non-dominated by a factor of ε. More formally, they consider that a path p dominates a path q if $d_i(p) < (1+\varepsilon)d_i(q), \forall i \in \{1 \dots m\}$. In the following, we will denote an approximated Pareto front as $\mathcal{P}_\varepsilon$.

Although not all weakly NP-Hard problems admit FPTASes, it has been shown that an approximated Pareto front $\mathcal{P}_\varepsilon$ can be computed through FPTAS for problems such as DCLC and MCOP [Hansen 1980, Papadimitriou & Yannakakis 2000]. Several FPTASes have been designed over the years. In general, they rely on the same core techniques, which leverage the weakly NP-Hard nature of the considered problems.

As the complexity of MCOP depends on the size of the Pareto front, which itself depends on the granularity of the metrics (and their bounds), one may sacrifice the precision of the data until the size of the Pareto front is polynomially bounded. Running pseudo-polynomial algorithms on this modified data thus solves MCOP in polynomial time.

Several general techniques, proposed and classified by Sahni, can be used to reduce the magnitude of the data and arrive at approximation algorithms for MCOP when tuned properly [Sahni 1977]. The first one, called *scaling and rounding*, simply consists (in short) in rounding each weight to an integer value. If the weights are bounded, the number of non-dominated paths becomes polynomially bounded. The second one, called *interval partitioning*, consists in dividing the space of the metrics into intervals and only keeping one value per interval. In practice, interval partitioning behaves very similarly to scaling and rounding. The main difference between the two is that scaling and rounding modify the actual problem data by reducing its precision, while interval partitioning keeps the actual data, but fits it into bins during the execution. Finally, *separation* consists in only keeping

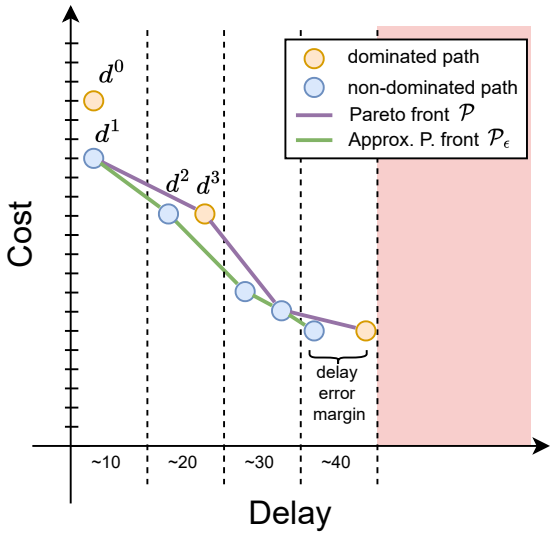


FIGURE 3.9: Reducing the precision of the delay only (while remaining exact on the cost) still allows to bound the number of non-dominated paths

distances that are *far enough apart*.

Note that in practice, not all metrics have to be transformed in such a fashion: only $m-1$ metrics are required to take bounded integer values for the problem to be solvable in polynomial time. This allows remaining exact for one metric of the considered problem. Fig. 3.9 shows this in the delay-cost space. Here, the delay has been discretized. Distances that fall within the same delay interval are considered to have the same delay. Thus, d^2 and d^3 are considered equal: they have the same delay and the same cost. It is thus possible that d^3, while being dominated, is considered non-dominated by the approximation algorithm. However, as the cost is not discretized, distances d^0 and d^1 can be discriminated correctly. Despite the cost being left unmodified, the maximum number of non-dominated distances in this approximated Pareto front $\mathcal{P}_\varepsilon$ is still bounded. At worst, there may exist a single non-dominated distance per delay interval. Consequently, only $m-1$ metrics have to be discretized to create polynomial time approximations.

Oftentimes, the optimization objective is approximated (*i.e.*, in our case, the cost), while the other metric is kept at its original granularity, to strictly enforce the constraint.

The techniques just described are at the core of most approximation algorithms, including our own. We will now describe the most well-known approximation algorithms, before positioning our work compared to the solutions described in this section.

Hansen noted that the exact label-setting scheme he described can be used to design an approximation scheme relying on scaling and rounding and consecutive runs of the original algorithm [Hansen 1980]. However, Warburton was the first to formalize the notion of approximated dominance, and to show that rounding could be used to solve MCOP in polynomial time [Warburton 1987]. One of the most well-known approximation schemes to solve DCLC is the one proposed by Hassin [Hassin 1992], which built upon Waburton's work. In short, Hassin's algorithm relies on the knowledge of an upper and lower bound on the optimal cost of the solution. These bounds are obtained through an algorithm that

refines these bounds through a test subroutine able to tell, for a given value x, if $d_2^* \geq x$ or if $d_2^* < x \times (1+\varepsilon)$. Once the two bounds are apart by a constant factor, the lower bound value obtained is used to scale down the costs in a fashion allowing for a pseudo-polynomial time algorithm run on said new costs to find an ε-approximation of the solution. This method was then improved, mainly through quicker ways to compute the upper and lower bounds [Lorenz & Raz 2001, Ergun *et al.* 2002]. Hassin also proposed an algorithm relying on interval partitioning, following an approach somewhat similar to BFM.

Chen and Nahrstedt proposed an algorithm to solve MCP that first scales down one metric to integers [Chen & Nahrstedt 1998]. The authors note that only the cost or the delay has to be scaled. The weights of the chosen metric, *e.g.*, the cost c, is changed to $\lceil c \times \frac{x}{c_2} \rceil$, c_2 being the cost constraint and x a chosen factor value. Note that this induces an error of $\frac{c_2}{x}$ per edge. They then solve MCP considering these scaled-down integer costs and x as the new constraint through an extended Dijkstra's algorithm or extended BFM. They prove that this is equivalent to solving the original MCP instance. However, depending on the value of x, the algorithm may not find a solution. It should be noted that their Dijkstra and BFM-like subroutine return a distance minimizing the first weight and respecting the $m-1$ remaining constraint, meaning that this scheme could be used to solve MCOP.

Goel et al. propose an approximation scheme to solve DCLC very similar to the one proposed by Chen and Nahrstedt [Goel *et al.* 2001]. However, the algorithm shares some core design choices with our contribution, BEST2COP. In particular, Goel et al. observe that it might be acceptable to violate the delay threshold by a constant fraction ε, and thus prefer the delay constraint instead of the costs. Second, the authors note that hop-by-hop routing may not be suitable to deploy DCLC paths (due to the loss of subpath optimality) and that source-routing is a more appropriate paradigm. However, they do not consider the technical constraints that ensue. The algorithm uses a standard dynamic programming algorithm to find distances by order of increasing delays, thus showcasing a complexity of $O(|E| \times c_1)$, c_1 being the delay constraint. Delays in the graph are scaled down very similarly to the Chen and Nahrstedt proposal, by multiplying them by x/c_1. Paths are then computed on the new graph, using x as an upper bound on the delay. The induced error of $\frac{c_1}{x}$ stacks at each edge, leading to an error of $H \times \frac{c_1}{x}$ at most, with H the diameter of the graph. Thus, a feasible path will have a real delay of at most $c_1 + H \times c_1/x = c_1(1+\frac{H}{x})$. If $x = \frac{H}{\varepsilon}$, then $c_1(1+H/H/\varepsilon) = c_1(1+\varepsilon)$. The algorithm thus yields a path whose cost is at most the shortest path cost, and whose delay is at most the constraint times $1+\varepsilon$ for all destinations. These guarantees are very close to one of our own solution, BEST2COP.

Yuan and Liu proposed the *Limited Granularity Heuristic* (LGH) and *Interval Partitioning Heuristic* (LPH) [Yuan & Liu 2001]. Both are based around an Extended Bellman-Ford Algorithm, which maintains the set of non-dominated distances to each node. LGH leverages the same concepts as the algorithms described previously by approximating $m-1$ metrics. Within each discretized interval, only the distance with the best remaining non-discretized weight is kept (similarly to what is shown in Fig 3.9). Non-dominated distances are extracted and extended. LPH bounds the number of non-dominated distances one may maintain, similarly to TAMCRA. Note that while LPH is often seen as similar to LGH, the former does not offer any guarantee on the quality of the solution, and is thus a heuristic. Its success hinges on the fact that the first non-dominated paths found are the most promising. As LPH is based on BFM, this may be likely, as paths of fewer edges are likely to be more interesting. Similarly to previous solutions, LPH and LGH can also be used to solve MCOP.

Song and Sahni proposed several algorithms based on interval partitioning and a multi-metric variant of BFM considering all non-dominated distances at each node [Song & Sahni 2006]. The first algorithm, *Internal Partitioning Heuristic* (IPH), works similarly to previous schemes, by dividing $m-1$ metrics into intervals. The second algorithm, *Generalized LPH* (GLPH), is a generalization of LPH. Simply put, while LPH limits the number of non-dominated paths to be maintained per node, GLPH imposes a global limit of non-dominated paths to maintain, which may be shared unevenly across all nodes. Finally, hybrid approaches are proposed (*Hybrid Interval Partitioning Heuristic* (HIPH)), which maintain up to k non-dominated paths per destination, and switch to interval partitioning if more than k paths need to be stored. Essentially, this design allows remaining exact on simple instances while ensuring a bounded complexity on more complex cases. The authors note that all the proposed heuristics (except GLPH) become ε-approximation schemes when the interval size is chosen correctly.

Finally, Tsaggouris and Zaroliagis propose a scheme that shares some design choices with BEST2COP [Tsaggouris & Zaroliagis 2009]. Their algorithm is a ε-approximation that relies on BFM and interval partitioning. Once again, $m-1$ metrics are approximated. At each iteration, each node v is considered, and new distances towards the latter are discovered by relaxing all existing edges (u, v). The intervals are set up so that weights close by a factor $(1+\varepsilon)^{|V|-1}$ fall within the same interval (only one distance is kept per interval, as usual), which, using their schemes, leads to a $1+\varepsilon$ approximation of the Pareto front. The contribution proposed by Tsaggouris and Zaroliagis remains mainly a theoretical result. It is thus not evaluated and does not consider any deployment constraint, be it the underlying technology or practical relevant features.

More recently, Hanusse et al. proposed a new algorithm to compute an approximated Pareto front $\mathcal{P}_\varepsilon$ [Hanusse *et al.* 2020]. Their algorithm assigns a *rank* to each distance, defined as the sum of their weight components. Recall that, following this definition, distances may not be dominated by distances with a higher rank. Distances are then explored by increasing rank. When several distances possess the same rank, only a subset is explored. If the subset is chosen properly, this algorithm becomes an ε-approximation. The authors then further improve their design to ensure that every path returned is non-dominated, *i.e.*, that $\mathcal{P}_\varepsilon \in \mathcal{P}$.

Interestingly, ε-approximations do not necessarily rely on the concept of dominance. Indeed, they often solely rely on the implicit reduction of the Pareto front brought by the discretization of the $m-1$ metrics and assume that the overhead implied by dominance checks is not profitable. Thus, in practice, such schemes may often extend and explore dominated distances. Finally, such schemes are often purely theoretical and may lack concrete evaluations.

3.3.4 Other approaches

Some other approaches may sometimes be used. Most notably, *implicit enumeration* aims at enumerating solutions (*e.g.*, following a breadth-first or recursive search approach) but pruning distances during the exploration when possible. Upon realizing that the LARAC-scheme they proposed was not exact Aneja et al. proposed an implicit enumeration scheme to solve DCLC, which this time was exact [Aneja *et al.* 1983]. More recently, an enumeration

scheme called *pulse* was proposed, also to solve DCLC [Duque *et al.* 2015]. Pulse explores the graph recursively. In short, the starting node sends a *pulse* through its outgoing edges. The distances discovered are pruned when possible. The remaining, non-pruned distances propagates through a new pulse until the end node is reached. Pulse is thus similar to a breadth-first search, coupled with aggressive pruning. To prune distances, Pulse removes distances containing cycles, distances exceeding the Nadir point [23], and distances that are already known to be dominated. Note that non-pruned distances may however still be discovered to be dominated at the end of the execution, due to the exploration order used. These approaches may thus fail to be competitive.

Finally, evolutionary algorithms offer an interesting framework to solve multiobjective problems [Coello & Veldhuizen 2007]. Some genetic algorithms have been designed to solve MCOP, *e.g.*, by combining different feasible paths to form the next generation of distances [Anh Quang *et al.* 2018], or by using ant-colony algorithms: ants, or probes, explore the graph pseudo-randomly according to a heuristic and retro-actively make efficient paths more attractive for the next generation of ants through the use of pheromones [Ghoseiri & Nadjari 2010]. Note that each probe could explore the graph in parallel. While interesting, the implementation complexity and unreliable nature of evolutionary algorithms may not be suitable for real-life deployments.

As we have seen so far, while many solutions exist, most possess certain drawbacks or lack certain features to reconcile both the practice and the theory. Heuristics do not always allow retrieving the existing paths enforcing strict SLAs, while exact solutions are not able to guarantee a reasonable maximum running time when difficult instances arise, although both features are essential for real-life deployment.

On the other hand, FPTASes can provide both strong guarantees and a polynomial execution time. However, they are often found in the field of operational research where, at best, possible networking applications and assumptions are discussed, but are not always thoroughly investigated. Because of this, the deployment of the computed paths, with SR and the MSD constraint in particular are not taken into consideration.

Our contribution, BEST2COP, is a labeling scheme aiming to close this gap by mixing the best existing features (such as providing both a limited execution time and strong guarantees in terms of precision in any cases) and adapting them for practical modern usage in IP networks deploying SR.

Table 3.1 summarizes some key features of a representative subset of the related work. Similarly to other FPTASes (for DCLC in particular), BEST2COP rounds one of the metrics of the graph (and leaves the other one untouched). However, conversely to most algorithms, BEST2COP does not sacrifice the accuracy of the cost metric, but rather of the measured delay. Because of the native inaccuracy of delay measurements (and the arbitrary nature of its constraint), this does not prevent BEST2COP from being *technically* exact in most practical cases. In addition, (and akin to HIPH), BEST2COP can easily be tuned to remain exact on all simple instances with a bounded Pareto front regardless of the accuracy of the metrics. Thus, BEST2COP can claim to return exact solutions in most scenarios and, at worst, ensure strict guarantees in others (for theoretical exponential instances). Finally,

[23] The Nadir point is defined as a theoretical distance composed of the cost of the LD path, and the delay of the LC path (the opposite of the *ideal* point). It is thus a very loose upper bound on the weights of efficient paths.

TABLE 3.1: Qualitative summary of a representative subset of DCLC-compatible algorithms showcasing their practicality, exactitude, and performance. In the *Practical Features* column, the green checkmark indicates whether the algorithm supports the corresponding feature (while the red cross denotes the opposite). In the *Exactitude vs Performance* column, the two subcolumns associated which each three scenarios show how the latter impact (*i*) the exactitude (exact, strong guarantees, no guarantees) and (*ii*) the performance of the algorithm (polynomial time or not). While the orange tilde denotes strong guarantees in terms of exactitude, green checkmarks (and red crosses respectively) either indicate exact results (no guarantees resp.) or polynomial-time execution (exponential at worst resp.) for performance. For both subcolumns *Bounded Pareto Front* and *Coarse Metric*, we consider the case where their spreading is polynomial with respect to the number of vertices in the input graph (and as such predictable in the design/calibration of the algorithm).

Algorithms	Practical Features			Exactitude vs Performance					
	Multi-Dest Single Run	SR Ready	Multi-thread Ready	Bounded Pareto Front		Coarse Metric		All Cases	
LARAC	×	×	×	×	✓	×	✓	×	✓
LPH	✓	×	×	✓	✓	∼	✓	×	✓
H_MCOP	×	×	×	×	✓	×	✓	×	✓
HIPH	✓	×	×	✓	✓	✓	✓	∼	✓
Hassin	×	×	×	∼	✓	✓	✓	∼	✓
Tsaggouris et al.	✓	×	×	∼	✓	✓	✓	∼	✓
Raith et al.	×	×	×	✓	✓	✓	✓	✓	×
A* Prune	×	×	×	✓	✓	✓	✓	✓	×
Corlay and Moon	✓	×	×	✓	✓	✓	✓	✓	×
SAMCRA	✓	×	×	✓	✓	✓	✓	✓	×
BEST2COP	✓	✓	✓	✓	✓	✓	✓	∼	✓

BEST2COP relies on the dominance property to reduce the number of paths to consider.

In addition, and conversely to several algorithms presented, BEST2COP exhibits several practical features. BEST2COP computes path (or rather, segment lists) to all destinations within the network. BEST2COP also leverages multi-threaded architectures, a feature rarely discussed, but increasingly important as such computations now tend to be performed by dedicated Path Computation Elements or even in the cloud. BEST2COP also leverages standard network architectures to deal with massive-scale networks of up to 100 000. Finally, BEST2COP was designed to efficiently consider the constraint brought by SR by encompassing the MSD constraint within thanks to the SR graph, despite the high density of the latter. Indeed, BEST2COP does not rely on our path encoding scheme LCA : rather, we designed BEST2COP to fully leverage the SR graph construct discussed in the previous section. Thanks to this graph transformation, BEST2COP can leverage the I-HOP property of BFM to take into account the number of segments *for free* (following a scheme close to the one proposed by Corlay and Moon). As a result, paths requiring more than MSD segments are naturally excluded from the exploration space by leveraging the I-HOP property [24].

[24] While we investigated using LCA on a bellman-ford like exploration schemes performed on the original graph (rather than the SR graph, the results were not as promising, are yet to be studied more extensively on different classes of graphs.)

3.3.5 Conclusion

In this chapter, we have reviewed the literature related to both OPTIC and BEST2COP.

OPTIC falls within the vast domain of FRR and improved convergence schemes. However, few solutions focus on the specific problem OPTIC tackles, *i.e.*, improving the convergence of BGP upon an internal event. OPTIC leverages the increased visibility brought by schemes such as Add-Path (which provides numerous other benefits) to allow routers to immediately fall back to the post-convergence route upon an event. Conversely, current solutions offer non-optimal fast reroute, and may not be able to deal with any internal event (even with increased visibility).

BEST2COP falls in the even vaster domain of TE and multi-constrained path computation. While numerous solutions exist, BEST2COP is, to the best of our knowledge, the first DCLC scheme to fully consider and leverage technical constraints and characteristics, both regarding path deployment, delay measurement, and topology design. Furthermore, few works discuss the benefit of parallelization. Finally, we test our algorithm on much larger topologies than usually considered in the literature and on various scenarios.

To consider the number of segments when computing paths, we propose two schemes, LCA and the SR graph. Other works in the literature propose similar approaches. However, they either (i) do not consider multiple metrics, (ii) do not follow the same conversion paradigm as ours (which leverages ECMP paths), or (iii) are not created to work hand-in-hand with the path computation algorithm, and may thus be inefficient when having to translate numerous paths. Finally, we also fit our conversion algorithm LCA on top of a MCOP algorithm, SAMCRA, and show that the latter remains competitive.

As both the background and context have been set, the following chapters of this thesis will describe my contributions. The next chapter will focus on BEST2COP, before moving on to describe OPTIC in Chapter 5.

Chapter 4

Deployable Multi-Constrained Tunnels

Contents

In this chapter, we describe in detail our contributions related to the deployment of multi-constrained paths. These contributions aim to bridge the gap between multi-constrained path computation algorithms, a heavily researched topic, and real-life deployment. Our solutions propose various ways to efficiently compute multi-constrained paths for SR domains, an important feature considering the strict latency requirements of emerging technologies.

To achieve our goal, we solve several challenges: (i) reducing the complexity of DCLC in a fashion suitable for operators networks, allowing to provide near-exact algorithms with bounded error margin and strong guarantees, (ii) efficiently considering the number of segments to take into account the packet manipulation overhead supported by routers, and (iii) scale with very large modern networks, despite the difficulty induced by considering three metrics (delay, cost, and number of segments).

Guarantees and practical concerns. While there exist several ways to solve DCLC, they usually do not consider the underlying deployment technologies and real-life deployment constraints, which often increases the complexity of the problem and limits the available data.

We keep such considerations at the core of our design. The nature of the paths we compute relies on a stable latency metric, the measured propagation delay (as recommended by [Giacalone *et al.* 2015, Ginsberg *et al.* 2019]), which is representative in our case dealing with few premium flows. Second, because of the inherent imprecision of these measurements and the arbitrary nature of the latency constraint, a small error margin regarding the delay of the computed path may be acceptable. We argue that the delay metric of the network may thus be discretized without loss of relevant information, or at least with a controlled loss of relevant information. As the number of non-dominated paths becomes bounded, exact DCLC/MCOP algorithms benefit from a reduced complexity while offering strong guarantees regarding the latency of the computed paths. Our structures and algorithms used to consider the number of segments may then be used in conjunction with this technique to efficiently solve DCLC in an SR domain.

Efficiently encompassing the Maximum Segment Depth constraint (MSD). An SR router can only prepend up to *MSD* routing instructions to a packet at line-rate, i.e., ≈ 10 with the best current hardware. Although we find that this limit does not prevent deploying most DCLC paths in practice, this constraint must still be taken into account. If ignored, the computed paths have no guarantees to be deployable, as they may exceed MSD.

To efficiently (and correctly) consider this new peculiar additive metric (the number of segments) when computing paths [1], we propose a new construct, the multi-metric SR Graph, which results from a transformation of the original graph, in which the number of segments necessary to encode a path is equal to the number of edges of the latter. As this construct encompasses all metrics, it can be relied upon to consider the number of segments when computing paths. We propose two ways to use this structure. First, as the SR Graph transforms the segment metric into an easily manageable metric (the hop count), it may be directly explored by any MCOP algorithm to solve DCLC in SR domain, by considering an

[1] Recall that one may not simply convert DCLC paths to segment list after their computation, as the computed paths may require more than MSD segment. As such, the number of segment required must be tracked while computing the paths.

additional constraint of MSD hops. Despite the high density of the SR Graph, we show that it is a viable option by designing an algorithm, BEST2COP, which explores the SR Graph and leverages its structure to natively remove non-deployable paths from the exploration space. BEST2COP offers competitive performances and has been designed for real-life deployment.

Second, for algorithms particularly sensitive to the density of the explored graph, we propose a way to keep exploring the original graph of the network but rely on the information within the SR Graph to perform on-the-fly conversions of paths to segment lists. This method, called LCA, does not only consist of an efficient multi-metric path encoding algorithm but also of a formally defined new kind of dominance relation which allows to correctly consider the segment metric and retrieve the 3D Pareto front.

Dealing with massive-scale networks efficiently. Current networks may be quite large, and growing. As such, modern computation schemes should maintain good performance even on large instances.

We thus extend BEST2COP to *BEST2COP-Extended* (BEST2COP-E), which leverages both multi-threaded architectures and the inherent structure of massive-scale networks (in particular, the logical and physical area-partitioning usually observed) to solve DCLC-SR in ≈ 1 second for $\approx 100\,000$ nodes. To evaluate our contributions, we create a topology generator, YARGG, able to construct massive-scale, multi-valuated, and multi-area topologies based on geographical data.

In the following, we will explain the motivation behind this work, before further detailing our solutions and discussing their performance. This work was published in *Network Computing and Applications* (NCA) [Luttringer *et al.* 2020b], in the French conference Algotel [Luttringer *et al.* 2021b] and extended in a journal version published in Computer Networks [Luttringer *et al.* 2022].

Note that while preliminary work on LCA was published in the Appendix of a prior publication, its in-depth description, formalization, and proofs are exclusive to this thesis.

4.1 Motivation & Context

Latency is critical in modern networks for various applications. The constraints on the delay are indeed increasingly stringent. For example, in financial networks, vast amounts of money depend on the ability to receive information in real-time. Likewise, technologies such as 5G slicing, in addition to requiring significant bandwidth availability, demand strong end-to-end delay guarantees depending on the service they aim to provide, e.g., less than 15ms for low latency applications such as motion control for industry 4.0, VR, or video games [Programme 2020]. For such interactive applications, the latency is as critical as the IGP cost.

As mentioned in Section 2.3, the IGP cost is defined as an additive metric that usually reflects both the link's bandwidth and the operator's load distribution choices on the topology. Paths within an IGP are computed by minimizing this cost. Thus, although delay constraints are increasingly important, they should not be enforced to the detriment of the IGP cost. With minimal IGP distances, the traffic benefits from high-bandwidth links and

follows the operator's intent in managing the network and its load. With bounded delays, the traffic can benefit from paths allowing for sufficient interactivity. It is thus relevant to minimize the IGP cost while enforcing an upper constraint on the latency. Computing such paths requires solving DCLC.

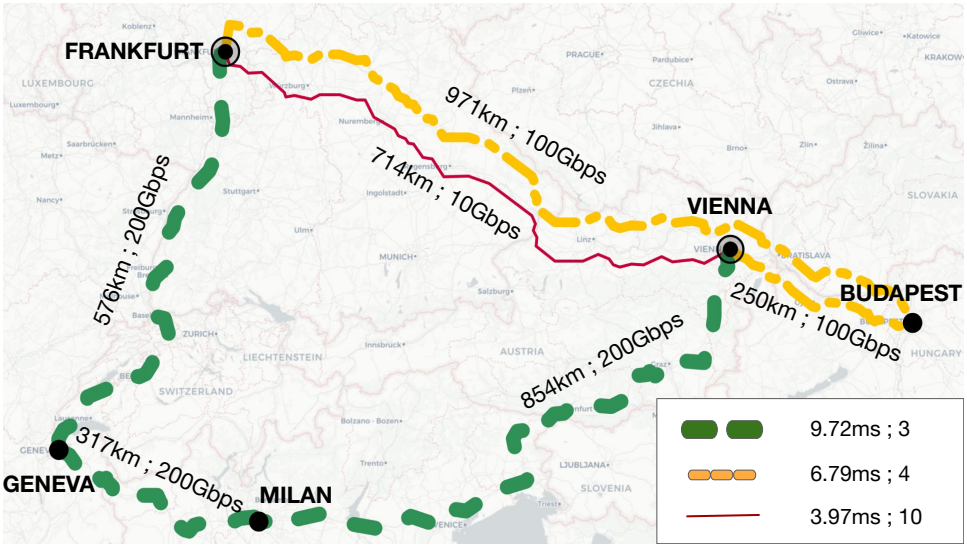


FIGURE 4.1: Practical relevance of DCLC in the GEANT network. IGP costs are deduced from the bandwidth of each link. Depending on their needs (in terms of delay and bandwidth), applications can opt for three non-comparable paths between Frankfurt and Vienna.

4.1.1 The DCLC problem in *Segment Routing* (SR) domains

In real networks, one may expect the two metrics to be strongly and positively correlated, and, as such, that computing the shortest path on any of the two metrics is sufficient. However, there are various cases where the delay and the IGP cost may be drastically different. For example, the IGP cost may have been tuned arbitrarily by the operator. Heterogeneous infrastructures between countries or geographical constraints may also create this effect. This can be illustrated on real networks, as displayed by Fig. 4.1. This map is a sample of the GEANT transit network [Nordic Gateway for Research and Education 2014]. As optic fibers often follow major roads, we rely on real road distances to infer the propagation delay of each link while the bandwidth, and so the estimated IGP cost, matches the indications provided by GEANT. A green link has an IGP cost of 1 while the IGP cost is 2 and 10 respectively for the yellow and pink ones.

Note that the two metrics are not correlated, hence all three distances shown between Frankfurt and Vienna offer diverse interesting options. In other words, all three distances are non-dominated and thus lie within the Pareto front of the distances between the two cities. Either solely the delay matters and the direct link (in pink) should be preferred, or the ISP prefers to favor high capacity links, and the green path, minimizing the IGP cost, should be used. The yellow path, however, offers an interesting compromise. Out of all paths offering a latency well-below 10ms, it is the one minimizing the IGP cost. Thus, it allows providing strict Service-Level Agreement ($<$ 9ms), while considering the

IGP cost. These kinds of paths, retrieved by solving DCLC, provide more options by enabling trade offs between the two most important networking metrics. Applications such as videoconferences, for example, can then benefit both from real-time interactive voice exchange (delay) and high video quality (bandwidth). In addition, IGP costs are also tuned to represent the operational costs. Any deviation from the shortest IGP paths thus results in additional costs for the operator. For all these reasons, there exist a clear interest for algorithms able to solve and deploy DCLC paths [Filsfils 2019]. However and so far, while this problem has received a lot of attention in the last decades from the network research community, no technologies were available for an efficient deployment of such paths.

Segment Routing (SR) is a vibrant technology gathering traction from router vendors, network operators and academic communities [Matsushima *et al.* 2022, Ventre *et al.* 2020]. Relying on a combination of strict and loose source routing, SR enables deviating the traffic from the shortest IGP paths through a selected set of nodes and/or links by prepending routing instructions to the packet itself. Such deviations may for example allow routing traffic through a path with lower latency. These deviations are encoded in the form of *segments* within the packet itself. To prevent any packet forwarding degradation, the number of deviations (*i.e.*, instructions) one can encode is limited to MSD, whose exact value depends on the hardware. While this technology is adequate to support a variety of services, operators mainly deploy SR in the hopes of performing fine-grained and ECMP-friendly tactical TE [Adams 2020], due to its reduced overhead compared to RSVP-TE [Filsfils *et al.* 2017]. Our discussion with network vendors further revealed a clear desire from operators to efficiently compute DCLC paths deployable with Segment Routing [Filsfils 2019]. Such a solution should thus not only encompass Segment Routing but also fare well on large-sized networks of several thousands of nodes, as already observable in current SR deployments [Matsushima *et al.* 2022].

Segment Routing implements source routing by prepending packets with a stack of up to MSD *segments*. In a nutshell, segments are checkpoints the packet has to go through. There are two main types of segments:

- **Node segments.** A node segment v indicates that the packet should (first) be forwarded to v with ECMP (instead of its final IP destination). Flows are then load-balanced among the best IGP next hops for destination v.

- **Adjacency segments.** Adjacency segments indicate that the packet should be forwarded through a specific interface and its link.

Once computed, the stack of segments encoding the desired path is added to the packet. Routers forward packets according to the topmost segment, which is removed from the stack when the packet reaches the associated intermediate destination.

Adjacency segments may be globally advertised, and thus be used the same way as node segments, or they may only have a local scope and, as such, can only be interpreted by the router possessing said interface. In this case, the packet should first be guided to the corresponding router, by prepending the associated node segment. In the following, as a worst operational case, we consider the latter scenario, as it always requires the highest number of segments.

As aforementioned, while operators seem to mainly deploy SR to perform fine-grained

TE, to the best of our knowledge, no DCLC variant exists for specifically tackling SR characteristics and constraints (except for our contribution). Using segments to steer particular flows allows however to deviate some TE traffic from the best IGP paths in order to achieve, for example, a lower latency (and by extension solve DCLC). A realistic example is shown in Fig. 4.1 where the node segment *Vienna*, as well as considering Vienna as the destination itself, would result in the packets following the best IGP path from Frankfurt to Vienna, *i.e.*, the green dashed path. To use the direct link instead (in plain pink) and so minimize the delay between the two nodes of this example, the associated adjacency segment would have to be used as it enforces a single link path having a smaller delay than the best IGP one (including here two intermediary routers). Finally, the yellow path, offering a non-dominated compromise between both metrics (and being the best option if considering a delay constraint of 8ms), requires the use of the node segment *Budapest* to force the traffic to deviate from its best IGP path in green. Before converting the paths to segment lists (and actually deploying them with SR), such non-dominated paths need first to be explored. Computing these paths while ensuring that the number of segments necessary to encode them remains under MSD is at least as difficult as solving the standard DCLC problem since an additional constraint now applies.

4.1.2 Problem statement : DCLC-SR and 2COP

In this section, we introduce and define preliminary notations and concepts, before describing the data structures used by our algorithms. We aim to solve DCLC in the context of an ISP deploying SR, leading to the DCLC-SR problem that considers the IGP cost, the propagation delay, and the number of segments.

For readability purposes, we denote:

- M_0 the metric referring to the number of segments, with the constraint $c_0 = \text{MSD}$ applied to it;
- M_1 the delay metric, with a constraint c_1;
- M_2 the IGP metric being optimized.

With these notations, we can now formally define the first problem we aim to solve, DCLC-SR.

> **Definition 4.1.1** (DCLC-SR)**.** *Given a source s, solving DCLC-SR consists in finding, for all destinations, a segment list verifying two constraints, c_0 and c_1, respectively on the number of segments (M_0) and the delay (M_1), while optimizing the IGP distance (M_2).*

On Fig. 4.1, we would have DCLC-SR$(\textit{Frankfurt}, 3, 8) \supset \textit{Frankfurt} - \textit{Budapest} - \textit{Vienna}$. This DCLC path (shown in yellow in Fig 4.1), is indeed the best option to reach Vienna when considering an arbitrary delay constraint of 8ms. Since the best IGP path from Frankfurt to Vienna (the green one) does not go through Budapest, encoding this DCLC path requires at least one detour, i.e. one segment (here, a node segment instructing the packet to go through Budapest first).

Solving DCLC-SR exactly requires, by definition, to visit the entirety of the Pareto front for all destinations. However, although only some of these paths are DCLC-SR solutions for a given delay constraint, all paths visited during this exploration may be of some practical interest. In particular, some of them solve problems similar to DCLC but with different optimization strategies and constraints. By simply memorizing the explored paths (*i.e.*, storing the whole Pareto front within an efficient structure), one can solve a collection of practically relevant problems. For instance, one may want to obtain a segment path that minimizes the delay, another the IGP cost, or the number of segments. Solving 2COP consists in finding, for all destinations, paths optimizing all three metrics independently, and respecting the given constraints (*i.e.*, returning the entire Pareto front) [2].

We formalize this collection of problems as 2COP. Solving 2COP enables more versatility in terms of optimization strategies and handles heterogeneous constraints for different destinations. Simply put, while DCLC-SR is a one-to-many DCLC variant taking MSD into account, 2COP is more general as it includes all optimization variants.

With initial constraints c_0, c_1, c_2, *Best Exact Segment Track for 2-Constrained Optimal Paths* (BEST2COP) solves 2COP, *i.e.*, returns in a single run paths that satisfy smaller constraints c'_0, c'_1, c'_2 for any $c'_i < c_i$, $i = 0, 1, 2$, offering more flexibility than simply returning the DCLC-SR solution. More precisely, the problem can be formulated as follows.

Definition 4.1.2 (*2-Constrained Optimal Paths* (2COP))**.** *Let $f(M_j, c_0, c_1, c_2, s, d)$ be a function that returns a feasible segment path from s to d (if it exists), verifying all constraints $c_i, 0 \leq i \leq 2$ and optimizing $M_j, j \in \{0, 1, 2\}$. For a given source s and given upper constraints c_0, c_1, c_2, we have*

$$2COP(s, c_0, c_1, c_2) = \bigcup_{\substack{\forall d \in V, \\ \forall j \in \{0,1,2\}, \\ \forall c'_j \leq c_j}} f(M_j, c'_0, c'_1, c'_2, s, d)$$

Solving 2COP and DCLC-SR requires returning (or, at least, visit) the entire Pareto front. As we have discussed in Section 2.2.2, this problem is weakly NP-Hard due to the size of the Pareto front. The added metric, the number of segments, further increases this complexity. In the following section, we discuss how we deal with this complexity to solve these problems efficiently.

4.2 Dealing with 2COP and DCLC-SR's complexity

In order to solve the problems described in the previous section efficiently, we rely on two main concepts. First (and similarly to the FPTASes presented in Section 3.3), we sacrifice the precision of one metric in order to bound the size of the Pareto front in a predictable

[2] The problem consisting in returning the entirety of the Pareto front is sometimes referred to as the *Bicriterion Shortest Paths* (BSP) problem in the literature. Thus, 2COP can be seen as an SR-variant of BSP

fashion. Second, we proposed schemes to efficiently consider the number of segments when computing paths, despite the peculiarities of the metric.

4.2.1 DCLC and true measured delays

DCLC is weakly NP-Hard and can be solved exactly in pseudo-polynomial time. In other words, as long as either the cost or the delay possesses only a limited number of distinct values (*i.e.*, only a limited number of distinct distances exist), the Pareto front of the paths' distances is naturally bounded in size as well, making DCLC tractable and efficiently solvable [3]. Such a metric thus has to be bounded and possess a coarse accuracy (i.e., be discrete). Although this has little impact when solving DCLC in a theoretical context, it can be strongly leveraged to solve DCLC efficiently thanks to the characteristics of real ISP networks.

4.2.1.1 Discretizing the delay

We argue that the metrics of real ISP networks do indeed possess a limited number of distinct values. Intuitively, the IGP costs already bound the size of the Pareto front. Indeed, such costs are integers that do not exceed 2^{16}, leading to a maximum of $|V| \times 2^{16}$ distinct distances (if there are no loops). However, this bound remains quite loose. Furthermore, further discretizing the IGP costs is not adequate when considering computing networks. Indeed, while some operators may rely on few spaced weights, others may possess intricate weight systems where small differences in weight may have a large impact. Thus, even a small error induced on the IGP costs may result in paths vastly different from the ones wished by the operators.

Rather, we argue that the *delay* can be leveraged to further bound the size of the Pareto front in a more predictable, tunable, and suitable fashion. The delay (i) is likely strongly bounded by the problem itself, and (ii) can be handled as if having a coarse accuracy in practice. Indeed, for TE paths, the delay constraint is likely to be very strict (10ms or less). Second, while the delay of a path is generally represented by a precise number in memory, the actual accuracy, *i.e.*, the *trueness* t of the measured delay is much coarser due to technical challenges [Almes *et al.* 2016, Almes *et al.* 1999]. In addition, delay constraints are usually formulated at the millisecond granularity with a tolerance margin, meaning that some loss of information is acceptable.

Thus, we argue that the loss of information resulting from sacrificing the precision of the delay metric does not prevent the algorithm from being *technically exact*, as the information lost is either not representative of the actual delay (due to the inaccuracy of the measurement) or irrelevant (due to the arbitrary nature of the constraint).

Consequently, floating numbers representing the delays can be truncated to integers, *e.g.*,, taking 0.1ms as unit. This allows to easily bound the number of possible non-dominated distances to $c_1 \times \gamma$, with γ being the desired level of accuracy of M_1 (the inverse of the unit of the delay grain, here 0.1ms). For example, with $c_1 = 100$ms and a delay grain of 0.1ms ($\gamma = \frac{1}{0.1} = 10$), we have only 1000 distinct (truncated) non-dominated pairs of distances to track at worst. This leads to a predictable and bounded Pareto front. One can then store

[3] Metric M_0 is omitted for now as this trivial distance is only required for SR and discussed in details later. While dealing with a three-dimensional Pareto front seems more complex at first glance, we will show that SR eventually reduces the exploration space because its operational constraint is very tight in practice and easy to handle efficiently.

non-dominated distances within a static array, indexed on the M_1-distance (as there can only be one non-dominated couple of distances (M_1, M_2) for a given M_1-distance).

The variable Γ denotes the size allocated in memory for this Pareto front array (i.e., $\Gamma = c_1 \times \gamma$). When t, *i.e.*, the real level of accuracy, is lower (or equal) than γ, the stored delay can be considered to be exact. More precisely, it is discretized but with no loss of relevant information. When t is too high, one can choose γ such that $\gamma < t$, to keep Γ at a manageable value. In this case, some relevant information can be lost, as the discretization is too coarse. While this sacrifices the exactitude of the solution (to the advantage of computation time), our algorithm is still able to provide predictable guarantees in such cases (*i.e.*, a bounded error margin on the delay constraint).

4.2.1.2 A note on the considered delay

Referring to a path's delay may be ambiguous. Indeed, this characteristic is not monolithic. The total delay is mainly composed of the propagation delay and the queuing delay. Both delays may play an important part in the overall latency, though none can be stated to be the main factor [Savage *et al.* 1999]. Although the propagation delay is stable, the queuing delay may vary depending on the traffic load. However, in order to compute TE paths, the delay metric must be advertised (usually within the IGP itself). For this reason, it is strongly recommended to use a stable estimate of the delay, as varying delay estimations may lead to frequent re-computations, control-plane message exchanges, and fluctuating traffic distribution [Giacalone *et al.* 2015, Ginsberg *et al.* 2019].

For this reason, we use the propagation delay, as recommended in [Giacalone *et al.* 2015, Ginsberg *et al.* 2019]. The latter is usually measured through the use of a priority queue, ignoring so the queuing delay. Its value is deduced as a minimum from a sampling window, increasing so its stability [Cisco Networks 2020]. Using this delay not only makes our solution practical (as we rely on existing measurements and respect protocol-related constraints), but is actually pertinent in our case. In practice, flows benefitting from DCLC paths benefit from a queue with high priority and experience negligible queueing delays. In addition, the amount of traffic generated by such premium interactive flows can be controlled to remain small enough if it is not limited by design. Consequently, not only is there no competition between premium flows and best-effort traffic, but these flows do not generate enough traffic to lead to significant competition between themselves. Thus, the experienced delay is actually agnostic of the traffic load for our use case, making the propagation delay a relevant estimate. Consequently, we use the discretized propagation delay, enabling both practical deployment and the limitation of the number of non-dominated distances, within our structure used to encompass Segment Routing natively, the SR graph.

4.2.2 A structure to efficiently consider the number of segments

To solve DCLC-SR efficiently, as well as its comprehensive generalization, 2COP, we rely on a specific construct used to encompass SR, the delay, and the IGP cost: the *multi-metric SR graph*.

This construct represents the segments as edges to natively deal with the M_0 metric and its constraint, $c_0 = \text{MSD}$. The valuation of each edge depends on the distance of the path encoded by each segment. While the weights of an adjacency segment are the weights of its associated local link, the weights of a node segment are the distances of the ECMP paths

it encodes: the (equal) IGP cost (*i.e.*, M2-distance), and the lowest guaranteed delay (*i.e.*, the M1-distance), *i.e.*, the worst delay among all ECMP paths. Hence, computing paths on the SR graph is equivalent to combining stacks of segments (and the physical paths they encode), as stacks requiring x segments are represented as paths of x edges in the SR graph (agnostically to its actual length in the raw graph). The SR graph can be built for all sources and destinations thanks to an *All Pair Shortest Path* (APSP) algorithm. Note that this transformation is inherent to SR and leads to a complexity of $O(|V|(|V|\log(|V|)+|E|))$, for a raw graph having $|V|$ nodes and $|E|$ edges, with the best-known algorithms and data structures. This transformation (or rather, the underlying APSP computation) being required for any network deploying TE with SR (the complexity added by our multi-metric extension being negligible), we do not consider it as part of our algorithm presented later.

This transformation is shown in Fig. 4.2b, which shows the SR counterpart of the raw graph provided in Fig. 4.1 (which is shown again in Fig. 4.2a for readability purposes). To describe this transformation more formally, we use the notations introduced in Section 2.1. We denote $G = (V, E)$ the original graph, where V and E respectively refer to the set of vertices and edges. As G can have multiple parallel links between a pair of nodes (u, v), we use $E(u, v)$ to denote all the direct links between nodes u and v. Each link (u, v) possesses two weights, its delay $w_1^G(u, v)$ and its IGP cost $w_2^G(u, v)$. The delay and the IGP cost being additive metrics, the M_1 and M_2 distances of a path p (denoted $d_1^G(p)$ and $d_2^G(p)$ respectively) are the sums of the weights of its edges.

From G, we create a transformed multigraph, the SR graph denoted $G' = (V, E')$. While the set of nodes in G' is the same as in G, the set of edges differs because E' encodes segments as edges representing either adjacency or node segments encoding respectively local physical link or sets of best IGP paths (with ECMP). The M_i-weight of an edge in G' is denoted $w_i^{G'}(u, v)$. Note that if G is connected, then G' is a complete graph thanks to node segments.

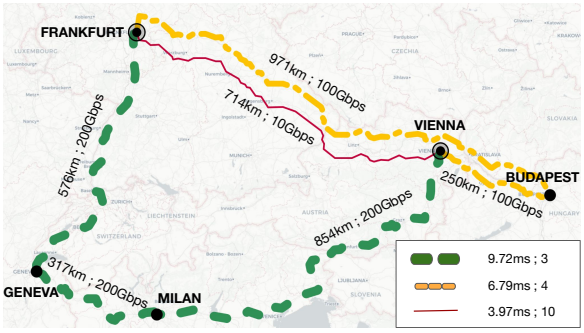
A node segment $(Node, u, v)$, encoding the whole set $P_2^{G*}(u, v)$ of ECMP best paths between two nodes u and v, is represented by exactly one edge in $E'(u, v)$. The M_2-weight $w_2^{G'}(u, v)$ of a node segment is the (equal) M_2-distance of $P_2^{G*}(u, v)$, *i.e.*, $d_2^*(u, v)$. Since, when using a node segment, packets may follow any of the ECMP paths, we can only guarantee that the delay of the path will not exceed the maximal delay out of all ECMP paths. Consequently, its M_1-weight $w_1^{G'}(u, v)$ is defined as the maximum M_1-distance among all the paths in $P_2^{G*}(u, v)$, *i.e.*, $d_1^{max}(P_2^{G*}(u, v))$ [4]. Links representing node segments in G' thus verify the following:

$$w_1^{G'}(u, v) = d_1^{max}(P_2^{G*}(u, v))$$
$$w_2^{G'}(u, v) = d_2^*(u, v)$$

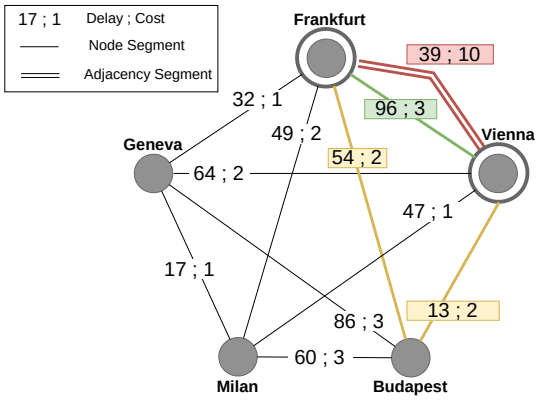
An adjacency segment (Adj, u, v, x) corresponds to a link in the graph G and is represented by an edge $(u, v)_x$ in $E'(u, v)$, whose weights are the ones of its corresponding link in G, only if it is not dominated by the node segment $(Node, u, v)$, *i.e.*, if $w_1^{G'}(u, v) > w_1^G((u, v)_x)$, or by any other non-dominated adjacency segments $(u, v)_y$, *i.e.*, if $w_1^G((u, v)_y) > w_1^G((u, v)_x)$ or $w_2^G((u, v)_y) > w_2^G((u, v)_x)$, where $(u, v)_x$ and $(u, v)_y$ are two different outgoing links of u in $E(u, v)$[5].

[4] In practice, this can be computed by first computing the shortest path DAG from u to v, and then finding the highest delay within this DAG, which can be done in polynomial time.

[5] If two links have exactly the same weights, we only add one adjacency segment in G'



(A) Extract of the Geant network.



(B) This figure shows the network from Fig. 4.2a translated into an SR Graph. The SR Graph encodes segments as edges. Plain edges represent node segments, *i.e.*, sets of ECMP paths. Double lines are adjacency segments, here only $(Frankfurt, Vienna)$, and are visible only if they are not dominated by other segments. Colored edges refer to the paths highlighted in Fig 4.2a.

FIGURE 4.2: Figures illustrating the translation of a network into an SR Graph.

We can now more precisely describe Fig. 4.2b, which illustrates the result of such a transformation: one can easily identify the three non-dominated paths between Frankfurt and Vienna, bearing the same colors as in Fig. 4.1. The green path (*i.e.*, the best M_2 path) is encoded by a single node segment. The pink, direct path (*i.e.*, the best M_1 path) is encoded by an adjacency segment (the double line in Fig. 4.2b). The yellow paths (the solution of DCLC-SR(Frankfurt, 3, 8) and an interesting tradeoff between M_1 and M_2) requires an additional segment, in order to be routed through Budapest. Note that in practice, the last segment is unnecessary if it is a node segment, as the packet will be routed towards its final IP destination through the best M_2 paths natively.

Using the SR Graph, we can now describe more intuitively the relevant subproblems that may be addressed when solving 2COP. As defined in Def. 4.1.2, solving $2COP(s, c_0, c_1, c_2)$ requires to retrieve the entirety of the constrained Pareto front, which not only contains solutions to DCLC-SR, but to other relevant problems considering the original, as well as tighter constraints.

Considering Fig. 4.2, solving $2COP(Frankfurt, 3, 100, 10)$ enables one to solve, for example, DCLC-SR considering $c_0 = 3$ and $c_1 = 100$ from Frankfurt to Vienna. Solving this problem is equivalent to retrieving the following output of 2COP (we rely on the first capital letter of the cities):

$$f(M_2, 3, 80, \infty, F, V) = (Node, F, B)|(Node, B, V) \ (67; 4)$$

One may also find solutions to *Constrained Cost Lowest Delay* (CCLD), *i.e.*, optimizing $M1$ while respecting a constraint on M_0 and M_2. For example,

$$f(M_1, 3, \infty, 10, F, B) = (Node, F, B)\ (54, 2)$$

Finally, solutions to *Delay-Cost Constrained with the Lowest #Segment* (DCCLS) are also available *i.e.*, optimizing M_0 while respecting constraints on M_1 and M_2. For example, we have

$$f(M_0, 10, 50, 50, F, V) = (Adj, F, V)\ (39, 10)$$

These solutions may all be retrieved by solving 2COP considering Frankfurt as a source, which can be achieved efficiently thanks to our SR graph.

Indeed, our multi-metric SR graph (or equivalent constructs gathering the multi-metric all-pair shortest path data) is mandatory to easily consider the number of segments necessary to encode the paths being explored. However, its usage can differ in practice. We envision two modes that allow considering this additional "off the graph" metric, using our SR Graph.

One of the two options is to run the path computation algorithm on the original topology and convert the paths being explored to segment lists during the exploration. Performing this conversion is however not trivial. One must return the minimal encoding of the given path (with respect to the number of segments) while correctly managing the (forced) path diversity brought by ECMP, which may exhibit heterogeneous delays. However, one can efficiently perform such conversion when relying on our SR graph. By summarizing the relevant information (*i.e.*, the worst-case delay within ECMP paths), the SR Graph allows to easily consider the ECMP nature of SR within a multi-metric context. However, the segment metric M_0 is peculiar. Extending a path does not always imply an increase in the number of necessary segments. We will see that because the way the number of segments evolve is not trivial to predict, pruning paths from the exploration requires further consideration. Because of these properties, the way to check paths for dominance must be revised. Indeed, we will show that distances that appear dominated may become non-dominated later on, if their number of segments evolve favorably. This extended dominance check may lead to an increased number of paths to extend, and thus to a higher worst-case complexity.

Another method, perhaps easier to implement, is to run the path computation algorithm directly on the SR graph we described. Note that this forces the algorithm to run on a complete graph, which may significantly increase the overall complexity. However, the segment metric M_0, originally an "off the graph" metric with singular properties, becomes a standard graph metric, as it is now expressed by the number of edges that compose the paths (a path encoded by x segments has x edges within the SR Graph). This method also allows using standard, known algorithms as-is to solve the DCLC-SR problem. However, the SR Graph possess (at least) $|V|^2/2$ edges. This high density may greatly increase the computation time of algorithms not designed to work hand-in-hand with the SR Graph. In some cases, the cost of relying on the SR Graph may thus outweigh the benefits.

In the following, we will describe schemes enabling to use the SR Graph in both fashions. First, we describe *Live Conversion Algorithm* (LCA), our algorithm which relies on the

SR Graph to perform live conversions of the paths being explored, and detail the revisited dominance property. Then, we will describe BEST2COP, which directly explores the SR Graph and leverages its characteristics to propose competitive performance.

4.3 Using the SR Graph to perform live conversions with LCA

When using the SR Graph to perform conversions, the algorithm can explore the original graph, which should be sparser and thus less computationally expensive to explore than the SR graph [6].

Using the SR graph in this fashion requires designing an efficient conversion algorithm, able to, given a path as input, output a minimal segment list encoding said path (or, at least, paths sharing the same distances). This conversion is not trivial, as one must take into account the (forced) path diversity brought by ECMP. Indeed, using a node segment implies that the packet may follow any of the ECMP paths, which may possess heterogeneous delays. On the other hand, relying too much on adjacency segments will not lead to the minimal segment encoding of the considered paths and limit load-balancing.

Furthermore, the number of segments, being an "off the graph" metric, exhibits peculiar behavior and requires to slightly extend the way paths are checked for dominancy and pruned from the exploration.

In this subsection, we will formally describe and detail both LCA, our conversion algorithm, and the notion of strong dominance, which allows to correctly consider the number of segments as a new metric of the Pareto front when performing on-the-fly conversions. We will show that LCA allows retrieving the minimal segment list of all non-dominated segment lists, and thus the 3D Pareto front.

While some preliminary work about LCA has been published in the Appendix of a previous publication [Luttringer *et al.* 2022], most of the work detailed in this section (such as the proofs, gadgets, and formalization) has not yet been published.

4.3.1 *Live Conversion Algorithm* (LCA)

The common goal of path encoding schemes is to find, given an input path, a segment list encoding said path with the minimum number of segments. However, there exist several paradigms one may follow to compute segment lists.

Most path encoding schemes aim to encode a single, specific path (as described in Section 3.2.2). In other words, the (usually minimal) segment list computed *solely* encodes the input path. We refer to this as *strict encoding*. When considering a single metric, strict encoding makes sense. Indeed, in this case, SR is usually used for goals such as path monitoring, failure resiliency, or function chaining. It is thus important that traffic follows exactly the desired path.

In our case, we argue that while the input path should be considered, its *distance* is what matters most. Load-balancing should thus be leveraged, as long as the delay constraint is respected. In this case, two main paradigms can be followed. The first one consists in providing a distance as input and returning a segment list encoding any path(s) with said

[6] Note that the information within the SR Graph may be stored in different fashions. For example, one may keep the DAGs computed by the APSP separated by source, rather than merging it in a single SR Graph. However, the multi-metric APSP computation itself is mandatory

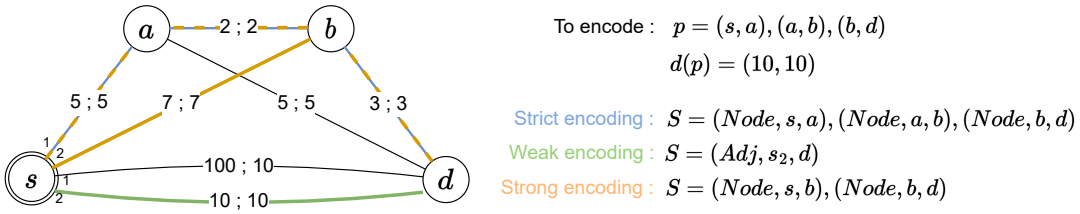


FIGURE 4.3: Figure illustrating the different type of encodings

distance (at most). We refer to this approach as *weak encoding*. The second approach consists in returning a segment list that *does* encode a path given as input, but may also encode other paths sharing the same or better distances. We refer to this approach as *strong encoding*.

The different types of encodings are illustrated in Fig. 4.3. Given the path to encode p, the segment lists S resulting from each type of encoding are shown. The paths encoded by S are shown in their respective color on the left side of the Figure.

We consider the path $p = (s,a),(a,b),(b,d)$. When relying on strict encoding, three segments are necessary to circumvent the effect of ECMP. Several segment lists weakly encode p, *i.e.*, encode path with distance $d = (10,10)$. Within these lists, the minimal one is composed of a single segment, encoding a direct edge between s and d. Note that the original path p is not contained within this segment list. Finally, the segment list that strongly encodes p does indeed encode p, as well as another path of equal distance.

While weak encoding may seem like the most natural option, it has several caveats. First, from a technical standpoint, its outcome is far less predictable, as it solely depends on distance with no paths acting as general anchor points. Second, weakly encoding a given distance $d(p = (s,t))$ may induce a high computational complexity. Intuitively, the lack of anchor points around which building the segment list requires finding all paths with said distance to find the associated minimal segment lists. In addition, note that relying on strong encoding does not necessarily prevent finding minimal encoding segment lists, as long as the proper path is considered. For example, considering Fig. 4.3, the encoding $S = (Adj,s_2,d)$ may also be found by strongly encoding the non-dominated path $p = (s_2,d)$. We leverage this when designing our conversion algorithm, in order to benefit from the more efficient strong encoding paradigm, while still ensuring that the minimal segment list is retrieved.

We thus design LCA as a conversion algorithm following the *strong encoding* paradigm. We first devise an offline conversion algorithm returning the minimal strongly encoding segment lists for a given path. We then adapt this algorithm for it to be usable online, *i.e.*, to encode paths *while* exploring the graph. We show that the peculiarities of the segment metric require particular care when maintaining and extending the distances. In particular, we show that some dominated distances must now be extended, as their number of segments may evolve differently despite being extended by the same edge. From there, we formally define how distances should be maintained (in particular when following the strong encoding paradigm), in order to ensure retrieving the actual 3D Pareto front, with the minimal encoding segment list of each non-dominated distance.

As the descriptions of LCA and strong encoding rely heavily on the notations introduced previously, we start by introducing (or reminding of) the following notations. We consider a network $G = (V, E)$. Let $p = ((x_0, x_1), (x_1, x_2), \ldots, (x_{l-1}, x_l))$ be a path within G, with $x_i \in V$. Let $p_{[x_i,x_j]} = ((x_i, x_{i+1}), \ldots, (x_{j-1}, x_j))$. We here consider the distance $d(p)$ as $d(p) = (d_1(p), d_2(p))$, as d_0 is unknown.

We denote $P_2^*(u, v)$ all the shortest paths between u and v in G with respect to the M_2 metric (*i.e.*, all ECMP paths between u and v). The edges of the paths $P_2^*(u, v)$ form a subgraph of the shortest path DAG rooted at u (considering M_2). We denote $Dag_{[u,v]}$ this set of edges.

A segment is characterized by a tuple (t, u, v), which encodes all ECMP paths from u to v when $t = Node$, or a specific link when $t = Adj$. We denote $d^{SR}(t, u, v)$ the best distances guaranteed when following the paths encoded by the segment (t, u, v). When $t = Node$, we have $d^{SR}(t, u, v) = (d_1^{max}(P_2^*(u, v)), d_2^*(u, v))$. When $t = Adj$, we have $d^{SR}(t, u, v) = w(u, v)$. Notice that the distance d^{SR} of a segment is thus exactly equal to the weight of the edge associated with this segment in the SR Graph.

Finally, we denote a list of segments $S = (S_i)_{0 \leq i \leq l-1}$. A segment $S_i \in S$ is thus defined as a tuple (t_i, s_i, s_{i+1}), with $s_i \in V$ and $t_i \in \{Adj, Node\}$. We defined the distance of a segment list $d^{SR}(S) = \sum_{n=0}^{l-1} d^{SR}(S_i)$.

Table 4.3.1 offers a summary and reminders of the notations used within this subsection

We now define formally the *strong encoding* of a path p.

Definition 4.3.1 (Strong Encoding)**.** *Let $p = ((x_0, x_1), \ldots, (x_{k-1}, x_k))$ be a path. Let $S = (S_i)_{0 \leq i \leq l-1}$ be a segment list.* S strongly encodes *the path p if both conditions are verified:*

- *$\forall i$, $\exists j$ such that $s_i = x_j$*
- *$\forall i$, either*

  (i) $p_{[s_i,s_{i+1}]} \in P_2^*(s_i, s_{i+1})$ and $d(p_{[s_i,s_{i+1}]})$ is dominated by $d^{SR}(S_i)$, or
  (ii) $t_i = Adj \wedge (s_i, s_{i+1}) \in p$.

Less formally, Def. 4.3.1 ensures that each segment within a segment list S either (i) does indeed encode a subpath of p, and that the paths encoded by the segment have equal or better distances than said subpath or (ii) encodes a specific link within p.

NOTATION	DEFINITION
(u, v)	An edge in E between node u and node v
$w(u, v)$	The weight vector of edge (u, v)
$w_i(u, v)$	The i^{th} component of $w(u, v)$
$p(s, t)$	A path between node s and t, defined as a list of edges
$p_{[x_i,x_j]}$	A subpath of p, defined as $(x_i, x_{i+1}) \dots (x_{j-1}, x_j)$.
$d(p)$	The distance vector of path p.
$d_i(p)$	The i^{th} component of $d(p)$, $i \leq m$.
$Dag_{[u]}$	Set of edges forming the shortest path DAG rooted at u considering the IGP cost
$Dag_{[u,v]}$	Edges of the paths from u to v within $Dag_{[u]}$
$P_i^*(s, t)$	All shortest path between node s and t when considering the i^{th} criterion.
$d_i^*(s, t)$	The best distance between node s and t when considering the i^{th} criterion.
$d_i^{max}(P(s, t))$	The maximum distance among all paths $p \in P$ considering the i^{th} criterion
S	A segment list
$S_i = (t, s_i, s_{i+1})$	A segment encoding a path between u and v with $t = Adj, Node$
$S_{[i,j]}$	Sublist of S consisting in $S_i, \dots, S_j$
$d^{SR}(t, s_i, s_{i+1})$	The best distances guaranteed by a segment
$d^{SR}(S)$	Distances of a segment list, defined as the sum of the distances of each segment within S.

FIGURE 4.4: Notations introduced within this section

Remark 4.3.1. *When considering condition (i) of Def. 4.3.1, $p_{[s_i,s_{i+1}]}$ and $d^{SR}(S_{[i,i+1]})$ necessarily share the same cost d_1 (the cost of the ECMP paths from s_i to s_{i+1}, $d_2^*(s_i, s_{i+1})$).*

Furthermore, by definition, the delay of $S_{[i,i+1]}$ is equal to the maximum *delay among all ECMP paths within $P_2^*(s_i, s_{i+1})$. Thus, stating that $d(p_{[s_i,s_{i+1}]})$ is dominated by $d^{SR}(S_{[i,i+1]})$, is equivalent to stating that $d_1(p_{[s_i,s_{i+1}]}) = d_1^{SR}(S_{[i,i+1]}) = d_1^{max}(P_2^*(s_i, s_{i+1}))$.*

Consequently, (i) can be alternatively expressed as :

$$d(p_{[s_i,s_{i+1}]}) = d^{SR}(S_{[i,i+1]})$$

.

We then define the *delay-cost path encoding problem*, which we aim to solve with our encoding scheme.

Definition 4.3.2 (Delay-cost path encoding problem)**.** *Given a path p, the delay-cost path encoding problem consists in finding a segment list S that strongly encodes p with the minimal number of segments.*

Before describing how we solve the Delay-cost path encoding problem in an online fashion (*i.e.*, while exploring the graph), we first describe how to solve this problem when the complete path is already known and provided as input.

We solve the delay-cost path encoding problem through a greedy algorithm. In short, this algorithm repeatedly finds, for a given path p, the longest prefix of p encodable by a single segment. This algorithm is presented in Alg. 1.

Algorithm 1: ENCODE(p)

```
1 s := x_0
2 S := [ ]
3 do
4 |   S_i = (t, s, v) = 1SegLongestPrefix(p_[s,x_l])
5 |   S.push(S_i)
6 |   s := v
7 while s ≠ x_l;
8 return S
```

Considering a path to encode p and a source s (the first node of the path), Alg. 1 starts by finding the longest prefix of p encodable in a single segment. Suppose that S_i encodes such a prefix $p_{[s,v]}$. The algorithm then adds the segment to the segment list S and repeats the procedure, considering the remainder of the path $p_{[v,x_l]}$ and v as the new source. This is repeated until the node considered as source is equal to the last node in p, *i.e.*, x_l.

Algorithm 2: 1SEGLONGESTPREFIX($p = (x_0, x_1), \dots (x_{l-1}, x_l)$)

```
1 if (x_0, x_1) ∉ Dag_[x_0,x_1] or w_1(x_0, x_1) ≤ d_1^SR(Node, x_0, x_1) then
2 |   return (Adj, x_0, x_1)
3 for i = 2 ... l do
4 |   if (x_{i-1}, x_i) ∉ Dag_[x_0,x_i] or d_1(p_[x_0,x_i]) < d_1^SR(Node, x_0, x_i) then
5 |   |   return (Node, x_0, x_{i-1})
6 return (Node, x_0, x_l)
```

To find the longest prefix encodable in one segment (and the associated segment), Alg 1 relies on another algorithm, 1SegLongestPrefix, described in Alg. 2. Intuitively, Alg. 2 tries to encode as many edges of p as possible in a single segment. Once the algorithm detects that a new segment would be required to properly encode an additional edge (*e.g.*, because it deviates from the ECMP paths), the algorithm returns.

More precisely, 1SegLongestPrefix starts by checking whether the first edge of p requires an adjacency segment (Line 1). Recall that a node segment may only encode ECMP paths, *i.e.*, paths within P_2^*. Consequently, a node segment (interpreted by x_0) may not encode an edge (x_0, x_1) which does not lie within $Dag_{[x_0,x_1]}$. In this case, the edge (x_0, x_1) must be encoded by an adjacency segment [7]. Note that since an adjacency segment only encodes a single link, encoding additional edges will necessarily require new segments. Hence, the algorithm returns immediately.

Otherwise, the algorithms checks, for each $i = 2 \dots l$, whether the subpath $p_{[x_0,x_i]}$ *remains* encodable in a single node segment when considering an additional edge (x_{i-1}, x_i). To remain encodable in a single node segment, the subpath must respect two conditions.

First, it must remain an ECMP path. This can be checked by verifying whether the new edge (x_{i-1}, x_i) remains within the shortest path DAG rooted at Dag_{x_0}, *i.e.*, if the edge belongs to Dag_{x_0,x_i}. Otherwise, the subpath induced by the added edge deviates from the shortest paths rooted at x_0 and thus cannot be encoded through a node segment interpreted by x_0. Second, if it is an ECMP path from x_0 to x_i, it must not possess a better delay than the one guaranteed by the node segment, *i.e.*, $d_1^{max}(P_2^*(x_0, x_i))$. Otherwise, the subpath exhibits a better delay than some ECMP path and must thus be encoded more precisely through additional segments.

Once one of these conditions is not met anymore, the longest prefix of p encodable by a single segment is found, and the algorithm returns. By following these steps, Alg. 1 returns a segment list that strongly encodes the path p given as input, as shown in Lemma 4.3.1.

Lemma 4.3.1. *Let $S = \text{ENCODE}(p)$. Then, S strongly encodes p.*

Proof. Any segment S_i respects the conditions of strong encoding, as defined in Def. 4.3.1.

- Notice from Alg. 2 that all segments $S_i \in S$ are necessarily equal to (t, s_i, s_{i+1}) with $s_i \in p$ and $s_{i+1} \in p$ (Line 2, 5, 6).
- If $S_i = (Adj, s_i, s_{i+1})$, then $(s_i, s_{i+1}) \in p$ following Alg. 1,Line 2.
- Finally, if $S_i = (Node, s_i, s_{i+1})$, then each edge $e \in p_{[s_i,s_{i+1}]}$ belongs to $Dag_{[s_i,s_{i+1}]}$ meaning that $p_{[s_i,s_{i+1}]} \in P_2^*(s_i, s_{i+1})$ (following Alg. 1, Line 4). Thus, by definition, $d_2(p_{[s_i,s_{i+1}]}) = d_2^{SR}(Node, s_i, s_{i+1})$.

  In addition, $d_1(p_{[s_i,s_{i+1}]}) \geq d_1^{SR}(Node, s_i, s_{i+1})$ (Line 4). Consequently, $d(p_{[s_i,s_{i+1}]})$ is dominated by $d^{SR}(S_i)$, following the definition of d^{SR}.

[7] If several adjacency segments exist between x_0 and x_1, the one that matches the weight of the considered edge should be used.

Consequently, S strongly encodes p, as per Def. 4.3.1.

□

Furthermore, we can show that the segment list returned by ENCODE encodes strongly the path given as input in the minimal number of segments.

Lemma 4.3.2. *Let S be a segment list that strongly encodes p. Then, $|S| \geq |\text{ENCODE}(p)|$, i.e., ENCODE finds the minimal strong encoding of p.*

Proof. Let us consider two segment lists $S = ((S_i)_{0 \leq i \leq l-1})$ and $S' = ((S'_i)_{0 \leq i \leq l'-1})$, both encoding p. Let $S' = \text{ENCODE}(p)$ (*i.e.*, S' is the result of Alg. 1).

For the sake of contradiction, let us assume that $l < l'$. Then, at some point, a segment S_k encoded more of p than $S'k$.

Recall that, per definition of strong encoding, $s_i \in p$.Then, there must exist k, $0 \leq k \leq l-1$, such that

1. s_k appears before, or at the same place as s'_k in path p
2. s_{k+1} appears strictly after s'_{k+1} in path p.

Let us start by considering that $s_k = s'_k$. Then, 2. contradicts with the fact that S'_k is a path encoding the longest prefix of p (recall that S'_k is the output of procedure 1SegLongestPrefix on s'_k).

Otherwise, s_k appears strictly before s'_k in path p. Notice that S_k is then necessarily a node segment, as it encodes more than an edge. Following the definition of strong encoding, $p_{[s_k, s_{k+1}]} \in P_2^*(s_k, s_{k+1})$. Furthermore, as $p_{[s'_k, s_{k+1}]}$ is a subpath of $p_{[s_k, s_{k+1}]}$, $p_{[s'_k, s_{k+1}]}$ is a shortest path, following the subpath optimality property. Thus, we have

$$(i)\ p_{[s'_k, s_{k+1}]} \in P_2^*(s'_k, s_{k+1})$$

When then have to prove that

$$(ii)\ d(p_{[s'_k, s_{k+1}]}) \text{ is dominated by } d^{SR}(s'_k, s_{k+1}).$$

Recall from Remark 4.3.1 that (ii) is equivalent to showing that $d(p_{[s'_k, s_{k+1}]}) = d^{SR}(Node, s'_k, s_{k+1})$. We show that this equality does indeed both for both M_1 and M_2.

1. It follows naturally from (i) and the definition of d^{SR} that $d_2(p_{[s'_k, s_{k+1}]}) = d_2^{SR}(Node, s'_k, s_{k+1})$.
2. By definition of d^{SR}, $d_1(p_{[s'_k, s_{k+1}]}) \leq d_1^{SR}(Node, s'_k, s_{k+1})$.

Let us suppose, for the sake of contradiction, that $d_1(p_{[s'_k,s_{k+1}]}) < d_1^{SR}(Node, s'_k, s_{k+1})$. Then, there exists $p' \in P_2^*(s'_k, s_{k+1})$ such that $d_1(p') > d_1(p_{[s'_k,s_{k+1}]})$. Then, the path $p_{[s_k,s'_k]} \oplus p'$ is a shortest path from s_k to s_{k+1} (as $d_2(p') = d_2(p_{[s'_k,s_{k+1}]}) = d_2^{SR}(Node, s'_k, s_{k+1})$) and $d_1(p_{[s_k,s'_k]} \oplus p') > d_1(p_{[s_k,s_{k+1}]})$. Consequently, we have $d_1^{SR}(Node, s_k, s_{k+1}) \geq d_1(p') > d_1(p_{[s_k,s_{k+1}]})$ (by definition of d^{SR}). Thus, as $d^{SR}(Node, s_k, s_{k+1})$ does not dominate $d(p_{[s_k,s_{k+1}]})$, $(Node, s_k, s_{k+1})$ does not encode $p_{[s_k,s_{k+1}]}$, which contradicts the definition of S.

We can then deduce that $d_1(p_{[s'_k,s_{k+1}]}) = d_1^{SR}(Node, s'_k, s_{k+1})$.

Given that (i) and (ii) must be true, $(Node, s'_k, s_{k+1})$ encodes $p_{[s'_k,s_{k+1}]}$, which contradicts the fact that $S'_k = (t, s'_k, s'_{k+1})$ encodes a longest prefix, as s_{k+1} appears after s'_{k+1} in p. □

Such a conversion algorithm must be called repeatedly when exploring the graph. Indeed, simply solving DCLC and then converting the non-dominated path to a segment list is not sufficient. For example, the DCLC path returned may require more than MSD segments to encode. Consequently, the number of segments necessary to encode a path must be maintained at each extension, to maintain the entire 3D Pareto front (considering segments, delay, and cost) and ensure finding the solution to DCLC-SR.

Considering this, we further optimize our encoding scheme. Note that the checks at Line 1 and 4 can be performed efficiently through the SR graph when keeping only the edges representing node segments. Indeed, the weight of an edge within this SR graph corresponds to the distance of the associated segment. Thus, checking whether a path $p(u, v)$ can be encoded through a node segment can be done by matching its distance with the weight of the edge $w(u, v)$ with the SR Graph, *i.e.*, $d^{SR}(Node, u, v)$.

More precisely, one may check if a path $p(u, v)$ is an ECMP path by checking whether it possesses the best cost from u to v, *i.e.*, if its cost is equal to $d_2^{SR}(Node, u, v)$ (*i.e.*, $w_2(u, v)$ in the SR Graph). Furthermore, checking whether a path $p(u, v)$ does not possess a better delay than other ECMP paths can be done by checking whether its delay is equal to $d_1^{SR}(Node, u, v)$.

Furthermore, while Alg. 1 and 2 assume that the whole path is given as input, the latter can be easily modified to consider paths edges by edges and correctly update the corresponding segment lists to encompass each new edge, instead of re-considering the entire path.

From these observations, we derive a new algorithm able to strongly encode paths in segment lists *during* the exploration of said path while relying on efficient tests performed via the SR Graph.

LCA (Alg. 3) takes as input a segment list strongly encoding a path p, and an edge e. It then returns a new segment list, strongly encoding $p \oplus e$. LCA follows the same general idea as Alg. 1 and 2, *i.e.*, encoding the longest possible prefix through a single segment.

More precisely, LCA starts by retrieving the last segment of the segment list $S_{l-1} = (t, s, u)$. It then checks if this segment can be updated to (t, s, v) to encompass the new edge e. Since an adjacency segment may only encode one segment, updating (t, s, u) to (t, s, v)

Algorithm 3: LCA(S, $e = (u, v)$)

```
1  (t, s, u) = S_{l-1}
2  if S = [ ] then
3  |   t = None
4  |   s = u
5  if t = Node and d^SR(S_{l-1}) + w(e) = d^SR(Node, s, v) then
6  |   S_{l-1} = (t, s, v)
7  |   return S
8  if w(e) ≠ d^SR(Node, s, v) then
9  |   S.push((Adj, u, v))
10 |   return S
11 S.push((Node, u, v))
12 return S
```

is only possible if $t = Node$. To ensure that the new, extended path is strongly encoded by $(Node, s, v)$, the distance of the path is matched against the distance of said node segment (see Line 5). The last segment is then modified accordingly.

Otherwise, a new segment is necessarily required. Line 8 checks whether this new segment should be an adjacency (*i.e.*, by matching the weight of e against the node segment $(Node, u, v)$) or a node segment. The new segment is then pushed onto S, which is returned. We now show that relying on LCA allows computing the same segment lists as ENCODE.

To do so, we will consider a simple wrapper to the LCA function, which calls LCA iteratively on each edge of a path and returns the resulting segment list.

Algorithm 4: LCA_WRAPPER($p = (x_0, x_1) \dots (x_{l-1}, x_l)$)

```
1 S = (Node, x_0, x_0)
2 for e ∈ p do
3 |   S = LCA(S, e)
4 return S
```

Lemma 4.3.3. *Let p be a path. Then,* LCA_WRAPPER*(p)* = ENCODE*(p).*

Proof. Let S^{lca} be the segment list obtained when calling LCA_WRAPPER with a given path p as argument, and S^{enc} the segment list obtained when calling ENCODE p as argument.

We will prove by induction that, when encoding a path p of n edges, $S^{lca} = S^{enc}$.

Base case (n = 1)

When $n = 1$, the path is composed of a single edge $e = (u, v)$. Any encoding segment will thus necessarily be equal to $S = (t, u, v)$. Consequently, it suffices to show that $t_0^{lca} = t_0^{enc}$.

Suppose that $t_0^{lca} = Adj$. By following Alg. 3, this implies $w(e) \neq w^{G'}(u, v)$, *i.e.*, $w(e) \neq d^{SR}(Node, u, v)$, by definition of the SR graph and d^{SR}. If $w(e) \neq w^{G'}(u, v)$,

either

1. $w_1(e) < d_1^{SR}(Node, u, v)$, *i.e.*, $w_1(e) < d_1^{max}(P_2^*(u, v))$
2. $w_2(e) > d_2^{SR}(Node, u, v)$ *i.e.*, $w_2(e) > d_2^*(u, v)$ and so $e \notin Dag_{[u,v]}$, by definition of d^{SR}.

Consequently, given Line 1 of Alg. 2, $t_0^{enc} = Adj$. Thus, $t_0^{lca} = Adj$ implies that $t_0^{enc} = Adj$.

If $t^{lca} = Node$, then $w(e) = d^{SR}(Node, u, v)$. Thus, $w_2(e) = d_2^*(u, v)$, meaning that $(u, v) \in Dag_{[u,v]}$, and $w_1(e) = d_1^{max}(P_2^*(u, v))$. Consequently, according to Line 1 of Alg. 2, $t_0^{enc} = Node$. Thus, $t_0^{lca} = Node$ implies that $t_0^{enc} = Node$. The property then holds for $n = 1$.

Inductive hypothesis

Let k be given and suppose that the property remains true for $n = k$, *i.e.*, that ENCODE(p)=LCA_WRAPPER(p) for any p of n edge. We show that this remains true when considering a path $p' = p \oplus e$.

Inductive step

When encoding $p + e$, both segment lists may gain one or zero additional segment.

Let $p = (x_0, x_1) \dots (x_{n-1}, x_n)$ be a path of n edge. By hypothesis, S^{enc} =ENCODE(p) is equal to S^{lca} =LCA_WRAPPER(p) and both segment lists possess the same number of segments, *i.e.*, $l^{enc} = l^{lca} = l$. Consequently, $S^{enc}_{[0,l-2]} = S^{lca}_{[0,l-2]}$.

Let us consider the path $p' = p \oplus e$ of $n + 1$ edges, with $e = (x_n, x_{n+1})$. Let $S^{enc'}$ =ENCODE(p') and $S^{lca'}$ =LCA_WRAPPER(p'). Note that both $S^{enc'}$ and $S^{lca'}$ have at most $l + 1$ segment. Indeed, if $l^{enc'} > l + 1$, then there exists a segment lists $S = S^{enc} \oplus (Adj, x_n, x_{n+1})$ which strongly encodes p' in $l+1$ segments, which contradicts Lemma 4.3.2. Furthermore, following Alg. 3, a new edge e may either add a single segment (Line 10 and 12) or leave the number of segments unchanged (Line 7). Thus, we have $l \leq l^{enc'} \leq l + 1$ and $l \leq l^{lca'} \leq l + 1$.

As e may only impact the last segment, showing that both algorithms choose the same last(s) segment(s) is sufficient.

Clearly from the pseudo-code of the algorithm, the edge e can only impact the last segment of $S^{enc'}$ and $S^{lca'}$ (*i.e.*, the l^{th} or $l + 1^{th}$ segment).

More precisely, considering e may either add a new segment to the list, or change the last segment. Previous segment are not impacted.

Thus, we have at least $S^{enc'}_{[0,l-2]} = S^{enc}_{[0,l-2]}$ and $S^{lca'}_{[0,l-2]} = S^{lca}_{[0,l-2]}$. Recall that $S^{enc}_{[0,l-2]} = S^{lca}_{[0,l-2]}$.

Consequently, $S^{lca'}_{[0,l-2]} = S^{enc'}_{[0,l-2]}$. It is thus sufficient to show that the last(s)

segment(s) of $S^{lca'}$ and $S^{enc'}$ are identical.

When $l^{enc'} = l$, then both ENCODE$(p \oplus e)$ and LCA_WRAPPER$(p \oplus e)$ output the same segment list.

Since we already know that $S^{lca'}_{[0,l-2]} = S^{enc'}_{[0,l-2]}$, it is sufficient to show that $S^{lca'}_{l-1} = S^{enc'}_{l-1}$ to prove that $S^{enc'} = S^{lca'}$, as both will then encode the entirety of p' and be of size l. By hypothesis, we know that $S^{lca}_{l-1} = (Node, x_i, x_n)$ (recall that $S^{lca}_{l-1} = S^{enc}_{l-1} =$ENCODE$(p)$). There is thus only one possibility : ENCODE chose to update its last (node) segment to encompass the new edge e. We will now show that in this case, LCA would follow the same behavior.

Let us suppose that $l^{enc'} = l$. Following Alg. 1, that means that the ultimate call to 1SegLongestPrefix reached Line 6, meaning that $S^{enc}_{l-1} = (Node, s_{l-1}, x_n)$ and $S^{enc'}_{l-1} = (Node, s_{l-1}, x_{n+1})$. We then know that $e \in Dag_{[s_{l-1},x_{n+1}]}$ and $d_1(p_{[s_{l-1},x_{n+1}]}) \geq d_1^{SR}(Node, x_i, x_{n+1})$. Given that $e \in Dag_{[x_i,x_{n+1}]}$, then $d_2^{SR}(S^{lca}_{l-1}) + w_2(e) = d_2^*(x_i, x_{n+1}) = d^{SR}(x_i, x_{n+1})$.

Furthermore, as $d_1(p_{[x_i,x_{n+1}]}) \geq d_1^{SR}(Node, x_i, x_{n+1})$, we know that $d_1(p_{[x_i,x_n]}) + w_1(e) = d_1^{SR}(Node, x_i, x_{n+1})$. Since $(Node, x_i, x_n)$ encodes $p_{[x_i,x_n]}$, it follows that $d_1^{SR}(p_{[x_i,x_n]}) + w_1(e) = d_1^{SR}(Node, x_i, x_{n+1})$. Consequently, from both equalities, we know that Line 5 is verified and that LCA also changes the last segment to $(Node, x_i, x_{n+1})$. No edges are left to consider.

When $l^{enc'} = l + 1$, then both ENCODE$(p \oplus e)$ and LCA_WRAPPER$(p \oplus e)$ output the same segment list.

Let us now suppose that $l^{enc'} = l + 1$. Since the last segment results from the addition of e to the path, said segment necessarily encodes e. Thus, $S^{enc'}_l$ may take only two values : either $S^{enc'}_l = (Node, x_n, x_{n+1})$ or $S^{enc'}_l = (Adj, x_n, x_{n+1})$. Consequently, the penultimate segment of $S^{enc'}$ may also only take two values : (i) $(Node, s_{l-1}, x_n)$ or (ii) (Adj, x_{n-1}, x_n).

(i) *If the penultimate segment of $S^{enc'}$, i.e., $S^{enc'}_{l-1}$ is a node segment, both* ENCODE$(p \oplus e)$ *and* LCA_WRAPPER$(p \oplus e)$ *output the same segment list*

Here, we have $S^{enc'}_{l-1} = (Node, s_{l-1}, x_n)$. Since e only added a new segment to the list, we know that $S^{enc'}_{l-1} = S^{enc}_{l-1}$, and so that $S^{enc'}_{l-1} = S^{enc}_{l-1} = S^{lca}_{l-1}$. Thus, $S^{lca}_{l-1} = (Node, s_{l-1}, x_n)$.

While both segment lists end with the same node segment, this does not mean that they remain the same after considering e, *i.e.*, it does not ensure that $S^{enc'}_{l-1} = S^{lca'}_{l-1}$. Indeed, while we know that ENCODE chose to add a new segment to the segment list (since $l^{enc'} = l + 1$) and leave $S^{enc'}_{l-1}$ unchanged, LCA could have chosen to update the node segment S^{lca}_{l-1} to encompass e, and not add any new segment after that. Thus, we have to prove that if ENCODE chose to add a new node or adjacency segment after $S^{enc'}_{l-1} = S^{lca}_{l-1}$, so did LCA.

If $S_l^{enc'} = (Node, x_n, x_{n+1})$, then two possible conditions could have been verified for ENCODE to add this segment.

Either, $(x_n, x_{n+1}) \notin Dag_{[s_{l-1}^{enc'}, x_{n+1}]}$, in which case $p_{[s_{l-1}^{enc}, x_{n-1}]} \oplus (x_{n-1}, x_n)$ is not a shortest path. Since $d^{SR}(S_{l-1}) = p_{[s_{l-1}^{enc}, x_n]}$ by definition, then $d^{SR}(S_{l-1}) + (x_n, x_{n+1}) \neq d^{SR}_{Node, s_{l-1}^{enc}, x_{n+1}}$, meaning that Line 5 of LCA is not verified.

Or, $d^1(p_{[s_{l-1}^{enc}, x_{n+1}]}) < d_1^{SR}(Node, s_{l-1}^{enc}, x_{n+1})$. In this case, $d_1(p_{[s_{l-1}^{enc}, x_n]}) + w_1(e) < d_1^{SR}(Node, s_{l-1}^{enc}, x_{n+1})$, meaning that $d_1^{SR}(Node, s_{l-1}^{enc}, x_n) + w_1(e) < d_1^{SR}(Node, s_{l-1}^{enc}, x_{n+1})$, *i.e.*, that $d^{SR}(S_{l-1}) + w_1(e) < d_1^{SR}(Node, s_{l-1}^{enc}, x_{n+1})$. Line 5 of LCA is consequently still not verified.

Finally, we know that $(x_n, x_{n+1}) \in Dag_{[x_n, x_{n+1}]}$ and $w_1(x_n, x_{n+1}) < d_1^{SR}(Node, x_n, x_{n+1})$ (else 1SEGLONGESTPREFIX would have returned an adjacency segment). Thus, clearly, we have $w_2(x_n, x_{n+1}) = d_2^*(x_n, x_{n+1}) = d_2^{SR}(Node, x_n, x_{n+1})$. Furthermore, we know that $w_1(x_n, x_{n+1}) \geq d_1^{SR}(Node, x_n, x_{n+1})$ for the same reason. Thus, $w_1(x_n, x_{n+1}) = d_1^{SR}(Node, x_n, x_{n+1})$, per definition of d^{SR}. Consequently, $w_(x_{n1}, x_{n+1}) = d^{SR}(Node, x_n, x_{n+1})$, meaning that Line 8 is not verified.

Consequently, if $S_l^{enc'} = (Node, x_n, x_{n+1})$, then $S_l^{lca'} = (Node, x_n, x_{n+1})$.

If $S_l^{enc'} = (Adj, x_n, x_{n+1})$, then two conditions had to be verified for ENCODE to add this segment. First, $(x_n, x_{n+1}) \notin Dag_{[x_n, x_{n+1}]}$ and second, $w_1(x_n, x_{n+1}) < d_1^{SR}(Node, x_n, x_{n+1})$.

Since $(x_n, x_{n+1}) \notin Dag_{[x_n, x_{n+1}]}$, it follows than $p_{[x_{l-1}, x_n]} \oplus (x_n, x_{n+1})$ is not a shortest path from x_{l-1} to x_n, and so that $d_2(p_{[x_{l-1}, x_n]}) + w_2(x_n, x_{n+1}) \neq d^{SR}(Node, x_{l-1}, x_n)$ by definition of d^{SR}. Since S_{l-1}^{lca} encodes $p_{[x_{l-1}, x_n]}$ (recall that $S_{l-1}^{enc'} = (Node, x_{l-1}, x_n) = S_{l-1}^{lca}$), then $d(p_{[x_{l-1}, x_n]}) = d^{SR}(S_{l-1}^{lca})$, meaning that $d^{SR}(S_{l-1}^{lca}) + w_2(x_n, x_{n+1}) \neq d^{SR}(Node, x_{l-1}, x_n)$. Thus, Line 5 of LCA is not verified.

Since $w_1(x_n, x_{n+1}) < d_1^{SR}(Node, x_n, x_{n+1})$, then $w(x_n, x_{n+1}) \neq d^{SR}(Node, x_n, x_{n+1})$, meaning that Line 8 of LCA is verified, and that LCA also adds (Adj, x_n, x_{n+1}) to its segment list.

(ii) *If the penultimate segment is an adjacency segment, both* ENCODE*$(p \oplus e)$ and lca_ wrapper$(p \oplus e)$ output the same segment list*

Similarly to the previous case, we necessarily have $S_{[0,l-1]}^{enc'} = S_{[0,l-1]}^{lca'} = (Adj, x_{n-1}, x_n)$. However, in this case, both algorithms have to add another segment to the list during the next step, as an adjacency segment can only encompass one edge.

Furthermore, note that when encoding the last edge $e = (x_n, x_{n+1})$, LCA will thus consider $s = x_n$, $t = Adj$ and thus behave exactly as if encoding a path of a single edge $p = (x_n, x_{n+1})$. Similarly, the last call of 1SEGLONGESTPREFIX, by

design, will behave exactly like if encoding a path $p = (x_n, x_{n+1})$. Consequently, the argument used for the case $n = 1$ is identical and holds.

Thus, we have ENCODE$(p + e)$ = LCA_WRAPPER$(p + e)$, meaning that ENCODE(p) = LCA_WRAPPER(p) for all p.

□

Lemma 4.3.4 follows naturally from Lemma 4.3.3.

Lemma 4.3.4. *LCA allows computing the minimal strong encoding of any given path p.*

As long as the segment lists encoding the paths being explored are maintained, LCA may be fitted onto any shortest path computation algorithm to encode paths into segment lists that ensure that their distance is respected. The associated overhead is negligible, as the required checks can be performed very efficiently if the SR graph was computed beforehand.

However, simply relying on LCA to track the size of the segment lists is not sufficient by itself to solve DCLC-SR. Because of the peculiarities of the segment metric, some *dominated* paths must be maintained and extended to retrieve the entirety of the three-dimensional Pareto front.

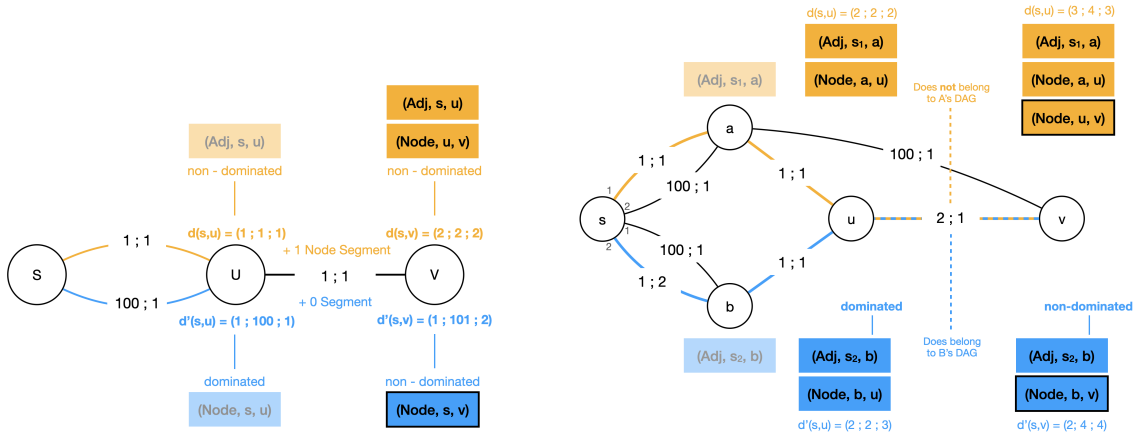
4.3.2 Keeping (some) dominated paths

Although the algorithms presented in the previous section allow finding the minimal number of segments to strongly encode a path, other modifications are required to correctly consider the number of segments as a new dimension when computing paths.

The notion of dominance is central to most DCLC schemes to limit the number of distances to extend. The standard definition of dominance is straightforward: a distance toward a node u is dominated if all the components of its weight vector are worse than or equal to another distance towards the same node (see Section 2.2.2). Dominated distances can usually be ignored, as they cannot lead to better distances overall. Indeed, as both distances may benefit from the same edges onwards, there is no way for the dominated distance to improve upon the non-dominated one.

This simple property does not behave in the same fashion when considering the number of segments as a new dimension of the paths' distance. Indeed, extending a distance by an edge (u, v) may or may not cost an additional segment, depending on the actual *path* used to reach u. As distances may be extended with no impact on the number of segments, distances that seem dominated may become non-dominated later on, or may even be the only feasible distances at the end of the exploration.

These peculiarities are illustrated in Fig. 4.5. Let us consider the COMEBACK gadget. Two delay-cost vectors $(d_1; d_2)$ exist to reach u from s : $(1; 1)$ and $(100; 1)$. Both can be encoded with a single segment ((Adj, s, u) and $(Node, s, u)$ respectively), leading to the distances $d(s, u) = (1; 1; 1)$ and $d'(s, u) = (1; 100; 1)$, where the first component d_0 is the



(A) COMEBACK gadget. (B) UNDERDAG gadget.

FIGURE 4.5: Figures illustrating the peculiarities of the number of segment metric.

number of segments. Note that following the standard definition of dominance, $d'(s,u)$ is dominated.

Both d' and d can be extended by (u,v), leading to the delay-cost vector $(2;2)$ and $(101;2)$. However, while the vector $(2;2)$ requires an additional node segment to be encoded (following the adjacency segment), the vector $(101;2)$ remains encodable through a single node segment $(Node,s,v)$. Thus, upon extension, the obtained distances are $d(s,v) = (2;2;2)$ and $d'(s,v) = (1;101;2)$.

While resulting from the extension of a dominated distance, $d'(s,v)$ is non-dominated (and the only feasible distance when considering a strict c_0 constraint of 1).

In the COMEBACK gadget, this effect results from the fact that an adjacency segment may only encode a single link. Hence, the following edges to encode will necessarily require additional segments. On the opposite, a node segment may encode any path lying within the shortest path DAG of the current source. Thus, as long as the next edges remain within said DAG, additional segments may not be required (depending on the weight of said edges). Rather, the current node segment may simply be updated to encode the following edges.

However, this effect may also occur even if both segment lists end with a node segment. This is illustrated by the UNDERDAG gadget. Let us consider the delay-cost vectors from s to u $(2;2)$ and $(2;3)$, from the paths $p(s,u) = (s_1,a),(a,u)$ and $p'(s,u) = (s_2,b),(b,u)$ respectively. These distances and paths can be encoded by the segments lists $S = (Adj,s_1,a)|(Node,a,u)$ and $S' = (Adj,s_2,b)|(Node,b,u)$ respectively. Thus, their distances are $d = (2;2;2)$ and $d' = (2;2;3)$. Note that d' is dominated by d, and that both S and S' are composed of segments of the same type.

However, notice that the edge (u,v) lies within the shortest path DAG of b. Hence, the last segment of S', $(Node,b,u)$, may simply be updated to $(Node,b,v)$ to encompass (u,v). On the contrary, the edge (u,v) does not lie within the shortest path DAG of a. Thus, the last segment of S, $(Node,a,u)$ may not be updated to encompass (u,v), and a new segment is required. Consequently, once updated, the distances to v are $d = (3;4;3)$ and $d' = (2;4;4)$. Once again, the non-dominated distance d' results from the extension of

a dominated distance.

From these observations, we derive and formally define the *strong dominance* relation. Strong dominance encompasses the peculiarities of the new metric when considering strong encoding, and should thus be followed when relying on LCA to correctly solve DCLC-SR.

Definition 4.3.3 (Strong dominance)**.** *Let d and d' be two distances encoding two paths p and p' from x_0 to x_k, composed of three components d_0, d_1, d_2, representing their number of segments, delay and cost respectively. Let $S = (t_i, s_i, s_{i+1})_{0 \leq i \leq l-1}$ and $S' = (t'_i, s'_i, s'_{i+1})_{0 \leq i \leq l'-1}$ be their respective (encoding) segment lists.*

The segment list S is strongly dominated by S' if d is dominated by d' (as per Def. 2.2.4), and *if either :*

1. $d_0 > d'_0$, *or*
2. $t_{l-1} = t'_{l'-1} \wedge d^{SR}(Node, s_{l-1}, s'_{l'-1}) + d^{SR}(Node, s'_{l'-1}, x_k) = d^{SR}(Node, s_{l-1}, x_k)$

While the importance of the conditions within Def. 4.3.3 are thoroughly detailed and leveraged within the following proofs, we will first describe their impact informally. Intuitively, if S is dominated by S', it will necessarily (by definition) remain dominated regarding its cost and delay, as both distances may be extended by the same edges, which have the same impact on both metrics. The additional conditions thus focus on the number of segment, which may evolve differently despite being extended by the same edge (as seen in Fig 4.5), and ensures that S remains dominated when considering this metric as well despite its odd behavior.

Less formally, we state that a segment list S is strongly dominated by S' if it is dominated by S', and if *one* of two conditions is verified.

The first condition is that the number of segments of S is strictly superior to that of S'. Otherwise, as the number of segments does not necessarily increase in a strictly monotonous fashion, it is possible for d_0 to become smaller than d'_0 (as we have seen in Fig. 4.5). However, we will prove that such a case (where S does not require an additional segment while S' does) may only happen once in a row. Hence, ensuring that $d_0 > d'_0$ is sufficient.

The second condition is perhaps less intuitive. Intuitively, the condition ensures that if S does not require another segment to encode an additional edge, neither does S', meaning that S is bound to remain dominated by S'.

Maintaining and extending only distances that are not strongly dominated allows pruning some distances from further exploration without compromising the integrity of the returned Pareto front, as stated and proved in Lemma 4.3.5.

Lemma 4.3.5. *A segment list that is strongly dominated remains strongly dominated after being extended.*

Proof. Let S be a segment list, and d the associated distance. Let S be strongly domi-

nated.

Since S is strongly dominated, there exists a segment list S' (having a distance d') such that S is dominated by S', and either (i) $t_{l-1} = t'_{l-1} \wedge d^{SR}(Node, s_{l-1}, s'_{l'-1}) + d^{SR}(Node, s'_{l'-1}, xn) = d^{SR}(Node, s_{l-1}, x_n)$ or (ii) $d_0 > d'_0$. Let us denote S^{+1} the segment list S once extended by an additional edge (u, v), and d^{+1} the associated distance.

If d is dominated by d' and $t_{l-1} = t'_{l-1} \wedge d^{SR}(Node, s_{l-1}, s'_{l'-1}) + d^{SR}(Node, s'_{l'-1}, x_n) = d^{SR}(Node, s_{l-1}, x_n)$, S^{+1} is strongly dominated when extended by an edge (u, v).

Clearly, when extending by the same edge $e = (u, v)$, the cost and the delay of d and d' evolves in the same fashion. We thus need to ensure that $d_0^{+1} \geq d_0'^{+1}$, and that (i) or (ii) is still verified.

d remains dominated

Note that when extending a distance and relying on the conversions schemes aforementioned, the number of segments may either stay the same or increase by one. The distances d_0 and d'_0 may thus evolve in four different fashion: either both increase by one, none increase by one, or one of them increases by one. We will refer to these cases as $+1/+1$ (both gain one segment), $+0/+0$ (both remain identical), $+1/+0$ (only d_0 gains a segment when extended) and $+0/+1$ (only d'_0 gains a segment when extended).

Note that in all cases but one, the dominance relationship remains unchanged. Solely the $+0/+1$ case may render d_0^{+1} non-dominated. However, recall that we mentioned earlier in this section that condition (i) ensures that if S does not require an additional segment, neither does S', meaning that this case may not occur if (i) is verified, as we will now show.

If S did not require an additional segment, then S_{l-1} is a node segment, meaning that $S'_{l'-1}$ is a node segment as well (since (i) holds).

Furthermore, this means that $d_2^{SR}(S_{l-1}) + w_2(u, v) = d_2^{SR}(Node, s^n_{l-1}, v)$. Thus, a shortest path from s_{l-1} to v goes through u. Since a shortest path is necessarily composed of shortest subpaths, we have $d_2^{SR}(s_{l-1}, v) = d_2^{SR}(s^n_{l-1}, u) + w(u, v)$. Since (i) hold, we have $d_2^{SR}(s_{l-1}, u) = d_2^{SR}(s_{l-1}, s'_{l'-1}) + d_2^{SR}(s'_{l'-1}, u)$. It follows than

$$d_2^{SR}(s_{l-1}, v) = d_2^{SR}(s_{l-1}, s'_{l'-1}) + d_2^{SR}(s'_{l'-1}, u) + w(u, v). \tag{4.1}$$

Notice that the triangle inequality hold for d_2. Consequently, we have $d_2^{SR}(s'_{l'-1}, u) \leq d_2^{SR}(s'_{l'-1}, u) + w_2(u, v)$. Similarly, we know that $d_2^{SR}(s_{l-1}, s'_{l'-1}) + d_2^{SR}(s'_{l'-1}, v) \geq d_2^{SR}(s_{l-1}, v)$, *i.e.*, that

$$d_2^{SR}(s'_{l'-1}, v) \geq d_2^{SR}(s^n_{l-1}, v) - d_2^{SR}(s_{l-1}, s'_{l'-1})$$

. Coupled with Eq. 4.1, we can deduce that

$$d_2^{SR}(s'_{l'-1}, v) \geq d_2^{SR}(s^n_{l-1}, s'_{l'-1}) + d_2^{SR}(s'_{l'-1}, u) + w_2(u, v) - d_2^{SR}(s^n_{l-1}, s'_{l'-1})$$

And so that $d_2^{SR}(s'_{l'-1}, v) \geq d_2^{SR}(s'_{l'-1}, u) + w_2(u, v)$. Consequently, we know that

$$d_2^{SR}(s'_{l'-1}, v) = d_2^{SR}(s'_{l'-1}, u) + w_2(u, v)$$

.

Furthermore, we know that $d_1^{SR}(s'_{l'-1}, v) = d_1^{SR}(s'_{l'-1}, u) + w_1(u, v)$. Let us suppose, for the sake of contradiction that $d_1^{SR}(s'_{l'-1}, v) \neq d_1^{SR}(s'_{l'-1}, u) + w_1(u, v)$. Then, as $d_1^{SR}(s'_{l'-1}, u) + w_1(u, v)$ is a shortest path, we know by definition that $d_1^{SR}(s'_{l'-1}, v) > d_1^{SR}(s'_{l'-1}, u) + w_1(u, v)$, meaning that there exist a shortest path p from $s'_{l'-1}$ to v with a delay higher than $d_1^{SR}(s'_{l'-1}, u)+w_1(u, v)$. However, in this case, we would have $d_1^{SR}(s_{l-1}, v) = d_1^{SR}(s^n_{l-1}, s'_{l'-1}) + d_1^{SR}(s'_{l'-1}, u) + w(u, v) < d_1^{SR}(s_{l-1}, s'_{l'-1}) + d_1(p)$, which contradicts the definition of d_1^{SR}.

Thus, for all *possible* cases, d^{+1} remains dominated if (i) is verified.

(i) or (ii) remains verified

Let us start by considering the $+0/+0$ case. Then, both segment lists S^n and S' ended with a node segment that has been updated. Let us denote S'^{+1} the segment list S' after being extended. Clearly, $t_l^{+1} = t_{l'}^{\prime+1} = Node$. Similarly, we have $s_l^{+1} = s_{l-1}$ and $s_{l'-1}^{\prime+1} = s'_{l'-1}$. Since the last (node) segment of S was updated, then we have

$$d^{SR}(Node, s_{l-1}, u) + w(u, v) = d^{SR}(Node, s_{l-1}, v) \tag{4.2}$$

Furthermore, as (i) was verified prior to the extension, we know that

$$d^{SR}(Node, s_{l-1}, s'_{l-1}) + d^{SR}(Node, s'_{l'-1}, u) = d^{SR}(Node, s_{l-1}, u) \tag{4.3}$$

It follows from Eq. 4.2 and Eq. 4.3 that

$$d^{SR}(Node, s_{l-1}, s'_{l'-1}) + d^{SR}(Node, s'_{l'-1}, u) + w(u, v) = d^{SR}(Node, s_{l-1}, v) \tag{4.4}$$

However, recall that S' also updated its last node segment, meaning that

$$d^{SR}(Node, s'_{l-1}, u) + w(u, v) = d^{SR}(Node, s'_{l'-1}, v) \tag{4.5}$$

It follows from Eq. 4.4 and Eq. 4.5 that

$$d^{SR}(Node, s_{l-1}, s'_{l'-1}) + d^{SR}(Node, s'_{l'-1}, v) = d^{SR}(Node, s_{l-1}, v)$$

, meaning that condition (i) remains verified.

Considering the $+1/+1$ case, then both segment list where either extended by one node segment $(Node, u, v)$ or one adjacency segment (Adj, u, v). Clearly, if both S and S' require a new segment, then the edge (u, v) does not lie on a shortest path from s_{l-1} to v, or had a better delay that other shortest paths. The similar reasoning can be had regarding $s'_{l'-1}$. The choice between an adjacency and a node segment thus boils down to whether (u, v) is the shortest path between u and v and, if it is, it possesses the highest delay among all shortest path between u and v. Clearly, this choice does not depend on

any previous segment, and both S and S' will be extended in the same fashion. Thus, we have $t_{l+1}^{+1} = t_{l'+1}'^{+1} = Node$. Since $s_l^{+1} = s_{l'}'^{+1} = u$, then (i) clearly remains verified, as the condition translates to $d^{SR}(Node, u, u) + d^{SR}(Node, u, v) = d^{SR}(Node, u, v)$.

Finally, we consider the last possible case, *i.e.*, the $+1/+0$ case. Note that by definition, we know that $d_0 \geq d'_0$ (as d is dominated by d'). Then, in the $+1/+0$ case, we necessarily have $d_0^{+1} > d_0'^{+1}$, meaning that (ii) is verified.

Thus, in all cases considered above, d remains dominated by d' after being extended. We now prove that the same is true when (ii) is verified.

If d is dominated by d' and $d_0 > d'_0$, d^{+1} is strongly dominated.

d^{+1} remains dominated

Similarly to the previous case, we simply need to ensure that after being extended, $d_0^{+1} \geq d_0'^{+1}$.

Let us consider the four possible cases. Clearly, for all possible cases $+0/+0$, $+1/+1$, $+1/+0$, $+0/+1$, we have either $d_0^{+1} > d_0'^{+1}$ or $d_0^{+1} = d_0'^{+1}$, meaning that d^{+1} is dominated by d'^{+1}.

(i) or (ii) remains verified

For cases $+0/+0$, $+1/+1$, $+1/+0$, $d_0^{+1} > d_0'^{+1}$ clearly remains true, meaning that (ii) remains verified and that d_0^{+1} remains strongly dominated.

Let us consider the case $+0/+1$. In this case, it is possible that $d_0^{+1} = d_0'^{+1}$, meaning that (ii) would not be verified anymore. However, recall that earlier in this section, we stated that such a case may only occur once in a row. Indeed, if this case occurs, then (i) becomes verified, meaning that the $+0/+1$ case cannot occur at the next iteration, as proved earlier. Let us show that (i) is now indeed verified.

In this case, we know that the last segment of S^{+1} is a node segment $(Node, s_{l-1}^{+1}, v)$, that has just been updated to encompass the edge (u, v). As a node segment may only encode shortest paths (with the highest delay), we know that the edge (u, v) must also be a shortest path from u to v, and possess the maximum delay among all these shortest paths. We can than deduce that $w(u, v) = d^{SR}(Node, u, v)$ meaning that the segment added to S' is also node segment. Thus, the last segment of S^{+1} and S'^{+1} share the same type.

Furthermore, we know that $d^{SR}(Node, s_{l-1}, u) + w(u, v) = d^{SR}(Node, s_{l-1}, v)$. As $w(u, v) = d^{SR}(Node, u, v)$, we know that $d^{SR}(Node, s_{l-1}, u) + d^{SR}(Node, u, v) = d^{SR}(Node, s_{l-1}, v)$. However, since the last segment of S'^{+1} is necessarily $(Node, u, v)$, then $s_{l'}'^{+1} = u$. Furthermore, we have $s_{l^{+1}-1}^{+1} = s_{l-1}^{n}$. Thus, we have $d^{SR}(Node, s_{l^{+1}-1}^{+1}, s_{l'}'^{+1}) + d^{SR}(Node, s_{l'}'^{+1}, v) = d^{SR}(Node, s_{l+1}^{+1}, v)$, meaning that condition (i) is verified.

Consequently, a strongly dominated distance d' remains strongly dominated.

□

From Lemma 4.3.5, we can deduce the following theorem :

Theorem 4.3.1. *Extending all non-strongly dominated segment list that have been computed through strong encoding allows to retrieve the entirety of the Pareto front of distances, considering cost, delay, and number of segments.*

Proof. As all non-dominated distances are segment lists, each non-dominated $(cost, delay)$ vector are found. We must then ensure that the minimal segment list will be found for each of these vectors.

We know that any given segment list encoding these distances strongly encodes a path (including the minimal segment list). Indeed, let S be a segment list. For any segment S_i, we know that there necessarily exists a path p_i from s_i to s_{i+1} such that $d(p_i) = d^{SR}(S_i)$, which is either the shortest path with the highest delay from s_i to s_{i+1} if $t_i = Node$, or a given link if $t_i = Adj$. Thus, S strongly encodes $p_0 \oplus p_1 \oplus \cdots \oplus p_l$.

Clearly, strongly encoding all paths of a given $(cost, delay)$ distance would then enable to find the minimal segment list encoding this distance.

While we do not strongly encode *all* of such paths (as not all paths are extended), we showed through Lemma 4.3.5 that the paths that are not extended (*i.e.*, strongly dominated paths) would necessarily have required a higher or equal number of segment to encode.

Consequently, we visit enough path to ensure that the minimal segment list encoding any given distance is found, and so that the entire Pareto front is retrieved.

□

In this section, we presented LCA, an SR conversion scheme that *strongly* encode paths, an encoding paradigm allowing to leverage load-balancing. Furthermore, we exhibited the peculiarities of the segment metric. From the latter, we derived the notion of strong dominance, which allows pruning paths from the exploration while ensuring to capture the entirety of the Pareto front. By relying on LCA and the concept of strong dominancy, multi-criterion path computation algorithms can be modified to correctly take into account the number of segments as a new dimension while exploring the original graph.

Note, however, that while the overhead induced by LCA is limited, the number of additional (dominated) distances to maintain may be quite high, with no guarantee that the latter will indeed end up on the Pareto front. For some algorithms, directly exploring the SR graph may be more efficient.

4.4 Exploring the SR Graph natively and efficiently with BEST2COP

The high density of the SR graph does not necessarily imply that exploring the latter directly is not viable. Indeed, the SR graph possesses characteristics that may be leveraged during the exploration to achieve competitive performance. Recall that within the SR graph, the number of segments is equal to the number of hops. The metric thus becomes far easier to manage, and the simpler relation of classic dominance can be used, reducing the number of

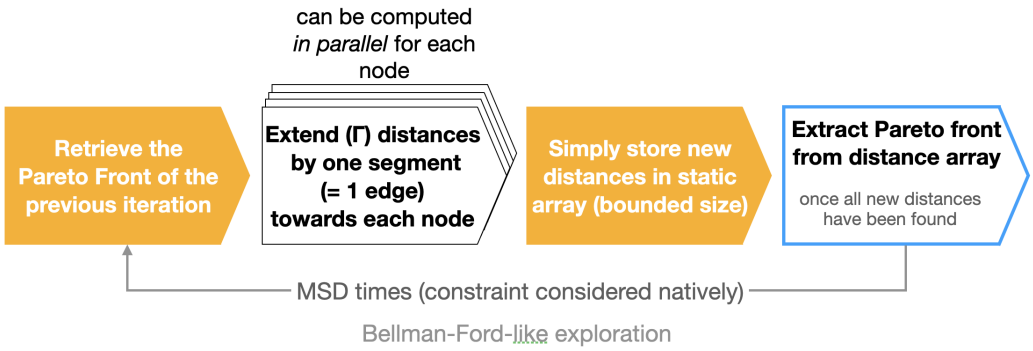


FIGURE 4.6: A simplified view of the BEST2COP algorithm.

distances to extend. Furthermore, by using a BFM-like exploration of the SR Graph, one can not only easily prune paths requiring more than MSD segments, but also benefit from efficient Pareto front management and multi-threading.

We designed our algorithm, BEST2COP, following this approach. These various features allow BEST2COP to efficiently solve both DCLC-SR and 2COP, despite the nature of the SR Graph. The implementation of BEST2COP is available online [8].

4.4.1 The general design of BEST2COP

BEST2COP's design is centered around two properties. First, the graph exploration is performed so that paths requiring i segments are found at the $i^{th}+1$ iteration, to natively tackle the MSD constraint. Second, BEST2COP's structure is easily parallelizable, allowing to benefit from multi-core architectures with low overhead. A simplified explanation of the way BEST2COP works can be seen in Fig. 4.6

The exploration scheme followed by BEST2COP is reminiscent of the one proposed by Corley and Moon [Corley & Moon 1985], or more generally, of BFM.

Simply put, at each iteration, BEST2COP starts by extending the known paths by one segment (one edge in the SR graph) in a Bellman-Ford fashion (a not-in-place version to be accurate, to benefit from the I-HOP property). When extended, the distances found during a given iteration are only checked loosely (and efficiently) for dominancy at first. This extension is performed in a parallel-friendly fashion that prevents data-races, allowing to easily parallelize our algorithm. Only once at the end of an iteration are the newly found distances filtered and thoroughly checked for dominance, to reflect the new Pareto front. The remaining non-dominated distances are in turn extended at the next iteration.

These steps only need to be performed $MSD \approx 10$ times, ignoring so all paths that are not deployable through SR. When our algorithm terminates, the results structure contains, for each segment number, all the distances of non-dominated paths from the source towards all destinations.

Note that interestingly, as the number of hops becomes a metric that must be considered, using a BFM-like exploration scheme now results in a label-setting approach, as distances

[8] https://github.com/talfroy/BEST2COP

discovered during previous iterations are necessarily non-dominated. This interesting consequence (as this type of exploration is usually used by label-correcting algorithms) is probably one of the reason behind the performance of our algorithm.

4.4.2 BEST2COP in more detail

BEST2COP's main procedure is shown in Alg. 5. The variable `pfront`, the end result returned by our algorithm, contains, for each iteration, the Pareto front of the distances towards each node `n`. In other words, `pfront[i]` contains, at the end of the i^{th} iteration, all non-dominated (M_1, M_2) distances of feasible paths towards each node `n`.

The variable `dist` is used to store, for each vertex, the best M_2-distance found for each M_1-distance to each node. Since the M_1-distance of any feasible path in G' is bounded by Γ, we can store these distances in a static array `dist[v]`. Note that during iteration i, `dist` will contain the Pareto front of the current iteration (non-dominated distances of i segments) in addition to distances that may be dominated. Keeping such paths in `dist` allows us to pre-filter paths before ultimately extracting the Pareto front of the current iteration from `dist` later on. This variable is used in conjunction with `pf_cand`, a boolean array to remember which distances within `dist` were found at the current iteration.

The variable `extendable` is a simple list that contains, at iteration i, all non-dominated distances discovered at iteration $i-1$. More precisely, `extendable` is a list of tuples (u, d_list), where d_list is the list of the best-known paths towards u. The variable `nextextendable` is a temporary variable allowing to construct `extendable`.

After the initialization of the required data structures, the main loop starts. This loop is performed MSD times, or until no feasible paths are left to extend. For each node v, we extend the non-dominated distances found during the previous iteration towards v (originally (0,0) towards `src`). Extending paths in this fashion allows to easily parallelize the main `for` loop (*e.g.*, through a single *pragma* Line 14). Indeed, each thread can manage a different node v towards which to extend the non-dominated paths contained within `extendable`. As threads will discover distances towards different nodes v (written in turn in structures indexed on v), this prevents data-races. Note that in raw graphs, this method may lead to uneven workloads, as not all paths may be extendable towards any node v. However, since an SR graph is (at least) complete, any path may be extended towards any node v, leading to similar workloads among threads.

The routine `ExtendPaths`, detailed in Alg. 6, takes the list of extendable paths, *i.e.*, non-dominated paths discovered at the previous iteration, and a node v. It then extends the extendable paths to u further towards v. The goal is to update `dist[v]` with new distances that may belong to the Pareto front. Before being added to `dist[v]`, extended distances go through a pre-filtering. Indeed, the newly found distance to v may be dominated or may be part of the Pareto front. While this check is performed thoroughly later, we can already easily prune some paths: if the new paths to v violate either constraint, there is no point in considering it. Furthermore, recall that `dist` stores, for all Γ M_1-distances towards a node, the best respective M_2-distance currently known. Thus, if the new M_2-distance is worse than the one previously stored in `dist` at the same M_1 index, this path is necessarily dominated and can be ignored. Otherwise, we add the distances to `dist` and update `pf_cand` to remember that a new distance that may be non-dominated was added during the current iteration. Note that `ExtendPaths` returns the number of paths updated within `dist`, as well

Algorithm 5: BEST2COP(G',src)

```
1  pfront := Array of size MSD
2  forall i ∈ [0..MSD] do
3      pfront[i] := Array of size |V| of Empty Lists
4  add (pfront[0][src], (0,0))
5  dist := Array of size |V|
6  forall n ∈ V do
7      dist[n] := Array of size Γ
8  dist[src][0] = (0,0)
9  optimal constrained extendable:= Empty List (of Empty Lists)
10 add (extendable, (src, [ (0,0) ]))
11 nextextendable:= Array of size |V| of Empty Lists
12 i:= 1, max_d1:= 0
13 while extendable ≠ [ ] and i ≤ MSD do
14     #pragma omp parallel for
15     forall v ∈ V do
16         pf_cand := Array of size Γ
17         nb, imax := ExtendPaths (v, extendable, pf_cand, dist[v])
18         max_d1 = max (imax, max_d1)
           // How to iterate on dist to get new PF
19         if nb log nb + nb + |pfront[i-1][v]| < max_d1 then
20             d1_it := mergesortd1 (pfront[i-1][v], pf_cand)
21         else
22             d1_it := [0...max_d1]
23         nextextendable[v] = [ ]
           // Extract new PF from dist
24         CptExtendablePaths (nextextendable[v], pfront[i][v], pf_cand, d1_it, dist[v])
       // Once each thread done, gather ext. paths
25     extendable = []
26     forall v ∈ V | nextextendable[v] ≠ [ ] do
27         add (extendable, (v,nextextendable[v]))
28     i = i + 1
29 return pfront
```

as the highest M_2-distance found. This operation is performed for efficiency reasons detailed here.

Once returned, `dist` contains distances either dominated or not. We thus need to extract the Pareto front of the current iteration. This operation is performed in a lazy fashion once for all new distances (and not for each edge extension). Since this Pareto front lies within `dist`, one can simply walk through `dist` by order of increasing M_1 distance from 0 to the highest M_1 distance found yet and filter all stored distances to get the Pareto front of the current iteration. This may not be effective as most of the entries of `dist` may be empty.

However, the precise indexes of all active distances that need to be examined (to skip empty entries) can be constructed by merging and filtering the union of the current Pareto front and the new distances (`pf_cand`). Thus, if the sorting and merging of the corresponding distance indexes is less costly than walking through `dist`, the former method is performed in order to skip empty or useless entries. Otherwise, a simple walk-through is preferred. The merging of the M_1 distances of the Pareto front and new M_1 distances is here showcased at high-level (Line 20, Alg. 5). The usage of more subtle data structures in practice allows performing this operation at the cost of a simple mergesort.

After the list of distances' indexes to check and filter is computed, the actual Pareto front is extracted during the `CptExtendablePaths` procedure, as shown in Alg. 7. This routine checks whether paths of increasing M_1 distance do possess a better M_2 distance than the one before them. If so, the path is non-dominated and is added to the Pareto front, as well as to the paths that are to be extended at the next iteration. Finally, once each thread is terminated, `nextextendable` contains $|V|$ lists of non-dominated distances toward each node. These lists are merged within `extendable`, to be extended at the next iteration.

Note that most approximations algorithms relying on interval partitioning or rounding do not bother with dominance checks. In other words, the structure they maintain is similar to our `dist`: the best M_2 distance for each M_1 distance to a given node. The latter may thus contain dominated paths that are considered and extended in future iterations. In contrast, by maintaining the Pareto front efficiently, we ensure to consider the minimum set of paths required to remain exact and thus profit highly from small Pareto fronts.

The output of BEST2COP. When our algorithm terminates, the `pfront` array contains, for each segment number, all the distances of non-dominated paths from the source s towards each destination d. To answer the 2COP problem, for each d and for all (stricter sub-)constraints $c'_0 \leq c_0$, $c'_1 \leq c_1$ and $c'_2 \leq c_2$, we can proceed as follows in practice:

- for $f(M_1, c'_0, \Gamma, c'_2, s, d)$, *i.e.*, to retrieve the distance from s to d that verifies constraints c'_0 and c'_2 minimizing M_1, we look for the *first* element in $\texttt{pfront}[c'_0 - 1][d]$ verifying constraint c'_2 (the first feasible distance is also the one minimizing M_1 because they are indexed on the latter).

- for $f(M_2, c'_0, c'_1, \infty, s, d)$, we look for the *last* element in $\texttt{pfront}[c'_0 - 1][d]$ verifying constraint c'_1. The path minimizing M_2 being, by design, the last element.

- to compute $f(M_0, \infty, c'_1, c'_2, s, d)$, let us first denote k the smallest integer such that $\texttt{pfront}[k][d]$ contains an element verifying constraints c'_1 and c'_2. The resulting image is then any of such elements in $\texttt{pfront}[k][d]$.

As one might notice, computing $f(M_j, c'_0, c'_1, c'_2, s, d)$, $j = 0, 1, 2$ cannot always be achieved

Algorithm 6: ExtendPaths(v, `extendable`, `pf_cand`, `dist_v`)

```
1  imax = 0, nb = 0
2  forall (u, d_list) ∈ extendable do
3      forall l ∈ E'(u,v) do
4          forall (d1u, d2u) ∈ d_list do
5              d1v = d1u + w1(l)
6              d2v = d2u + w2(l)
               // Filters:  constraints and dist
7              if d1v ≤ c1 and d2v ≤ c2
8              and d2v < dist_v[d1v] then
9                  dist_v[d1v] = d2v
10                 if not pf_cand[d1v] then
11                     nb ++
12                     pf_cand[d1v] = True
13                 if d1v > imax then
14                     imax = d1v
15 return nb, imax
```

Algorithm 7: CptExtendablePaths (`nextextendable_v`, `pfront_iv`, `pf_cand`, `d2_it`, `dist_v`)

```
1  last_d2 = ∞
2  forall d1 ∈ d1_it do
3      if dist_v[d1] < last_d2 then
4          add (pfront_iv, (d1,d2))
5          last_d2 = dist_v[d1]
6          if pf_cand[d1] then
7              add (nextextendable_v, (d1,d2))
```

in constant time (for $j = 0$ and sub-constraints in particular). Indeed, we favor a simple data structure. A search in an ordered list of size Γ is needed for stricter constraints (and may be performed $\log(MSD)$ times when optimizing M_0). To improve the time efficiency of our solution, each $\texttt{pfront}[i][d]$ may be defined as or converted into a static array in the implementation.

Finally, for simplicity, we did not show in our pseudo-code the structure and operations that store and extend the lists of segments. In practice, we store one representative of the best predecessors and a posteriori retrieve the lists using a backward induction for each destination.

Several aspects allow BEST2COP to reach good performance. First, the fact that paths requiring more than MSD segments are natively excluded from the exploration space. Second, well-chosen data structures benefit from the limited accuracy of the delay measurements to limit the number of paths to extend. This allows manipulating arrays of fixed size because the Pareto front of distances towards each node is limited to Γ at each step (enabling very efficient read/write operations). Third, using a Bellman-Ford approach allows not only to easily parallelize our algorithm but also to perform lazy efficient updates of the Pareto front. Indeed, a newly found path may only be extended at the *next* iteration. Thus, we can efficiently extract the non-dominated paths from all paths discovered during the current iteration in a single pass, once at the end of the iteration. Conversely, other algorithms tend to either check for dominancy whenever a path is discovered (as the latter may be re-extended immediately), or not bother to check for dominancy at all, *e.g.*, by relying solely on interval partitioning to limit the number of paths to extend.

4.4.3 For Massive Scale, Multi-Area Networks

Despite these aspects, the design of BEST2COP implies a dominant factor of $|V|^2$ in terms of time complexity[9] (the SR graph being complete), recent SR deployments with more than 10 000 nodes would not scale well enough. The sheer scale of such networks, coupled with the inherent complexity of TE-related problems, makes 2COP very challenging if not impossible to practically compute at first glance. In fact, BEST2COP originally exceeds 20s when dealing with ≈15 000 nodes.

In this section, we describe how we extend BEST2COP in order to deal efficiently with such massive-scale networks. By leveraging the physical and logical partitioning usually performed in such networks, we manage to solve 2COP in ≈ 1s even in networks of 100 000 nodes.

4.4.3.1 Scalabity in Massive Network & Area decomposition

The scalability issues in large-scale networks do not arise solely when dealing with TE-related problems. Standard intra-domain routing protocols encounter issues past several thousands of nodes. Naive network design creates a large, unique failure domain resulting in numerous computations and message exchanges, as well as tedious management. Consequently, networks are usually divided, both logically and physically, in *areas*. This notion exists in both major intra-domain routing protocols (OSPF and IS-IS). In the following, we consider the

[9] The detailed complexity is given in section 4.4.4.1

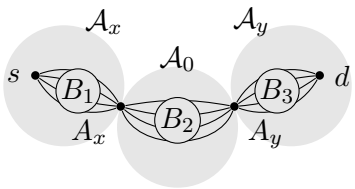


FIGURE 4.7: The set of solutions across areas is obtained from the cartesian product of the solutions in each area.

standard OSPF architecture and terminology but our solution can be adapted to fit any one of them.

Areas can be seen as small, independent sub-networks (usually of around 100 - 1000 nodes at most). As explained in Section 4.4.3, within OSPF, routers within an area maintain a comprehensive topological database of their own area only. Stub-areas are centered around the *backbone*, or area 0. *Area Border Routers*, or ABRs, possess an interface in both the backbone area and a stub area. Being at the intersection of two areas, they are in charge of sending a summary of the topological database (the best distance to each node) of one area to the other. There are usually at least two ABRs between two areas [10]. Summaries of a non-backbone area are sent through the backbone. Upon reception, ABRs inject the summary within their own area. In the end, all routers possess a detailed topological database of their own area and the best distances toward destinations outside their own area.

4.4.3.2 Leveraging Area Decomposition

This partitioning creates obvious separators within the graph, the ABRs. Thanks to the latter, we can leverage this native partition in a similar divide-and-conquer approach, adapted to the computation of 2COP paths, by running BEST2COP at the scale of the areas before exchanging and combining the results. We do not only aim to reduce computation time, but also to keep the number and size of the exchanged messages manageable.

We now explain how we perform this computation in detail. For readability purposes, we rely on the following notations: $\mathcal{A}_x$ denotes area x. A_x denotes the ABR between the backbone and $\mathcal{A}_x$. When necessary, we may distinguish the two ABR $A1_x$ and $A2_x$. Finally, b2cop$(\mathcal{A}_x, s, d)$ denotes the results (the non-dominated paths) from s to d within $\mathcal{A}_x$. When d is omitted, we consider all routers within $\mathcal{A}_x$ as destination. Figure 4.7 illustrates a network with three areas, x, y and 0, the backbone area.
We here chose to detail a simple distributed and router-centric variant of our solution. However, our solution may well be deployed in other ways, e.g. relying on controllers, or even a single one. In such cases, the computation could be parallelized per area if needed. Such discussion is left for future work.

Due to the area decomposition, routers do not possess the topological information to compute a full, complete SR graph of the whole network. Thus, we make routers only compute the SR graph of their own area(s). Because exchanging the SR graphs themselves

[10] We here (and in the evaluation) consider two ABRs, but the computations performed can be easily extended to manage more ABRs.

implies a large volume of information to share, we instead make the ABRs exchange their 2COP paths (i.e., the non dominated paths to all destinations of their areas) since we limit their numbers to Γ at worst. This exchange still provides enough information for all routers to compute all 2COP paths for every destination.

More formally, each ABR A_x computes b2cop($\mathcal{A}_x, A_x$) and exchange the results with $A_y, \forall y \neq x$. Areas being limited to a few hundred routers on average, this computation is very efficient. Note that ABRs also compute b2cop($\mathcal{A}_0, A_x$), but need not exchange it, as all ABRs perform this computation. Exchanging the computed 3D Pareto front has a message complexity of $|V| \times c_0 \times \Gamma$ at worst in theory. In practice, we expect both the size of Pareto fronts and the number of relevant destinations to consider to be fairly low ($<< \Gamma$ and $<< |V|$ resp.). In the case of non-scalable Pareto fronts, one can opt for sending only part of them but at the cost of relaxing the guarantees brought by BEST2COP.

After exchanging messages, any ABR A_x should know the non-dominated paths from itself to $A_y, \forall y \neq x$, and the non-dominated paths from A_y to all nodes within $\mathcal{A}_y$. By combining this information, we can compute the non-dominated paths from A_x to all nodes within $\mathcal{A}_y$, as we will now detail.

Since ABRs act as separators within the graph, to reach a node within a given area $\mathcal{A}_y$, it is necessary to go through one of the corresponding ABRs A_y. It thus implies that non-dominated paths to nodes within $\mathcal{A}_y$ from A_x can be found by combining bcop($\mathcal{A}_0, A_x, A_y$) with bcop($\mathcal{A}_y, A_y$). In other words, by combining, with a simple cartesian product, the local non-dominated paths towards the ABRs of a given zone with the non-dominated paths from said ABRs to nodes within the corresponding distant areas, one obtains a superset of the non-dominated paths towards the destinations of the distant area. In practice, since several ABR can co-exist, it is necessary to handle the respective non-dominated paths (bcop($\mathcal{A}_y, A1_y$) and bcop($\mathcal{A}_y, A2_y$)) with careful comparisons to avoid incorrect combinations.

To ensure that the results obtained through the cartesian product aforementioned are correct, some post-processing is required. When combining segment lists, the latter are simply concatenated. More precisely, the resulting segment list necessarily possesses the following structure: $(u_0, u_1)|\ldots|(u_i, A)|(A, v_0)|\ldots|(v_{j-1}, v_j)$, with A denoting an ABR. However, A being a separator, it is likely that the best IGP path from u_i to v_0 natively goes through A without the need of an intermediary segment. Thus, segments of the form $(u_i, A)|(A, v_0)$ can often be replaced by a single segment (u_i, v_0). Such anomalies should be corrected, as an additional useless segment may render the path falsely unfeasible, even though it actually fits the MSD constraint. This correction can be performed easily. Let $A1$ be the separator, if $(u_i, A1)$ and $(A1, v_0)$ are node segments, and all best IGP paths from u_i to v_0 go through $A1$ (or possess the same cost and delay as the best IGP ones going through $A2$), the two node segments can be replaced by a single one.

This correction is performed quickly and relies solely on information available to the router (the local SR graph and the received distances summary). Finally, after having performed and corrected the cartesian products for all the ABRs of the area, the latter are merged in a single Pareto front.

Once performed for all areas, an ABR A_x now possesses all 2COP paths to all considered destinations within the network. These can then be sent to routers within $\mathcal{A}_x$, who will

need to perform similar computations to compute non-dominated paths to all routers within a different area. Note that the 2COP paths for each destination can be sent as things progress, so that routers can process such paths progressively (and in parallel) if needed.

4.4.4 A Limited Complexity with Strong Guarantees

4.4.4.1 An Efficient Polynomial-Time Algorithm

The flat BEST2COP. In the worst-case, for a given node v, there are up to $degree(v) \times \Gamma$ paths that can be extended towards it. Observe that $degree(v)$ is at least $|V|$ (because G' is complete) and depends on how many parallel links v has with its neighbors. With L being the average number of links between two nodes in G', on average we thus have $degree(v) = |V| \times L \times \Gamma$ paths to extend to a given node, at worst. These extensions are performed for each node v and up to MSD times, leading to a complexity of

$$O(c_0 \cdot \Gamma \cdot |V|^2 \cdot L)$$

Using up to $|V|$ threads, one can greatly decrease the associated computation time. Note that, while the algorithm is easily parallelizable with regular loads between cores, the exhibited speedup ultimately depends on the underlying hardware characteristics and the difficulty of the problem instance. As such, while we will see that the speedup observed is significant, it is unlikely that the theoretical (quasi-linear) speedup can be reached in practice.

The Cartesian Product. Its complexity is simply the size of the 2COP solution space squared, for each destination, thus at worst $O((c_0 \cdot \Gamma)^2 \cdot |V|)$. Note that we can reach a complexity of $O(c_0^2 \cdot \Gamma^2)$, again with the use of $|V|$ threads since each product is independent. This worst case is not expected in practice as metrics are usually mostly aligned to result in Pareto fronts whose maximal size is much smaller than $c_0 \cdot \Gamma$.

Overall, `BEST2COP-E` (multi-area) exhibits a complexity of

$$O\left(c_0 \cdot \Gamma \cdot \left(c_0 \cdot \Gamma + L \cdot max_{\forall i \in [1..m]}(|V_i|)\right)\right)$$

with V_i denoting the set of nodes in each area i (m being their number) and the use of $|V|$ threads and sufficient CPU resources (this bound is achievable ideally because the load is perfectly balanced and bottlenecks negligible). Note that the cartesian product dominates this worst-case analysis as long as the product $V_i \cdot L$ remains small enough. However, with realistic weighted networks, we argue that the contribution of the Cartesian product is negligible in practice, so BEST2COP-E is very scalable for real networking cases.

4.4.4.2 What are the Guarantees one can expect when the Trueness exceeds the Accuracy, *i.e.*, if $t > \gamma$?

If propagation delays are measured with a really high trueness (*e.g.*, with a delay grain of 1 μs or less), BEST2COP (and so, BEST2COP-E) can either remain exact but slower, or, on the contrary, rapidly produce approximated results. In practice, if one prefers to favor performance by choosing a fixed discretization of the propagation delay (to keep the computing

time reasonable rather than returning truly exact solutions), this may result in an array not accurate enough to store all non-dominated delay values, *i.e.*, two solutions might end up in the same cell of such an array even though they are truly distinguishable. Nevertheless, we can still bound the margin errors, relatively or in absolute, regarding constraints or the optimization objective of the 2COP variant one aims to solve.

In theory, note that while no exact solutions remain tractable if the trueness of measured delays is arbitrarily high (for worst-case DCLC instances), it is possible to set these error margins to extremely small values with enough CPU power. If $t < \gamma$, each iteration of our algorithm introduces an absolute error of at most $\frac{1}{\gamma}$ for the M_1 metric, *i.e.*, the size of one cell in our array (recall that $\gamma = \frac{\Gamma}{c_1}$ is the accuracy level and is the inverse of the delay grain of the static array used by BEST2COP). So our algorithm may miss an optimal constrained solution p^* (for a given destination) only if there exists another solution p such that $d_1(p) \geq d_1(p^*)$ but the M_1 distance of both solutions associated with the same integer (that is stored in the same cell of the `dist` array) *i.e.*, only if $d_1(p) \leq d_1(p^*) + \frac{c_0}{\gamma}$. In this case, we have $d_2(p) \leq d_2(p^*)$ because otherwise, p^* would have been stored instead of p. From this observation, depending on the minimized metric, BEST2COP ensures the following guarantees.

If one aims to minimize M_0 or M_2 (*e.g.*, when solving DCLC), then BEST2COP guarantees a solution p that optimizes the given metric, but this solution might not satisfy the given delay constraint $c \leq c_1$. As an example, for DCLC-SR (optimizing M_2), we have

$$d_0(p) \leq c_0$$
$$d_1(p) < c + \frac{c_0}{\gamma}$$
$$d_2(p) \leq d_2(p^*)$$

With p^* denoting the optimal constrained solution. When minimizing M_1, the solution returned by BEST2COP for a given destination, p, will indeed verify the constraints on M_0 and M_2, and we have $d_1(p) < d_1(p^*) + \frac{c_0}{\gamma}$. The induced absolute error of c_0/γ regarding the delay of paths becomes negligible as the delay constraint increases. If $c \approx c_1$, the latter translates to a small relative error of c_0/Γ. Conversely, it becomes significant if $c << c_1$. When minimizing M_0 or M_2, it is thus recommended to set c_1 as low as possible regarding the relevant sub-constraint(s) $c \leq c_1$ if necessary. Similarly, to guarantee a limited relative error when minimizing M_1, it is worth running our algorithm with a small c_1 as we can have $d_1(p^*) << c_1$. However, note that this later and specific objective (in practice less interesting than DCLC in particular) requires some a priori knowledge, either considering the best delay path without any c_2 and c_0 constraints, or running twice BEST2COP to get $d_1(p)$ as a first approximation to avoid set up c_1 blindly initially (here c_1 is not a real constraint, only c_2 and c_0 apply as bounds of the problem, c_1 just represents the absolute size of our array and, as such, the accuracy one can achieve).

For other variants, like DCCLS (*i.e.*, $2COP(s, 10, c_1, c_2, M_0)$), the guarantee turns out to be similar. The discretized delay is also constrained, leading to identical guarantees on the latter. Since BEST2COP prefers paths of lower IGP cost, the paths found are bound to respect any feasible constraint on the IGP cost. Finally, regarding the number of segments, we have $d_0(sp') = d_0(sp)$ as the whole M_0 dimension is visited, regardless of the delay accuracy.

The CCLD-SR variant (*i.e.*, $2COP(s, c_0, c_1, \infty, M_1)$) is different, as the discretized value is here optimized. The guarantee is thus absolute for the M_1 metric (propagation delay). Formally, for each destination, BEST2COP returns sp' that verifies at worst: $d_1(sp') < d_1(sp) + MSD/t$, $d_2(sp') < c_2$ and $d_0(sp') < c_0$, where sp is an optimal solution.

Focusing back on DCLC, even though BEST2COP exhibits strong and tunable guarantees, it may not return exact solutions once two paths end up in the same delay cell, which may happen even with simple instances exhibiting a limited Pareto front. Fortunately, a slight tweak in the implementation is sufficient to ensure exact solutions for such instances. Keeping the original accuracy of M_1 distances, one can rely on truncated delays only to find the cell of each distance. Then, one possible option consists of storing up to k distinct distances in each cell[11]. Thus, some cells would form a miniature, undiscretized Pareto front of size k when required. This trivial modification allows the complexity to remain bounded and predictable: as long as there exists less than k distances within a cell, the returned solution is exact. Otherwise, the algorithm still enforces the aforementioned guarantees. While this modification increases the number of paths we have to extend to $k \cdot \Gamma$ at worst, such cases are very unlikely to occur in average. Notably, our experiments show that 3D Pareto fronts for each destination contain usually less than ≈ 10 elements at most on realistic topologies, meaning that a small k would be sufficient in practice. In summary, BEST2COP is efficient and exact to deal with simple instances and/or when $t \geq \gamma$, while it provides approximated but bounded solutions for difficult instances if $t < \gamma$ to remain efficient and so scalable even with massive scale IP networks.

4.5 Evaluating BEST2COP, LCA, and SR

In this section, we evaluate the computation time of our solution. However, as both BEST2COP and LCA are able to compute DCLC paths and their minimal segment lists, we first use these algorithms to evaluate whether SR is an adequate technology to deploy DCLC paths.

4.5.1 DCLC paths can indeed be deployed through SR

Given the MSD constraint, one may question the choice of SR for deploying DCLC paths in practice. Indeed, in some cases, in particular if the metrics are not aligned[12], constrained paths may require more than MSD detours to satisfy a stringent latency constraint.

While it has been shown that few segments are required for most current SR usages (*e.g.*, for FRR or when considering mono-metric paths such as least-delay ones) [Filsfils 2020, Aubry 2020], to the best of our knowledge, there is no similar study for our specific use-case, *i.e.*, massive scale networks with two valuation functions (delay and IGP cost). This is probably one of the most exciting challenge for SR as DCLC is a complex application. Since such massive-scale computer network topologies are not available publicly, we rely on our

[11] In practice, note that several implementation variants are possible whose one consists of using the array only when the stored Pareto front exceeds a certain threshold. Moreover, k can be set up at a global scale shared for all cells or even all destinations, instead of a static value per cell, to support heterogeneous cases more dynamically. These approaches were also evoked by Song and Sahni et al. [Song & Sahni 2006]

[12] The delay and the IGP costs in particular. Since node segments represent best IGP paths, the IGP cost and the number of segments will most likely be aligned by design

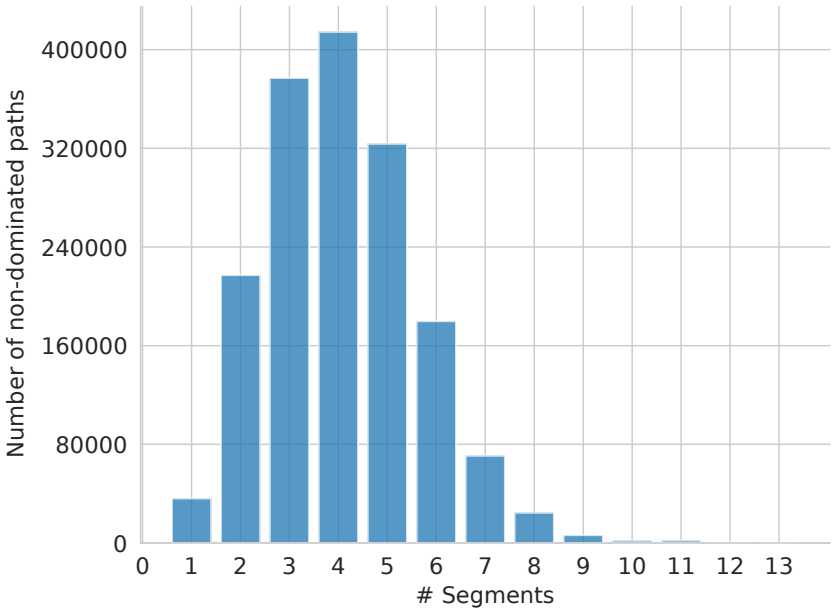


FIGURE 4.8: Required number of segments for all DCLC solutions, in a network of 45 000 nodes generated by YARGG, with delay constraints of up to 100ms.

own topology generator (presented in the following section) to perform this study. For this analysis, we opt for a worst-case graph having $\approx 45\,000$ nodes and $\approx 92\,000$ edges scattered in 140 areas.

For this analysis, we keep track, for each destination, of all the solutions solving DCLC for all delay constraints up to 100ms, and extract the necessary number of segments. In other words, we show the number of segments required to encode all non-dominated (and thus practically relevant for some given constraint) paths, considering all delay constraints up to 100ms. The results are shown in Fig. 4.8.

One can see that most paths require less than 10 segments, meaning that performant hardware should be able to deploy most DCLC paths. However, some corner-cases requiring more than 10 segments do exist, probably arising from stringent delay constraints. In addition, less performant hardware (*e.g.*, with MSD ≈ 5), while able to deploy the majority of DCLC paths, can not deploy *any* DCLC path. Note that several mechanisms discussed in Section 2.4.2 exist to bypass this limit, such as Flexible Algorithms [Psenak *et al.* 2020] and Binding Segments [Filsfils *et al.* 2018]. However, both techniques increase the message exchange, number of states to maintain, and overall complexity. Their usage should thus be limited to a few corner cases.

Consequently, our analysis exhibits two main points. First, SR is appropriate to deploy TE paths. Indeed, the majority of DCLC paths should be deployable within the MSD constraint, if not all when using performant hardware. Second, since there may however exist DCLC paths requiring more than MSD segments, this limit must be considered to compute feasible paths correctly. Otherwise, a single non-feasible path dominating feasible ones is enough to lead to an incorrect algorithm. The underlying path computation algorithm must then efficiently consider delay, IGP cost and the number of segments to ensure its correctness.

4.5.2 Performance Evaluation

We now evaluate the computation time of our solutions. We start by evaluating BEST2COP, which explores the SR Graph directly on various flat network instances, ranging from worst-case scenarios to real topologies. We then compare these results to another existing algorithm SAMCRA, modified to benefit from (both) our methods used to consider the segment metric. Then, after having introduced our multi-area topology generator, we evaluate the extended variant of BEST2COP-E, on massive scale networks.

Recall from Chapter 3 that SAMCRA is a multi-metric algorithm able to deal with more than two metrics and can thus be used to solve DCLC-SR through the SR Graph or LCA. SAMCRA is a label-setting Dijkstra-like algorithm that relies on a priority queue indexed according to a non-linear aggregated metric (referred to as LENGTH), and which extends all non-dominated distances.

The delays fed to SAMCRA are *also* discretized in the same fashion as for BEST2COP, allowing the number of non-dominated distances that SAMCRA has to consider to be bounded by Γ. Consequently, SAMCRA is also able to rely on optimized static structures to store and manage the Pareto front [13]. Overall, SAMCRA has to consider (at worst) $MSD \times \Gamma \times |V|$ distances at most within the queue [14], meaning that extracting the distance with minimal LENGTH bears a complexity of $O(MSD \times \Gamma \times |V| \times (log(MSD \times \Gamma \times |V|)))$. Each time a new edge (u, v) is considered, the $MSD \times \Gamma$ distances of node u must be extended, and checked for dominance against the $MSD \times \Gamma$ distances of node v, leading to a complexity of $(MSD \times \Gamma^2 \times E) \times log(MSD \times \Gamma)$ when relying on a binary heap, or $(MSD \times \Gamma^2 \times E)$ when relying on a Fibonacci heap [15]. Overall, the theoretical complexity of SAMCRA (when exploring the SR Graph) is either (when relying on a binary heap)

$$O(MSD \times \Gamma \times |V| \times (log(MSD \times \Gamma \times |V|)) + (MSD \times \Gamma^2 \times E) \times log(MSD \times \Gamma))$$

or

$$O(MSD \times \Gamma \times |V| \times (log(MSD \times \Gamma \times |V|)) + (MSD \times \Gamma^2 \times E))$$

when relying on a Fibonacci Heap.

As SAMCRA is not designed with the SR Graph in mind, it is difficult to know whether it is best to make the algorithm explore the SR Graph directly, or rely on LCA. Thus, we compare ourselves to both variants. First, we run SAMCRA on the fully-meshed SR Graph, which allows using the SAMCRA algorithm nearly as-is (by adding the number of hops with a constraint of MSD as a metric considered by the LENGTH). We call this variant SAMCRA+*SR Graph* (SRG). Second, we implement LCA on top of SAMCRA. This method requires however further modification of the SAMCRA algorithm, not only by adding the conversion algorithm but also by extending its dominance checks to strong dominance ones.

[13] As SAMCRA does now rely on static structure of size Γ, its design is arguably closer to its heuristic variant, TAMCRA (which extended only k path per node) with $k = \Gamma$. We here use the name SAMCRA to remind that a k value of Γ enables the algorithm to be exact.

[14] This complexity considers the direct exploration of the SR Graph. When relying on LCA, the number of non-*strongly*-dominated distances to extend may reach $\Gamma \times |V|$ *per node*, although this high bound is very unlikely to be reached.

[15] In practice, we have observed that the binary heap leads to similar, if not better results. We thus rely on a binary heap in our evaluation.

We refer to this variant as SAMCRA+LCA [16].

We thus re-implemented SAMCRA in C [17] as described. Note that our implementation of SAMCRA (both SAMCRA+SRG and SAMCRA+LCA) is purely sequential. While it may be possible to parallelize some inner loops (or the outer ones by adapting methods used to parallelize the Dijkstra algorithm [Crauser *et al.* 1998b]), doing so raises several challenges to verify the correctness and actual efficiency of the resulting algorithm in our context.

In the following, we consider our discretization to be exact (*i.e.*, Γ is high enough to prevent loss of relevant information) and consider :

- $c_0 = MSD = 10$, as it is close to the best hardware limit;
- $L = 2$: while some pairs of nodes may have more than two parallel links connecting them in G, we argue that, on average in G', one can expect that the total number of links in E' is lower than $2|V|^2$.
- $\Gamma = 1000$, although this value is tunable to reflect the expected product trueness-constraint on M_1, we consider here a fixed delay grain of 0.1ms (so an accuracy level of $\gamma = 10$) regarding a maximal constraint $c_1 = 100$ms. This Γ limitation is realistic in practice and guarantees the efficiency of BEST2COP even for large complex networks as it becomes negligible considering large $|V|$.

All our experiments are performed on an Intel(R) Core(TM) i7-9700K CPU @ 3.60GHz × 8. We measure the execution time starting from the allocation of the structures, up to the return of the results.

4.5.2.1 Computing Time & Comparisons for Flat Networks

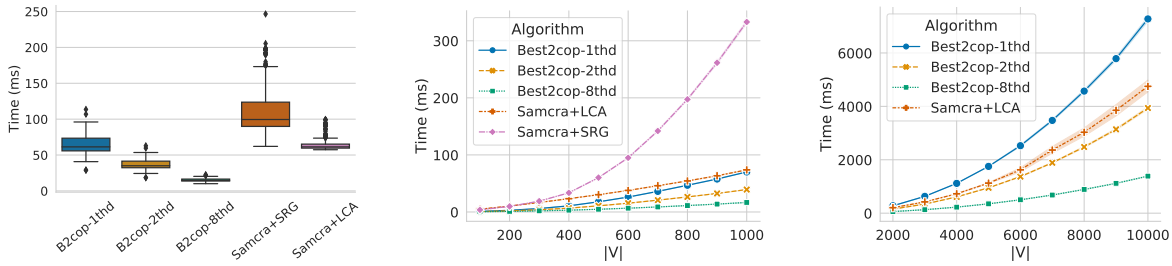
This section illustrates the performance of our BEST2COP and SAMCRA (both SAMCRA+LCA and SAMCRA+SRG) using three flat network scenarios: real networks, random networks, and large-scale random networks. Note that we do not take advantage of any area decomposition to mitigate the computing time.

Real networks. We start by considering a real IP network topology. We use our largest available ISP topology, consisting of more than 1100 nodes and 4000 edges. This topology describes the network of a Tier-1 operator and is not available to the public [18]. While the IGP costs of each link were available, we do not have their respective delays. We thus infer delays thanks to the available geographical locations we do possess: we set the propagation delays as the orthodromic distances between the connected nodes divided by the speed of light, and run both algorithms on the obtained topology. The execution times are then

[16] While we investigated whether a BEST2COP-like exploration scheme would lead to interesting performance when relying on LCA (and exploring the original graph), preliminary results did not seem promising. This may be because, by not exploring the original graph, BEST2COP becomes a label-correcting algorithm. Indeed, as the number of hops becomes (at least partially) irrelevant, BEST2COP may extend dominated distances. Coupled with the strong dominance requirement, which requires extending distances that may be dominated as well, this could lead BEST2COP to consider too many irrelevant distances. However, further investigation considering various graph instances is yet to be done.

[17] https://github.com/talfroy/BEST2COP

[18] While public topology datasets exist, these topologies are often too small for our use-case and/or do not possess any link valuation.



(A) Computation time of BEST2COP (1, 2, 8 threads) & SAMCRA on a real network with real link valuation.

(B) Computation time of BEST2COP & SAMCRA on random networks with random valuation for $|V| = 100$ to 1000

(C) Computation time of BEST2COP & SAMCRA on random networks with random valuation for $|V| = 1000$ to 10 000

FIGURE 4.9: Computation time of BEST2COP and SAMCRA on various experiments. Although the results can be close when considering mono-threaded BEST2COP and SAMCRA, BEST2COP outperforms SAMCRA when using multi-threading.

shown in Fig. 4.9a. BEST2COP (1, 2, 8 threads) and SAMCRA (with LCA and SRGs) are run for every node as source, resulting in the distributions showcased.

One can see that SAMCRA+SRG (*i.e.*, SAMCRA run directly on the SR Graph) exhibits the worst execution times out of all the algorithms and variants presented, averaging at 100ms, and reaching 250ms at worst. Interestingly, this shows that exploring the SR Graph itself may be detrimental to some algorithms (in particular priority-queue-based ones) due to its high density. Hence, algorithms not designed to take advantage of its features may fare better by exploring the original, sparser topology, and using the information within the SR Graph to compute the number of necessary segments to encode the paths being explored. This is visible on SAMCRA+LCA computation times. Our construct, coupled with our conversion algorithm, allowed SAMCRA+LCA to reach computation times very similar to the mono-threaded variant of BEST2COP, with an average execution time of $\approx$ 60ms. Note that BEST2COP, which runs on the SR Graph itself, shows equivalent execution time when relying on a single thread. However, when relying on multiple threads, BEST2COP outperforms SAMCRA in all runs, reaching a computation time of $\approx$ 25ms at worst when using 8 threads, *i.e.*, three times faster than SAMCRA.

These low execution times are not only due to the efficiency of the algorithms presented, but also to the realistic link valuations, which tend to be correlated in practice. In realistic cases, BEST2COP can thus work with $\Gamma > 1000$ and so with a supported accuracy $t >> 0.1$ms (to deal with a micro-second grain) for small enough delay constraint (*i.e.*, $<<$ 100ms), while keeping the execution time in the hundreds of milliseconds. One may notice that (almost) perfectly aligned metrics reduce the usefulness of any DCLC-like algorithm, but such metrics are not always aligned for all couples in practice (even with realistic cases, we observe that the average size of the 3D Pareto front is strictly greater than 1, typically ≈ 4). Our algorithm deals efficiently with easy cases and remains exact[19] and efficient for more complex cases, *e.g.*, with random graphs.

[19] Or at least near exact for difficult instances having both high trueness and exponential increasing Pareto fronts.

Random networks The number of publicly available large topologies being limited, we continue our evaluation with random scenarios to assess the computation time of the aforementioned algorithms on a larger number of scenarios.

We generate raw connected graphs of $|V|$ nodes by using the Erdos-Rényi model. The generated topologies have a degree of $log(|V|)$. Both the delays and the IGP weights are picked uniformly at random. IGP weights are chosen within the interval $[1, 2^{32}/|V|/10]$, to ensure that no paths possesses a cost higher than 2^{32}. Delays are chosen within the interval $[0, 0.01 \times c_1]$, with $c_1 = 100ms$, to ensure that a high number of feasible paths exist.

We start by running BEST2COP and SAMCRA for $|V|$ ranging from 100 to 1000 (with steps of 100). To account for the randomness of both valuation functions, we generate 30 differently weighted distinct topologies for each value of $|V|$. We run BEST2COP and SAMCRA for 30 nodes selected as representative sources (randomly picked uniformly). Computing times are shown in Fig 7.5b.

While the computation times are slightly higher (due to the random valuations which lead to a higher number of non-dominated paths), the results are similar to the previous experiment. These results display more clearly that SAMCRA does not benefit from exploring the SR Graph. Indeed, on random networks, SAMCRA+SRG is about 7 times slower than the other algorithms displayed. However, as on real networks, SAMCRA+LCA shows results close (if not equal) to BEST2COP execution time. Nevertheless, even on random networks, BEST2COP remains three times faster than SAMCRA when relying on 8 threads.

Interestingly, BEST2COP mono-threaded and SAMCRA+LCA computation times get closer as $|V|$ increases. Thus, we continue our comparison on networks of 2000 to 10 000 nodes. Given the long computation times of SAMCRA+SRG, we here only consider SAMCRA+LCA. The results are shown in Fig. 4.9c. On such networks, BEST2COP (mono-threaded) exhibits an execution time of $7s$, while SAMCRA+LCA remains under $5s$. The quadratic complexity of BEST2COP (whose main factor is $|V|^2$) is here clearly visible. SAMCRA+LCA exhibits a less steep growth. However, when relying on multiple-thread, BEST2COP remains far more efficient. While two threads already allow reaching an execution time slightly lower than SAMCRA (4s), 8 threads allow BEST2COP to remain ≈ 3.3 times faster than SAMCRA.

The way to use the SR Graph has a high impact on the underlying algorithm. As the SR Graph is not at the core of SAMCRA's design, exploring the latter leads to high execution time due to its density. However, adding our conversion algorithm (which relies on the SR Graph data) within SAMCRA allowed the latter to reach competitive execution times while solving 2COP. BEST2COP, which explores the SR Graph directly, exhibits an execution similar to SAMCRA+LCA when relying on a single thread. When using multi-threading, BEST2COP outperforms its SAMCRA in all scenarios.

In any case, it is interesting to note that even BEST2COP takes more than one second on networks of 10 000 nodes. To showcase the performance of our contribution on massive-scale networks, we now evaluate the execution time of its extension, BEST2COP-E, which supports and leverages OSPF-like area division. This version is adapted to tackle TE problems in massive-scale hierarchical networks. In the following, we only consider our approach. Indeed, the latter showcased better performance than SAMCRA even without relying on multiple threads when considering a topology size $|V| < 1000$, which encompasses the size of standard

OSPF areas.

However, before continuing to analyze the computing time results, we first introduce our generator for massive-scale, multi-areas, realistic networks having two valuations (IGP cost & delays).

4.5.2.2 Massive Scale Topology Generation

To the best of our knowledge, although such networks exist in the wild, there are no massive scale topologies made *publicly* available which exhibit IGP costs, delays, and area subdivision. For example, the graphs available in the topology zoo (or sndlib) datasets do not exceed 700 nodes in general. Moreover, the ones for which the two metrics can be extracted, or at least inferred, are limited to less than 100 nodes. Thus, at first glance, performing a practical massive-scale performance evaluation of BEST2COP-E is highly challenging if not impossible. There exist a few topology generators [Quoitin *et al.* 2009, Medina *et al.* 2001] able to generate networks of arbitrary size with realistic networking patterns, but specific requirements must be met to generate topologies onto which BEST2COP-E can be evaluated, in particular the need for two metrics and the area decomposition.

Topology generation requirements. First, the experimental topologies must be large, typically between 10 000 and 100 000 nodes. Second, they must possess two valuation functions as realistic as possible, one for the IGP cost and the other modeling the delay. Third, since the specific patterns exhibited by real networks impact the complexity of TE-related problems, the generated topologies must possess realistic structures (*e.g.*, with respect to redundancy in the face of failures in particular). Finally, for our purposes, the topology must be composed of different areas centered around a core backbone, typically with two ABRs between each to avoid single points of failure. Since we do not know any generator addressing such requirements, we developed YARGG, a python tool(Code available online [20]) which allows one to evaluate its algorithm on massive-scale realistic IP networks. In the following, we describe the generation methods used to enforce the required characteristics.

High-level structure. One of the popular ISP structure is the three-layers architecture [Cisco Networks 2008], illustrated in Fig. 4.11. The *access layer* provides end-users access to the communication service. Traffic is then aggregated in the *aggregation layer*. Aggregation routers are connected to the *core routers* forming the last layer. The aggregation and access layers form an area and usually cover a specific geographical location. The core routers, the ABRs connecting the backbone other areas, and their links, form the backbone area that interconnect the stub areas, *i.e.*, the aggregation and access layers of the different geographical locations. Core routers are ABRs and belong both to a sub-area, per couple of 2 for redundancy. Thus, while the access and aggregation layers usually follow standard structures and weight systems recommended by different network vendors, the backbone can vastly differ among different operators, depending on geographical constraints, population distribution, and pre-existing infrastructure. Taking these factors into account, YARGG

[20] https://github.com/JroLuttringer/YARGG

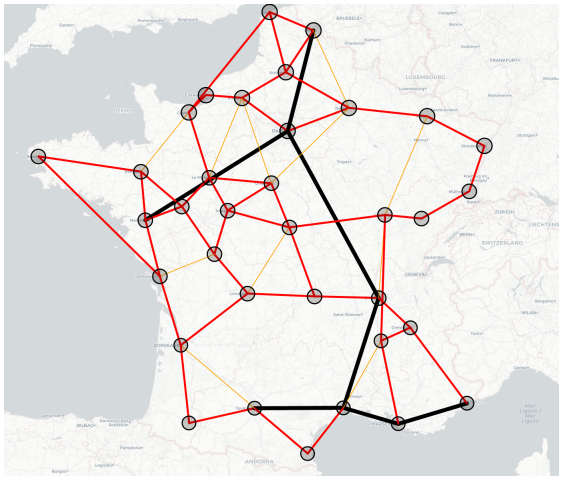


FIGURE 4.10: Core network (before step 5) generated by YARGG in France. While we consider the road distances, we represent the links in an abstract fashion for readability purposes. The color and width of the links represent their bandwidth (and thus their IGP costs).

generates large networks by following this 3-layer model, given a specific geographical location (*e.g.*, a given country).

Generating the core network and the areas. YARGG is a heuristic that generates the core network by taking the aforementioned considerations into account: existing infrastructures, population, and geographical constraints. An example of a core network as generated by YARGG may be seen in Fig. 4.10. Given a geographical location (e.g., a country or a continent), YARGG builds the structure of the core network by

1. *Extracting the x most populated cities in the area.* Close cities are merged into a single entity. The merge trigger value may change (the exact values used here can be found in [Luttringer *et al.* 2021a]).
2. *Constructing a minimum Spanning-Tree covering all cities of the area*, using road distances as the metric. Links between cities totaling more than 30% of the total population are normalized in order to be highly prioritized.[21]
3. *Removing articulation-points.* YARGG picks one bi-connected component and adds the smallest link (in terms of road distances) that bridges this component with another. This process is repeated until no articulation point remains (*i.e.*, the topology is bi-connected).
4. *Adding links increasing the connectivity and resilience for a limited cost.* YARGG considers all links meeting certain criteria. The two cities/nodes must be closer than 20% of the largest road distance. Their current degree must be lower than 4. The link, if added, should reduce the distance (and so, the delay) between the nodes by at least 25%. Among these links, YARGG adds the one with the highest attractiveness, expressed as the sum of the distance reduction and the population of the cities (normalized).
5. *Doubling the obtained topology.* The topology is doubled. There are now two nodes (/routers) per city. Links are added between the two routers of the same city, making the

[21] The parameters tuned may be easily modified. For now, the latter are purely empirical.

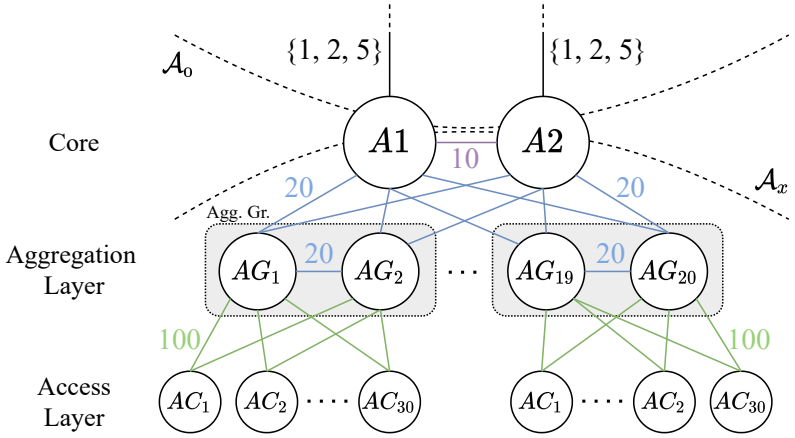


FIGURE 4.11: Weights and structures of an area generated by YARGG.

topology tri-connected.

The couple of routers located at each city within this generated backbone area become the ABRs between the backbone and their area, which is generated next.

Access & aggregation layers. These last two layers make up a non-backbone area and span a reduced geographical area. Thus, one access and one aggregation layer are located in each city considered by YARGG. Several network equipment vendors recommend a hierarchical topology, such as the three-layer hierarchical model [Cisco Networks 2014]. An illustration can be seen in Fig. 4.11. Simply put, there should be two core routers (the ABRs) at the given location (a city in YARGG's case). Each core router is connected to all aggregation routers. For better resiliency, the aggregation layer is divided into aggregation groups, composed of two connected routers. Finally, routers within an aggregation group are connected to access-layer routers. To achieve areas of $\approx$ 300 nodes, we consider 30 access routers per aggregation group. This results in a large, dense, and realistic graph.

Weights. In the backbone, the weights generated by YARGG are straightforward. The delays are extracted from the road distances between the cities, divided by 60% the speed of light (close to the best performing fiber optic). The IGP cost is 1 for links between large cities since these links usually have a high bandwidth (in black in Fig. 4.10), 2 for standard links, necessary to construct a tri-connected graph (added at step 3, in red in Fig. 4.10), and 5 for links that are not mandatory, but that increase the overall connectivity (added at step 4, in orange in Fig. 4.10).

Within an area, the IGP costs follow a set of realistic constraints, according to two main principles: (i) access routers should not be used to route traffic (except for the networks they serve), (ii) links between routers of the same hierarchical level (*e.g.*, between the two core routers or the two aggregation routers of a given aggregation group) should not be used, unless necessary (*e.g.*, multiple links or node failures). These simple principles lead to the IGP costs exhibited in Fig. 4.11. The delays are then chosen uniformly at random. Since access routers and aggregation routers are close geographically, the delay of their links

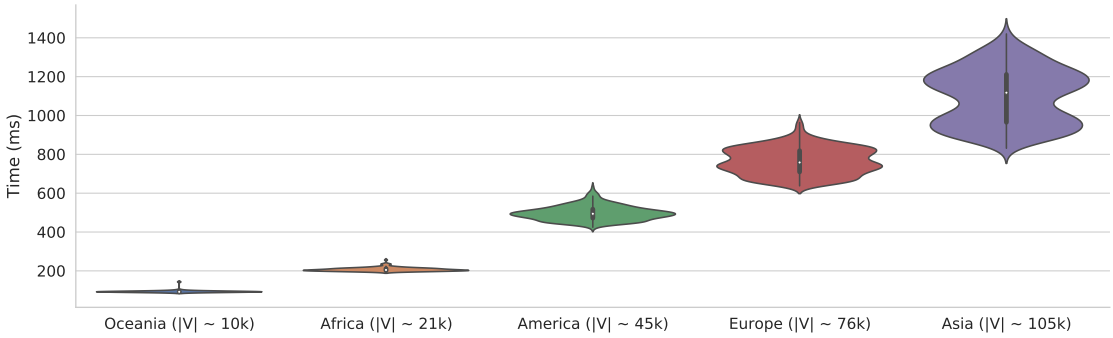


FIGURE 4.12: BEST2COP-E computation time on 5 continent-wide topologies generated by YARGG.

is chosen between 0.1 and 0.3ms. The delay between aggregation routers and core routers is chosen between 0.3ms and the lowest backbone link delay. Thus, links within an area necessarily possess a lesser delay than core links.

In summary, YARGG computes a large, realistic, and multi-area topology. The backbone spans a given geographical location and possesses simple IGP weights and realistic delays. Other areas follow a standard three-layer hierarchical model. Weights within a stub area are chosen according to a realistic set of usual ISP constraints. Delays, while chosen at random within such areas, remain consistent with what should be observed in practice.

4.5.2.3 Computing Time for Massive Scale Multi-Areas Networks

Using YARGG, we generate five massive scale, continent-wide topologies, and run BEST2COP on each one of them. The topologies ranges from 10 000 to 100 000 nodes. Each non-backbone area possess around 320 nodes. The topologies, their geographical representations and some of the associated network characteristics can be found online [Luttringer *et al.* 2021a].

We run BEST2COP on each ABR as a source (around $|V|/320 \times 2$ sources). The time corresponding to the message exchange of the computed Pareto front (step 2 of BEST2COP-E) is not taken into consideration. Thus, the computation time showcased is the sum of the average time taken by ABRs to perform the preliminary intra-area BEST2COP (and the distances to segment lists conversions) plus the time taken to perform the $|V|/320 \times 2 - 2$ Cartesian products (for all other ABRs of all other areas).

Note that we consider an ABR as a source and not an intra-area destination. In practice, the ABR would send the computed distances to the intra-area nodes, who in turn would have to perform a Cartesian product of these distances with its own distances to said ABR. However, both the ABR and the intra-area node have to consider the same number of destinations ($|V|$), and the results computed by the ABR can be sent as they are generated (destination per destination), allowing both the ABR and the intra-area nodes to perform their Cartesian product at the same time. In addition, intra-area nodes may benefit from several optimizations regarding their Cartesian product. For these reasons, we argue that the time measured here, using an ABR as a source, is representative of the total actual time required, *i.e.*, the overall worst time for the last treated destination at each source.

The results of this experiment are shown in the violin plot of Fig. 4.12. By leveraging

the network structure, BEST2COP-E exhibits very good performance despite the scale of the graph. For 10 000 nodes, BEST2COP-E exhibits a time similar to the one taken by its flat variant for $|V| = 2000$. Furthermore, BEST2COP-E seems to scale linearly with the number of nodes, remaining always under one second for $|V| \approx 75\,000$. Even once the network reaches a size of $\approx 100\,000$, BEST2COP-E is able to solve 2COP in less than one second for a non-negligible fraction of the sources, and never exceeds 1.5s.

Note that the times showcased here rely on a single thread. While BEST2COP-E's Cartesian product can be parallelized locally (both at the area and the destination scale), this parallelization hardly has any effect. This is explained by the fact that these individual computations are in fact fairly efficient, hence the overhead induced by the creation and management of threads is heavier than their workload. In addition, since BEST2COP deals with very large topologies, some complex memory-related effects might be at play. Indeed, we notice these results to surprisingly vary depending on the underlying system, operating system, and architecture due to the differences in terms of memory management.

Thus, while massive scale deployments seem to a priori prevent the usage of fine-grained TE, their structures can be leveraged, making complex TE problems solvable in less than one second even for networks reaching 100 000 nodes. The computations performed for each area can also be distributed among different containers within the cloud, if handled by a controller.

4.6 Conclusion, Limitations & Perspectives

While the overhead of MPLS-based solutions led to a TE winter in the past decade, Segment Routing marked its rebirth. SR encouraged operators to ask for the ability to perform fine-grained, large-scale TE, in particular the ability to deploy DCLC paths. While computing DCLC path is already NP-Hard, this complexity is further increased by the technical constraint of SR. Through a novel evaluation, we showed that the number of segment required to implement DCLC paths is manageable, showcasing that SR is indeed a relevant technology to deploy DCLC paths.

We propose several contributions enabling such deployments. Through a novel construct, the SR graph, we propose two ways to encompass the number of segments when computing paths.

LCA relies on the information contained within the SR graph to enable multi-criterion algorithms to translate paths in segment list on the fly, during the exploration of the original graph. We formalize this algorithm, as well as how this new dimension (whose behavior is more peculiar than the cost or the delay) must be handled by multi-constrained path algorithms. LCA may be fitted onto existing DCLC algorithm (along with the aforementioned guidelines) to solve DCLC-SR efficiently and correctly. We implemented LCA on top of a state-of-the-art algorithm, SAMCRA.

We also propose BEST2COP, an algorithm that directly explores the SR Graph, and leverage the characteristics of the latter to efficiently solve DCLC-SR. A specific graph exploration, a parallelizable structure, and an efficient management of the Pareto front allow BEST2COP to exhibit good performance. We implemented BEST2COP in C, and showed that the latter exhibits good performances, in particular when relying on multi-threading. By leveraging multi-area partitioning, BEST2COP is able to solve DCLC-SR on networks

of up to 100 000 nodes in about one second.

The work behind BEST2COP and LCA offers several interesting and challenging perspectives that are yet to be tackled. For example, the effect of Flex-Algo can be more thoroughly investigated, in particular when considering conversion schemes. While the SR Graph enables to deal with Flex-Algo natively, its impact on the complexity of LCA has not been studied. Similarly, the impact of SRv6 (and adjacency segment with global meaning) should be supported natively by LCA, but the best way to deal with this paradigm when relying on the SR graph does not seem trivial. Indeed, one could either allow segment lists to be extended two times per iteration (as using a node segment followed by an adjacency segment could be replaced by a single adjacency segment) or by representing the global meaning of adjacency segment within the SR Graph, which would further increase its density. The complexity of both approaches has not been extensively discussed yet. Finally, the notion of strong dominance can be further investigated. While a stricter definition may allow reducing the number of distances one has to extend, checking for the associated conditions in practice may be more computationally expensive, resulting in an interesting trade-off to evaluate.

Furthermore, I plan to extend YARGG to provide a tool allowing to evaluate path computation algorithms on massive-scale multi-metric computer networks. While showing whether the resulting topologies are realistic or not may prove arduous (if not impossible) given that little data is available, YARGG could at least offer the community a way to easily perform reproducible experimentation. As YARGG was at first programmed only to evaluate BEST2COP-E, a lot of room is left for improvement or additional features. For example, the performance of YARGG leaves a lot to be desired but could be enhanced by relying on probabilistic choices within the construction of the topology. This perspective offers interesting and varied tasks. Algorithmically, efficiently generating numerous massive-scale topologies is quite challenging. In addition, such a tool should be easily usable, which raises interesting questions design-wise. Finally, a tool can be envisioned to translate the generated topologies into OSPF advertisements (similarly to the tool LSAD proposed by Thomas Alfroy [Alfroy 2020]). This would, for example, allow the evaluation of path computation algorithms when implemented on real routing software such as Free-Range Routing.

YARGG would also enable us to revisit the current state-of-the-art. Modern hardware coupled with the realistic overall structure of the networks generated by YARGG may have had a drastic impact on the previous results of the literature regarding the design of multi-constrained path computation algorithms (node-based/label-based selection, label-setting/label-correction, type of queues...). Revisiting these results could thus prove interesting and lead to perhaps surprising results. However, while YARGG already offers interesting tasks to be tackled, our multi-constrained path computation algorithms leads to even more compelling perspectives.

For example, BEST2COP was designed assuming a source routing paradigm, but a hop-by-hop variant could be envisioned. As mentioned in Section 2.2.2, the subpath optimality property does not hold per se when considering DCLC paths, making hop-by-hop routing a less adequate paradigm. More precisely, a subpath of a DCLC path is a DCLC path, but considering a *different* constraint. More formally, with $p = (x_0, x_1) \dots (x_n - 1, x_n)$ a DCLC path with a delay constraint c_1, $p_{[x_i,x_n]}$ is a DCLC path considering a delay constraint $c'_1 = c_1 - d_1(p_{[x_0,x_i]})$. To perform hop-by-hop DCLC routing, routers should thus maintain several

entries per destination within the FIB, offering different delays, and select the adequate entries according to the current delay already experienced by the packet.

While routing packets in this fashion is not possible in standard architectures, programmable data-planes could enable such hop-by-hop DCLC routing. A timestamp could be maintained at line-rate within the transiting packet. Furthermore, P4 tables allow maintaining several entries (c'_1, n) per destination, and to select the appropriate one based on the timestamp of the packet and the original c_1 constraint. While maintaining enough entries to ensure optimal routing may not be possible due to the size of the Pareto front, one may only push some entries (a sample of the Pareto front, *e.g.*, as computed by 2COP) within the P4 data plane, sufficient to ensure some sort of degraded but controlled guarantees. This type of routing could be interesting, as it would be guided by the actual delay experienced by the packet rather than pre-established measurements. The main challenge is here technical, as a full proof-of-concept would require implementing both the data-plane and the control-plane while ensuring proper communication between the two. Finding the best way(s) to sample the Pareto front to select appropriate FIB entries also raises interesting algorithmic questions. Note that SR is here not required anymore, meaning that BEST2COP would be used in a different context not reliant on the SR Graph. Preliminary experiments have shown that BEST2COP can offer interesting and competitive performances even without considering SR (*i.e.*, when run on the original graph with no constraint on the number of hops) in particular cases, and could thus be used in this context. These encouraging results also plead for further investigation regarding the relative performance of different exploration schemes and priority when relying on LCA.

LCA could also be extended to additional use-cases. More precisely, the general concept behind this contribution (strong dominance and loose encoding) may very well be used as-is to augment other path computation algorithms for SR. However, the concrete steps that should be taken to fit LCA within this much larger context have to be investigated in deppth.

One can also envision reducing the theoretical complexity of BEST2COP (or any similar algorithms) by modifying the IGP weights of the network. Indeed, currently, we state that the delay is the most adequate metric to discretize in order to limit the size of the Pareto front predictably. However, for operators relying on few IGP weights, the number of possible IGP distances between each couple of routers may be even lower than our currently proposed Γ. More generally, a tool could be created (*e.g.*, relying on linear programming) to propose new IGP weights minimizing the number of possible IGP distances without impacting the original routing performed within the network.

Similarly, IGP weights could be modified to reduce the number of required segments to deploy DCLC paths, if the MSD constraint of the underlying hardware proves too tight. However, as this implies increasing the correlation of the IGP weights and the propagation delays, routing would be modified. In this case, relying on Flex-Algo or Binding segment may be more appropriate.

Some further discussions may also be had regarding the behavior of BEST2COP in a real, dynamic network undergoing frequent changes.

First, it should be noted that FRR mechanisms such as LFAs do not consider any TE objectives when computing backup paths. One could thus envision computing backup paths

that still respect the original delay constraint (if they exist). Note that such paths could, in some cases, already be found within the output of 2COP.

However, recall that to reduce the scale of the required pre-computation, each node usually computes backup paths only in the event of local failures. If a failure occurs, traffic is re-routed by the adjacent node. When considering SR, this would require (i) removing the current segment list from the packet and (ii) pushing a new list encoding the new backup path. This operation may be quite heavy, which could make the concept of delay-aware FRR self-defeating in a segment routing environment. However, this may prove interesting when relying on hop-by-hop DCLC routing, as the underlying required programmable data-plane could already offer the tools necessary to implement such intricate FRR.

One could however argue that a network is not (and should not be) expected to fulfill its TE requirements upon failure, and that the induced suboptimal state should be transient and short enough to not cause heavy *Service Level Agreement* (SLA) violations. Critical flows, which would benefit from the stronger guarantees brought by delay-aware FRR, are in addition likely to be deployed on resilient resources specifically allocated to this effect.

Perhaps more interestingly, one could envision ways to reduce the impact of the SR Graph computation time. Indeed, after an internal event, the recomputation performed by BEST2COP (or any other algorithm fitted with LCA) should be quick. The computation of the required SR graph is however expensive by nature. Several optimizations exist to reduce the computation time when solving APSP [Demetrescu & Italiano 2004, Demetrescu & Italiano 2006]. However, I believe that the most interesting option is to keep the computation as-is (running one Dijkstra's algorithm per source in parallel), but minimizing the number of re-computations. Ideally, only changes that impact the results of one (or several) of said runs of Dijkstra's algorithm would trigger a partial (or full) recomputation. Efficiently predicting (or, at least, approximating) the effect of any possible failures is an interesting algorithmic challenge. However, one could also rely on fine-grained active monitoring to detect when the APSP should be recomputed, a perspective that I find particularly interesting.

Indeed, just as SR may steer traffic, it can also be used to steer probes and perform network monitoring [Aubry *et al.* 2016b, Ventre *et al.* 2020]. Probes could then periodically be sent over the DCLC paths that have been deployed (using the segment lists computed beforehand). These probes would enable us to track the actual delay experienced by the traffic. The recomputation of the SR Graph (and the segment lists) would thus not be triggered by IGP events, but rather by an actual measured violation of the delay constraint regarding one flow (or a sample of them). Note however that while this ensures that the delay constraint is still respected, the IGP cost of the path may not be optimal anymore. One may envision a hybrid design, where the APSP is periodically re-computed to ensure that DCLC paths do not deviate too much (or too long) from the shortest IGP paths, and are re-computed immediately upon strong measured SLA violations. Performing such fine-grained distributed monitoring of the latency raises several technical challenges, in particular to ensure the accuracy of the measurements. In addition, this design requires a centralized controller to monitor said measurement and trigger the re-computation. From an operator's viewpoint, this architecture enables greater transparency and detailed history regarding the service provided to the client but may have a significant cost.

BEST2COP and LCA are two algorithmic contributions aiming to tackle a well-known practical routing problem: the computation and deployment of TE paths in large-scale networks. However, TE-related problems are only part of the challenges one has to tackle in modern-day networks. In the following chapter, we visit the other extremity of the qualitative routing spectrum and consider another type of challenge: resiliency. More precisely, we examine how to ensure quick optimal routing for transit traffic after internal events.

Chapter 5

Reliable Hot-Potato Routing

Contents

In Section 2.3, we have seen that the Internet is composed of independent domains known as ASes. IGPs provide intra-domain connectivity within each AS, while BGP allows ASes to trade transit traffic. ASes exchange routes through eBGP, while iBGP enables their dissemination among internal routers.

Since several distinct routes may exist for a given BGP prefix, internal routers determine the best route towards each prefix by running the BGP *decision process*, a process that may be long given the scale at which BGP operates [Filsfils *et al.* 2011, Holterbach 2021]. Furthermore, as the ranking of the routes varies depending on the state of the IGP topology (through hot-potato routing), this process is triggered whenever an IGP event occurs. Since intra-domain changes are frequent [Merindol *et al.* 2018], this interaction becomes a challenging issue.

In this chapter, we present OPTIC, a multi-scale routing scheme mitigating the impact of such IGP changes on the BGP convergence while enforcing both hot- and cold-potato routing. OPTIC fulfills two design requirements, effectively making the transit traffic impervious to any IGP event.

OPTIC re-routes traffic quickly towards the new optimal BGP route. Upon an internal event, OPTIC allows the transiting traffic to benefit from the new optimal route in a time equivalent to the IGP convergence (without running a full BGP decision process). We say that OPTIC *optimally protects* BGP prefixes, meaning that whatever the IGP event, the BGP prefixes are almost immediately reachable again through their best BGP route.

To achieve this requirement, OPTIC efficiently computes backup sets of gateways[1] guaranteed to contain the optimal gateway after any internal event. The new optimal gateway can be found quickly within said set, and the latter can be constructed without having to consider every possible event independently.

The computed sets are shared in memory by a group of prefixes, allowing to quickly switch to the new optimal gateway for all prefixes within the same group.

OPTIC's background processing performed to anticipate any next future IGP event is manageable. Once the traffic is optimally re-routed, the current set may be obsolete if the internal event impacted the overall connectivity level of the network. In this case, groups and their associated sets should be updated in background to *anticipate* any *next* future event.With OPTIC, this process is manageable, *i.e.*, at worst similar to BGP, but negligible when the event does not hamper the connectivity level of the network.

We start by reminding of the interaction between BGP and the IGP, exhibited on a more intricate example than the one used in Section 3.1. We then define the notions behind OPTIC and present the associated structures and algorithms in detail, before evaluating the feasibility of OPTIC through a theoretical model.

OPTIC is a continuation of my Master's thesis internship (supervised by Dr. Pascal Mérindol). Since my Master's thesis, OPTIC has been considerably refined, in particular to better encompass the effects of the MED, and more thoroughly evaluated through an extensive theoretical evaluation. Furthermore, OPTIC is currently being implemented in a routing software, as well as in P4. OPTIC has been published in INFOCOM'21 [Luttringer *et al.* 2021d] and in CoRes'21 [Luttringer *et al.* 2021c].

5.1 BGP/IGP: an Intimate Relationship

We start by reminding the IGP-BGP interplay, on a new example, which will be used in the following chapter to illustrate the concepts of OPTIC.

Recall that BGP routes are characterized by a collection of attributes $\beta \circ \alpha$ of decreasing importance, as explained in Section 2.3.2.2 (the Table referencing the BGP attributes is

[1]Since we also consider the failure of an external gateway, the term *gateway* refers to the external gateway of the neighboring AS by default. Thus, gateway and route may be used interchangeably.

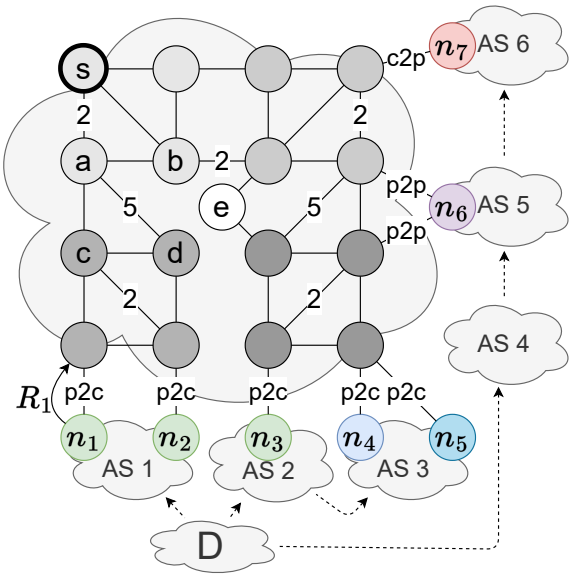


FIGURE 5.1: This example consists of an AS that learns routes towards D via several border routers, focusing on the point of view of s. Each link from an internal border router to the BGP NH is labeled with the type of relation between the two ASes (p2c means provider-to-customer, p2p and c2p, peer-to-peer and customer-to-provider respectively, modeled in practice by a decreasing local preference). A route R_x is advertised by the BGP NH n_x. The routes announced by n_4 and n_5 are discriminated through the MED attribute. Unlabeled edges weight one.

shown again in Table 5.1 for readability purposes.). Some attributes can be locally modified by each router. The BGP decision process elects the best route by ordering them lexicographically, according to their attributes.

When the inter-domain related attributes β = [LOCAL-PREF, AS-PATH, MED] of two routes are equal, routers considering the following attributes α = [IBGP/EBGP, IGP COST, ID]. Since the attribute after the IGP cost is a simple tie-break, and since preferring eBGP routes over iBGP routes can be seen as an extension to the IGP cost (an eBGP route has an IGP cost of 0), we can refer to α simply as the IGP distance towards the IGP NH.

It becomes clear that IGP events may affect the ranking of BGP routes. Let us consider Fig. 5.1. We state that $R_x \prec R_y$ if R_x is better than R_y according to the BGP decision process. We consider the routes R_1, R_2, R_3 and R_4 towards the prefix D announced by n_1, n_2, n_3 and n_4 respectively. The MED being irrelevant to the point, we consider that these routes have no MED for now. R_4 originates from a client and has an as-path length of 2, leading to the attributes $\beta(R_4)$ = [p2c, 2, -, -]. R_1, R_2 and R_3 are all characterized by the same $\beta(R_1,R_2, R_3)$ = [p2c, 1, -, -] and so are discriminated through their α distances (4 vs 5 vs 6). All have a better β than R_4. Thus, overall, $R_1 \prec R_2 \prec R_3 \prec R_4$ from the point of view of s.

While the inequality $R_1 \prec R_2 \prec R_3$ holds initially, this order is reversed after the failure of the link a→c as the IGP distances, taken into account by BGP, go from 4, 5 and 6 to 9, 8 and 6 respectively. After the failure, $R_3 \prec R_2 \prec R_1$, requiring to wait for the BGP decision process to find the new best route. However, note that inter-domain related attributes (β) are left unaffected by an IGP event, meaning that R_4 will remain less preferred than the other three routers after **any** IGP event in any cases.

TABLE 5.1: An example of the set of attributes taken into consideration by BGP, as already shown in Section 2.3.2.2.

	Step	*Criterion*	
β	1	Highest `local-pref` LP	(economical relationships)
	2	Shortest `as-path`	
	3	Lowest `origin`	
	4	Lowest MED	(cold-potato routing)
α	5	eBGP over iBGP	
	6	Lowest IGP distance	(hot-potato routing)
	7	Lowest `router-id` rid	(arbitrary tie-break)

Fig. 5.1 also illustrates the shortcomings of PIC [Filsfils *et al.* 2011] and Add-path [Uttaro *et al.* 2016a], already described in Section 3.1.

First, after the failure of the link a→c, PIC's HFIB would restore the connectivity to n_1 by updating the IGP NH used to reach n_1, allowing the transit traffic to reach n_1 again. However, the effect of the IGP event in the α ranking of some routes is ignored. Restoring the connectivity to n_1 without considering such changes leads to the use of a sub-optimal route until BGP re-converges, violating so the hot-potato routing policy. Besides, the traffic may first be re-directed after the IGP convergence, and then re-directed once again after the BGP convergence, potentially leading to flow disruptions [Teixeira & Rexford 2006].

Second, recall that PIC stored the two best BGP NH, in case the preferred exit point were to fail. Here, even if both n_1 and n_2 were stored, both become unreachable if a fails (due to the network not being node-biconnected), leading to a loss of connectivity until BGP re-converges and finds the new best available gateway, n_3.

In both scenarios, retrieving the correct new optimal path requires the BGP decision process which does not scale well. Furthermore, even events that do not impact the connectivity state lead to useless computations. For example, if $s \to a$ fails, BGP still re-converges, and PIC re-considers its sets of gateways even though the latter are still viable, given the nature of the event.

With OPTIC, we aim at dealing with these cases more elegantly and efficiently, by guaranteeing to be able to switch immediately to the new best BGP NH whatever the internal event, for a low maintenance cost and while limiting unnecessary computations.

5.1.1 How to Reach a Symbiotic Coupling?

We present here the necessary operational condition to untie the BGP convergence from IGP events. The question to address is: how to efficiently pre-compute the subset composed of every BGP route that may become the new *best* route upon *any* IGP change? We state that prefixes need to be *optimally protected*, as per Definition 5.1.1.

> **Definition 5.1.1** (Optimal Protection)**.** *Let D be an external destination. We state that D is* ***optimally protected*** *by a set O, if both pre- and post-convergence BGP NHs are*

> *stored within O. More precisely, O should verify the two following properties for any IGP change c:*
>
> - *(i) It contains the best BGP NH n towards D before c occurs (pre-convergence NH for D);*
> - *(ii) It contains the BGP NH of the new best path towards D after c occurs (post-convergence NH for D). It should be true for any kind of c: link or node event, n included, such as an insertion, deletion or weight update.*

Computing such sets naively may look costly, as predicting the optimal gateway for each specific possible failure and weight change is time-consuming. However, OPTIC computes and maintains these sets efficiently by *rounding* them. We will show that the size and number of such rounded sets are limited in most cases. Finding the new optimal gateway among these sets is performed through a simple min-search within each set (with no additional computation), updating so each group of prefixes depending on this set, and thus every prefix. Depending on how such sets of gateways (per group of prefixes) are designed, OPTIC can protect the traffic transiting in a BGP network against any link, node, or even SRLG (*i.e.*,, links sharing a common fate) single failure.

5.2 *Optimal Protection Technique for Inter-intra-domain Convergence* (OPTIC)

This section explains our solution in detail. OPTIC mitigates the impact of IGP events on the BGP convergence without hampering neither hot- or cold- potato routing. It pre-computes sets of gateways bound to contain the current and future *optimal* gateways after any single IGP failure or change, *optimally protecting* every prefix[2]. For the sake of simplicity, we only consider single node and link failures (including the gateway) as well as weight changes, but OPTIC can be extended for more general failure scenarios at the cost of complexifying the group management overhead.

5.2.1 Sorting and Rounding BGP Routes

First, we show that optimal protecting sets can be efficiently computed and maintained by sorting and *rounding* BGP routes in a specific way. We start by explaining this concept in a high-level fashion before formally detailing our solution.

5.2.1.1 General idea

Using our β (inter-domain attributes) and α (IGP distance) attribute separation, we can compute optimal protecting sets easily. Indeed, β is of higher importance than α within the BGP decision process, and IGP events can only affect α, leaving β unchanged. Thus, given the current optimal route, denoted R^{st}, with $\beta(R^{st}) = \beta^{st}$, the new optimal route after an

[2]The addition of a gateway, taken into account in our set-maintaining algorithms, is signaled by a BGP message and thus considered a BGP event.

IGP event is among the ones with the same best β^{st} – we simply need to find the one with the new best α.

We can then easily avoid predicting which gateway will be the optimal one for a specific event; whatever the IGP event is (except the gateway failure possibly requiring to look for more gateways), the new optimal route is among $\{R \mid \beta(R) = \beta^{st}\}$. We thus create a *rounded* set that includes all routes sharing the same β. After the IGP event, since β attributes are unaffected, we simply need to consider that α may have changed and find within this rounded set the gateway with the lowest α (*i.e.*,, with a simple min-search). In Fig. 5.1, such a set would be composed of n_1, n_2, n_3 as they share the same (best) β attributes. We indeed showed in Section 5.1 that any of these three gateways may become the new optimal gateway.

This is however not sufficient to deal with all failures. In particular, if the first rounded set only contains one gateway, a single failure (that of the gateway itself) may render all routes within the set unusable. If this scenario occurs (because there are no two node-disjoint paths toward the external prefix – see Section 5.2.1.4), more gateways are needed to optimally protect the prefix. Since β attributes are unchanged by an internal event, the new best route is, a priori, among the ones with second-to-best β attributes β^{nd}.

To form an optimal protecting set, we add rounded sets of β-tied gateways up until there is enough path diversity to ensure that no single failure may render all of them unreachable (there are two node-disjoint paths between the border routers and prefix D). By never adding less at a time than all gateways sharing the same β, we ensure that the final set contains all potential optimal gateways (as only α can be affected by internal events). This final set (composed of the union of rounded sets) is thus optimally protecting, and the new optimal gateway can be found through a simple walk-through of this set after any IGP event.

If two prefixes share an equal optimal protecting set, they belong in the same group and share the same set in memory, reducing both the memory consumption and the number of entries to go through and update upon an event (as covering all shared sets covers all prefixes). In Fig. 5.1, n_1, n_2, and n_3 provided enough path-diversity to ensure the prefix was protected, and shared the same best β. Thus, the optimal gateway after any internal event is bound to be one of these three, which then form an optimal protecting set (for all single possible failures).

We can now present formally the data structures allowing OPTIC to compute and maintain optimal protecting sets easily. Our solution requires to re-design both the control- and the data-plane. The control-plane here refers to all learned BGP routes. It is restructured to ease the handling of the routes, their comparison in particular, for efficient computation of optimal protecting sets. The data-plane only contains the information necessary for the optimal forwarding and protection of all prefixes (*i.e.*, the optimal protecting sets). The resulting structures are illustrated in Fig. 5.2, which shows how the network depicted in Fig. 5.1 would be translated within OPTIC's control-plane (left) and data-plane (right). To better illustrate our data structures, we assume here that n_4 has a better MED than n_5, while other routes do not possess any MED. The next sections describe the control-plane structure, how we construct optimal protecting sets from it, and how they are used in the data-plane.

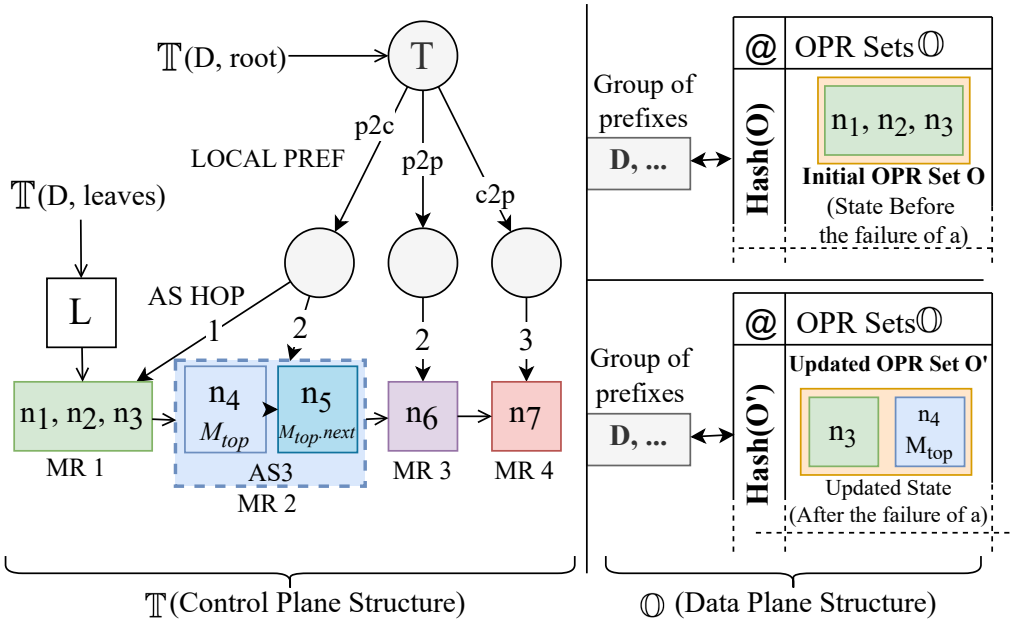


FIGURE 5.2: OPTIC's data-plane and control-plane data structures. In the control-plane, routes are sorted within a prefix tree T whose leaves form a linked list L of structured BGP NH. MED-tied routes from the same AS are chained within a linked list inside their leaf. Only a sufficient optimally protecting subset O of routes is pushed to the data-plane.

5.2.1.2 OPTIC's control-plane

At the control-plane level, OPTIC stores every BGP routes learned within a sorted prefix-tree referred to as T, whose leaves form an ordered linked-list L, which contains rounded sets of routes sorted by decreasingly preferred β attributes. Both T and L are per-prefix structures. The set of all trees, for all prefixes, is referred to as $\mathbb{T}$. It is important to observe that, since α is not considered, the tree and the list stay stable upon IGP changes and that routes sharing the same β attributes are stored within the same leaf.

This observation implies that when an IGP event occurs, the BGP NH of the new optimal route belongs to the first leaf of the list L that contains at least one reachable gateway. In addition, a route from another leaf cannot be preferred to the routes of the first leaf. **While any route within the first leaf can become optimal after an internal change, the order of the leaves themselves can not be modified by an IGP change**.

The MED attribute can only be used to compare routes originated from the same AS, hence we cannot use it as a global, generic attribute. One can only consider a route with a greater MED if the route with a better one from the same AS becomes unreachable. Thus, routes discriminated by their MED (*MED-tied* routes) for each AS are stored within a sub-linked-list inside their leaf. This is illustrated in Fig. 5.2 with n_4 and n_5. Both BGP NH share the same three first BGP attributes and are thus stored within the same blue leaf of T (MR2). As they originated from the same AS, we store them in a sorted linked list depending on their MED attribute. By doing so, we consider only the first route in the *MED-tied* list that is reachable (referred to as M_{top}), respecting the MED's semantics. Peculiar situations, such as routes not having a MED while others do, can be resolved by applying the standard ISP practices (*e.g.*, ignoring such routes or using a default MED value). The leaves of the

tree T thus form a sequence of rounded sets of gateways stable upon IGP changes. We call each set a *MED-aware Rounded* (MR) set.

Definition 5.2.1. MED-aware Rounded *(MR) For a given prefix, a leaf of its prefix tree T is called a MED-Aware Rounded set. In particular, it contains all the routes having the same β attributes (MED-Excluded).*

5.2.1.3 Getting optimal-protecting sets from the control-plane

We now explain how this construct eases the computation of optimal protecting sets.

For each prefix D, the first MR set contains, by construction, the optimal pre-convergence route. As stated in Section 5.2.1.2, any BGP NH within the same MR set may offer the new optimal post-convergence route after an internal event. However, this first MR set is not always sufficient to protect D. In this case, the new optimal BGP NH is bound to be within the first MR set in L which contains a gateway that is still reachable. Consequently, OPTIC constructs an optimally protecting set by considering the union of the best MR sets, in order, until the destination prefix D is protected from any failure, *i.e.*, there exist two node-disjoint paths. The union of such MR sets is referred to as an Optimal-Protecting Rounded (OPR) set for D. The formal definition is given in Theorem. 5.2.1. Due to lack of space and its intuitive design, its proof is not presented here (but available in [Luttringer & Mérindol 2019]).

Definition 5.2.2 (OPR set)**.** *Let D be a destination prefix, and $M_1, M_2, \ldots$ be the sequence of MR sets in the list L. The OPR set of D is defined as $O = \bigcup_{i=1}^{x} M_i$, with x minimal such that there exist two node-disjoint paths towards D (passing through O)*

Theorem 5.2.1. *Let D be a prefix, and O an OPR set as defined by Def. 5.2.2. Then, O is optimally protecting D.*

Adding MR sets until the prefix D is protected means that there now exists enough path diversity to protect D from any single event. The number of routes necessary to protect a prefix depends on the resilience of the network. In bi-connected networks, two gateways are enough.

OPR sets computation does not require any (prior) knowledge of the IGP graph to cover all possible IGP events. Verifying the existence of two node-disjoint paths between the border router and D via O is enough and the lightest possible processing to test the protection property. Unless the protection property is affected by the event, OPR sets stay stable.

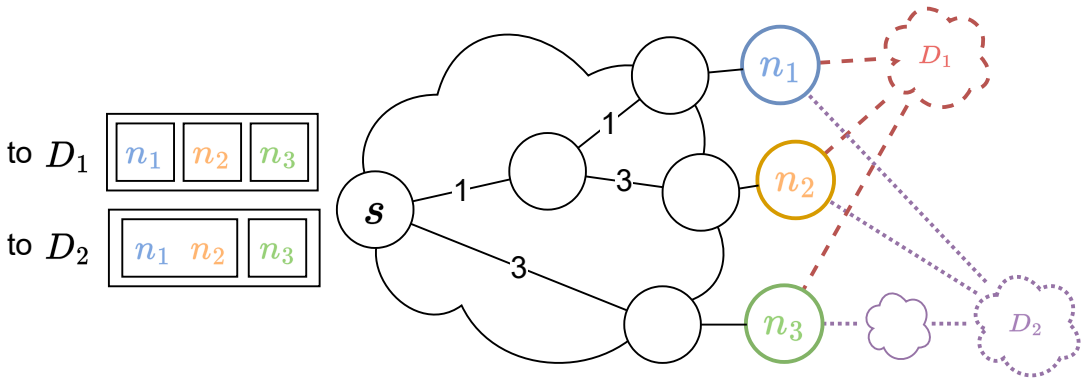


FIGURE 5.3: For prefixes to share the same OPR set, the latter must not only possess the same gateways, but also the same MR set decomposition.

5.2.1.4 Using OPR sets in the data-plane

Once OPR sets are extracted from the control-plane, we push them to the data-plane. The bottom part of Fig. 5.2 shows OPTIC's data-plane. For a given prefix, only the OPR set O (and not the whole list L) that optimally protects D is pushed to the data-plane. The data-plane contains the meta-set $\mathbb{O}$ of all OPR sets for all groups of prefixes, indexed by their hash, as shown in Fig. 5.2. Prefixes, when sharing the same OPR set, point towards the same set O. The hash index is content-based (see next sections for more details) and eases the management of $\mathbb{O}$. Allowing prefixes to share the same O reduces the amount of data that has to be stored within the data-plane, as well as the scale of the updates. Note that, since O is constructed from a subset of L, prefixes can share the same OPR set O while having different control-plane structures L. Note, however, that in order to share an OPR set, prefixes must not only share the same best gateways but also the same MR set decomposition within the OPR set. Indeed, when retrieving the new best gateway within an OPR set after an internal event, one does not have to consider all the gateways of the OPR set. More precisely, one should only consider the first MR set (*i.e.*, find the gateway with the minimum IGP cost within the first MR set of the OPR). Thus, prefixes that don't share the same MR set decomposition of their OPR sets must not share the same OPR set in memory, as the min-search should not be performed on the same routes. This issue is illustrated in Fig. 5.3.

Fig. 5.3 shows the OPR sets (and their respective decomposition into MR sets) of the prefixes D_1 and D_2, from the point of view of s. If n_1 fails, the new best gateway to D_1 becomes n_3, while the new best gateway to D_2 becomes n_2. Thus, one must look for the new best gateway in the first MR set that contains a reachable gateway only, and not in the whole OPR set. Hence, the MR-decomposition of each OPR set is relevant at the data-plane level and should be kept in O (not only in T and L). Prefixes then share their OPR set in memory if they contain the same gateways *and* the same MR-decomposition.

To manage OPR sets within the data-plane, we first require a utility function able to extract, from L, the current OPR set. This function is described in Alg. 8. In short, the required MR sets are computed by first *(i)* creating a graph G' from G where we add a virtual node representing a destination prefix D, then *(ii)* connecting in G' the gateways from MR sets, MR per MR to this virtual node, until there exists two node-disjoint paths

towards the virtual node. Once this condition is verified, we know that the OPR set should be composed of the union of these MR. EXTRACT_OPR thus returns an OPR set as defined by Theorem 5.2.1. The *start* variable is useful if we know that the OPR set should contain more than x MR, in which case we can set this variable to x in order to skip the first steps of the main loop.

Another way of finding the number of required MR sets, more intuitive, would rely on checking the existence of a path from the current node to the destination for any possible failure on the best path towards p. However, this method is far less efficient than the one proposed here and does not allow to deal easily with ECMP. With this function, we can know design algorithms allowing to manage OPTIC's data-plane.

Algorithm 8: EXTRACT_OPR

```
1  ──────────────────────────────────────────────────────────────────
   Data:
     L, G, start = 0 (Index of the start of the search)
   Result:
     OPR set required for protection
2  ──────────────────────────────────────────────────────────────────
3  Function extractOPRSet→
5      V' = V ∪ D;
7      E' = E ∪ {(u,p) | u=L[i][n], ∀ i ∈ [0,start], ∀ n ∈ L[i]});
9      G' = (V', E');
11     ρ = any_path(G', D);
13     RG_ρ(V', E") = residual_graph (G',ρ);
15     while ∄ path (RG_ρ, D) do
17         RG_ρ(V', E") = (V', E' ∪ {(u,D) | u=L[j][n], ∀ n ∈ L[j]}));
19         j++;
21     O = ∅;
23     for i = 0...j do
25         O = O ∪ L[i];
27         i++;
29     return O;
30 ──────────────────────────────────────────────────────────────────
```

Algorithm 9 shows how the OPR sets are updated in the data-plane when necessary. The optimal protection property may require adding gateways from the data-plane structure O (while removals are performed for efficiency). We start by extracting the OPR set O from the control-plane structure L (Line 5). We then add the IGP distances towards each gateway (contained within d_{IGP}) to the structure (Line 9). This is done for each gateway, including MED-tied ones (Line 13). Finally, OPTIC retrieves the current optimal gateway within O, O_{top}, *i.e.*, the one with the lowest IGP distance (Line 15) within the first MR set that still contains at least one gateway.

Once the OPR set O is updated, we compute its hash to check its existence within $\mathbb{O}$ and insert it if required (Line 21). Finally, if no prefixes still use the previous O, it is removed from the data-plane (Line 27). This algorithm maintains the data-plane in an optimal-protecting state. Its limited complexity is often bypassed (after the bootstrap), as we expect OPR sets to stay stable in bi-connected networks. Unused OPR sets could be kept transiently within the data-plane to mitigate the effects of intermittent failures.

Algorithm 9: UPDATE_OPR

```
Data: L, 𝕆, oldH, D, G, d_IGP
Result: updates OPR sets and D_BGP
Function update_OPR →
    O = extractOPRSet (L, G);
    for M_top ∈ O do
        while M_top not = ∅ do
            M_top.α = d_IGP[M_top];
            M_top = M_top.next;
    O_top = min_α(O[0]);
    newH = hash (O);
    if O ∉ 𝕆 then
        𝕆[newH] = O;
    D_BGP(D) = newH;
    if not∃ D | D_BGP(D) = oldH then
        remove 𝕆[oldH] from 𝕆;
```

5.2.2 Dealing with BGP and IGP events

We describe here how OPR sets are updated upon a BGP or an IGP event to achieve optimal protection of all destinations.

Algorithm 10: BGP_Update

```
Data: 𝕋, 𝕆, D_BGP, R = (D, n, β), G, d_IGP
Result: Update of 𝕋, 𝕆, D_BGP
Function BGP_Update →
    T = 𝕋(D, tree);
    H = D_BGP(D); // H = hash (O)
    L = 𝕋(D, leaves);
    if Event = Add then
        rMR = add R in T;
    else
        rMR = remove R from T;
    if rMR ∈ useful_MR (L) then
        update_OPR (L, 𝕆, H, D, G, d_IGP);
```

5.2.2.1 BGP updates

Algorithm 10 showcases how to maintain OPR sets upon a BGP update, being either an *Add* (*i.e.*,, a new route is learned) or a *Down* event (*i.e.*,, a withdraw that cancels a route)[3]. As a BGP update concerns a given prefix, only one OPR set O (the one that optimally protects

[3] A modified route can be handled through a *Down* followed by an *Add*.

D) is modified when necessary. Intuitively, checking whether the route R belongs (or should belong) to the leaves of T extracted to create the current O (*i.e.*, if R belongs to the current O) is enough to know if the update is necessary.

First, Alg. 10 retrieves the route-tree T of the updated prefix D (Line 5). Depending on the nature of the update, we update the control-plane structure T (and implicitly L) by either adding (Line 13) or removing (Line 17) the updated route. When performing these operations, we store the rank of the MR set containing the route R, *rMR*.

Using rMR, one can check whether R belongs (or should belong) to O (Line 19), *e.g.*, by memorizing the number of MR sets used to form O. If R is not good enough to belong to the current OPR set, there is no need to consider it and the algorithm ends. Otherwise, if R is a newly added (resp. withdrawn) route, it must be added (resp. removed) from the data-plane structure O which can be found in $\mathbb{O}$ through its hash. In both cases, O has to be updated (Line 21). One can see that OPTIC's behavior when dealing with a BGP update is pretty straightforward and that BGP events are likely to have no bearing on the data-plane.

Algorithm 11: IGP_Change

```
1  ──────────────────────────────────────────────────────────────
   Data: 𝕋, 𝕆, D_BGP, G= (E, V), l, w
   Result: Update of 𝕋, 𝕆, D_BGP, G
2  ──────────────────────────────────────────────────────────────
3  Function Change →
       // get new IGP distances
5      d_IGP = spt (G, l, w);
7      forall O ∈ 𝕆 do
9          iMR = 0; i = 0; forall M_top ∈ O do
13             while M_top not = ∅ ∧ d_IGP[M_top] = ∞ do
15                 M_top = M_top.next;
17             if M_top = ∅ then
19                 remove M_top from O;
21             else
23                 M_top.α = d_IGP[M_top];
25                 iMR = min (i, iMR);
27             i += 1;
           // Switch to optimal route
29         O_top = min_α(O[iMR]);
           // Anticipate next igp event
31         if l ∈ E ∧ w not = ∞ then
               // Just a valuation change
33             continue;
35         else if 2disj_routes (O, G) ∧ is_min(O) then
37             continue;
39         add hash (O) in OPR_to_update;
41     forall H ∈ OPR_to_update do
43         forall D | D_BGP(D) = H do
45             L = 𝕋(D, list);
47             update_OPR (L, 𝕆, H, D, G, d_IGP);
48 ──────────────────────────────────────────────────────────────
```

5.2.2.2 IGP changes

Algorithm 11 showcases the behavior of OPTIC upon an IGP change, which can be a modification on the existence (insertion or deletion – modeled by an infinite weight) or on the weight w of a link l (a node wide change can be modeled through its multiple outgoing links).

Upon a change, the new IGP distances d_{IGP} are recovered. OPTIC then considers each O, covering so every BGP prefixes (Line 7). For each relevant gateway (with the best MED for each AS, M_{top}) within O, we first check whether it is still reachable (Line 13). Unreachable gateways are replaced by the next MED-tied route when possible (Line 15) or removed (Line 19) otherwise. Reachable gateways are first updated with their new best IGP distances (Line 23). The whole group of prefixes using the set benefits from its new optimal path as soon as possible (Line 29). Afterward, if necessary, we update OPTIC's structures in the background to anticipate any future internal event.

If the updated link l is a valuation change (Line 31), there is no loss of reachability. Thus, O still contains two disjoint paths towards D and remains stable. For other kinds of events, O may need to be updated, as connectivity may have evolved due to the insertion or deletion of a link. If a link was added, the network connectivity may have increased and useless MR sets can be removed if O is not already minimal (*e.g.*, containing two gateways). If a link was removed, O may have lost its protecting property and may have to be updated. These two scenarios, leading to the update of O, are visible in the condition Line 35. This update is used to prepare for a future event. We perform it in the background afterwards (Line 41) and continue to walk through $\mathbb{O}$ to restore the optimal forwarding state of all groups of prefixes quickly.

The update aforementioned (Line 41) is performed at the prefix granularity (*i.e.*,, for each prefixes that used O that will be updated). Indeed, while these prefixes share the same O before the change, they do not necessarily share the same L. Since O may be updated by fetching information from L, they may point to distinct OPR sets after the update. Recall that this is a background processing phase where OPTIC may fall back to the prefix granularity to anticipate the next change only if node-bi-connectivity is not granted anymore. The fast switch to the new optimal post-convergence gateway was already performed at Line 29. This switch is not done at the prefix granularity, it is performed only for each O instead.

In short, most BGP and IGP updates both result in simple operations. A BGP update just triggers a prefix tree manipulation: a single OPR set is re-computed only if the updated route is, or should be, part of the set. An IGP weight-change results in the walk-through of all OPR sets ($\mathbb{O}$) and a min-search to converge to the new optimal forwarding state followed by a background processing if necessary. We argue that the cardinal of $\mathbb{O}$ will be orders of magnitudes lower than the number of BGP prefixes in most networks. The failure or addition of a link or node results in the same walk-through, but could also require the background update of some OPR sets to prepare for a future event. More precisely, only when the network gains or loses its bi-connected property could some OPR sets be affected. New OPR sets then need to be re-computed for the prefixes of the groups that depended on the affected OPR sets. Instead of the number of prefixes, OPTIC convergence scales with the number and the size of the OPR sets. Consequently, to assess the viability of OPTIC, we aim at limiting their size (and so number). While Sec. 5.3.3 analyzes that aspect in detail, the

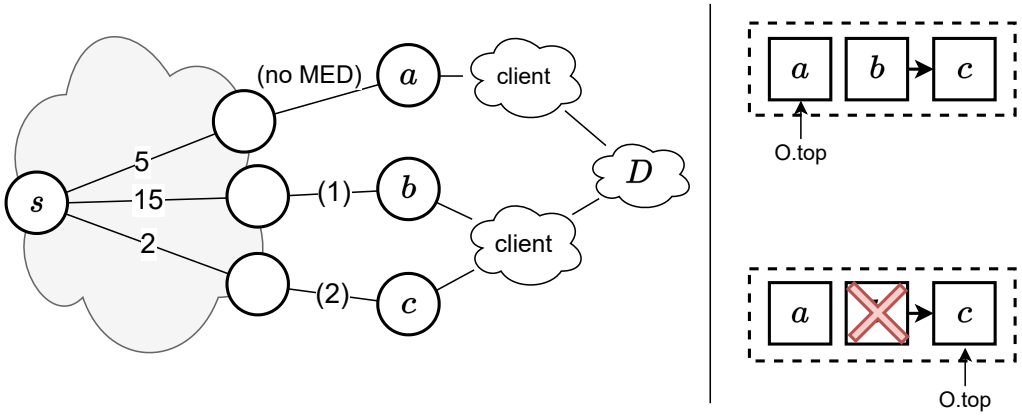


FIGURE 5.4: SURPRISE DOOM-MED. Example illustrated while all the gateways within a MED linked list should be kept within the first MR.

next subsection explores conditions on the graph properties allowing to use smaller optimally protecting sets.

5.2.3 Optimizations

In this section, we introduce optimizations that allow reducing the size of the OPR sets used by OPTIC.

Let us start with a fairly reasonable assumption: well-designed networks should offer bi-connectivity between border routers. Based on this realistic hypothesis, we can consider two kinds of reductions: (i) removing some MED-tied entries from an OPR set and (ii) discarding all gateways in the second MR set except the best one (when the first MR set includes only one gateway).

Let us consider the first optimization. Recall that the MED is an attribute of higher importance than the IGP cost. Hence, a gateway that does not have the best MED has no chance of becoming the best gateway after an IGP event, as the one with the best MED (M_{top}) will always be preferred by the decision process (excluding if M_{top} itself fails).

Now, let us consider O_{top} as the current optimal gateway, and as the only gateway within the first MR set. Clearly, the only possible internal event that may impact the optimal status of O_{top} is the failure of O_{top} itself, meaning that the new best gateway is in the second MR set. However, this internal event cannot impact the MED-attribute of these gateways. Consequently, the new optimal gateway is necessarily one of the gateways with the best MED pre-failure, *i.e.*, the head M_{top} of each list. Consequently, there is no point if pushing to the data-plane more than the head of each list of any MR set after the first one.

A legitimate question would be to ask whether this optimization could also be performed in the first MR set of the OPR set. Indeed, following similar reasoning, the failure of O_{top} (or a change of distance towards the latter) does not change the MED, and the new best gateway should thus be a head of a linked list. However, in this case, a perhaps surprising effect may occur depending on the underlying BGP configuration, as illustrated by the SURPRISINGLY DOOM-MED gadget in Fig. 5.4.

Here, the OPR set is composed of a single MR set containing a, b and c, with b having

a lower MED than c. Recall that the route comparison is usually done two-by-two, for efficiency purposes. Thus, it is possible for a to be chosen as the best gateway (if c was eliminated by b prior because of the MED). In this case, if b were to fail, c would become the best gateway, meaning that it is necessary to keep the entirety of the list within the MR set. This effect is here due to the fact that the MED breaks the total ordering between routes (similarly to what was shown on the MEDEVIL gadget in Section 2.3.2.2), *hiding* so the route with the actual lowest IGP distance. This effect should thus not occurs when relying on the ALWAYS COMPARE MED option, which changes the MED semantics to that of a standard attribute. Nevertheless, this showcases that particular care must be taken regarding the hypothesis behind such optimizations.

The second optimization is perhaps more general, and can be performed if the first MR set is composed of a single gateway. Then, clearly, an internal event will not make this gateway lose its optimal status, except if the gateway itself becomes unreachable. As the network is supposed to be bi-connected, the only failure which may render the gateway unreachable is the failure of the gateway itself. Let us suppose that such a failure occurs. Then, the new optimal gateway should be found within the second MR set. However, if the failed gateway was not part of the path used to reach the best gateway of the second MR set, then the best gateway of the second MR set is the same before and after the failure (and is the new optimal gateway).

Thus, in such a case where the first MR set is composed of a single gateway *and* this gateway is not on the path used to reach the best gateway of the second MR set, the best gateways of the first two MR set form an OPR set.

Such optimizations can be important, as, to react to an IGP event, OPTIC only operates a min-search in all OPR sets. OPTIC's performances thus mainly depend on the number of OPR sets ($|\mathbb{O}|$) and their sizes.

5.3 Theoretical Evaluation

5.3.1 Preliminary Model: counting #OPR sets

To react to an IGP event, OPTIC only operates a min-search in all OPR sets. OPTIC's performances thus mainly depend on the number of OPR sets ($|\mathbb{O}|$) and their sizes.

We present here a theoretical model capturing a wide variety of scenarios. This analytical approach is more suitable than experiments, as it is more general and provides a pessimistic order of magnitude of OPTIC's potential. This approach yields the same results as a simulation, but allows to easily explore numerous scenarios. It highlights what an ISP can expect by running OPTIC given only a few structural parameters on their networks. We investigate several ASes profiles (constructed from [Luckie *et al.* 2013]'s data), varying the number of gateways, peers, clients, and providers, as well as the number of prefixes learned through each of the latter. We show that $|\mathbb{O}|$ remains manageable and/or close to the lower bound, being 99% smaller than the number of prefixes for stub networks.

5.3.2 Preliminary Model: counting #OPR sets

We consider an AS (or a portion of it), with B bi-connected gateways advertising P prefixes in total. Each prefix D is advertised by a subset of $b \leq B$ of those gateways, chosen uniformly at random. The β of each prefix is represented by a value between 1 and ps (policy spreading) also chosen uniformly at random. For a given D, this implies that any subset of gateways of a given size $n \leq b$ all have the same probability to be the OPR set for D. Our model analyzes the number $|\mathbb{O}| = |\mathbb{O}_{B,P,ps}|$ of unique OPR sets depending on the number B of gateways, the number P of prefixes, and on the policy spreading ps. In practice, we decide to set b to a constant value (*e.g.*, $b = 5$) greater than the median in [Luckie *et al.* 2013].

In bi-connected networks, OPR sets can only take two possible forms: they are either composed of one MR set (all best β-tied gateways) or two MR sets, if the first one only contains a single gateway. Indeed, more than 2 MR sets are not necessary, as two gateways will by definition protect the prefix, and two MR sets will thus do so optimally. Similarly, a single MR set of two gateways or more will always be sufficient. Recall that OPR sets are shared if they share both the same gateways *and* the same MR set decomposition. Both cases thus need to be treated independently.

We first compute the probability p_n of a prefix to have an OPR composed of a single MR set of n gateways. In this case, all n gateways must have the same best β, an event with a probability of $\frac{1}{ps^n}$. In addition, all other $b-n$ gateways must not have a better β. This event has a probability of $\left(1 - \frac{i}{ps}\right)^{b-n}$, with i the β value of the n best gateways. This may happen for each possible ps values, so the probability must be summed for $i = 1$ to ps. Finally, there are $\binom{B}{n}$ such possible sets of size n. Thus, the probability that a prefix has an OPR set composed of a single MR of n gateways is

$$p_n = \sum_{i=1}^{ps} \binom{b}{n} \frac{1}{ps^n} \left(1 - \frac{i}{ps}\right)^{b-n} \tag{5.1}$$

An OPR set of size n may also be composed of 2 MR sets, if the first one only contains a single gateway (the second MR set will thus contains $n-1$ gateways). We thus compute the probability p'_n of a given prefix to have an OPR set of size n composed of two MR sets. For this to happen, $n-1$ gateways must have the same β attributes, while a single gateway must have better β attributes. In addition, all other $b-n$ gateways must have worst β attributes, leading to a probability of $\frac{1}{ps^{n-1}} \frac{i-1}{ps} \left(1 - \frac{i}{ps}\right)^{b-n}$, with i the β value of the $n-1$ gateways of the second MR set. Any one of the b gateways may possess the best β attribute, and there are $\binom{b-1}{n-1}$ possible second MR set of $n-1$ gateways. Thus, this event occurs $b\binom{b-1}{n-1}$ times. Summing all the cases (for each possible ps), we have

$$p'_n = \sum_{i=1}^{ps} b \binom{b-1}{n-1} \frac{1}{ps^{n-1}} \frac{i-1}{ps} \left(1 - \frac{i}{ps}\right)^{b-n} \tag{5.2}$$

From Eq. 5.1 and 5.1, we can compute the probability that a given OPR set of size n is associated with at least one prefix.

First, let us compute the probability that a given OPR set of size n composed of a single MR is associated with at least one prefix. There are $\binom{B}{n}$ such sets. On average, the number of prefixes associated with OPR sets of size n (with a single MR set) is $p_n P$. Thus, the

probability that a given set of size n is *not* associated with any prefix is $\left(1-\binom{B}{n}^{-1}\right)^{p_n P}$ (all prefixes chose another OPR set). So, the probability that a given OPR set is associated with at least one prefix is

$$\mathbb{P}_{B,ps,P,n} = 1 - \left(1 - \binom{B}{n}^{-1}\right)^{p_n P} \tag{5.3}$$

A similar reasoning can be done to find the probability that a given OPR set of size n composed of two MR is associated with at least one prefix, the main difference being that there are $B\binom{B-1}{n-1}$ such sets (B possible first MR set, and $\binom{B-1}{n-1}$ possible second MR set), leading to

$$\mathbb{P}'_{B,ps,P,n} = 1 - \left(1 - \left(B\binom{B-1}{n-1}\right)^{-1}\right)^{p'_n P} \tag{5.4}$$

From $\mathbb{P}$ and $\mathbb{P}'$, we can compute $|\mathbb{O}_{B,P,ps}|$, the number of distinct OPR sets. The quantity $|\mathbb{O}_{B,P,ps}|$ can be seen as the sum of distinct OPR sets of different sizes. From our assumptions, OPR sets of size n ($2 \leq n \leq b$) are in $\mathbb{O}_{B,P,ps}$ with the same probability. A particular subset of gateways of size n can be the OPR of a given prefix through its one MR set variant (with a probability $\mathbb{P}_{B,P,ps,n}$) or through its two MR set variant (with a probability $\mathbb{P}'_{B,P,ps,n}$), leading to a probability $\mathbb{P}_{B,P,ps,n} + \mathbb{P}'_{B,P,ps,n}$, to account for both possible configurations. There are respectively $\binom{B}{n}$ and $B\binom{B-1}{n-1}$ such sets, for $n = 2$ to b (as b gateways announce the prefix). Thus, we have

$$|\mathbb{O}_{B,P,ps}| = \sum_{n=2}^{b} \binom{B}{n}\mathbb{P}_{B,P,ps,n} + B\binom{B-1}{n-1}\mathbb{P}'_{B,P,ps,n} \tag{5.5}$$

We can also easily compute the number of distinct OPR sets when applying the optimizations mentioned in the previous sections. In particular, recall that when the first MR contains a single gateway, then the OPR set can be reduced to a size of two (two MR with one gateway each), no matter how many gateways actually are in the second MR set. This optimization is very likely in practice, as the properties required are pretty lenient [4]. With this optimization, the probabilities do not change when considering OPR sets composed of one MR set (all gateways must have the same β attributes, as expressed by the left side of equation (5)). However, the probabilities slightly change when considering OPR sets composed of two MR sets (right sight of equation (5)): as long as only one gateway has the best β attributes, the OPR set will be of size 2, and there are $B(B-1)$ possible sets of size two. Thus, for the optimized version, we have

$$|\mathbb{O}^{opt}{}_{B,P,ps}| = \sum_{i=1}^{ps} B(B-1)\frac{b}{ps}\left(1-\frac{i}{ps}\right)^{b-1} + \sum_{n=2}^{b}\binom{B}{n}\mathbb{P}_{B,P,ps,n} \tag{5.6}$$

5.3.3 Towards a Realistic Evaluation

5.3.3.1 Break Down Into Classes

In practice, neighboring ASes are partitioned in several classes (*eg.*, clients, peers, and providers), usually represented by the LOCAL-PREF attribute. At the end of the decision

[4] The best path to the best gateway does not contain the second-best gateway

TABLE 5.2: Number of distinct OPR sets ($|\mathbb{O}|$) for several scenarios.

Type of AS	# gateways per class	# prefix per class	# distinct OPR sets	OPR sets median size	Lower bound
Stub	$(10; 20; 0)$	$(700K; 100K; 0K)$	3945	4	235
Tier 4	$(10; 25; 25)$	$(500K; 200K; 100K)$	11 879	3	645
Tier 3	$(10; 50; 100)$	$(500K; 200K; 100K)$	46 010	3	6219
Large Tier 3	$(10; 100; 500)$	$(500K; 200K; 100K)$	127 433	2	73 781
Tier 2	$(5; 500; 2000)$	$(500K; 200K; 100K)$	263 219	2	197 194
Tier 2 (other)	$(5; 500; 2000)$	$(450K; 250K; 100K)$	294 886	2	205 484
Tier 1	$(0; 50; 5000)$	$(0K; 600K; 200K)$	232 180	2	199 633
Tier 1 (other)	$(0; 50; 5000)$	$(0K; 400K; 400K)$	425 786	2	394 891

process, we know that a prefix D is associated with a single class. Indeed, the LOCAL-PREF depends only on the set of advertising neighbors for D: it belongs to the class of the neighbor having the highest LOCAL-PREF.

This allows us to split the analysis by class. With this assumption, OPR sets are included inside a unique class of gateways, but as a counterpart, the policy spreading in each class is reduced (because gateways have the same LOCAL-PREF inside a class). We use our former model to compute the number of distinct OPR sets in each class with $ps = b = 5$. This calibration is pessimistic enough as it only takes into account a limited AS length dispersion and always 5 learning gateways in the best class.

B_1, B_2 and B_3, denote respectively the number of gateways with LOCAL-PREF 1, 2 and 3. Similarly, P_1, P_2 and P_3, denote respectively the number of prefixes originating from a gateway with LOCAL-PREF 1, 2 and 3. We have $P_1 + P_2 + P_3 = P = 800\,000$ and $B_1 + B_2 + B_3 = B$. We can now compute $|\mathbb{O}|$ by assuming each class follows our basic model:

$$|\mathbb{O}| = |\mathbb{O}_{B_1,P_1,5}| + |\mathbb{O}_{B_2,P_2,5}| + |\mathbb{O}_{B_3,P_3,5}|$$

This sum gives the theoretical performance of OPTIC as it is the number of OPR sets each router has to manage.

5.3.3.2 Definition of the Lower Bound

We define here the best theoretical performance an optimally protecting scheme could reach, to compare it with OPTIC. Such a scheme would have to store sets of at minima two gateways (less cannot ensure protection). This lower bound also provides an estimation of the performances of techniques just aiming at providing (non-optimal) protection like PIC [Filsfils *et al.* 2011]. In other words, with P_i prefixes and B_i gateways in a given class, the average minimum number of optimally protecting sets is the average number of distinct sets obtained when choosing P_i random subsets of two gateways (such sets are chosen uniformly at random).

5.3.3.3 Evaluation on Fixed Break Down

We now compute $|\mathbb{O}|$ for several AS categories; a *Stub* has few peers and even fewer providers from where most the prefixes originate; a *Transit* (Tier 2, 3 and 4) has a limited number of providers but from where the majority of the prefixes originate, more peers and possibly

numerous customers; a *Tier 1* has few peers and a large number of customers. For *Transit* and *Tier 1*, we present different class and prefix break down. Note that our model is pessimistic, as, for *Tier 1* in particular, ASes may have more classes with $ps > 5$ (*e.g.*,, gateways can be geographically grouped). The number of gateways, and their partition into classes, are rounded upper bounds of realistic values obtained from [Luckie *et al.* 2013]. Moreover, we did not assume any specific popularity of certain gateways. Using our complementary material [Bramas *et al.* 2020], $|\mathbb{O}|$ can be computed for any parameters.

Table 7.1 shows that the number of OPR sets is more than reasonable for Stubs and small Transit. For large transit, the distribution of the prefixes into classes has a great impact on $|\mathbb{O}|$. As expected, for Tier 1, the number of OPR sets is high, but OPTIC is close to the lower bound (there is not much room for possible improvements). The number of routes contained within each OPR set is limited, meaning that the min-search applied upon an IGP event has a limited computational cost. Finally, it is worth recalling that our analysis is pessimistic because uniform. Regional preferences or gateway popularity can strongly reduce the size of $\mathbb{O}$ in practice.

5.3.3.4 Evaluation on Variable Break Down

In addition to previous specific cases, we here show how $|\mathbb{O}|$ evolves depending on the sizes of the classes and their relation. For that, we introduce the variable δ that represents the ratio between the sizes of two successive classes. More precisely, when a break down considers a ratio of δ, then it means $(B_1, B_2, B_3) = (B_1, B_1 \times \delta, B_1 \times \delta^2)$. Similarly, we also assume that the number of prefixes learned by each class verifies $(P_1, P_2, P_3) = (P_1, P_1/\delta, P_1/\delta^2)$.

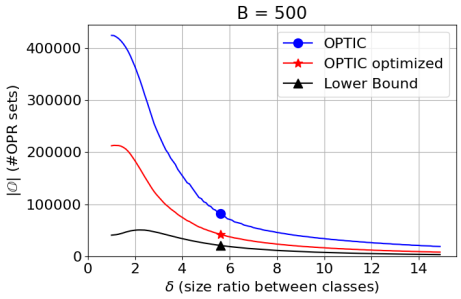
We present in Figure 5.5a the number of distinct OPR sets depending on δ for $B = 500$; δ varies from 1 (all the classes have the same size) to 15 (each class has 15 times more gateways that the previous class, but learns 15 times fewer prefixes). When the number of gateways is low, the management cost is obviously limited (even when all the classes have the same size). With $B = 500$, OPTIC's performance is limited for small δ but gets better quickly. When $\delta \geq 8$, our optimization performs as well as the lower bound.

We now investigate the case where δ equals 5 and look at how $|\mathbb{O}|$ evolves depending on B. We see in Fig. 5.5b that our optimized OPR reduction outperforms the non-optimized one. For less than 1500 gateways, the number of OPR sets is smaller than 100 000 with our optimized version. Then, $|\mathbb{O}|$ increases quickly to reach 200 000 when there are around 4000 gateways, with a lower bound at 125 000 sets. After, the growth is linear, so the proportional overhead of our solution compared to the lower bound tends to one.

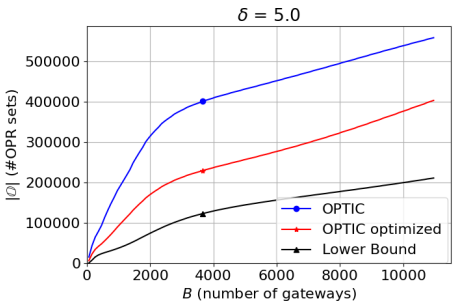
The management cost of OPTIC is reasonable, especially for networks having a limited number of border gateways where OPTIC exhibits very good performances. For large networks having numerous gateways, the size of our data-plane structures remains limited regarding the number of prefixes, and there does not exist many room for theoretical improvements.

5.4 Conclusion & Perspectives

Because the IGP and BGP are entangled to enforce hot-potato routing at the AS scale, an IGP change triggers the full and slow BGP convergence. With OPTIC, we aim at re-designing this IGP/BGP coupling. We proposed efficient MED-aware algorithms and data-structures



(A) $|\mathbb{O}|$ depending on the ratio δ between classes with $B = 500$.



(B) $|\mathbb{O}|$ depending on the number of gateways, with $\delta = 5$.

FIGURE 5.5: Evolution of $|\mathcal{O}|$ depending on various factors.

to anticipate and quickly react to any single IGP event (weight change, link, or node failure, including the outage of BGP border routers). At the data-plane level, OPTIC ensures a fast and optimal re-convergence of the transit traffic. In the control-plane, OPTIC updates its constructs in the background to anticipate a future event when necessary (only after changes modifying the 2-node-connectivity network property). This process has the same computational as BGP at worst to deal with one event in advance (the optimal connectivity being already restored prior).

Since nearly all calculations are performed per group of prefixes, OPTIC scales orders of magnitudes lower than the number of BGP prefixes. Our analytical evaluation shows that the number of entries to manage in the FIB is at worst 50% of the full Internet table for large Tier-1. It scales down to 25% for large Tier-2s and less than 1% for Stub AS, which represents 84% of all ASes in the current Internet.

Several short- to mid-term perspectives can be envisioned for OPTIC. New ways to construct OPR sets can be investigated, *e.g.*, considering (some) concurrent failures, leveraging other network characteristics, or even considering the operators' arbitrary requirements. Our tool (used to predict the number of OPR set in the evaluation) could then be extended to propose these various schemes and predicts the resulting $\mathcal{O}$, allowing operators to choose the best protecting scheme remaining under a given constraint on the number of OPR sets. One may also envision computing sets whose objectives differ from the one presented here. For example, one may compute sets with TE objectives, *e.g.*, aiming to protect the destination prefix (if possible) and ensuring that the gateways within the set can be reached within a certain delay constraint, performing a kind of delay-aware hot-potato routing. Such a set could be constructed by solving 2COP, if paths towards the gateway are to be deployed with SR [5]. This may however require deep modifications to the decision process, as the latter may currently eliminate too many routes before reaching the hot-potato rule to ensure that routes satisfying such intricate requirements are still available.

Further discussion regarding the actual behavior of OPTIC within an operator's networks may also be required. For example, the behavior of OPTIC considering intricate iBGP topologies or communities [Li *et al.* 1996] is worthy of additional investigation. More realistic

[5] Note that ensuring such constraints within an AS would however not allow ensuring end-to-end guarantees

evaluations may also be envisioned, relying on data gathered through projects such as BGP Streams [6] or RIPE RIS [7].

Furthermore, the interaction of OPTIC with other FRR mechanisms can be worthy of discussion. In theory, such interaction should not raise any troublesome issues. Nevertheless, after an IGP event, FRR mechanisms such as LFAs could direct the traffic towards the pre-event optimal gateway through backup paths, even though said gateway may be obsolete. Similar cases may occur when relying on tunneling. The use of this transient sub-optimal gateway and the switch over to the new optimal one could lead to desequencing [8]. Investigating whether other detrimental effects may occur and whether they may be mitigated is a further step to show that OPTIC can be deployed within real operators' networks.

Similarly, further discussions and considerations regarding the deployment of OPTIC may also prove interesting. In particular, partially deploying OPTIC incrementally may lead to further heterogeneity and could increase the chances of transient forwarding loops or deflection. Indeed, OPTIC-routers may make use of the new optimal route earlier than non-OPTIC routers. Should a non-OPTIC router be used as the new optimal gateway, the latter may not yet be aware of its new status and thus forward traffic to another gateway (or even back to the source). This effect can however be mitigated when relying on tunneling mechanisms, in particular if such tunnels are extended one hop past the gateway, enforcing traffic to enter a specific neighboring AS (this can be done through technologies such as BGP *Egress Peer Engineering* (EPE) with SR [Filsfils *et al.* 2021b]). When such tunnels are deployed, deployed OPTIC-routers could enforce their choices, allowing traffic going through OPTIC-routers to benefit from the new optimal route quicker with no ill-effects. This argument is however worthy of further discussions.

Currently, OPTIC must rely on either a fully-meshed iBGP topology or Add-Path All, the only mode of Add-Path ensuring enough route diversity to construct OPR sets. However, one may envision a new mode of Add-Path built upon and for OPTIC, consisting in exchanging the OPR set associated with each prefix. As OPR sets are computed solely following the β attributes, which are identical among all routers, routers should agree regarding whether a given route belongs within an OPR set or not. Consequently, exchanging these routes should ensure that all routers possess enough information to compute OPR sets while reducing the scale of the message exchanges compared to Add-Path All. Other architectures may be envisioned, more or less centralized. For example, routers may send all their routes to a given node in charge of the computation and re-distribution of the OPR set, akin to a route-reflector.

In addition, exchanging OPR sets should not only ensure that all routers are aware of the post-convergence route after any IGP event but should also prevent MED and IGP-related oscillations within the AS. Indeed, by definition, routers (i) would not hide routes to other routers because of local preferences (as α is not considered), and (ii) the route with the best MED is necessarily available to all routers (when the optimizations presented in Sec. 5.2.3

[6] https://bgpstream.caida.org

[7] https://ris-live.ripe.net

[8] Note that while this effect could also occur without any underlying FRR schemes, between the IGP convergence and OPTIC's convergence. However, as will be detailed later on, OPTIC's data-plane can be programmed in a fashion allowing an immediate switch to the new optimal gateway, without going through any transient state between the IGP and OPTIC's convergence.

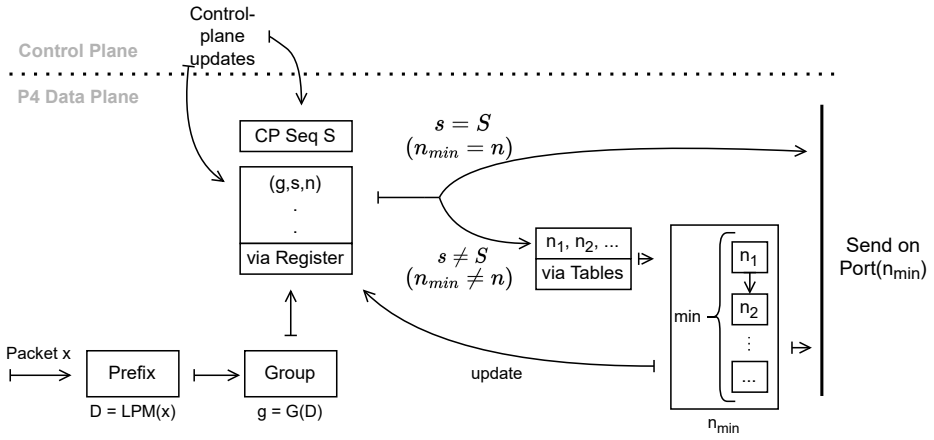


FIGURE 5.6: Simplified illustration of the OPTIC data-plane pipeline.

are not implemented). Recall that such conditions were the root cause of both types of oscillations presented in Sec. 2.3.2.2. These claims however require additional proof and further investigation.

OPTIC was implemented in Free-Range Routing by Thomas Alfroy [Alfroy 2020]. However, this implementation exhibited some limitations when implementing OPTIC fully within the control-plane. Mainly, once the new optimal gateway is computed, updating the FIB accordingly takes a non-negligible amount of time. This comes as no surprise, given the results within the literature [Boucadair 2005, Filsfils *et al.* 2011, Holterbach 2021].

New technologies now allow implementing OPTIC as envisioned and described, *i.e.*, through modifications both within the control-plane and the data-plane. In particular, P4 [Bosshart *et al.* 2014] allows modifying the data-plane of a routing device. OPTIC was indeed imagined as a hardware-software co-design, where the control-plane would only be in charge of the more tedious operators. Implementing the data-plane of OPTIC through P4 would allow updating the new optimal BGP NH (*i.e.*, finding the best gateway within an OPR set) directly within the data-plane, instead of relying on the control-plane to perform such operations. To ensure that packets are still processed at line-rate, P4 does, however, have strong limitations (*e.g.*, not allowing control loops) which makes such implementation far from trivial.

While the current implementation of OPTIC in P4 is a work in progress, this perspective seems promising [9] and preliminary tests with a minimal control-plane demonstrated the feasibility of this approach.

Fig.5.6 shows an illustration of the currently implemented P4 pipeline for OPTIC. Given a destination prefix D, we retrieve a set (g, s, n) composed of the group of the prefix, a sequence number, and the current optimal BGP NH n, all stored within registers. Along with the BGP NH is a sequence number s. After an IGP event, the control plane (i) updates the IGP distances towards each gateway within the FIB (P4 tables in this case) and (ii) increments a global sequence number S. The control-plane then re-computes and updates the best BGP NH n associated to each prefix group and sets the associated sequence number s to S. If, for a given packet, we have $s = S$, then n is optimal, and the packet can

[9] https://icube-forge.unistra.fr/jr.luttringer/optic-p4

be forwarded through the associated interface. If s differs from S, then n is not optimal. In this case, the OPR set (stored within tables in the data-plane) is retrieved. The gateway with minimal IGP distance is found within the OPR set, n and s are set accordingly within the registers, and the packet is forwarded through the associated port.

In practice, since there are no control loops, we look for the minimal IGP distance within the first k value of the set. If the size of the OPR set is equal or inferior to k, the optimal gateway has been found. Otherwise, the packet can be resubmitted to the start of the pipeline, carrying states allowing to start the search for the minimal distance from k onward.

By allowing both the control-plane *and* the data-plane to update the current optimal gateway for each group, we aim to get the best of both worlds (or in this case, planes). As updates are triggered by data packets, the latter immediately benefit from the new optimal gateway once the FIB has been updated with the new IGP distances: there are no transient suboptimal states. Furthermore, groups with less popular prefixes will still be updated by the control-plane, which may update the latter through a clever prioritized ordering.

Currently, the P4 implementation of OPTIC was coded for BMv2, a software environment simulating a P4 switch. We plan on implementing both the control and data-plane of OPTIC on actual hardware and perform in-depth experimentations. This perspective gives rise to several technical challenges (in particular, the actual update and modification of the OPR sets are one of the more complex aspects of OPTIC which is yet to be implemented). Furthermore, some steps of OPTIC could potentially be implemented by leveraging TCAM memory to drastically reduce their computational complexity, in the fashion of PURR [Chiesa *et al.* 2019]. All of these challenges remain to be explored and tackled.

Chapter 6

Conclusion

Over the course of this thesis, we have explored different topics related to qualitative routing within computer networks. In particular, we discussed the topic of computing DCLC paths deployable within SR domains with BEST2COP and LCA (Chapter 4), and improving the convergence of the BGP protocol upon intra-domain events, mitigating so the ill-effects of hot-potato routing with OPTIC (Chapter 5).

The recent surge of interest for SR incited operators to strive for the ability to compute and deploy more intricate paths, suited for premium flows with strict latency requirements. While numerous algorithms exist to compute these DCLC paths, most do not consider the underlying deployment technology and its operational constraint, *i.e.*, the number of segments required to encode the path, a metric that cannot be handled natively by most algorithms. This prompted the need for efficient algorithms, able to compute DCLC paths deployable with SR, *i.e.*, solving DCLC-SR. To this effect, we proposed two schemes that rely on our specific construct, the SR Graph : BEST2COP and LCA.

BEST2COP is a parallel algorithm which explores the SR Graph to solve DCLC-SR. By leveraging the characteristics of the SR Graph, BEST2COP computes deployable DCLC paths in less than 100ms on networks of 1000 nodes. By taking advantage of obvious graph separators within operator networks, we show on topologies generated by a generator of our own design that BEST2COP can solve DCLC-SR in about 1 second within networks of 100 000 nodes. This showcases the feasibility of deploying DCLC paths in modern networks, that are slowly reaching such scales [Matsushima *et al.* 2022].

LCA is a conversion algorithm enabling multi-criterion path computation algorithms to keep track of the number of segments required to encode the distances they are currently exploring. We formally define (and prove) for the first time in this thesis how distances should be maintained to find the minimal encoding segment list of each distance and retrieve the entirety of the Pareto front.

Similarly, while improving the convergence of IP routing protocols or providing fast reroute capabilities has been researched extensively, few works focus on mitigating the detrimental effects of multi-protocol interactions, which yet lies at the heart of routing.

Our goal was to re-visit the interaction between BGP and the IGP and propose a new, more appropriate way to fit these protocols together. OPTIC, the scheme described in this thesis, efficiently pre-computes optimal protecting sets of backup routes (which may be shared in memory by several prefixes) for any internal event. The standard BGP convergence is thus replaced by a simple walk-through of each set, updating so the route to every remote prefixes quickly. We detail algorithms enabling the management and update of these sets. Furthermore, we showed, through a theoretical evaluation, that the number of entries to update is drastically reduced compared to vanilla BGP for most ASes within the Internet,

in particular for Stub and enterprise networks.

In the chapters describing these contributions, I have already described and discussed several short- to mid-term perspectives that are closely related to my contributions. However, the work I did during these past three years made me eager to investigate several other subjects and challenges related to the ones mentioned in this thesis.

6.1 Perspectives

When working on BEST2COP, LCA and OPTIC, I have discovered and come to think about several research questions or domains that seem deeply interesting to me, and that I plan to investigate further.

After having studied some aspects of BGP in-depth through OPTIC, I believe that the way inter-domain routing is performed could be revisited, in particular to open up new extension possibilities. The development of OPTIC's data-plane in P4 also made me eager to investigate the possibilities offered by new programmable devices. Furthermore, while I have studied path computations problems mainly within computer networks, other contexts and networks offer very compelling challenges and new ways to approach these problems. Finally, the teachings I had the chance to give in the past years made me realize how stimulating pedagogical challenges and the creation of pedagogical tools could be.

6.1.1 Revisiting the Inter-domain routing paradigm

The drastically different paradigm that inter-domain routing followed is, nowadays, very rarely challenged or even discussed, aside from contributions such as HLP [Subramanian *et al.* 2005b] and SCION [Barrera *et al.* 2017]. In the future, I plan to investigate whether (and how) a more viable kind of inter-domain routing could be achieved by relying on the link-state paradigm.

Indeed, research around BGP has shown that this protocol exhibits several issues, ranging from exponential message complexity to long convergence time or even correctness issues. Furthermore, I believe that some arguments usually put forward to justify the design choices behind BGP do not necessarily still hold, in particular given the current capabilities of routing devices.

I am especially interested in the starting point proposed by HLP. In particular, I believe that link-state routing could scale well enough (if not better), even in an inter-domain routing context. While maintaining a full router-level map of the Internet is impossible, maintaining a coarser, AS-level connectivity map may be achievable. The hierarchical structure of the Internet may allow the presence of natural graph separators, enabling one to rely on schemes similar to multi-area IGPs (a concept also explored in HLP). As there are currently (in total) around 100 000 ASes, the computation of shortest paths should scale (as we showed with BEST2COP). While this scheme does show some limitations for now (in particular regarding the type of supported policies), I believe that it is worthy of further exploration and discussion.

In particular, I also want to investigate what further benefits (in addition to scalability) such a routing scheme would enable, were it to be deployed. Relying on the link-state paradigm would allow for (at least partial) knowledge of the connectivity map, which would for example enable the design of more efficient FRR schemes, similar to the ones currently

proposed at the intra-domain scale. Furthermore, additional information could be added to the link of the connectivity map. For example, the latency induced by going from domain A to domain B could be advertised, without revealing precise topological information. As this latency depends on the entry and exit points used at a given AS, this would however require a connectivity map with finer information and increase the size of the resulting graph. Further investigation is then required regarding the feasibility of this approach (in particular its scalability) or possible security concerns.

Overall, if doable, a re-design of the current inter-domain routing paradigm may not only remove (some) of its current plagues and caveats but also opens the path toward new routing capabilities.

I also plan to investigate the ability to perform end-to-end, inter-domain TE without drastic changes to the current inter-domain routing paradigm. While this problem has become less and less visited in the past years, I believe that SR could be leveraged to perform inter-domain TE with no drastic modifications to the existing protocols.

In particular, *Binding Segments* (BSIDs) allow one to advertise specific distances while keeping detailed topological information mostly opaque. One could thus envision adding Binding Segment to advertised BGP routes, which could be used by the neighboring AS to forward data across a specific path (*e.g.*, guaranteeing a given latency) within the remote network. By correctly maintaining this information across ASes and advertisement, implementing inter-domain TE paths could be achievable. However, the actual logistics of how such attributes should be exchanged and behave, as well as the possible security implications require further investigation.

6.1.2 Programmable routing devices

While starting to implement the data-plane of OPTIC in P4, I have come to realize that programmable routing hardware offers a myriad of paths to investigate. Through their abilities to mitigate network ossification, deploy personalized data-plane behavior or protocols, and by being carried by both academia and the industry alike, I believe that programmable switches will become more and more prevalent as time goes on.

Being intrigued as well by the capabilities of these devices, I plan to further investigate how to best leverage the latter. Programmable data-planes (with P4 in particular) may not only allow improving the performance of existing solutions but also pursuing new goals. For example, the fine control that these new architectures could be leveraged to deploy new routing strategies, aiming for example to reduce the energy consumption of the routing devices, *e.g.*, by limiting the use of energy-consuming components such as TCAM. Consequently, interesting challenges also arise regarding the optimization of P4 programs. While low-level optimizations mainly lie in the realm of compilation, high-level optimizations (be it computation-wise or energy-wise) could be performed automatically depending on the characteristics of the underlying network, and be implemented as possible strategies within the P4 compiler. My advisors and I recently started to discuss these perspectives with the ICube laboratory team working on the compilation and optimization of computer programs, to better investigate such prospects.

6.1.3 Path computation in various contexts

Path computations and routing problems have been a cornerstone of my thesis, and a subject that I have come to deeply enjoy. While I, for now, mainly focused on path computations within computer networks, I have found that other contexts bring additional challenges and interesting variations to these problems.

In particular, the computation of paths within dynamic networks is a challenging and interesting subject. Indeed, while computer networks are dynamic, the path computation algorithms used therein rely on a snapshot of the network.

Dynamic networks, on the other hand, are a perhaps more theoretical object that thus require different approaches when designing path computation algorithms. In particular, the theoretical framework defined around dynamic networks allows setting various hypotheses on these entities, to better study the effects they may have on the complexity of well-known problems. I believe that this framework (with the right set of hypotheses) could very well be used to study some specific, real networks with specific behaviors (*e.g.*, satellite constellations) to offer strong theoretical results which could be relevant in practice.

I have had the opportunity to already work on some of these challenges with Quentin Bramas and Sebastien Tixeuil, by discussing the complexity of delay-constrained, least cost path computation in dynamic networks when considering route planning with time travel (a kind of network which is, to the best of my knowledge, hypothetical for now). We showed that some multi-constrained path computation problems can be solved in polynomial time in such dynamic graphs.

Similarly, the Lightning Network, a recent addition to the Bitcoin Blockchain, is a multi-metric network with peculiar metrics [Poon & Dryja 2016]. The Lightning Network allows performing transactions through a peer-to-peer network, minimizing the usage of the blockchain. A direct connection is not necessary to perform this transaction: the latter can be routed along intermediary nodes (which may ask for a small fee in exchange) if the capacity of the links supports the transaction. Interestingly, the transaction can be split across various paths, effectively introducing flow-related problems within a multi-metric network. This very rich and complex environment leads to problems that are intrinsically computationally expensive, even when considering apparently simple and relevant mono-optimization path computation (*e.g.*, minimizing the total amount of fees). This new ecosystem thus offers a novel challenging playground into which to study path computation.

Finally, two other contexts, with which I am currently not as familiar as the previous ones, deeply interest me.

First, the study of routing problems within satellite networks. Low Earth and Medium Earth Orbit satellite constellations are dynamic, but deterministic networks. As the computing capabilities of these devices may be limited, designing efficient routing algorithms fitter to this context is complicated. However, as orbits are predictable, the periodicity of the constellation can be leveraged to optimize routing decisions (and could be modeled and studied as a periodic dynamic graph) or even prevent congestion, by factoring in the non-uniform geographical distribution of the population [Wood *et al.* 2001]. I plan to become more familiar with satellite networks and their peculiarities and study the possibility of implementing

QoS or even FRR within these constellations that, despite their specificities, share numerous concepts with terrestrial networks.

Second, routing problems within quantum networks seem particularly fascinating. Although I do believe that the main challenges regarding quantum networks are still mainly related to physics, it seems that enough is known to start discussing and anticipating what a quantum routing protocol should be. As qubits cannot be cloned to be sent over a communication link, exchanging qubits between two distant parties A and B is done through quantum teleportation. Quantum teleportation requires not only classic communication links (meaning that the communication cannot be faster than classical communication, only more secure as it is resilient to quantum attacks) but also the creation of an entangled state between A and B (*i.e.*, both nodes should possess one half of an entangled pair of particles). This state can be achieved simply by having A prepare two entangled particles and send one to B, or, more interestingly, two entangled states $A-B$ and $B-C$ can be transformed into a single-hop entanglement $A-C$ through entanglement swapping. This allows A and C to communicate a qubit even if they do not share any physical link, thus essentially creating a quantum virtual overlay over the physical network [Schoute *et al.* 2016, Caleffi 2017]. These virtual links cannot be kept up for too long, and can only be used once. This begs to investigate *how* routing could be performed in such an environment. For example, whether (and how) virtual links should be pre-established (and refreshed) depending on the communications usually occurring within these networks [Schoute *et al.* 2016]. As quantum communication between satellites has been discussed, perhaps the periodicity and predictability of these constellations could be leveraged to this effect. If new virtual links should be constructed on the fly, choosing how the latter should be constructed requires the design of a complex routing metric, taking into consideration the stochastic nature of the process [Caleffi 2017]. The new problems and considerations as well as the ability to create new routing paradigms from scratch in such a peculiar environment makes the novel context of quantum networks very appealing to me.

6.1.4 Pedagogical tools

I've had the chance to be able to teach varied courses throughout the entirety of my thesis, to students within different cursus. This experience was very insightful and made me realize that some pedagogical challenges may prove as (if not more) stimulating than research ones. Consequently, I plan to get further involved in the conception and implementation of pedagogical tools. To this effect, I am currently working with a former Ph.D. student of the ICube laboratory, Amine Mohammed Falek, to conceive a tool assisting the teaching of path computation algorithms. This tool, (soberly) called Graph Theory Algorithm, allows students to code and visualize any algorithms on personalized graphs, to better understand their behavior. In time, we plan on extending this tool to support multi-metric graphs and dynamic graphs, or even serve as a support to ease the understanding of other types of network-related problems, such as intra or inter-domain routing.

Since my first research internship in 2016, I have had the opportunity to work on various research topics, such as distributed computing, topology discovery, network measurements, and (of course) resiliency and quality of service. Each time, the new challenges that I discovered were more intriguing than the last. In particular, networks offer problematics

that I find particularly enticing, as they often lead to interesting theoretical problems that are yet deeply anchored in reality.

These past three (to six) years have made me enthusiastic to continue exploring this field, and perhaps venture a bit beyond it, as distributed systems as a whole present other tempting areas such as autonomous robots and population protocols.

I, however, am under no illusion that I will be able to explore all of the interesting aspects that computer science has to offer. If anything, I have learned that proper research takes time and that it is possible to get (pleasantly) lost within even one square inch of these research areas. Nevertheless, I do hope that I will be able to explore (and continue wandering through) some of them as time goes on.

Chapter 7

Résumé français

7.1 Introduction

Au cours des dernières décennies, les défis que les réseaux informatiques ont dû relever ont considérablement évolué. Au début, ces défis tournaient principalement autour de la connectivité. Permettre la communication entre les machines d'un réseau est en effet déjà une tâche considérable, notamment en raison de la nature dynamique de ces réseaux qui peuvent subir des pannes ou d'autres changements topologiques. Lorsque les réseaux informatiques ont commencé à devenir de plus en plus prévalents, il est devenu évident que la connectivité ne devait pas seulement être assurée entre les dispositifs d'un même réseau, mais également entre différents réseaux indépendants, afin d'établir une connectivité globale entre tous les terminaux. Cet ensemble de réseaux indépendants (également appelés domaines ou ASes) interconnectés formera plus tard l'Internet moderne.

Pour obtenir une connectivité constante, des protocoles de routage ont été conçus, tels que OSPF [Moy 1998], IS-IS [rfc 1990], ou BGP [Rekhter *et al.* 2006]. Les protocoles de routage dictent la manière dont le routage doit être effectué, c'est-à-dire comment les informations topologiques doivent être échangées entre les équipements chargés de transmettre les données, et comment les meilleurs chemins de transmission de données doivent être calculés.

Les protocoles de routage appelés IGPs spécifient comment le routage *intra-domaine* doit être effectué. En général, le routage intra-domaine repose sur des chemins qui minimisent le coût IGP, une métrique additive fixée sur chaque lien par l'opérateur du réseau en fonction de ses souhaits de conception.

Bien que le routage entre réseaux (*i.e.*, *inter-domaine*) soit également spécifié par des protocoles de routage, ces protocoles diffèrent fortement des IGPs, car ils évoluent dans un contexte radicalement différent. En effet, chaque AS est gouverné par des entités administratives distinctes, avec des points de vue différents (ou même contradictoires) concernant les meilleurs chemins, en fonction des relations politiques ou économiques entre chaque domaine. Par conséquent, le routage inter-domaines est régi par un protocole de routage différent, BGP, conçu pour fonctionner dans un tel environnement.

Les communications sur Internet dépendent donc de l'interaction entre différents protocoles de routage : alors que les domaines traversés par le trafic sont décidés par le protocole BGP, son chemin au sein de chaque domaine est décidé par le protocole IGP qui y est déployé.

Avec la mise en place d'une connectivité stable et globale, l'Internet a commencé à devenir l'épine dorsale d'un nombre croissant de communications, allant de flux triviaux et sans importance à des flux critiques ayant des exigences strictes concernant les caractéristiques des communications, telles que la latence expérimentée. En conséquence, les défis à relever se sont orientés vers le contexte du routage qualitatif, afin d'améliorer le service procuré aux utilisateurs. Dans cette thèse, nous allons étudier plus en détail deux types principaux de

défis de routage qualitatif : la résilience et la qualité de service.

En effet, la popularité croissante des services en temps réel demande une disponibilité des services quasi permanente, malgré les nombreux changements non planifiés que subissent les réseaux informatiques. Une telle résilience n'est pas facile à obtenir, car les changements topologiques obligent à recalculer les meilleurs chemins de transmission. Cette reconvergence place le réseau dans un état transitoire et incohérent au cours duquel la connectivité peut être perdue ou le routage effectué peut être sous-optimal.

De nombreuses améliorations ont été proposées et mises en œuvre pour améliorer le temps de convergence des protocoles de routage. Des améliorations ont été apportées au protocole IGPs pour mieux gérer les événements survenant dans son propre domaine, ou au protocole BGP, pour offrir une meilleure résilience aux événements distants survenant dans des domaines éloignés.

Ce défi devient encore plus intéressant si l'on considère que les deux types d'événements et de protocoles sont enchevêtrés. Plus précisément, BGP doit re-converger même après des événements intra-domaine. En effet, lorsque plusieurs routes inter-domaines semblent intéressantes, celle permettant au trafic de sortir du domaine le plus rapidement possible (en suivant les meilleurs chemins intra-domaine) est sélectionnée. En raison de ce routage dit *de la patate chaude*, les routes interdomaines doivent être réévaluées même lors d'événements intra-domaines, un processus intrinsèquement long qui peut conduire à un routage sous-optimal ou à des interruptions de service de longue durée. Naturellement, la question suivante, à laquelle nous répondons dans ces travaux, se pose :

Peut-on assurer une reconvergence rapide de BGP après un événement interne ?

De plus, même des réseaux parfaitement résilients peuvent ne pas offrir des performances ou une QoS suffisantes pour certains types de trafic. En effet, les réseaux informatiques suivent généralement le paradigme *best-effort*, ce qui signifie que le réseau ne fournit aucune garantie quant à la qualité réelle des chemins d'acheminement empruntés. Bien que ce paradigme soit suffisant pour la plupart du trafic, certains flux modernes ont des exigences plus strictes. Par exemple, les flux de transactions financières échangent des données temps-réel sensibles qui impliquent des enjeux monétaires importants [Giacalone *et al.* 2015], obligeant les clients à payer un montant considérable pour des latences faibles. Bien que ces flux aient rarement à traverser l'Internet, ils peuvent s'appuyer sur des VPNs pour communiquer. Les ISPs fournissant de tels services VPN sont donc confrontés à des SLAs encore plus strictes que d'habitude.

S'assurer que le trafic concerné bénéficie effectivement de chemins d'acheminement respectant ces SLAs est une tâche complexe, tant d'un point de vue informatique que technique. Ces-dernier doivent en effet respecter la latence requise, mais devraient aussi idéalement viser à minimiser les coûts IGP, car ces derniers reflètent des choix de conception importants. Trouver des chemins qui minimisent le coût IGP tout en respectant une contrainte supérieure sur le délai nécessite de résoudre DCLC, un problème NP-Hard.

Ce problème devient encore plus intéressant si l'on considère les contraintes opérationnelles entrant en jeu. En effet, étant donné que les chemins calculés peuvent s'écarter des chemins standard de type "best-effort" (toujours utilisés par la majorité du trafic), des technologies supplémentaires sont nécessaires pour garantir que les flux concernés suivent effectivement ces chemins DCLC, c'est-à-dire pour *deployer* ces chemins. Actuellement, SR

(une implémentation du paradigme de routage à la source) est la technologie la plus populaire offrant de telles capacités. Au sein des réseaux déployant SR (parfois appelés domaines SR), des chemins spécifiques peuvent être spécifiés sous la forme de détours (appelés segments) ajoutés au paquet lui-même. Ces segments, prenant la forme d'instructions de routage interprétées par les routeurs sur le chemin, sont limités en nombre en fonction du matériel sous-jacent. Par conséquent, les chemins doivent non seulement respecter la latence requise, mais aussi pouvoir être déployés en moins de segments que la limite supportée par le du matériel. Cette métrique supplémentaire présente un comportement particulier par rapport aux métriques additives standard. En particulier, elle ne peut pas être représentée comme un poids statique supplémentaire sur le graphe et, si considérée naïvement, brise l'optimalité des plus courts chemins (une propriété fondamentale utilisée par la majorité des algorithmes de calcul de chemins). Plus précisément, il n'est pas trivial de prédire la manière dont une liste de segment va évoluer. Il est donc nécessaire d'être particulièrement prudent lorsque des distances sont élaguées de l'espace d'exploration. Il est donc nécessaire de concevoir des méthodes supplémentaires pour considérer cette métrique correctement, ce qui augmente encore la complexité d'un problème théorique déjà complexe. Ces défis donnent lieu à la question suivante, à laquelle nous répondons également dans cette thèse :

> Peut-on efficacement calculer des chemins DCLC pour SR ?

7.1.1 Contributions

Dans cette thèse, je présente plusieurs contributions, tournant autour des défis centrés sur la résilience et les QoS mentionnés ci-dessus. Bien qu'ils soient liés par le thème commun du routage qualitatif, ces deux défis (et donc, les contributions qui leur sont associées) sont distincts et indépendants.

Premièrement, nous fournissons plusieurs mécanismes permettant un calcul efficace des chemins DCLC déployables dans les domaines SR. Pour concevoir ces méthodes, nous proposons d'abord de nous appuyer sur l'imprécision des mesures de latence pour réduire la complexité théorique de DCLC.

Pour considérer correctement le nombre de segments, nous proposons une structure, le Graph SR, qui nous permet de considérer cette métrique correctement en englobant les trois métriques pertinentes (nombre de segments, délai et coût) et en transformant le nombre de segments en une métrique plus facile à gérer (le nombre de sauts). Nous proposons deux façons de s'appuyer sur cette construction pour calculer les chemins DCLC déployables.

La première consiste à explorer directement le graphe SR afin de considérer facilement les trois métriques. En effet, sur ce graphe augmenté, le nombre de segments redevient une métrique standard, permettant l'utilisation d'algorithmes et de techniques habituelles. Bien que le graphe SR soit particulièrement dense, nous montrons que cette méthode est viable en concevant un algorithme, BEST2COP, qui résout DCLC pour les domaines SR en explorant le SR Graph. En tirant parti des caractéristiques du SR et du multithreading, nous montrons que BEST2COP peut calculer des chemins DCLC déployables efficacement, même dans des réseaux à grande échelle. En particulier, nous étendons également BEST2COP pour tirer parti de la structure des réseaux d'opérateurs à grande échelle. Nous concevons un générateur de topologies multi-métriques large échelle, YARGG, pour évaluer notre algorithme, et

montrons que cette extension de BEST2COP est capable de résoudre DCLC pour SR en environ 1s dans des réseaux de 100 000 nœuds.

Explorer le graphe SR pouvant être coûteux (de par sa densité), nous proposons une autre méthode permettant de résoudre DCLC pour SR. En s'appuyant sur les informations contenues dans le graphe SR, il est possible pour un algorithme de calcul de chemin de calculer des chemins DCLC-SR tout en explorant le graphe originel. Il est cependant nécessaire de modifier l'algorithme originel, afin que ce dernier puisse convertir les chemins explorés en liste de segments à la volée. Il est également nécessaire de modifier la manière dont les chemins sont relaxés afin de considérer l'effet de cette métrique particulière sur la propriété de sous-optimalité des chemins. Nous concevons ainsi un algorithme de conversion, LCA, permettant aux algorithmes de calcul de chemins multi-métriques de connaître le nombre de segments nécessaires pour encoder les chemins à la volée, tout en explorant le graphe originel du réseau. Comme cette nouvelle métrique se comporte différemment des métriques standard, nous décrivons et prouvons formellement les modifications nécessaires à apporter pour que les algorithmes de calcul de chemins gèrent correctement ces particularités lorsqu'ils s'appuient sur LCA, et récupèrent correctement tous les chemins pertinents. Nous mettons en œuvre ces modifications sur un algorithme multi-métrique existant et évaluons ce dernier dans divers scénarios.

Deuxièmement, pour assurer une meilleure résilience du réseau lors d'événements intra-domaine, nous proposons OPTIC, qui rend les événements intra-domaine transparents pour le trafic de transit passant par le domaine malgré les effets néfastes du routage de la patate chaude. OPTIC parvient à ces résultats grâce à des structures de données et des algorithmes adéquats. En exploitant la manière dont les routes inter-domaines sont sélectionnées, OPTIC peut précalculer efficacement des ensembles de routes de secours *garanties* de contenir la nouvelle meilleure route vers une destination distante après *n'importe quel* événement interne. Lors d'un événement interne, la nouvelle route optimale peut être trouvée efficacement dans ces ensembles, au lieu de s'appuyer sur la lente convergence BGP. De plus, les destinations distantes partageant les mêmes ensembles de routes de secours partagent la même entrée en mémoire, ce qui permet des mises à jour groupées efficaces. Nous montrons par une évaluation théorique que le nombre d'ensembles distincts (et donc d'entrées à gérer) est largement inférieur au nombre d'entrées gérées traditionnellement par BGP. Comme OPTIC a été conçu avec le contexte émergeants des réseaux programmables à l'esprit, nous discutons en détail de la façon dont notre solution s'intègre dans ces architectures modernes grâce à une co-conception matérielle-logicielle intelligente et les avantages qui en découlent.

Plan

Cette thèse est organisée en six chapitres. Dans le chapitre 2, nous commençons par fournir le contexte nécessaire pour discuter de nos contributions et des travaux connexes. Nous introduisons d'abord les concepts fondamentaux de la théorie des graphes. Nous discutons ensuite du calcul de chemin et, plus généralement, du routage. Nous présentons les algorithmes de calcul des plus courts chemins mono-métriques les plus connus, ainsi que les concepts généraux nécessaires pour comprendre comment des métriques multiples affectent le calcul des plus courts chemins. Nous décrivons ensuite comment ces algorithmes s'intègrent dans les protocoles de routage avant d'aborder les technologies de TE, en mettant l'accent

sur SR. Dans le chapitre 3, nous détaillons les travaux connexes à notre contribution. Nous commençons par discuter de la résilience et examinons les solutions visant à améliorer le temps de convergence des protocoles ou à fournir des capacités de reroutage rapide, que ce soit dans un contexte intra-domaine ou inter-domaine. Nous passons ensuite en revue les travaux et propositions centrés sur SR. Enfin, nous passons en revue les algorithmes de calcul des chemins contraints, en visitant des heuristiques, des approximations, des schémas exacts et d'autres approches. Ces chapitres ne seront pas présentés dans ce résumé. Cependant, les concepts clés seront inclus dans les résumés des chapitres des contributions.

Dans le chapitre 4, nous présentons nos contributions liées au calcul de chemins multi-métriques déployables dans les domaines SR. Nous décrivons comment nous traitons la complexité de DCLC d'une manière adaptée aux réseaux d'opérateurs. Nous décrivons en détail notre construction, le graphe SR, et les deux algorithmes qui en dépendent, BEST2COP et LCA. Nous évaluons ensuite leurs performances, avant de conclure ce chapitre et de discuter des perspectives possibles à court et moyen termes.

Dans le chapitre 5, nous présentons OPTIC. Nous expliquons comment OPTIC construit et maintient des ensembles de routes de secours, permettant à BGP de réagir presque instantanément aux événements intra-domaines. Nous présentons ensuite notre évaluation théorique de OPTIC, montrant que son coût opérationnel est gérable, avant de discuter des perspectives possibles.

Enfin, nous concluons cette thèse dans le chapitre 6 en résumant nos résultats et en discutant des perspectives à long terme. Notez que le code de nos contributions et le code permettant de reproduire facilement nos expériences est mis à disposition en ligne (les liens étant listés au début de ce document).

7.2 Résumé du chapitre 4 : calcul de tunnels multi-contraints

Certains services de communication exigent des garanties fortes sur plusieurs métriques, notamment le délai de propagation. Bien que la minimisation du *coût IGP* soit essentielle pour satisfaire les intentions de l'opérateur – en prenant en compte les ressources consommées comme la bande passante relative, il peut également être nécessaire de garantir un délai maximal. Cependant, calculer un chemin respectant une contrainte sur une première métrique additive tout en en minimisant une seconde est un problème NP-Difficile connu sous le nom de DCLC, *Delay Constrained, Least Cost*. Résoudre DCLC requiert de visiter l'intégralité des distances non-dominées, *i.e.*, le front de Pareto [Deb 2005], pouvant croître de manière exponentielle avec le nombre de nœuds.

À cette difficulté inhérente au problème s'ajoute une difficulté technique : le déploiement des chemins calculés. Les opérateurs peuvent maintenant utiliser SR, *Segment Routing* [Filsfils *et al.* 2017], une technologie extensible de routage à la source. SR permet d'encoder la route *dans* le paquet sous forme d'instructions appelées *segments*.

En pratique, le nombre de segments est contraint à MSD *Maximum Segment Depth*, ≈ 10. À première vue, SR complexifie ainsi DCLC avec une dimension et une contrainte additionnelles. Cette métrique est ignorée par les solutions existantes qui ne prennent pas en considération la contrainte technique liée au déploiement des segments.

Dans cette section, nous commençons par présenter le problème DCLC plus en détail, avant de présenter la technologie SR et les difficultés induite par la nouvelle métrique que

cette technologie rajoute au problème. Nous détaillerons ensuite nos contributions, permettant de résoudre DCLC pour SR efficacement : BEST2COP et LCA.

7.2.1 Delay-constrained Least-Cost : le problème de calcul de chemins multi-métriques par excellence

Les problèmes de calcul de chemins multi-critères sont une généralisation des problèmes de calcul de chemins mono-critère bien connus. Comme leurs homologues mono-critères, ces problèmes prennent en entrée un graphe pondéré $G = (V, E, w)$, contenant $|V|$ nœuds et $|E|$ arêtes. Cependant, la fonction de coût w associe à une arête un vecteur de poids, représentant par exemple le coût IGP et la latence du lien en question. Cette fonction peut donc être défini comme $w : E \to \mathbb{N}^m$, où m est le nombre de composants du vecteur.

Lorsque l'on considère un chemin, défini comme une suite d'arête, les éléments des vecteurs de poids de chaque arête se combinent différemment, selon la métrique que ces-derniers suivent.

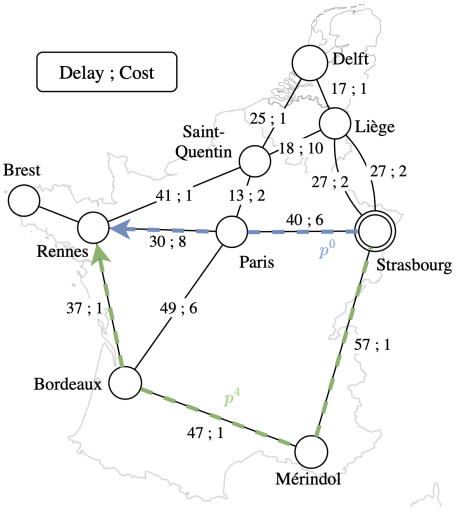
Nous nous concentrons ici sur des métriques additives. Dans ce cas, la *distance* d'un chemin est égale à la somme des vecteurs de poids de chaque arête qui le compose. Les métrique additives permettent notamment de modéliser la latence (car le délai d'un chemin est égale à la somme du délai de ses arêtes). Ils existent plusieurs algorithmes connus permettant de trouver un chemin minimisant une unique métrique additive. Cependant, il peut être intéressant de considérer plusieurs métriques additives. Par exemple, un flux premium peut nécessiter des chemins qui offrent de fortes garanties de latence tout en minimisant une métrique additive telle que le coût IGP. Étonnamment, la prise en compte d'un tel ensemble de métriques a un impact considérable sur la complexité des problèmes de calcul des chemins.

Cette augmentation drastique de la complexité de calcul peut s'expliquer par le fait que, contrairement au calcul de chemins monocritères, il n'existe pas de moyen naturel de définir un ordre total parmi un ensemble de chemins multicritères. Par conséquent, la notion d'optimalité devient floue, et le choix du chemin à étendre devient moins évident. Par exemple, il n'est pas possible de comparer deux vecteurs de distances $(1, 10)$ et $(10, 1)$ de manière non-arbitraire. Ces distances sont donc non-comparables. Dans le cadre des problèmes multi-critères, chaque distance non-comparable est considérée optimale et doit être explorée. Un nombre exponentiel de ces distances pouvant exister, ces problèmes sont en général NP-Difficile.

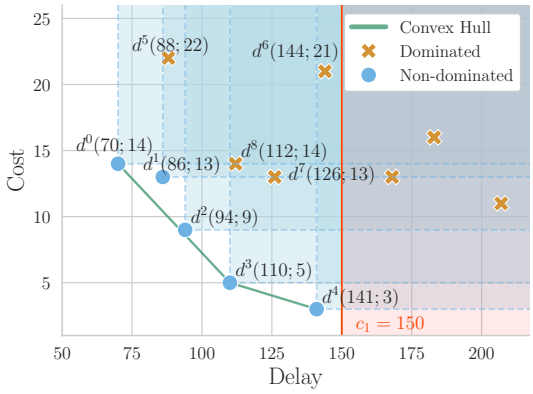
Un des problèmes multi-métriques les plus connu est DCLC, dans lequel le délai doit respecter une borne supérieure, et un coût (*e.g.*, le coût IGP) doit être optimisé. Denotons le délai M_1, et le coût M_2. Denotons la distance d'un chemin d, avec d_1 son délai et d_2 son coût. Enfin, notons le poids d'un lien w, avec w_1 son délai et w_2 son coût. Une contrainte c_1 s'applique au délai. Malgré la présence d'un objectif à optimiser (M_2), les distances explorées peuvent rester incomparables, comme illustré par la Figure 2.3.

La figure 7.1 montre un réseau hypothétique composé de 9 nœuds, couvrant plusieurs pays. Chaque lien est caractérisé par un vecteur de poids $(delai; couts)$. Le coût est fixé arbitrairement tandis que le délai est représentatif de la distance réelle entre les villes correspondantes.

Considérons le problème DCLC, de Strasbourg à Brest avec $c_1 = 150$. Plusieurs chemins simples vers Rennes peuvent être prolongés pour atteindre Brest, qui offrent différents compromis entre le délai et le coût. Une manière intuitive de visualiser leur distance est dans



(A) Réseaux hypothétique traversant la France, Belgique et les Pays-Bas. Les arêtes sont annotés par le délai et le coût des liens. Le coût IGP est arbitraire, mais le délai est représentatif de la distance géographique (ici, euclidienne) entre chaque ville.



(B) Distances des chemins simples de Strasbourg à Rennes, affichées en fonction de leur délai et coût. Certaines distances ont été retirées pour rendre la figure plus agréable à lire. Les distances dans la zone rouge viole la contrainte sur le délai. Les distances dans une ou plusieurs zones bleues sont dominées.

FIGURE 7.1: Exemple illustrant les notions de front de Pareto et de dominance.

l'espace *délai-coût*, montré dans la Figure 7.1b.

La distance d^0, par exemple, offre le meilleur délai, mais pour un coût élevé. Inversement, d^4 offre le meilleur coût, mais avec un délai élevé. Enfin, d^1, d^2, et d^3 offrent des compromis alternatifs. Bien que d^4 soit la distance dont le chemin associé résout cette instance de DCLC (avec $c_1 = 150$) en considérant Rennes comme destination, elle ne résout pas nécessairement la même instance de DCLC en considérant Brest comme destination. En effet, étendre cette distance pourrait violer la contrainte fixée après extension. Pour s'assurer qu'une solution est trouvée, des distances supplémentaires doivent être explorées. Ici, la distance d^3 devrait également être découverte et maintenue, au cas où d^4 finirait par violer la contrainte. Suivant le même raisonnement, les distances d^2, d^1, et d^0 devraient également être découvertes et maintenues, car n'importe laquelle d'entre elles pourrait devenir la distance DCLC vers Brest (ou tout autre nœud) en fonction des poids des liens restants à explorer. Le nombre de distances non-comparables à conserver et à étendre peut augmenter rapidement, rendant DCLC intractable.

Il est important de noter que certaines distances peuvent être élaguées de l'exploration. Puisque seuls les poids strictement positifs sont considérés, il est inutile d'étendre des distances qui violent déjà une contrainte. De plus, toutes les distances oranges visibles sur la Fig. 2.3b sont pires sur *tous les* critères qu'au moins une des distances indiquées en bleu. On dit qu'elles sont *dominées*. Par exemple, d^7 possède un délai et un coût plus mauvais que d^2 pour atteindre le même nœud (Rennes). La distance d^7 ne peut donc pas devenir une meilleure distance que d^2 après avoir été étendue. On dit que la distance d^7 est dominée par d^2. La Fig 2.3b représente cette relation de dominance. Toutes les distances situées dans une zone bleue sont dominées par la distance bleue d'où provient la zone bleue.

L'ensemble des distances non dominées, représenté en bleu sur la Fig. 2.3b, forme le *front de Pareto* de la solution [Deb 2005]. Pour résoudre DCLC, (ou MCOP/MCP), il

est nécessaire d'étendre *toutes* ces distances non dominées qui forment le front de Pareto. Ainsi, bien que la résolution de DCLC nécessite de retourner un seul chemin par destination, la totalité du front de Pareto doit toujours être explorée. Les distances dominées peuvent cependant être élaguées de l'exploration, réduisant drastiquement la complexité du problème en pratique.

Dans le pire des cas, toutes les distances sont non dominées, ce qui conduit à un front de Pareto de taille exponentielle, rendant le problème intractable.

7.2.2 Segment Routing et DCLC-SR

SR est une implémentation du paradigme de routage à la source : la source (ou le nœud le plus proche d'elle) calcule et choisit comment le trafic entrant doit être dirigé à travers le réseau. Contrairement au routage saut par saut, le chemin est donc entièrement décidé par le routeur en amont et non par une séquence de décisions de routage indépendantes. Le SR est, à l'heure actuelle, l'une des technologies TE les plus largement déployées et les plus populaires. En tant que tel, SR est au cœur de nos contributions.

SR implémente le routage à la source en ajoutant des instructions dans l'en-tête du paquet lui-même sous la forme d'une liste de *segments*. Les routeurs en aval ne prennent en compte que les instructions contenues dans le paquet pour prendre des décisions de routage.

Pour encoder ces instructions, les segments utilisés sont généralement des segments IGP. Comme leur nom l'indique, les segments IGP sont des instructions de routage liées aux informations relatives à l'IGP. Il existe de nombreux types de segments IGP. Dans cette thèse, nous nous concentrons sur les *segment de préfixe* (en particulier, les *segment de nœud*) et les *segment d'adjacence*, car ils fournissent un cadre suffisant pour déployer des chemins multicritères.

Les *segments de nœud* sont un type particulier de segments qui identifient un nœud au sein du réseau. En d'autres termes, l'utilisation d'un segment de nœud permet de demander à un routeur de transmettre le paquet à un nœud spécifique du réseau via les chemins les plus courts vers ce nœud (le trafic peut donc suivre n'importe lequel de ces plus courts chemins, si plusieurs existent). Nous notons un segment de nœud s ayant pour instruction d'envoyer un packet à v lu par le u par $s = (Node, u, v)$. Notez que si plusieurs chemins existent et que le routage multi-chemins (ECMP) est activé, le seul délai pouvant être garanti par l'utilisation d'un segment de nœud est le délai *maximal* des chemins encodés.

Les *segments d'adjacence* permettent d'ordonner aux routeurs de transmettre le paquet par une interface spécifique. Ils peuvent donc être utilisés pour imposer l'utilisation d'un lien spécifique le long du chemin. Nous notons un segment d'adjacence imposant un lien (u, v) par $s = (Adj, u, v)$. Si des liens parallèles existent, l'identifiant du lien x est rajouté et noté $s = (Adj, u, v, x)$

La Fig 7.2 illustre comment SR peut être utilisé pour encoder plusieurs chemin. La distance d^4, par exemple, était la plus courte distance IGP vers Rennes. En tant que tel, un seul segment de nœud est suffisant pour coder un chemin avec ces distances.

La distance d^0 était particulièrement intéressante, car c'était la distance minimisant la latence. Notons cependant que le fait qu'utiliser des segments de nœuds pour encoder ce chemin n'est pas suffisant pour garantir que la distance d^0 sera respectée. En effet, les

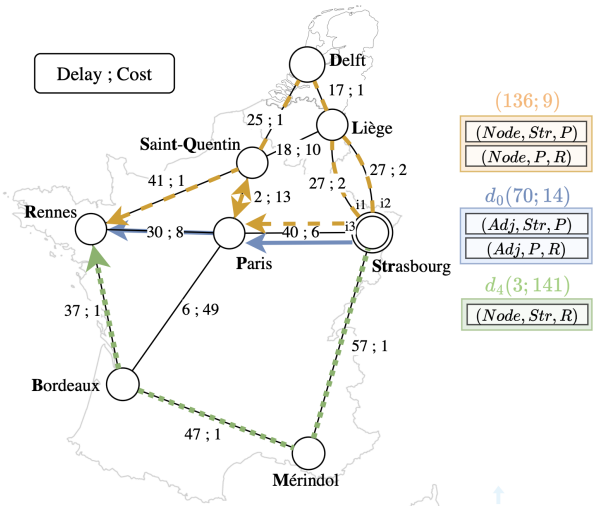


FIGURE 7.2: Encodage de chemins multi-critères via SR.

paquets peuvent prendre n'importe quel chemin ECMP de Strasbourg à Paris (indiqué en orange dans la Fig 7.2). Ainsi, les paquets peuvent prendre le lien direct ou passer par Liège. Une fois arrivés à Parisn le segment suivant est considéré, et les paquets sont envoyés à Rennes via les plus courts chemins. La *seule* latence qui peut être garantie est la *plus mauvaise latence parmi tous ces chemins possibles*, ici 136.

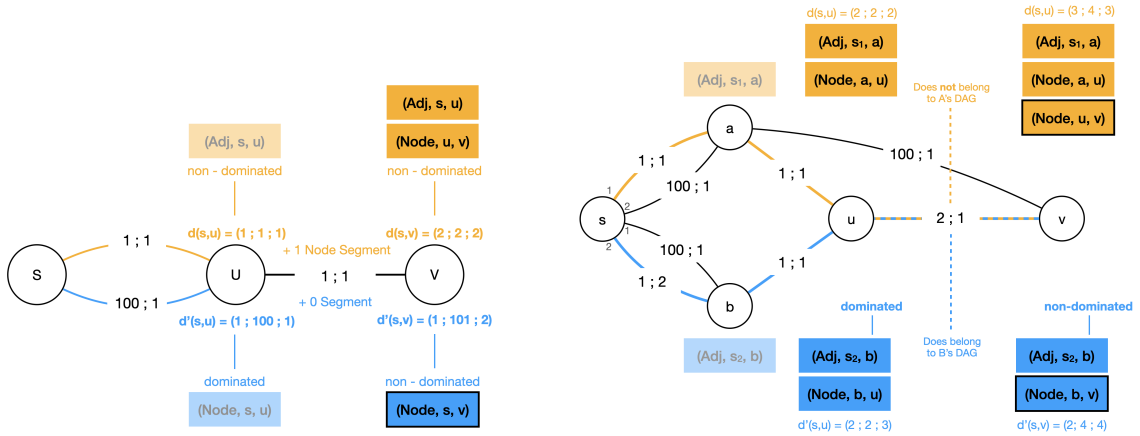
Pour imposer le chemin avec le meilleur délai (70), le lien entre Strasbourg et Paris ainsi que le lien entre Paris et Rennes doivent être explicitement imposés par l'utilisation de segments d'adjacence. Par conséquent, la liste de segments requise est $(Adj, Str, P) \mid (Adj, P, R)$.

En pratique, l'insertion de segments dans les paquets de données implique nécessairement une surcharge supplémentaire. Ainsi, le nombre de segments que l'on peut ajouter à un paquet est limité. Cette limitation dépend fortement du matériel sous-jacent. Les routeurs haut de gamme peuvent permettre des listes de segments allant jusqu'à environ 10 segments. Les équipements moins performants, en revanche, peuvent n'autoriser que 3 à 5 segments [Guedrez *et al.* 2016a]. Cette limite, appelée MSD, doit donc être prise en considération lors de l'encodage des chemins.

De ce fait, afin de résoudre DCLC tout en considérant le contexte opérationnel sous-jacent, il est nécessaire de considérer également le nombre de segments nécessaires pour encoder ces chemins, et de s'assurer que ce nombre reste sous la contrainte imposée par le matériel utilisé.

Il existe de nombreux algorithmes de calculs de chemins multi-métriques pouvant gérer une métrique supplémentaire, que nous notons M_0. Résoudre DCLC pour SR semble donc être une extension naturelle, demandant de rajouter une distance additionnelle d_0 dans le vecteur de distance des chemins.

Cependant, cette nouvelle métrique (le nombre de segments) est particulièrement délicate à considérer. En effet, cette dernière brise la propriété d'optimalité des sous-chemins utilisée par les algorithmes multi-critères. Plus précisément, nous avons vu que les distances dominés peuvent être élagués de l'exploration. En effet, lorsque des métriques traditionnelles sont considérées, une distance dominée restera dominée. Cette dernière n'offrant un compromis intéressant sur aucune métrique, il n'est donc pas nécessaire de l'étendre. Cependant, cette



(A) COMEBACK gadget. (B) UNDERDAG gadget.

FIGURE 7.3: Figures illustrant les péculiarités de la métrique SR.

propriété essentielle pour limiter la complexité des algorithmes utilisé doit être ré-visitée pour être adaptée à SR.

Ce défi est illustré par les gadgets UNDERDAG et COMEBACK, illustrés en Fig. 7.3a et 7.3b.

Considérons le gadget COMEBACK. Deux vecteurs délai-coût $(d_1; d_2)$ existent pour atteindre u depuis s : $(1; 1)$ et $(100; 1)$. Les deux peuvent être encodés avec un seul segment ((Adj, s, u) et $(Node, s, u)$ respectivement), conduisant aux distances $d(s, u) = (1; 1; 1)$ et $d'(s, u) = (1; 100; 1)$, où la première composante d_0 est le nombre de segments. Notez que suivant la définition standard de la dominance, $d'(s, u)$ est dominée, car pire sur toutes les métriques.

Les distances d' et d peuvent être étendues par (u, v), ce qui conduit au vecteur délai-coût $(2; 2)$ et $(101; 2)$. Cependant, alors que le vecteur $(2; 2)$ nécessite un segment de nœud supplémentaire (après le segment d'adjacence), le vecteur $(101; 2)$ reste encodable par un seul segment de nœud $(Node, s, v)$. Ainsi, après extension, les distances obtenues sont $d(s, v) = (2; 2; 2)$ et $d'(s, v) = (1; 101; 2)$.

Bien que résultant de l'extension d'une distance dominée, $d'(s, v)$ est non-dominée (et la seule distance ne violant pas une éventuelle contrainte stricte de c_0 de 1).

Dans le gadget COMEBACK, cet effet résulte du fait qu'un segment d'adjacence ne peut encoder qu'un seul lien. Par conséquent, les arêtes suivantes à encoder nécessiteront nécessairement des segments supplémentaires. A l'inverse, un segment de nœud peut encoder tout chemin se trouvant dans le DAG des plus court chemin de la destination intermédiaire courante. Ainsi, tant que les bords suivants restent à l'intérieur dudit DAG, utiliser des segments supplémentaires n'est pas forcément nécessaire.

Cependant, cet effet peut également se produire même si les deux listes de segments se terminent par un segment de nœud. Ceci est illustré par le gadget UNDERDAG. Considérons les vecteurs délai-coût de s à u $(2; 2)$ et $(2; 3)$, à partir des chemins $p(s, u) = (s_1, a), (a, u)$ et $p'(s, u) = (s_2, b), (b, u)$ respectivement. Ces distances et chemins peuvent être encodés par les listes de segments $S = (Adj, s_1, a)|(Node, a, u)$ et $S' = (Adj, s_2, b)|(Node, b, u)$ respectivement. Ainsi, leurs distances sont $d = (2; 2; 2)$ et $d' = (2; 2; 3)$. Notez que d' est dominée

par d, et que S et S' sont toutes deux composées de segments de même type. Ces deux listes de segments permettent actuellement de joindre u en utilisant comme destinations intermédiaires a et b respectivement.

Cependant, on remarque que l'arête (u, v) se trouve dans le DAG des plus court chemins de b. Par conséquent, le dernier segment de S', $(Node, b, u)$, peut simplement être modifié en $(Node, b, v)$ pour englober (u, v). Au contraire, l'arête (u, v) ne se trouve pas dans le plus court chemin DAG de a. Ainsi, le dernier segment de S, $(Node, a, u)$ ne peut pas être mis à jour pour englober (u, v), et un nouveau segment est nécessaire. Par conséquent, une fois mises à jour, les distances à v sont $d = (3; 4; 3)$ et $d' = (2; 4; 4)$. Une fois encore, la distance non dominée d' résulte de l'extension d'une distance dominée.

Cette découverte a un impact non-négligeable : les distances d'apparence dominées ne peuvent plus, à présent, être élaguées de l'exploration. De prime abord, ceci implique que la plupart (si pas toutes) les distances dominées doivent être étendues, rendant le problème intractable même sur des petits réseaux.

Pour résoudre ce défi et calculer des chemins DCLC pour SR efficacement, nous proposons trois contributions. Premièrement, nous définissons une transformation de graphe. Le graphe résultant de cette transformation, appelé le graphe SR, contient toutes les informations nécessaires au calcul de ces chemins (délai, coût et nombre de segments). De plus, sur ce graphe, le nombre de segments se transforme en une métrique traditionnelle. Nous définission ensuite deux manière d'utiliser ce graphe augmenté : l'explorer directement (BEST2COP) ou s'en servir afin de considérer le nombre de segments tout en explorant le graphe originel (LCA).

7.2.3 Le graphe SR

Le graphe SR représente les segments sous forme d'arcs dans un graphe augmenté : les arcs entre les nœuds u et v représentent les segments pouvant être utilisés pour guider le paquet vers v depuis u. Sur notre construction [Luttringer *et al.* 2020a], un arc peut représenter l'unique segment de nœud encodant les meilleurs chemins ECMP de u vers v. Dans ce cas, le poids IGP de l'arc est la distance IGP commune aux chemins encodés par le segment, et son délai est le temps de propagation maximum parmi ces chemins. Des segments d'adjacence peuvent s'ajouter, auquel cas les poids des arcs les représentant sont exactement égaux aux arcs initiaux qu'ils encodent. Seuls les arcs non-dominés sont conservés en cas d'arcs multiples.

Afin de réduire la complexité du problème, les délais des arêtes sur le graphe SR sont discrétisés. En effet, en pratique, il s'avère que M_1 permet aisément de borner la taille du front de Pareto. La métrique délai est par construction bornée par une contrainte c_1 stricte. De plus, bien que représentés avec une grande précision, les mesures réelles ont une fidélité, ou exactitude, beaucoup plus limitée à cause des défis techniques liés à la mesure du temps de propagation (e.g., synchronisation des instruments). Il est alors facile de discrétiser les distances M_1 sans perte d'information pertinente. Par exemple, avec une fidélité de 0.1ms, le nombre maximal de distances sur le front de Pareto est de $\Gamma = c_1 \times \frac{1}{0.1}$. La fidélité avérée des délais de propagation mesurés permet donc de stocker l'intégralité du front de Pareto dans une structure efficace car pré-bornée, par exemple un tableau indexé sur M_1 de taille Γ, et ce sans sacrifier l'exactitude de la solution.

Sur le graphe SR multi-métrique, les chemins originaux encodables en x segments sont

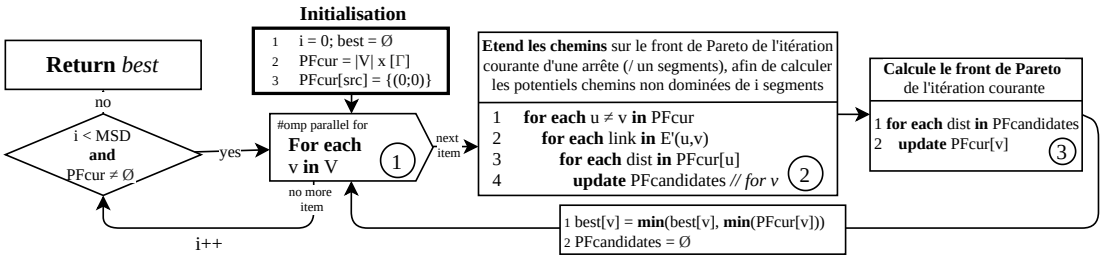


FIGURE 7.4: Diagramme représentant notre algorithme BEST2COP dans sa résolution de DCLC-SR

représentés par des chemins de x arcs. En explorant le graphe SR de manière similaire à l'algorithme de Bellman-Ford, l'exploration des chemins itère donc sur le nombre de segments. Cette exploration permet de ne visiter que les chemins compatibles avec SR (en s'arrêtant à l'itération MSD) et de découvrir rapidement les chemins DCLC : les segments sont construits sur l'IGP, que DCLC cherche à minimiser. Si les coût IGP sont majoritairement alignés sur le délai, il est vraisemblable que la solution requière peu de segments.

7.2.4 BEST2COP : Résoudre DCLC-SR sur le Graphe SR

Grâce au graphe SR, BEST2COP explore les chemins par nombre croissant de segments, *i.e.*, par nombre croissant d'arcs, et mémorise toutes les distances non-dominées : leur nombre est borné par $|V| \times \Gamma$ à chaque itération. BEST2COP est décrit à haut niveau sur la Fig. 7.4. Le front de Pareto de l'itération courante est mémorisé dans un tableau de taille $|V| \times \Gamma$, et contient initialement la distance $(0, 0)$ vers la source. A chaque itération i, BEST2COP parcourt tous les nœuds v depuis chaque nœud u. En combinant les distances vers u non-dominées découvertes à l'itération $i-1$ avec les poids des arcs (u, v), BEST2COP calcule tous les nouveaux chemins de i segments vers v, *candidats* au front de Pareto. Bien que filtrés, certains de ces chemins candidats peuvent être dominés en fin d'itération. La vérification de leur appartenance au front de Pareto n'est faite qu'une fois que toutes les distances ont été calculées (boîte 3) pour amortir la complexité de mise à jour du front de Pareto; cette vérification n'étant réalisée qu'une seule fois par itération et pas à chaque ajout/modification. Ces nouvelles distances sont à leur tour étendues d'un arc et ainsi de suite. Ce processus est répété MSD fois, afin d'explorer l'intégralité du front de Pareto 3D, mais uniquement celui des distances encodables avec SR : la solution à DCLC-SR est alors fournie vers l'ensemble des destinations.

Outre le maintien efficace du front de Pareto, les bonnes performances de BEST2COP proviennent également des structures utilisées. La taille du front étant bornée et prédictible grâce à la discrétisation, BEST2COP peut bénéficier de structure efficaces, par exemple des tableaux statiques de taille Γ. Cerise sur le gâteau, les itérations de la boîte 3 sont indépendantes (chaque itération explore des distances vers des nœuds v distincts), permettant de facilement paralléliser cette boucle. Le graphe SR Multi-Métrique étant a minima une clique, les charges des différents threads sont aisément distribuables de manière équitable.

Enfin, BEST2COP a également été étendu afin de profiter de l'existence de séparateurs dans les graphes de réseaux réalistes. En effet, la division physique et logique des réseaux large échelle en *aires* permet d'utiliser une approche *diviser pour mieux régner* afin de résoudre DCLC-SR efficacement. Plus précisément, DCLC-SR peut être résolu dans chacunes

des aires du réseau. Les distances trouvées peuvent ensuite être combinées au niveau des séparateurs via un simple produit cartésien. Cette approche permet à BEST2COP d'être utilisable et efficace même sur des réseaux modernes vastes. Cette variante est appelée BEST2COP-E.

BEST2COP possède une complexité de $O(MSD \times |V|^2 \times L \times \Gamma)$. Pour les $|V|$ voisins de $|V|$ nœuds, BEST2COP étend au pire Γ distances par L liens parallèles, le tout MSD fois. Ce coût peut être divisé par le nombre de threads utilisés. Dans l'évaluation, nous considérons $\Gamma = 1000$ et $L = 2$ sur le graphe SR (davantage que le nombre que nous avons observé en pratique).

7.2.5 Live Conversion Algorithm

Bien que BEST2COP soit efficace, l'utilisation du graphe SR n'est pas nécessairement idéale. En effet, BEST2COP a été conçu avec ce graphe à l'esprit. Il est donc conçu pour profiter de ces charactéristiques sans souffrir (excessivement) de sa forte densité.

Cependant, pour certains algorithmes, explorer directement le graphe SR peut induire un coût non-négligeable et impacter négativement les performances natives de l'algorithme. Pour cette raison, nous concevons également LCA, permettant à un algorithme de calculs de chemins multi-critère de résoudre DCLC-SR efficacement.

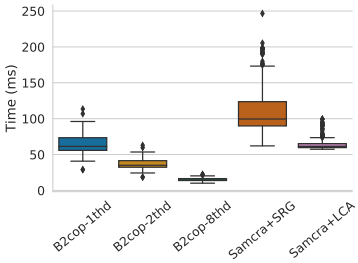
Pour ce faire, deux principaux défis doivent être résolus. Premièrement, là où le nombre de segments était connu nativement quand le graphe SR était exploré (car il était égal au nombre de sauts), il est ici nécessaire de convertir les chemins en segments lors de l'exploration, afin de connaitre le nombre de segments nécessaires pour l'encoder. Deuxièmement, si le graphe originel est exploré, le problème initial amené par la métrique SR (la perte d'optimalité des sous-chemins) doit être également être considéré. Pour résoudre ces défis, nous proposons deux solutions.

7.2.5.1 Une traduction lâche en liste de segments

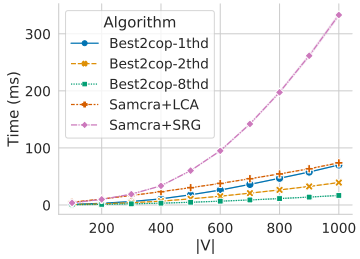
Nous proposons une méthode de traduction des chemins explorés en liste de segments de manière *lâche*. Plus précisément, cet algorithme ne vise pas à encoder un chemin précis, mais des chemins possédant une distance (coût et délai) équivalent au chemin considéré. Adopter une telle stratégie d'encodage permet de conserver les propriétés importantes des chemins (leurs métriques) tout en utilisant un nombre de segments réduit (car encoder un chemin unique précis requiert généralement davantage de segments, à cause d'ECMP).

Notre algorithme d'encodage suit une approche *greedy*, en tentant d'encoder itérativement le plus long sous-chemin possible en un unique segment. Intuitivement, un sous-chemin peut être encodé en un unique segment tant que c'est un plus court chemin et qu'il possède le coût maximal parmi ces chemins (*i.e.*,, la meilleure distance pouvant être garantie par un segment de nœud). Si un sous-chemin viole ces propriétés, un segment additionnel est nécessaire pour l'encoder. Nous définissons formellement cet algorithme, qui est ensuite prouvé.

Enfin, afin de garantir que la solution optimale est trouvée, nous redéfinissons les conditions permettant d'élaguer un chemin de l'espace d'exploration. Intuitivement, deux listes de segments peuvent évoluer différement si ces-dernières évoluent depuis une destination intermédiaire différente. Par exemple, si l'on considère la Fig. 7.3b, la liste de segments correspondant à la distance d de s à u évoluait actuellement depuis le nœud intermédiaire a. Á l'inverse, la liste de segments de la distance d' évoluait depuis le nœud intermédiaire



(A) Temps d'exécution de BEST2COP (1, 2, 8 threads) & SAMCRA sur un réseau réel.



(B) Temps d'exécution de BEST2COP & SAMCRA sur réseaux aléatoires pour $|V| = 100$ à 1000

FIGURE 7.5: Temps d'exécution de BEST2COP and SAMCRA dans divers scénarios.

b. L'arête (u, v) n'appartenant pas aux plus courts chemins de a vers v, mais appartenant aux plus courts chemins de b vers v, le nombre de segments de ces listes peuvent évoluer différent. Il est donc nécessaire d'étendre les deux distances.

En considérant les distances dont les listes de segments évoluent depuis des nœuds intermédiaires différents comme incomparables, suffisamment de chemins sont étendus pour résoudre DCLC-SR optimalement. Nous définissons formellement cette nouvelle relation, appelée *strong dominance*. Nous prouvons qu'étendre les chemins n'étant pas *strongly dominated* suffit à résoudre le problème optimalement, pour un surcoût polynomial en les dimensions du graphe originel.

Ces algorithmes sont ensuite implémentés dans un algorithme de calcul de chemins multi-critères existant, SAMCRA [Van Mieghem & Kuipers 2003], et évalué.

7.2.6 Évaluation

Réseaux réels Nous commençons par considérer une topologie réelle. Nous utilisons notre plus grande topologie disponible, composée de plus de 1100 nœuds et 4000 arêtes. Cette topologie décrit le réseau d'un opérateur de niveau 1 et n'est pas disponible pour le public [1]. Les temps d'exécution sont ensuite présentés dans la Fig. 4.9a. BEST2COP (1, 2, 8 threads) et SAMCRA (avec LCA et avec le graphe SR) sont exécutés pour chaque nœud comme source, ce qui donne les distributions présentées.

On peut voir que SAMCRA+SRG (*i.e.*, SAMCRA exécuté directement sur le graphe SR) présente les pires temps d'exécution parmi tous les algorithmes, avec une moyenne de 100ms, et atteignant 250ms au pire. Il est intéressant de noter que cela montre que l'exploration du graphe SR peut être préjudiciable à certains algorithmes. Par conséquent, les algorithmes qui ne sont pas conçus pour tirer parti de ses caractéristiques peuvent mieux s'en sortir en explorant la topologie originale, plus clairsemée, et en utilisant les informations contenues dans le graphe SR pour calculer le nombre de segments nécessaires. Ceci est visible sur les temps de calcul SAMCRA+LCA. Notre construction, couplée à notre algorithme de conversion, a permis à SAMCRA+LCA d'atteindre des temps de calcul très similaires à la variante mono-thread de BEST2COP, avec un temps d'exécution moyen de $\approx$ 60ms. Notez que BEST2COP, qui fonctionne sur le graphe SR lui-même, présente un temps d'exécution

[1] Bien que des ensembles de données topologiques publiques existent, ces topologies sont souvent trop petites et/ou ne possèdent pas d'évaluation de liens

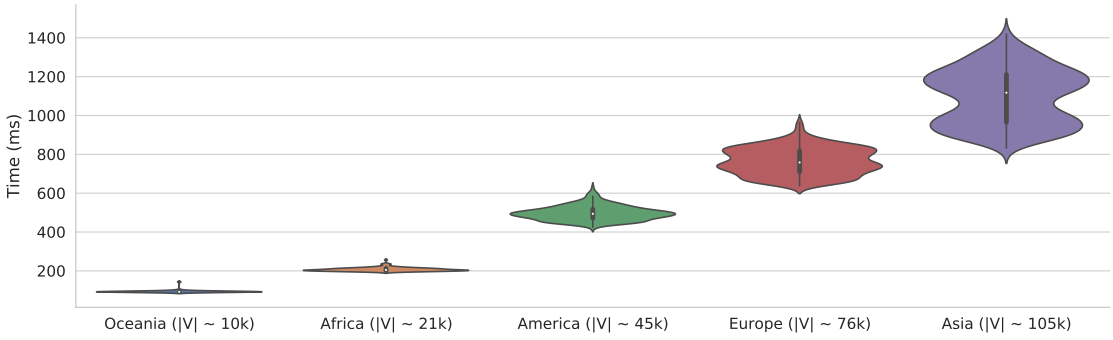


FIGURE 7.6: Temps d'exécution de BEST2COP-E sur des réseaux à très large échelle.

équivalent lorsqu'il s'appuie sur un seul thread. Cependant, lorsqu'il s'appuie sur plusieurs threads, BEST2COP surpasse SAMCRA dans toutes les exécutions, atteignant un temps de calcul de $\approx$ 25ms au pire en utilisant 8 threads, *i.e.*, trois fois plus rapide que SAMCRA.

Réseaux aléatoires Nous générons ensuite des graphes connectés bruts de $|V|$ nœuds en utilisant le modèle d'Erdos-Rényi. Les topologies générées ont un degré de $log(|V|)$. Tandis que les délais et les poids IGP sont choisis uniformément au hasard.

Bien que les temps de calcul soient légèrement plus élevés (en raison des évaluations aléatoires qui conduisent à un plus grand nombre de chemins non dominés), les résultats sont similaires à l'expérience précédente. Comme sur les réseaux réels, SAMCRA+LCA montre des résultats proches (sinon égaux) du temps d'exécution de BEST2COP. Néanmoins, même sur des réseaux aléatoires, BEST2COP reste trois fois plus rapide que SAMCRA lorsqu'il s'appuie sur 8 threads.

La manière d'utiliser le SR a un impact important sur l'algorithme sous-jacent. Étant donné que le SR n'est pas au cœur de la conception de la conception de SAMCRA, l'exploration de ce dernier entraîne un temps d'exécution élevé. Cependant, l'ajout de notre algorithme de conversion au sein de SAMCRA a permis à ce dernier d'atteindre des temps d'exécution compétitifs tout en résolvant 2COP. BEST2COP, qui explore directement le Graph SR directement, présente une exécution similaire à celle de SAMCRA+LCA lorsqu'elle repose sur un seul thread. En utilisant le multithreading, BEST2COP surpasse son SAMCRA dans tous les scénarios.

Topologie large-échelle À notre connaissance, bien que de tels réseaux existent, il n'y a pas de topologies à grande échelle disponibles publiquement. De plus, bien qu'il existe quelques générateurs de topologies [Quoitin *et al.* 2009, Medina *et al.* 2001] capables de générer des réseaux de taille arbitraire, ils ne permettent pas de générer de très grandes topologies multi-valuées (coût et délai).

Nous avons donc créé YARGG, une heuristique qui génère de tels réseaux tout en prenant des données géographiques en considération afin de rendre ces topologies réalistes. YARGG calcule une topologie large, réaliste et multi-zone. Le cœur du réseau s'étend sur un lieu géographique donné et possède des poids IGP simples et des délais réalistes. Les autres zones suivent un modèle hiérarchique standard à trois couches.

En utilisant YARGG, nous générons cinq topologies à grande échelle, à l'échelle du con-

tinent, et nous exécutons BEST2COP sur chacune d'entre elles. Les résultats de cette expérience sont présentés dans le diagramme en violon de la Fig. 7.6. En tirant parti de la structure du réseau, BEST2COP présente de très bonnes performances malgré l'échelle du graphe. Pour 10 000 nœuds, BEST2COP-E affiche un temps similaire à celui pris par sa variante plate pour $|V| = 2000$. De plus, BEST2COP-E semble évoluer linéairement avec le nombre de nœuds, restant toujours inférieur à une seconde pour $|V| = 75\,000$. Même lorsque le réseau atteint une taille de $\approx 100\,000$, BEST2COP-E est capable de résoudre 2COP en moins d'une seconde pour une fraction non négligeable des sources, et ne dépasse jamais 1,5s.

7.2.7 Conclusion et Perspectives

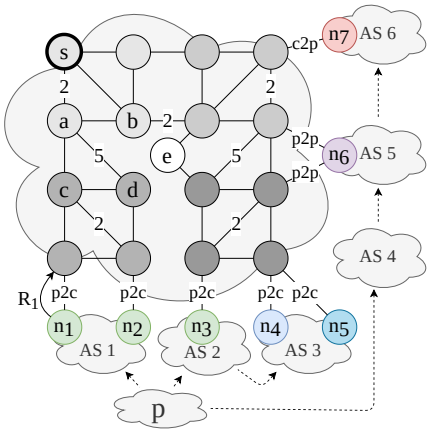
Nous proposons plusieurs contributions permettant de résoudre DCLC-SR. Grâce à une nouvelle construction, le graphe SR, nous proposons deux façons de considérer le nombre de segments lors du calcul des chemins. LCA s'appuie sur les informations contenues dans le graphe SR pour permettre aux algorithmes multicritères de traduire les chemins en liste de segments à la volée, pendant l'exploration du graphe original. BEST2COP explore directement le graphe SR et exploite les caractéristiques de ce dernier pour résoudre efficacement DCLC-SR.

Plusieurs défis intéressants peuvent encore être relevés. Plus particulièrement, le concept de LCA peut être adapté à d'autres défis que le calcul de DCLC-SR. En effet, la notion de *strong dominance* pourrait être utilisée pour adapter des algorithmes de calcul de chemins arbitraires pour SR. YARGG pourrait également être retravaillé et étendu, afin de devenir un outil à part entière utilisable par la communauté. Enfin, on pourrait étudier si ce type de routage peut être implémenté directement dans le plan de données, en utilisant des langages tels que P4. En particulier, plusieurs next-hop pourraient être poussés dans le plan de données pour chaque destination, afin de les sélectionner en direct en fonction du retard subi par le paquet.

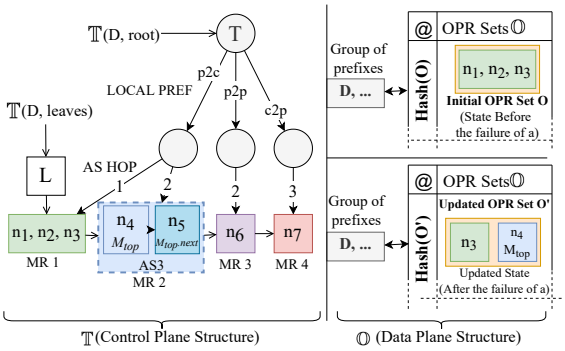
7.3 Résumé du chapitre 5 : une patate cuite à la perfection

Une des briques essentielles à Internet est l'acheminement des paquets en transit à travers les domaines qui le composent. Dans un domaine (ou système autonome, AS), les routeurs commutent les paquets en fonction du préfixe IP auquel la destination du paquet appartient. Lorsque le préfixe n'appartient pas à l'AS où le paquet est commuté, on dira qu'il s'agit de trafic en transit. Ce trafic est acheminé via un mécanisme de routage appelé patate chaude : l'opérateur de l'AS cherche à se débarrasser du paquet le plus vite possible selon la route interne la plus courte.

Cette route est obtenue en appliquant le processus de décision BGP dont l'une des dernières règles stipule que la meilleure route est celle qui minimise la distance dans l'AS — meilleure route parmi celles dont les critères strictement inter-domaines (préférence économique et longueur en saut d'AS en particulier) sont égaux. En d'autres termes, la sélection des routes BGP dépend des changements de l'IGP. Malheureusement, BGP converge lentement, même de manière interne avec iBGP. Pour augmenter la visibilité des routes à l'intérieur du domaine et diminuer le temps de convergence, les opérateurs peuvent utiliser *addPath* [Uttaro *et al.* 2016b] mais la mise à jour du meilleur prochain saut reste



(A) Exemple illustrant l'opportunité d'utiliser plusieurs passerelles aux caractéristiques différentes.



(B) Représentation des routes apprises dans le plan de contrôle et de données.

FIGURE 7.7: Des routes vers les prochains sauts de bordure : représentation et structuration.

relativement lente car il existe beaucoup de préfixes BGP à traiter (plus de 800K) : en cas de panne, beaucoup d'entrées doivent être mises à jour. PIC (Prefix Independent Convergence) [Filsfils *et al.* 2011] permet, grâce à une table de routage hiérarchique, de grouper les préfixes BGP pour réduire le nombre d'entrées et de basculer rapidement, en cas de changement IGP, vers une route de secours pré-calculée pour chaque groupe. Cependant la route de secours n'est pas nécessairement la route optimale (post-convergence BGP). De plus, PIC suppose la bi-connexité du réseau comme hypothèse garantie. Comme l'alternative de secours n'est ni nécessairement active ni optimale, PIC se repose encore et toujours sur BGP pour finalement re-converger à son tour. Or les groupes de PIC ne sont d'aucune utilité pour cela : la re-convergence BGP est à nouveau nécessaire pour l'ensemble des préfixes.

Cet aspect est critique : comment anticiper *et* garantir l'optimalité face à *tous* les événements sans les considérer un à un ? La réponse est plus simple qu'il n'y parait et est expliquée en détail dans la partie 7.3.1. Un ensemble de passerelles possédant les mêmes attributs inter-domaines, et suffisant pour assurer la 2-connectivité, contient nécessairement la meilleure passerelle après un changement IGP. Nous exploitons cette propriété pour construire facilement des ensembles de passerelles protecteurs après *n'importe quel* changement IGP. Ces ensembles, partagés en mémoire par groupe de préfixes, permettent de basculer très vite vers la nouvelle meilleure route BGP pour tous les préfixes d'un groupe.

Néanmoins, ces ensembles doivent potentiellement être mis à jour après un événement. Cette opération se doit donc également d'être efficace. Pour mettre en évidence la faisabilité d'OPTIC, et avant même de montrer que le nombre et la taille de ces ensembles sont limités en théorie comme en pratique (partie 7.3.3), nous allons expliquer comment facilement maintenir ces ensembles à jour lors de changement BGP ou IGP (partie 7.3.2). En d'autres termes, nous allons montrer qu'OPTIC est capable de maintenir efficacement ses groupes de préfixes ayant des ensembles de passerelles en commun pour toujours avoir un coup d'avance sur BGP. Alors que la convergence vers la nouvelle passerelle est quasi instantanée, la mise à jour des groupes pour garantir la protection face à un prochain événement à un coût au pire égal à celui de BGP.

7.3.1 Garantir une protection optimale face à tout évènement IGP

Dans cette partie et dans la suite de l'article, nous ferons l'hypothèse que la visibilité des routes est suffisante grâce à une solution comme AddPath et une architecture iBGP très simple, par exemple avec un seul route reflector. Des hypothèses plus réalistes sont abordées dans [Luttringer *et al.* 2021d], et OPTIC peut s'y adapter. Sur l'exemple donné en figure 7.7a, nous montrons les limites d'une solution comme PIC. Avec cette fonctionnalité de re-routage, le routeur s mémoriserait uniquement les routes via n_1 (optimal) et n_2 (secours si n_1 tombe en panne) vers le préfixe p. Après un événement IGP, PIC se contente de restaurer rapidement la connectivité vers p via la meilleure route mémorisée encore active. Or, si le lien $a - c$ tombe en panne, la route via n_3 devient optimale. PIC se contente de restaurer la connectivité vers n_1, qui est toujours joignable, et offre donc une route sous-optimale vers p jusqu'à ce que BGP converge. Pire encore, si le routeur a tombe en panne, PIC n'a pas de route de secours, car il fait l'hypothèse d'un réseau 2-connexe sans vérification préalable ni ajustement : s est alors déconnectée de p jusqu'à la re-convergence. **PIC n'apporte ni une protection complète, ni un re-reroutage optimal, et n'assure donc pas seul la re-convergence de BGP.**

Notre objectif est d'assurer une re-convergence optimale et immédiate avec une méthode efficace. Notre solution consiste à trouver, pour chaque préfixe p, un ensemble de passerelles garanti de contenir la nouvelle meilleure route après n'importe quel événement IGP. Pour calculer cet ensemble, (i) nous groupons les passerelles ayant les mêmes attributs inter-domaine (cad sans considérer la distance IGP) et (ii) nous empilons ces ensembles (en commençant par les passerelles avec les meilleurs attributs inter-domaine) jusqu'à ce que les passerelles de l'union de ces ensembles offrent deux chemins disjoints vers le préfixe p. L'union ainsi formée, qu'on appelle ensemble protecteur, contient donc suffisamment de passerelles pour tolérer tout changement IGP (chemins disjoints vers p). Comme les routes ont été considérées suivant leurs attributs inter-domaine, ces dernières resteront les meilleures routes après n'importe quel changement IGP car ces changements ne modifient pas les attributs inter-domaine des routes. Après un événement IGP, pour basculer sur la nouvelle route optimale de manière quasi-immédiate, il suffit donc de sélectionner dans l'ensemble protecteur la passerelle avec le poids IGP le plus faible. Les préfixes ayant un ensemble protecteur identique le partagent en mémoire : ainsi, une unique mise à jour de l'ensemble protecteur bénéficie à tous les préfixes.

Sur la Fig. 7.7a, l'ensemble des passerelles possédant les meilleurs attributs inter-domaine $\{n_1,n_2,n_3\}$ suffit à offrir deux chemins disjoints vers p. Si le nœud a est supprimé, la nouvelle meilleure passerelle est celle possédant le plus petit poids IGP : n_3. Cependant, après la panne de a, cet ensemble composé uniquement de n_3 n'offre plus deux chemins disjoints vers p : il est nécessaire de rajouter l'ensemble des passerelles possédant les deuxièmes meilleurs attributs inter-domaine afin de re-créer un ensemble protecteur.

7.3.2 Toujours un coup d'avance, ou la gestion efficace des groupes

La principale difficulté à surmonter est de garantir une reconstruction efficace des groupes après un changement IGP : la bascule vers la meilleure passerelle étant déjà réalisée, comment préparer efficacement les prochains groupes protecteurs pour anticiper n'importe quel futur événement IGP ? Sur la Fig. 7.7, on peut observer que le plan de contrôle d'OPTIC est construit sur base d'un arbre de préfixes $\mathbb{T}$ dont les feuilles $\mathbb{L}$ sont triées selon les attributs

inter-domaine. Les premières (meilleures) feuilles permettent de construire les ensembles protecteurs, transférés dans le plan de données $\mathbb{O}$. C'est sur ces ensembles réduits que pointent les groupes de préfixes, une fonction de hachage étant appliquée aux contenus des ensembles afin de les identifier de manière unique. Ainsi, dans le plan de données, un groupe de préfixes est associé à un ensemble de passerelles contenant la meilleure route courante et la meilleure route après tout changement IGP.

Après un changement IGP OPTIC opère une bascule vers la meilleure passerelle en appliquant un simple minimum dans chaque ensemble. Le changement IGP est donc pris en compte de manière optimale et quasi-instantanée. Ensuite, OPTIC se met à jour afin d'être prêt pour le *prochain* changement IGP. Pour chaque groupe de préfixe, si les passerelles de l'ensemble protecteur associé offrent toujours deux chemins disjoints vers p, ce-dernier reste inchangé. En revanche, si cette propriété n'est plus vérifiée, le groupe est mis à jour via les informations contenues dans $\mathbb{L}$. Cette mise à jour s'opère préfixe par préfixe dans le groupe concerné (car ils ne partagent pas nécessairement le même $\mathbb{L}$). **OPTIC opérant à la granularité des groupes de préfixes, son coût de mise à jour pour anticiper le prochain événement IGP est inférieur ou au pire égal au coût de BGP pour réagir à l'événement courant.** Sur la Fig. 7.7a, seule la panne du nœud a provoque une modification du groupe (pour le préfixe p). Son effet est visible dans la Fig. 7.7b : la passerelle n_4 doit être ajoutée car n_1 et n_2 sont inaccessibles. Dans tous les autres cas, l'ensemble IGP arrondi n_1, n_2, n_3 est suffisant pour assurer la protection, même après une panne.

Après une annonce BGP Quand une route BGP est apprise/modifiée, il suffit de l'insérer (resp. la modifier) dans $\mathbb{T}$ et d'appliquer les éventuels changements associés dans $\mathbb{O}$ si la route modifie effectivement l'ensemble protecteur. Le coût de cette mise à jour d'OPTIC est équivalent à celui de BGP.

7.3.3 Analyse du nombre de groupes : un plan de données compact

Nous fournissons ici[2] un modèle d'analyse assez défavorable car ne prenant pas en compte les préférences régionales limitant le nombre d'annonces à considérer dans la réalité. Soit un AS avec B passerelles bi-connectées annonçant P préfixes au total. Chaque préfixe D est annoncé par un sous ensemble $b \leq B$ de passerelles, choisies aléatoirement selon une loi uniforme. Pour chaque préfixe, l'étalement de la politique BGP est représenté par un entier entre 1 et ps choisi aléatoirement selon une loi uniforme. Cela implique que chaque sous ensemble de taille $n \leq b$ donnée a la même probabilité d'existence. Notre modèle calcule le nombre $|\mathbb{O}| = |\mathbb{O}_{B,P,ps}|$ d'ensembles uniques en fonction de B, P, et ps. La quantité $|\mathbb{O}_{B,P,ps}|$ est donc le nombre d'ensembles de passerelles distinctes, i.e. le nombre de groupes de préfixes.

Suivant sa configuration interne, un ensemble de taille n ($2 \leq n \leq b$) est dans $\mathbb{O}_{B,P,ps}$ avec une probabilité $\mathbb{P}_{B,P,ps,n}$ ou $\mathbb{P}'_{B,P,ps,n}$. Comme il existe respectivement $\binom{B}{n}$ ou $B\binom{B-1}{n-1}$ de ces ensembles, nous en déduisons (avec p_n et p'_n la probabilité pour un ensemble d'une certaine

[2]Les résultats présentés sont issus d'une version optimisée d'OPTIC expliquée ici : https://optic-icube.github.io/

Table 7.1: Nombre de groupes distincts ($|\mathbb{O}|$) selon plusieurs configurations.

Type of AS	# gateways per class	# prefix per class	# distinct OPR sets	OPR sets median size	Lower bound
Stub	$(10; 20; 0)$	$(700K; 100K; 0K)$	3945	4	235
Tier 3	$(10; 50; 100)$	$(500K; 200K; 100K)$	46 010	3	6219
Tier 2	$(5; 500; 2000)$	$(500K; 200K; 100K)$	263 219	2	197 194
Tier 1	$(0; 50; 5000)$	$(0K; 600K; 200K)$	232 180	2	199 633

configuration interne de taille n avec b passerelles par préfixe et des distances inter-domaines entre 1 et ps) :

$$|\mathbb{O}_{B,P,ps}| = \sum_{n=2}^{b} \binom{B}{n} \mathbb{P}_{B,P,ps,n} \; + \; B\binom{B-1}{n-1} \mathbb{P}'_{B,P,ps,n} \tag{7.1}$$

$$\mathbb{P}_{B,ps,P,n} = 1 - \left(1 - \binom{B}{n}^{-1}\right)^{p_n P} \quad , \quad \mathbb{P}'_{B,ps,P,n} = 1 - \left(1 - \left(B\binom{B-1}{n-1}\right)^{-1}\right)^{p'_n P} \tag{7.2}$$

$$p_n = \sum_{i=1}^{ps} \binom{b}{n} \frac{1}{ps^n} \left(1 - \frac{i}{ps}\right)^{b-n} \quad , \quad p'_n = \sum_{i=1}^{ps} b\binom{b-1}{n-1} \frac{1}{ps^{n-1}} \frac{i-1}{ps} \left(1 - \frac{i}{ps}\right)^{b-n} \tag{7.3}$$

Le tableau 7.1 décrit le nombre de groupes obtenus en raffinant l'analyse. Les AS étant triés en fonction de la dispersion de leurs préférences locales en trois classes (fournisseurs, paires, clients) et de leur structure topologique — nombre de passerelles et préfixes appris par classes de voisins. **Pour la majorité des AS, tels que les réseaux Stubs et les réseaux de transit avec un nombre de passerelles inférieur à la centaine, le nombre de groupes $|\mathbb{O}|$ est très réduit.** Pour les gros réseaux de transit, le nombre de groupe distincts $|\mathbb{O}|$ est fortement dépendant de la décomposition en sous classes : pour les grands Tier 1, $|\mathbb{O}|$ est relativement élevé mais OPTIC est de toute façon proche de la borne minimale pour la protection. Globalement, comme le nombre de groupes est (bien) plus faible que le nombre d'entrées BGP, OPTIC est à même de répondre rapidement à la plupart des changements IGP très efficacement.

7.3.4 Conclusion

OPTIC découple l'IGP de BGP à l'aide de groupes de préfixes BGP peu nombreux, petits et stables. Chaque groupe pointe vers un ensemble de routes protecteur commun offrant deux chemins disjoints vers le préfixe et contenant les meilleures routes pre- et post-convergence pour n'importe quel événement IGP. Afin de protéger le trafic de transit en cas de nouveau changement, c'est à dire pour anticiper la *prochaine* panne avec de nouveaux ensembles protecteurs, le coût de la mise à jour des groupes d'OPTIC est limité pour être inférieur – ou au pire égal bien que généralement très inférieur – au temps pris par BGP pour se "remettre de la panne précédente" ! Lors de changements internes ou de pannes de bordure, OPTIC a un coup d'avance sur BGP pour un coût moindre : la bascule vers la route post-convergence est quasi-immédiate et les prochains groupes protecteurs reconstruits efficacement si nécessaire.

Á l'heure actuelle, OPTIC reste à l'état d'idée, bien qu'une implémentation partielle de son plan de contrôle ait été faite sur Free Range Routing. L'étape suivante et les perspectives directes de cette contribution concerne donc son implémentation et son évaluation dans un environnement réaliste. De plus, OPTIC est parfaitement compatible avec les architectures modernes des réseaux programmables. Une implémentation de son plan de données en P4

est actuellement en cours de réalisation, afin de démontrer la faisabilité d'OPTIC dans un réseau d'opérateur réel.

7.4 Conclusion

Les travaux présentés dans cette thèse se centraient autour du routage qualitatif.

Nous avons motivé l'intérêt de résoudre DCLC, ainsi que la complexité additionnelle apportée par le contexte opérationnel souvent négligé par les algorithmes de plus courts chemins. Nous avons conçu différentes méthodes et algorithmes permettant de résoudre DCLC efficacement dans ce contexte opérationnel.

Nous avons ensuite étudié les effets néfastes des interactions entre BGP et l'IGP, entraînant des temps de convergence long. Avec OPTIC, nous avons proposé un moyen de ramener ce temps de convergence à un temps marginal.

Ces contributions ont été formalisées et prouvées. Ces-dernières ont également été évaluées, théoriquement ou via des évaluations pratiques. Le code de ces contributions a été mis en ligne. Enfin, des perspectives possibles ont été énoncées.

List of Figures

List of Tables

Glossary

2COP 2-Constrained Optimal Paths xi, 95, 100, 101, 103, 105, 106, 128, 131, 133–137, 143, 148, 150, 151, 172, 197, 198

ABR Area Border Router 24, 30, 31, 61, 134, 135, 144, 147, 205

APSP All Pair Shortest Path 11, 76, 104, 107, 151

AS Autonomous System vi, vii, 1, 4, 26, 33–45, 47, 54, 60–65, 68, 153–155, 159, 165, 167–173, 177–179, 183, 198, 201, 202, 205–207

ASBR Autonomous System Border Router 38, 65

ASIC Application-Specific Integrated Circuit 25

BEST2COP Best Exact Segment Track for 2-Constrained Optimal Paths ix, 3–5, 12, 14, 23, 26, 54–56, 70, 71, 76, 77, 83, 84, 87, 89–93, 95, 97, 101, 107, 127–129, 131, 133–138, 140–143, 147–152, 177, 178, 185–188, 193–198, 206, 207

BEST2COP-E BEST2COP-Extended 97, 136, 140, 143, 144, 147–149, 195, 197, 198, 207

BFM Bellman-Ford-Moore 10, 12–16, 21, 22, 54, 71, 79, 80, 84, 89, 90, 92, 128, 205

BGP Border Gateway Protocol vi, vii, xi, 1, 2, 4, 5, 7, 24, 33–45, 47, 49, 54–56, 59–69, 93, 153–160, 163–166, 171–174, 177–179, 183, 184, 186, 187, 198–203, 205–208

BSD Berkeley Standard Distribution 32

BSID Binding Segment 53, 54, 179

BSP Bicriterion Shortest Paths 101

CBF Constrained Bellman-Ford 85

CCLD Constrained Cost Lowest Delay 105, 138

DAG Directed Acyclic Graph 16, 27–29, 53, 54, 57, 72–75, 104, 107, 109, 110, 113, 122, 192, 193, 205, 206

DCBF Delay-Constrained Bellman-Ford 79

DCCLS Delay-Cost Constrained with the Lowest #Segment 106, 137

DCCR Delay-Cost Constrained Routing 82

DCLC Delay-Constrained, Least-Cost ix, xi, 3–5, 18–22, 47, 54–56, 69–72, 76–78, 80, 82–85, 87–91, 93, 95–103, 105, 106, 115, 121, 123, 128, 137–140, 142, 148–151, 177, 184–191, 193–196, 198, 203, 206

DCR Distributed Delay Constrained Routing 78

DCUR Delay Constrained Unicast Routing 78

DDC Data Drive Connectivity 59

DEB Dual Extended Bellman-Ford 79

eBGP External BGP 36–42, 44, 45, 63, 153, 155, 205

ECMP Equal Cost Multi Paths 16, 24, 25, 28, 29, 49–52, 57, 72–74, 76, 93, 99, 103–109, 111, 113, 115, 162, 190, 191, 193, 195, 206

EGP Exterior Gateway Protocol 33, 34, 39

EPE Egress Peer Engineering 173

EPTAS Efficient Polynomial Time Approximation Scheme 22

FIB Forwarding Information Base 25–29, 31, 32, 35, 37, 40, 41, 45, 47, 56, 57, 59, 62, 65–68, 150, 174, 175, 206, 208

FIFO First In First Out 84

FIR Failure Insensitive Routing 59

FPTAS Fully Polynomial Time Approximation Schemes 22, 23, 87, 91, 101

FRR Fast ReRoute 31, 44, 45, 55–59, 61, 65–67, 69, 93, 138, 150, 151, 173, 178, 181, 206

GGP Gateway to Gateway Protocol 33, 34

GLPH Generalized LPH 90

GWINFO Gateway Information Protocol 32

H_DCC Heuristic for Delay-Cost Constraint 82

H_MCOP Heuristic for Multi-Constrained Optimal Path 82, 92

HFIB Hierarchical FIB 66, 67, 156

HIPH Hybrid Interval Partitioning Heuristic 90–92

HLP Hybrid Link-state Path-vector 60, 178

iBGP Internal BGP 7, 36–45, 47, 56, 62–64, 66, 68, 69, 153, 155, 172, 173, 198, 200, 205, 206

IETF Internet Engineering Task Force 33

IGP Internal Gateway Protocol vi, vii, xi, 1, 2, 10, 17, 18, 24, 26–29, 32–36, 38–45, 47, 48, 50, 52–54, 57–59, 62, 64–73, 76, 97–104, 110, 135, 137–139, 141, 143–147, 150, 151, 153–167, 171–175, 177, 178, 183, 184, 188, 190, 193, 194, 197–203, 205–208

IPH Internal Partitioning Heuristic 90

IS-IS Intermediate System to Intermediate System 1, 29, 133, 183

ISP Internet Service Provider 2, 102, 141, 147, 167, 184

LARAC LAgrange Relaxation-based Aggregated Cost 80, 90, 92

LC Least-Cost 78, 86, 91

LCA Live Conversion Algorithm ix, 4, 5, 70, 72, 75, 76, 92, 93, 97, 106–109, 115, 116, 119–121, 123, 127, 138, 140–143, 148–152, 177, 178, 186–188, 193, 195–198

LD Least-Delay 78, 82, 86, 91

LDP Label Distribution Protocol 48, 49

LFA Loop Free Alternate 58, 150, 173, 206

LGH Limited Granularity Heuristic 89

LIFO Last In First Out 84

LOUP Link-Ordered Update Protocol 62

LPH Interval Partitioning Heuristic 89, 90, 92

LSA Link-State Advertisement 29, 31

LSDB Link-State Database 29

MCOP Multi Constrained Optimal Path 17–23, 75–77, 79, 81–83, 87–89, 91, 93, 96, 189

MCP Multi Constrained Path 17, 18, 20, 77, 79, 81, 82, 89, 189

MED Multi-Exit Discriminator 39–43, 63–65, 154, 155, 158–160, 162, 165–167, 171, 173, 205–207

MLU Maximum Link Utilization 70, 71

MPLS Multi-Protocol Label Switching 38, 48, 50, 51, 57, 148

MR MED-aware Rounded 160–162, 164–169, 207

mRPC Metrics and Routing Policies Compliant 57, 60

MSD Maximum Segment Depth 52, 53, 72, 75, 76, 91, 92, 96, 97, 99–101, 115, 128–130, 133, 135, 136, 138–140, 150, 187, 191, 194

NCA Network Computing and Applications 97

NH Next-Hop 24, 39, 40, 66, 67, 155–157, 159, 160, 174, 207

NR-DCLC Non-Linear Relaxation for DCLC 82, 225

OPR Optimal-Protecting Rounded set 153, 160–175, 207

OPTIC Optimal Protection Technique for Inter-intra-domain Convergence ix, xi, 4, 5, 26, 36, 45, 55–57, 60–63, 68, 69, 93, 153, 154, 156–167, 170–175, 177–179, 186, 187, 199–203, 207

OSPF Open Shortest Path First 1, 24, 29–33, 44, 45, 60, 72, 133, 134, 149, 183

PIC Prefix Independent Convergence 65–69, 156, 170, 199, 200

PQ Priority Queue 85

PTAS Polynomial Time Approximation Scheme 22

QoS Quality of Service 2, 3, 29, 181, 184, 185

RIB Routing Information Base 23–26, 28, 29, 32, 35, 37, 40, 41, 45, 48, 49, 60, 208

RIP Routing Information Protocol 32–34

RSVP Resource Reservation Protocol 48, 49

RSVP-TE RSVP - Traffic Engineering 48, 49, 69, 99

SAMCRA Self-Adaptive Multiple Constraints Routing Algorithm ix, 75, 85, 92, 93, 140–143, 148, 196, 197, 206, 207

SCION Scalability, Control, and Isolation On Next-Generation Networks 60

SF-DCLC Selection Function Based DCLC 78, 82

SLA Service Level Agreement 2, 22, 23, 83, 87, 151, 184

SOUP Simple Ordered Update Protocol 62

SPA Shortest Path Algorithms 11, 12, 15, 16, 79–81, 83, 86

SPC Shortest Path Computation 23, 84

SPF Shortest Path First 29, 31, 32

SPP Shortest Path Problem 10, 11

SR Segment Routing vi, vii, ix, xi, 3–5, 38, 49–56, 58, 69–73, 76–78, 91–93, 95–107, 109, 115, 117, 121, 123, 127–129, 133–135, 137–140, 142, 143, 148–151, 172, 173, 177, 179, 184–188, 190–198, 206, 207

SRG SR Graph 140–143, 196

SSR+DCCR Search Space Reduction + DCCR 82

TAMCRA Tunable Accuracy Multiple Constraints Routing algorithm 75, 82, 83, 85, 89, 140

TCAM Ternary Content-Addressable Memory 25, 59, 175, 179

TE Traffic-Engineering 4, 5, 47–49, 54–56, 69–72, 77, 93, 99, 100, 102–104, 133, 139, 143, 144, 148, 150–152, 172, 179, 186, 190

TI-LFA Topology Independent LFA 58, 206

VPN Virtual Private Network 2, 184

YARGG Yet Another Realistic Graph Generator ix, 3, 97, 139, 144–147, 149, 185, 197, 198, 206, 207

Bibliography

[6net 2008] 6net. IPv6 deployment guide. Javvin, 2008.

[Adams 2020] Heidi Adams. *Segment routing in the 5G era.* https://www.juniper.net/assets/de/de/local/pdf/articles/3200083-en.pdf, 2020.

[Alfroy 2020] Thomas Alfroy. *Towards a PoC for an Optimal Protection Technique for Inter-intra- domain Convergence.* PhD thesis, Master's thesis, University of Strasbourg, 2020.

[Almes *et al.* 1999] G. Almes, S. Kalidindi and M. Zekauskas. *A Round-trip Delay Metric for IPPM.* RFC 2681, RFC Editor, September 1999.

[Almes *et al.* 2016] G. Almes, S. Kalidindi, M. Zekauskas and A. Morton. *A One-Way Delay Metric for IP Performance Metrics (IPPM).* STD 81, RFC Editor, January 2016.

[Aneja & Nair 1978] Y. P. Aneja and K. P. K. Nair. *The constrained shortest path problem.* Naval Research Logistics Quarterly, vol. 25, no. 3, page 549–555, 1978.

[Aneja *et al.* 1983] Y. Aneja, V. Aggarwal and K. Nair. *Shortest chain subject to side constraints.* Networks, 1983.

[Anh Quang *et al.* 2018] Pham Tran Anh Quang, Jean-Michel Sanner, Cedric Morin and Yassine Hadjadj-Aoul. *Multi-objective multi-constrained QoS Routing in large-scale networks: A genetic algorithm approach.* In 2018 International Conference on Smart Communications in Network Technologies (SaCoNeT), pages 55–60, 2018.

[Atlas & Zinin 2008] Alia Atlas and Alex D. Zinin. *Basic Specification for IP Fast Reroute: Loop-Free Alternates.* RFC 5286, September 2008.

[Aubry *et al.* 2016a] François Aubry, David Lebrun, Stefano Vissicchio, Minh Thanh Khong, Yves Deville and Olivier Bonaventure. *SCMon: Leveraging segment routing to improve network monitoring.* In IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications, pages 1–9, 2016.

[Aubry *et al.* 2016b] François Aubry, David Lebrun, Stefano Vissicchio, Minh Thanh Khong, Yves Deville and Olivier Bonaventure. *SCMon: Leveraging segment routing to improve network monitoring.* In IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications, pages 1–9. IEEE, 2016.

[Aubry *et al.* 2018] François Aubry, Stefano Vissicchio, Olivier Bonaventure and Yves Deville. *Robustly disjoint paths with segment routing.* In Proceedings of the 14th International Conference on emerging Networking EXperiments and Technologies, page 204–216, Heraklion Greece, Dec 2018. ACM.

[Aubry 2020] Francois Aubry. *Models and Algorithms for Network Optimization with Segment Routing.* 2020.

[Awerbuch *et al.* 1994] B. Awerbuch, A. Bar-Noy and M. Gopal. *Approximate distributed Bellman-Ford algorithms.* IEEE Transactions on Communications, vol. 42, no. 8, pages 2515–2517, 1994.

[Baker *et al.* 1998] Fred Baker, David L. Black, Kathleen Nichols and Steven L. Blake. *Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers.* RFC 2474, December 1998.

[Balon & Leduc 2007] Simon Balon and Guy Leduc. *Can Forwarding Loops Appear When Activating iBGP Multipath Load Sharing?* In Serge Fdida and Kazunori Sugiura, editors, Sustainable Internet, pages 213–225, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[Balon & Leduc 2008] Simon Balon and Guy Leduc. *Combined Intra- and Inter-Domain Traffic Engineering Using Hot-Potato Aware Link Weights Optimization.* In Proceedings of the 2008 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, SIGMETRICS '08, page 441–442, New York, NY, USA, 2008. Association for Computing Machinery.

[Bannister & Eppstein 2012] Michael J. Bannister and David Eppstein. Randomized speedup of the bellman–ford algorithm, pages 41–47. 2012.

[Barrera *et al.* 2017] David Barrera, Laurent Chuat, Adrian Perrig, Raphael M. Reischuk and Pawel Szalachowski. *The SCION Internet Architecture.* Commun. ACM, vol. 60, no. 6, page 56–65, may 2017.

[Bashandy *et al.* 2019] Ahmed Bashandy, Clarence Filsfils, Stefano Previdi, Bruno Decraene, Stephane Litkowski and Rob Shakir. *Segment Routing with the MPLS Data Plane.* RFC 8660, December 2019.

[Bellman 1958] R. Bellman. *ON A ROUTING PROBLEM.* Quarterly of Applied Mathematics, vol. 16, no. 1, pages 87–90, 1958.

[Bertsekas & Gallager 1992] D.P. Bertsekas and R.G. Gallager. Data networks. Prentice-Hall international editions. Prentice Hall, 1992.

[Bökler & Mutzel 2017] Fritz Bökler and Petra Mutzel. *Tree-Deletion Pruning in Label-Correcting Algorithms for the Multiobjective Shortest Path Problem.* In Sheung-Hung Poon, Md. Saidur Rahman and Hsu-Chun Yen, editors, WALCOM: Algorithms and Computation, pages 190–203, Cham, 2017. Springer International Publishing.

[Bosshart *et al.* 2014] Pat Bosshart, Dan Daly, Glen Gibb, Martin Izzard, Nick McKeown, Jennifer Rexford, Cole Schlesinger, Dan Talayco, Amin Vahdat, George Varghese and David Walker. *P4: Programming Protocol-Independent Packet Processors.* SIGCOMM Comput. Commun. Rev., vol. 44, no. 3, page 87–95, jul 2014.

[Boucadair 2005] Mohamed Boucadair. Solutions for sustaining scalability in internet growth. Parlux, 2005.

[Braden *et al.* 1994] Robert T. Braden, Dr. David D. Clark and Scott Shenker. *Integrated Services in the Internet Architecture: an Overview.* RFC 1633, June 1994.

[Braden *et al.* 1997] Robert T. Braden, Lixia Zhang, Steven Berson, Shai Herzog and Sugih Jamin. *Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification.* RFC 2205, September 1997.

[Bramas *et al.* 2020] Quentin Bramas, Pascal Mérindol, Cristel Pelsser and Jean-Romain Luttringer. *A Fast-Convergence Routing of the Hot-Potato: The Tool to Perform your own Evaluation*, February 2020.

[Bremler-Barr *et al.* 2003] A. Bremler-Barr, Y. Afek and S. Schwarz. *Improved BGP convergence via ghost flushing.* In IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No.03CH37428), volume 2, pages 927–937 vol.2, 2003.

[Bremler-Barr *et al.* 2009] Anat Bremler-Barr, Nir Chen, Jussi Kangasharju, Osnat Mokryn and Yuval Shavitt. *Bringing order to BGP: Decreasing time and message complexity.* Computer Networks, vol. 53, no. 12, pages 2241–2256, 2009.

[Brenes *et al.* 2020] Juan Brenes, Alberto García-Martínez, Marcelo Bagnulo, Andra Lutu and Cristel Pelsser. *Power Prefixes Prioritization for Smarter BGP Reconvergence.* IEEE/ACM Transactions on Networking, vol. 28, no. 3, pages 1074–1087, 2020.

[Breugem *et al.* 2017] Thomas Breugem, Twan Dollevoet and Wilco van den Heuvel. *Analysis of FPTASes for the multi-objective shortest path problem.* Computers & Operations Research, vol. 78, pages 44–58, 2017.

[Brumbaugh-Smith & Shier 1989] J. Brumbaugh-Smith and D. Shier. *An empirical investigation of some bicriterion shortest path algorithms.* European Journal of Operational Research, vol. 43, no. 2, page 216–224, Nov 1989.

[Brundiers *et al.* 2021] Alexander Brundiers, Timmy Schüller and Nils Aschenbruck. *On the Benefits of Loops for Segment Routing Traffic Engineering.* In 2021 IEEE 46th Conference on Local Computer Networks (LCN), pages 32–40, 2021.

[Bryant *et al.* 2015] Stewart Bryant, Clarence Filsfils, Stefano Previdi, Mike Shand and Ning So. *Remote Loop-Free Alternate (LFA) Fast Reroute (FRR).* RFC 7490, April 2015.

[Buob *et al.* 2007] Marc-Olivier Buob, Mickael Meulle and Steve Uhlig. *Checking for optimal egress points in iBGP routing.* In 2007 6th International Workshop on Design and Reliable Communication Networks, pages 1–8, 2007.

[Buob *et al.* 2008] Marc-Olivier Buob, Steve Uhlig and Mickael Meulle. *Designing Optimal iBGP Route-Reflection Topologies.* In Amitabha Das, Hung Keng Pung, Francis Bu Sung Lee and Lawrence Wai Choong Wong, editors, NETWORKING 2008 Ad Hoc and Sensor Networks, Wireless Networks, Next Generation Internet, pages 542–553, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[Buob *et al.* 2016] Marc-Olivier Buob, Anthony Lambert and Steve Uhlig. *iBGP2: A scalable iBGP redistribution mechanism leading to optimal routing.* In IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications, pages 1–9, 2016.

[Caleffi 2017] Marcello Caleffi. *Optimal routing for quantum networks*. IEEE Access, vol. 5, pages 22299–22312, 2017.

[Cavendish & Gerla 1998] D. Cavendish and M. Gerla. Internet qos routing using the bellman-ford algorithm, page 627–646. IFIP — The International Federation for Information Processing. Springer US, 1998.

[Chang *et al.* 2015] Michael Chang, Thomas Holterbach, Markus Happe and Laurent Vanbever. *Supercharge me*. ACM SIGCOMM Computer Communication Review, vol. 45, pages 341–342, 08 2015.

[Chekuri 2021] Chandra Chekuri. *Shortest Paths with Negative Lengths and DP*, 2021.

[Chen & Nahrstedt 1998] Shigang Chen and K. Nahrstedt. *An overview of quality of service routing for next-generation high-speed networks: problems and solutions*. IEEE Network, vol. 12, no. 6, page 64–79, Nov 1998.

[Chen *et al.* 2006] Enke Chen, Tony J. Bates and Ravi Chandra. *BGP Route Reflection: An Alternative to Full Mesh Internal BGP (IBGP)*. RFC 4456, April 2006.

[Chen *et al.* 2007] Mo Chen, Rezaul Alam Chowdhury, Vijaya Ramachandran, D L Roche and Lin Tong. *Priority Queues and Dijkstra ' s Algorithm*. 2007.

[Cheng & Ansari 2003] Gang Cheng and N. Ansari. *A new heuristics for finding the delay constrained least cost path*. In GLOBECOM '03. IEEE Global Telecommunications Conference (IEEE Cat. No.03CH37489), volume 7, page 3711–3715 vol.7, Dec 2003.

[Chiesa *et al.* 2019] Marco Chiesa, Roshan Sedar, Gianni Antichi, Michael Borokhovich, Andrzej Kamisiński, Georgios Nikolaidis and Stefan Schmid. *PURR: A Primitive for Reconfigurable Fast Reroute: Hope for the Best and Program for the Worst*. In Proceedings of the 15th International Conference on Emerging Networking Experiments And Technologies, CoNEXT '19, page 1–14, New York, NY, USA, 2019. Association for Computing Machinery.

[Chiesa *et al.* 2020] Marco Chiesa, Andrzej Kamisiński, Jacek Rak, Gábor Rétvári and Stefan Schmid. *A Survey of Fast Recovery Mechanisms in the Data Plane*. 6 2020.

[Chiesa *et al.* 2021] Marco Chiesa, Andrzej Kamisiński, Jacek Rak, Gábor Rétvári and Stefan Schmid. *A Survey of Fast-Recovery Mechanisms in Packet-Switched Networks*. IEEE Communications Surveys & Tutorials, vol. 23, no. 2, pages 1253–1301, 2021.

[Cisco Networks 2008] Cisco Networks. *Campus Network for High Availability Design Guide*. https://www.cisco.com/c/en/us/td/docs/solutions/Enterprise/Campus/HA_campus_DG/hacampusdg.html, May 2008. (Accessed on 07/12/2021).

[Cisco Networks 2014] Cisco Networks. *Hierarchical Network Design Overview (1.1) > Cisco Networking Academy Connecting Networks Companion Guide: Hierarchical Network Design | Cisco Press*. https://www.ciscopress.com/articles/article.asp?p=2202410&seqNum=4, May 2014. (Accessed on 09/08/2021).

[Cisco Networks 2020] Cisco Networks. *Cisco Content Hub - Performance Measurement for Traffic Engineering.* https://content.cisco.com/chapter.sjs?uri=/searchable/chapter/content/en/us/td/docs/ios-xml/ios/seg_routing/configuration/xe-17/segrt-xe-17-book/performance_measure_TE.html.xml, September 2020. (Accessed on 06/21/2021).

[Cisco-Systems 2022] Cisco-Systems. *iBGP Multipath Load Sharing*, Accessed July 2022.

[Clad *et al.* 2014] François Clad, Pascal Mérindol, Jean-Jacques Pansiot, Pierre Francois and Olivier Bonaventure. *Graceful Convergence in Link-State IP Networks: A Lightweight Algorithm Ensuring Minimal Operational Impact.* IEEE/ACM Transactions on Networking, vol. 22, no. 1, pages 300–312, 2014.

[Clad 2014] François Clad. *Disruption-free routing convergence : computing minimal link-state update sequences. (Convergence du routage sans perturbation : calcul de séquences minimales de mises à jour d'états des liens).* PhD thesis, University of Strasbourg, France, 2014.

[Coello & Veldhuizen 2007] Carlos Coello and David Veldhuizen. Evolutionary algorithms for solving multi-objective problems second edition. 01 2007.

[Cohon *et al.* 1979] Jared L. Cohon, Richard L. Church and Daniel P. Sheer. *Generating multiobjective trade-offs: An algorithm for bicriterion problems.* Water Resources Research, vol. 15, no. 5, pages 1001–1010, 1979.

[Corley & Moon 1985] H. W. Corley and I. D. Moon. *Shortest paths in networks with vector weights.* Journal of Optimization Theory and Applications, vol. 46, no. 1, page 79–86, May 1985.

[Cormen *et al.* 2009] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein. Introduction to algorithms, third edition. The MIT Press, 3rd édition, 2009.

[Crauser *et al.* 1998a] Andreas Crauser, Kurt Mehlhorn, Ulrich Meyer and Peter Sanders. *A parallelization of Dijkstra's shortest path algorithm.* In International Symposium on Mathematical Foundations of Computer Science, pages 722–731. Springer, 1998.

[Crauser *et al.* 1998b] Andreas Crauser, Kurt Mehlhorn, Ulrich Meyer and Peter Sanders. *A Parallelization of Dijkstra's Shortest Path Algorithm.* In MFCS, 1998.

[D'Angelo *et al.* 2014] Gianlorenzo D'Angelo, Mattia D'Emidio and Daniele Frigioni. *A loop-free shortest-path routing algorithm for dynamic networks.* Theoretical Computer Science, vol. 516, pages 1–19, 2014.

[Dantzig & Thapa 2003] George Dantzig and Mukund Thapa. *Linear Programming: 2: Theory and Extensions.* 01 2003.

[Davoli *et al.* 2015] Luca Davoli, Luca Veltri, Pier Luigi Ventre, Giuseppe Siracusano and Stefano Salsano. *Traffic Engineering with Segment Routing: SDN-Based Architectural Design and Open Source Implementation.* In 2015 Fourth European Workshop on Software Defined Networks, pages 111–112, 2015.

[De Neve & Van Mieghem 2000] H. De Neve and P. Van Mieghem. *TAMCRA: a tunable accuracy multiple constraints routing algorithm.* Computer Communications, vol. 23, no. 7, pages 667–679, 2000.

[De Queirós Vieira Martins *et al.* 1999] Ernesto De Queirós Vieira Martins, Marta Margarida Braz Pascoal and José Luis Esteves Dos Santos. *Deviation algorithms for ranking shortest paths.* International Journal of Foundations of Computer Science, vol. 10, no. 03, page 247–261, Sep 1999.

[Deb 2005] Kalyanmoy Deb. Multi-objective optimization, pages 273–316. Springer US, 2005.

[Demetrescu & Italiano 2004] Camil Demetrescu and Giuseppe F. Italiano. *A New Approach to Dynamic All Pairs Shortest Paths.* J. ACM, vol. 51, no. 6, page 968–992, nov 2004.

[Demetrescu & Italiano 2006] Camil Demetrescu and Giuseppe F Italiano. *Experimental analysis of dynamic all pairs shortest path algorithms.* ACM Transactions on Algorithms (TALG), vol. 2, no. 4, pages 578–601, 2006.

[Dijkstra 1959] Edsger W Dijkstra. *A note on two problems in connexion with graphs.* Numerische mathematik, vol. 1, no. 1, pages 269–271, 1959.

[Dong *et al.* 2022] Jie Dong, Stewart Bryant, Takuya Miyasaka, Yongqing Zhu, Fengwei Qin, Zhenqiang Li and Francois Clad. *Introducing Resource Awareness to SR Segments.* Internet-Draft draft-ietf-spring-resource-aware-segments-06, Internet Engineering Task Force, October 2022. Work in Progress.

[Doyle & Carroll 2005] Jeff Doyle and Jennifer Dehaven Carroll. Routing TCP/IP, volume 1. CCIE professional development. Cisco Press, Indianapolis, IN, 2 édition, October 2005.

[Dugeon *et al.* 2017] Olivier Dugeon, Rabah Guedrez, Samer Lahoud and Géraldine Texier. *Demonstration of Segment Routing with SDN based label stack optimization.* In ICIN 2017 : 20th Conference on Innovations in Clouds, Internet and Networks, pages 143 – 145, Paris, France, March 2017.

[Duque *et al.* 2015] Daniel Duque, Leonardo Lozano and Andrés L. Medaglia. *An exact method for the biobjective shortest path problem for large-scale road networks.* European Journal of Operational Research, vol. 242, no. 3, pages 788–797, 2015.

[E. Hart *et al.* 1968] Peter E. Hart, Nils J. Nilsson and Bertram Raphael. *A Formal Basis for the Heuristic Determination of Minimum Cost Paths.* IEEE Transaction on Systems Science and Cybernetics, Jul 1968.

[Ergun *et al.* 2002] Funda Ergun, Rakesh Sinha and Lisa Zhang. *An improved FPTAS for Restricted Shortest Path.* Information Processing Letters, vol. 83, no. 5, page 287–291, Sep 2002.

[Feng *et al.* 2002] G. Feng, C. Douligeris, K. Makki and N. Pissinou. *Performance evaluation of delay-constrained least-cost QoS routing algorithms based on linear and nonlinear Lagrange relaxation.* In 2002 IEEE International Conference on Communications.

Conference Proceedings. ICC 2002 (Cat. No.02CH37333), volume 4, page 2273–2278. IEEE, 2002.

[Filsfils *et al.* 2011] Clarence Filsfils, Pradosh Mohapatra, John Bettink, Pranav Dharwadkar, Peter De Vriendt, Yuri Tsier, Virginie Van Den Schrieck, Olivier Bonaventure and Pierre Francois. *BGP prefix independent convergence (PIC) technical report*. Cisco, Tech. Rep, Tech. Rep, 2011.

[Filsfils *et al.* 2015] Clarence Filsfils, Nagendra Kumar Nainar, Carlos Pignataro, Juan Camilo Cardona and Pierre Francois. *The Segment Routing Architecture*. In 2015 IEEE Global Communications Conference (GLOBECOM), pages 1–6, 2015.

[Filsfils *et al.* 2017] Clarence Filsfils, Kris Michielsen and Ketan Talaulikar. Segment routing part i. CreateSpace Independent Publishing Platform, North Charleston, SC, USA, 1st édition, 2017.

[Filsfils *et al.* 2018] C. Filsfils, S. Previdi, L. Ginsberg, B. Decraene, S. Litkowski and R. Shakir. *Segment Routing Architecture*. RFC 8402, RFC Editor, July 2018.

[Filsfils *et al.* 2021a] Clarence Filsfils, Pablo Camarillo, John Leddy, Daniel Voyer, Satoru Matsushima and Zhenbin Li. *Segment Routing over IPv6 (SRv6) Network Programming*. RFC 8986, February 2021.

[Filsfils *et al.* 2021b] Clarence Filsfils, Stefano Previdi, Gaurav Dawra, Ebben Aries and Dmitry Afanasiev. *Segment Routing Centralized BGP Egress Peer Engineering*. RFC 9087, August 2021.

[Filsfils 2019] C. Filsfils. *Segment Routing - Cisco Live Barcelona 2019*. https://www.segment-routing.net/conferences/2019-01-30-CLEUR-3122/, January 2019. (Accessed on 08/30/2021).

[Filsfils 2020] Clarence Filsfils. *Network Programming with SRv6*. https://orbi.uliege.be/bitstream/2268/245292/4/network-programming-phd-final.pdf, April 2020. (Accessed on 06/28/2021).

[Ford 1956] L. R. Ford. Network flow theory. RAND Corporation, Santa Monica, CA, 1956.

[Fortz & Thorup 2000] B. Fortz and M. Thorup. *Internet traffic engineering by optimizing OSPF weights*. In Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No.00CH37064), volume 2, pages 519–528 vol.2, 2000.

[Francois & Bonaventure 2007] Pierre Francois and Olivier Bonaventure. *Avoiding Transient Loops During the Convergence of Link-State Routing Protocols*. IEEE/ACM Transactions on Networking, vol. 15, no. 6, pages 1280–1292, 2007.

[Francois *et al.* 2005a] Pierre Francois, Clarence Filsfils, John Evans and Olivier Bonaventure. *Achieving Sub-Second IGP Convergence in Large IP Networks*. SIGCOMM Comput. Commun. Rev., vol. 35, no. 3, page 35–44, jul 2005.

[Francois *et al.* 2005b] Pierre Francois, Clarence Filsfils, John Evans and Olivier Bonaventure. *Achieving Sub-Second IGP Convergence in Large IP Networks.* SIGCOMM Comput. Commun. Rev., vol. 35, no. 3, page 35–44, jul 2005.

[Francois *et al.* 2013] Pierre Francois, Clarence Filsfils, Ahmed Bashandy, Stefano Previdi and Bruno Decraene. *Segment Routing Fast Reroute.* Internet-Draft draft-francois-sr-frr-00, Internet Engineering Task Force, July 2013. Work in Progress.

[Francois 2007] Pierre Francois. *Improving the Convergence of IP Routing Protocols.* PhD thesis, Catholic University of Louvain, Louvain-la-Neuve, Belgium, 2007.

[Fredman & Tarjan 1987] Michael L. Fredman and Robert Endre Tarjan. *Fibonacci Heaps and Their Uses in Improved Network Optimization Algorithms.* J. ACM, vol. 34, no. 3, page 596–615, jul 1987.

[Gafni & Bertsekas 1981] E. Gafni and D. Bertsekas. *Distributed Algorithms for Generating Loop-Free Routes in Networks with Frequently Changing Topology.* vol. 29, no. 1, pages 11–18, 1981.

[Gang *et al.* 2018] Yirou Gang, Pei Zhang, Xiaohong Huang and Tianle Yang. *Throughput Maximization Routing in the Hybrid Segment Routing Network.* In Proceedings of the 2nd International Conference on Telecommunications and Communication Engineering, ICTCE 2018, page 262–267, New York, NY, USA, 2018. Association for Computing Machinery.

[Gao & Rexford 2001] Lixin Gao and J. Rexford. *Stable Internet routing without global coordination.* IEEE/ACM Transactions on Networking, vol. 9, no. 6, pages 681–692, 2001.

[Garey & Johnson 1979] M. R. Garey and D. S. Johnson. Computers and intractability: A guide to the theory of np-completeness. W. H. Freeman, first edition édition, 1979.

[Garroppo *et al.* 2010] Rosario G. Garroppo, Stefano Giordano and Luca Tavanti. *A survey on multi-constrained optimal path computation: Exact and approximate algorithms.* Computer Networks, vol. 54, no. 17, page 3081–3107, Dec 2010.

[Gay *et al.* 2017] S. Gay, Renaud Hartert and Stefano Vissicchio. *Expect the unexpected: Sub-second optimization for segment routing.* IEEE INFOCOM 2017 - IEEE Conference on Computer Communications, pages 1–9, 2017.

[Ghoseiri & Nadjari 2010] Keivan Ghoseiri and Behnam Nadjari. *An ant colony optimization algorithm for the bi-objective shortest path problem.* Applied Soft Computing, vol. 10, no. 4, pages 1237–1246, 2010. Optimisation Methods & Applications in Decision-Making Processes.

[Giacalone *et al.* 2015] Spencer Giacalone, David Ward, John Drake, Alia Atlas and Stefano Previdi. *OSPF Traffic Engineering (TE) Metric Extensions.* RFC 7471, March 2015.

[Gill *et al.* 2014] Phillipa Gill, Michael Schapira and Sharon Goldberg. *A Survey of Inter-domain Routing Policies.* SIGCOMM Comput. Commun. Rev., vol. 44, no. 1, page 28–34, dec 2014.

[Ginsberg *et al.* 2019] Les Ginsberg, Stefano Previdi, Spencer Giacalone, David Ward, John Drake and Qin Wu. *IS-IS Traffic Engineering (TE) Metric Extensions.* RFC 8570, March 2019.

[Goel *et al.* 2001] Ashish Goel, K. G. Ramakrishnan, Deepak Kataria and Dimitris Logothetis. *Efficient Computation of Delay-sensitive Routes from One Source to All Destinations*, 2001.

[Goldberg *et al.* 2006] Andrew V. Goldberg, Haim Kaplan and Renato F. Werneck. *Reach for A*: Efficient Point-to-Point Shortest Path Algorithms.* In ALENEX, 2006.

[Griffin & Wilfong 2002a] Timothy Griffin and Gordon T. Wilfong. *Analysis of the MED Oscillation Problem in BGP.* In Proceedings of the 10th IEEE International Conference on Network Protocols, ICNP '02, page 90–99, USA, 2002. IEEE Computer Society.

[Griffin & Wilfong 2002b] Timothy G. Griffin and Gordon Wilfong. *On the Correctness of IBGP Configuration.* SIGCOMM Comput. Commun. Rev., vol. 32, no. 4, page 17–29, aug 2002.

[Gross & Rekhter 1995] Phillip G. Gross and Yakov Rekhter. *Application of the Border Gateway Protocol in the Internet.* RFC 1772, March 1995.

[Gross 1992] Phillip G. Gross. *Choosing a Common IGP for the IP Internet.* RFC 1371, October 1992.

[Guck *et al.* 2018] Jochen W. Guck, Amaury Van Bemten, Martin Reisslein and Wolfgang Kellerer. *Unicast QoS Routing Algorithms for SDN: A Comprehensive Survey and Performance Evaluation.* IEEE Communications Surveys & Tutorials, vol. 20, no. 1, page 388–415, 2018.

[Guedrez *et al.* 2016a] Rabah Guedrez, Olivier Dugeon, Samer Lahoud and Géraldine Texier. *Label encoding algorithm for MPLS Segment Routing.* In NCA 2016 : IEEE 15th International Symposium on Network Computing and Applications, pages 113 – 117, Cambridge, Ma, United States, October 2016.

[Guedrez *et al.* 2016b] Rabah Guedrez, Olivier Dugeon, Samer Lahoud and Géraldine Texier. *Label encoding algorithm for MPLS Segment Routing.* In 2016 IEEE 15th International Symposium on Network Computing and Applications (NCA), pages 113–117, 2016.

[Guerin *et al.* 1997] R.A. Guerin, A. Orda and D. Williams. *QoS routing mechanisms and OSPF extensions.* In GLOBECOM 97. IEEE Global Telecommunications Conference. Conference Record, volume 3, page 1903–1908 vol.3, Nov 1997.

[Guerriero & Musmanno 2001] F. Guerriero and R. Musmanno. *Label Correcting Methods to Solve Multicriteria Shortest Path Problems.* Journal of Optimization Theory and Applications, vol. 111, no. 3, page 589–613, Dec 2001.

[Guo & Matta 1999] Liang Guo and I. Matta. *Search space reduction in QoS routing.* In Proceedings. 19th IEEE International Conference on Distributed Computing Systems (Cat. No.99CB37003), page 142–149, Jun 1999.

[Gvozdiev *et al.* 2013] Nikola Gvozdiev, Brad Karp and Mark Handley. *LOUP: The Principles and Practice of Intra-Domain Route Dissemination.* In 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13), pages 413–426, Lombard, IL, April 2013. USENIX Association.

[Halpern & Pignataro 2015] Joel M. Halpern and Carlos Pignataro. *Service Function Chaining (SFC) Architecture.* RFC 7665, October 2015.

[Handler & Zang 1980] Gabriel Y. Handler and Israel Zang. *A dual algorithm for the constrained shortest path problem.* Networks, vol. 10, no. 4, page 293–309, 1980.

[Hansen 1980] Pierre Hansen. *Bicriterion Path Problems.* In Günter Fandel and Tomas Gal, editors, Multiple Criteria Decision Making Theory and Application, Lecture Notes in Economics and Mathematical Systems, page 109–127. Springer, 1980.

[Hanusse *et al.* 2020] Nicolas Hanusse, David Ilcinkas and Antonin Lentz. *Framing Algorithms for Approximate Multicriteria Shortest Paths.* In Dennis Huisman and Christos D. Zaroliagis, editors, 20th Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2020), volume 85 of *OpenAccess Series in Informatics (OASIcs)*, pages 11:1–11:19, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik.

[Hart *et al.* 1968] Peter E. Hart, Nils J. Nilsson and Bertram Raphael. *A Formal Basis for the Heuristic Determination of Minimum Cost Paths.* IEEE Transactions on Systems Science and Cybernetics, vol. 4, no. 2, pages 100–107, 1968.

[Hartert *et al.* 2015] Renaud Hartert, Stefano Vissicchio, Pierre Schaus, Olivier Bonaventure, Clarence Filsfils, Thomas Telkamp and Pierre Francois. *A Declarative and Expressive Approach to Control Forwarding Paths in Carrier-Grade Networks.* SIGCOMM Comput. Commun. Rev., vol. 45, no. 4, page 15–28, aug 2015.

[Hassin 1992] Refael Hassin. *Approximation Schemes for the Restricted Shortest Path Problem.* Mathematics of Operations Research, vol. 17, no. 1, page 36–42, 1992.

[Hauser *et al.* 2021] Frederik Hauser, Marco Häberle, Daniel Merling, Steffen Lindner, Vladimir Gurevich, Florian Zeiger, Reinhard Frank and Michael Menth. *A Survey on Data Plane Programming with P4: Fundamentals, Advances, and Applied Research*, 2021.

[Holterbach *et al.* 2017] Thomas Holterbach, Stefano Vissicchio, Alberto Dainotti and Laurent Vanbever. *SWIFT: Predictive Fast Reroute.* In Proceedings of the Conference of the ACM Special Interest Group on Data Communication, SIGCOMM '17, page 460–473, New York, NY, USA, 2017. Association for Computing Machinery.

[Holterbach *et al.* 2019] Thomas Holterbach, Edgar Costa Molero, Maria Apostolaki, Alberto Dainotti, Stefano Vissicchio and Laurent Vanbever. *Blink: Fast Connectivity*

Recovery Entirely in the Data Plane. In 16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19), pages 161–176, Boston, MA, February 2019. USENIX Association.

[Holterbach 2021] Thomas Holterbach. *A Framework To Fast Reroute Traffic Upon Remote Outages*. PhD thesis, ETH Zurich, Zurich, 2021.

[Hopps 2000] C. Hopps. *RFC2992: Analysis of an Equal-Cost Multi-Path Algorithm*, 2000.

[Hou *et al.* 2018] Xiaolan Hou, Muqing Wu and Min Zhao. *An Optimization Routing Algorithm Based on Segment Routing in Software-Defined Networks*. Sensors, vol. 19, page 49, 12 2018.

[Hribar *et al.* 1995] Michelle Hribar, Valerie E. Taylor and David E. Boyce. Choosing a shortest path algorithm. 1995.

[Iori *et al.* 2010] Manuel Iori, Silvano Martello and Daniele Pretolani. *An aggregate label setting policy for the multi-objective shortest path problem*. European Journal of Operational Research, vol. 207, no. 3, pages 1489–1496, 2010.

[Ishida *et al.* 1998] K. Ishida, K. Amano and N. Kannari. *A delay-constrained least-cost path routing protocol and the synthesis method*. In Proceedings Fifth International Conference on Real-Time Computing Systems and Applications (Cat. No.98EX236), page 58–65, Oct 1998.

[Jabloner 2016] Paula Jabloner. *The two-napkin protocol*, Jun 2016.

[Jadin *et al.* 2019] Mathieu Jadin, Francois Aubry, Pierre Schaus and Olivier Bonaventure. *CG4SR: Near Optimal Traffic Engineering for Segment Routing with Column Generation*. In IEEE INFOCOM 2019 - IEEE Conference on Computer Communications, pages 1333–1341, 2019.

[Jaffe 1984] Jeffrey M. Jaffe. *Algorithms for finding paths with multiple constraints*. Networks, vol. 14, no. 1, page 95–116, 1984.

[Jia & Varaiya 2006] Zhanfeng Jia and P. Varaiya. *Heuristic methods for delay constrained least cost routing using /spl kappa/-shortest-paths*. IEEE Transactions on Automatic Control, vol. 51, no. 4, page 707–712, Apr 2006.

[Juttner *et al.* 2001a] A. Juttner, B. Szviatovski, I. Mecs and Z. Rajko. *Lagrange relaxation based method for the QoS routing problem*. In Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No.01CH37213), volume 2, pages 859–868 vol.2, 2001.

[Juttner *et al.* 2001b] A. Juttner, B. Szviatovski, I. Mecs and Z. Rajko. *Lagrange relaxation based method for the QoS routing problem*. In Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No.01CH37213), volume 2, page 859–868 vol.2, Apr 2001.

[Katz & Ward 2010] Dave Katz and David Ward. *Bidirectional Forwarding Detection (BFD) for IPv4 and IPv6 (Single Hop).* RFC 5881, June 2010.

[Kergosien *et al.* 2022] Yanic Kergosien, Antoine Giret, Emmanuel Neron and Gaël Sauvanet. *An Efficient Label-Correcting Algorithm for the Multiobjective Shortest Path Problem.* 2022.

[Kfoury *et al.* 2021] Elie F. Kfoury, Jorge Crichigno and Elias Bou-Harb. *An Exhaustive Survey on P4 Programmable Data Plane Switches: Taxonomy, Applications, Challenges, and Future Trends.* IEEE Access, vol. 9, pages 87094–87155, 2021.

[Korkmaz & Krunz 2001] T. Korkmaz and M. Krunz. *Multi-constrained optimal path selection.* In Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No.01CH37213), volume 2, page 834–843 vol.2, Apr 2001.

[Kozierok 2005] Charles M Kozierok. The tcp/ip guide. No Starch Press, San Francisco, CA, May 2005.

[Kuipers & Van Mieghem 2003] Fernando A. Kuipers and Piet Van Mieghem. *Bi-directional Search in QoS Routing.* In Gunnar Karlsson and Michael I. Smirnov, editors, Quality for All, Lecture Notes in Computer Science, page 102–111. Springer, 2003.

[Kuipers & Van Mieghem 2005] F.A. Kuipers and P. Van Mieghem. *Non-dominance in QoS routing: an implementational perspective.* IEEE Communications Letters, vol. 9, no. 3, pages 267–269, 2005.

[Kuipers *et al.* 2002] F. Kuipers, P. Van Mieghem, T. Korkmaz and M. NR-DCLC. *An overview of constraint-based path selection algorithms for QoS routing.* IEEE Communications Magazine, vol. 40, no. 12, page 50–55, Dec 2002.

[Kushman *et al.* 2007] Nate Kushman, Srikanth Kandula and Bruce M. Maggs. *R-BGP: Staying Connected in a Connected World.* In 4th USENIX Symposium on Networked Systems Design & Implementation (NSDI 07), Cambridge, MA, April 2007. USENIX Association.

[Kvalbein *et al.* 2009] Amund Kvalbein, Audun Fosselie Hansen, Tarik Cicic, Stein Gjessing and Olav Lysne. *Multiple Routing Configurations for Fast IP Network Recovery.* IEEE/ACM Transactions on Networking, vol. 17, no. 2, pages 473–486, 2009.

[Labovitz *et al.* 2000a] Craig Labovitz, Abha Ahuja, Abhijit Bose and Farnam Jahanian. *Delayed Internet Routing Convergence.* In Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, SIGCOMM '00, page 175–187, New York, NY, USA, 2000. Association for Computing Machinery.

[Labovitz *et al.* 2000b] Craig Labovitz, Abha Ahuja, Abhijit Bose and Farnam Jahanian. *Delayed Internet Routing Convergence.* SIGCOMM Comput. Commun. Rev., vol. 30, no. 4, page 175–187, aug 2000.

[Lambert *et al.* 2009] Anthony Lambert, Marc-Olivier Buob and Steve Uhlig. *Improving Internet-Wide Routing Protocols Convergence with MRPC Timers.* In Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies, CoNEXT '09, page 325–336, New York, NY, USA, 2009. Association for Computing Machinery.

[Larkin *et al.* 2014] Daniel H. Larkin, Siddhartha Sen and Robert E. Tarjan. A back-to-basics empirical study of priority queues, pages 61–72. 2014.

[Lazzeri *et al.* 2015] F. Lazzeri, Gianmarco Bruno, J. Nijhof, A. Giorgetti and P. Castoldi. *Efficient label encoding in segment-routing enabled optical networks.* 2015 International Conference on Optical Network Design and Modeling (ONDM), pages 34–38, 2015.

[Leduc *et al.* 2006] G. Leduc, H. Abrahamsson, S. Balon, S. Bessler, M. D'Arienzo, O. Delcourt, J. Domingo-Pascual, S. Cerav-Erbas, I. Gojmerac, X. Masip, A. Pescapè, B. Quoitin, S. P. Romano, E. Salvadori, F. Skivée, H. T. Tran, S. Uhlig and H. ímit. *An Open Source Traffic Engineering Toolbox.* Comput. Commun., vol. 29, no. 5, page 593–610, mar 2006.

[Lee & Sheu 2016] Ming-Chieh Lee and Jang-Ping Sheu. *An efficient routing algorithm based on segment routing in software-defined networking.* Computer Networks, vol. 103, pages 44–55, 2016.

[Lee *et al.* 1995] W. C Lee, M. G. Hluchyi and P. A. Humblet. *Routing Subject to Quality of Service Constraints in Integrated Communication Networks.* 1995.

[Li *et al.* 1996] Tony Li, Ravi Chandra and Paul S. Traina. *BGP Communities Attribute.* RFC 1997, August 1996.

[Litkowski *et al.* 2022] Stephane Litkowski, Ahmed Bashandy, Clarence Filsfils, Pierre Francois, Bruno Decraene and Daniel Voyer. *Topology Independent Fast Reroute using Segment Routing.* Internet-Draft draft-ietf-rtgwg-segment-routing-ti-lfa-08, Internet Engineering Task Force, January 2022. Work in Progress.

[Liu & Ramakrishnan 2001] Gang Liu and K. G. Ramakrishnan. *A*Prune: An Algorithm for Finding K Shortest Paths Subject to Multiple Constraints.* 2001.

[Liu *et al.* 2005] W Liu, W Lou and Y Fang. *An efficient quality of service routing algorithm for delay-sensitive applications.* Computer Networks, vol. 47, no. 1, page 87–104, Jan 2005.

[Liu *et al.* 2013] Junda Liu, Aurojit Panda, Ankit Singla, Brighten Godfrey, Michael Schapira and Scott Shenker. *Ensuring Connectivity via Data Plane Mechanisms.* In 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13), pages 113–126, Lombard, IL, April 2013. USENIX Association.

[Lorenz & Raz 2001] Dean H. Lorenz and Danny Raz. *A simple efficient approximation scheme for the restricted shortest path problem.* Operations Research Letters, vol. 28, no. 5, page 213–219, Jun 2001.

[Luckie *et al.* 2013] Matthew Luckie, Bradley Huffaker, Amogh Dhamdhere, Vasileios Giotsas and kc claffy. *AS Relationships, Customer Cones, and Validation.* In Proceedings of the 2013 conference on Internet measurement conference - IMC '13, pages 243–256, Barcelona, Spain, 2013. ACM Press.

[Luo *et al.* 2002] Jiazeng Luo, Junqing Xie, Ruibing Hao and Xing Li. *An approach to accelerate convergence for path vector protocol.* In Global Telecommunications Conference, 2002. GLOBECOM '02. IEEE, volume 3, pages 2390–2394 vol.3, 2002.

[Luttringer & Mérindol 2019] Jean-Romain Luttringer and Pascal Mérindol. *OPTIC: An Efficient BGP Protection Technique For Optimal Intra-domain Convergence*, August 2019.

[Luttringer *et al.* 2020a] J. R. Luttringer, T. Alfroy, P. Mérindol, Q. Bramas, F. Clad and C. Pelsser. *Computing Delay-Constrained Least-Cost Paths for Segment Routing is Easier Than You Think.* In 2020 IEEE 19th International Symposium on Network Computing and Applications (NCA), pages 1–8, 2020.

[Luttringer *et al.* 2020b] Jean-Romain Luttringer, Thomas Alfroy, Pascal Mérindol, Quentin Bramas, François Clad and Cristel Pelsser. *Computing Delay-Constrained Least-Cost Paths for Segment Routing is Easier Than You Think.* In 2020 IEEE 19th International Symposium on Network Computing and Applications (NCA), pages 1–8, 2020.

[Luttringer *et al.* 2020c] Jean-Romain Luttringer, Yves Vanaubel, Pascal Mérindol, Jean-Jacques Pansiot and Benoit Donnet. *Let There Be Light: Revealing Hidden MPLS Tunnels With TNT.* IEEE Transactions on Network and Service Management, vol. 17, no. 2, pages 1239–1253, 2020.

[Luttringer *et al.* 2021a] Jean-Romain Luttringer, Thomas Alfroy, Quentin Bramas, François Clad, Pascal Mérindol and Cristel Pelsser. *Deploying Near Optimal Delay Constrained Paths with Segment-Routing in Massive Scale Networks*, September 2021.

[Luttringer *et al.* 2021b] Jean-Romain Luttringer, Thomas Alfroy, Pascal Mérindol, François Clad and Cristel Pelsser. *Le Problème à trois Contraintes : Calcul et Déploiement de Segments de Routage.* In ALGOTEL 2021 - 23èmes Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications, La Rochelle, France, June 2021.

[Luttringer *et al.* 2021c] Jean-Romain Luttringer, Quentin Bramas, Cristel Pelsser and Pascal Mérindol. *L'Art d'Anticiper les Changements IGP pour Acheminer Optimalement la Patate en Transit.* In CORES 2021 – 6ème Rencontres Francophones sur la Conception de Protocoles, l'Évaluation de Performance et l'Expérimentation des Réseaux de Communication, La Rochelle, France, September 2021.

[Luttringer *et al.* 2021d] Jean-Romain Luttringer, Quentin Bramas, Cristel Pelsser and Pascal Mérindol. *A Fast-Convergence Routing of the Hot-Potato.* In IEEE INFOCOM 2021 - IEEE Conference on Computer Communications, pages 1–10, 2021.

[Luttringer *et al.* 2022] Jean-Romain Luttringer, Thomas Alfroy, Pascal Mérindol, Quentin Bramas, François Clad and Cristel Pelsser. *Deploying near-optimal delay-constrained paths with Segment Routing in massive-scale networks.* Computer Networks, vol. 212, page 109015, 2022.

[Madkour *et al.* 2017] Amgad Madkour, Walid G Aref, Faizan Ur Rehman, Mohamed Abdur Rahman and Saleh Basalamah. *A survey of shortest-path algorithms.* arXiv preprint arXiv:1705.02044, 2017.

[Malkin 1998] Gary S. Malkin. *RIP Version 2.* RFC 2453, November 1998.

[Marques *et al.* 2012] Pedro Marques, Rex Fernando, Enke Chen, Prodosh Mohapatra and Hannes Gredler. *Advertisement of the best external route in BGP.* Internet-Draft draft-ietf-idr-best-external-05, Internet Engineering Task Force, January 2012. Work in Progress.

[Martins & Santos 1999] Ernesto Queirós Vieira Martins and José Luis Esteves Dos Santos. *The labelling algorithm for the multiobjective shortest path problem.* 1999.

[Martins *et al.* 2007] Ernesto Queiro Martins, José Manuel Paixão, Mario Silva Rosa and Jose Luis Santos. *Ranking Multiobjective Shortest Paths.* page 23, 2007.

[Martins 1984a] Ernesto Queirós Vieira Martins. *An algorithm for ranking paths that may contain cycles.* European Journal of Operational Research, vol. 18, no. 1, page 123–130, Oct 1984.

[Martins 1984b] Ernesto Queirós Vieira Martins. *On a multicriteria shortest path problem.* European Journal of Operational Research, vol. 16, no. 2, page 236–245, May 1984.

[Matsushima *et al.* 2022] Satoru Matsushima, Clarence Filsfils, Zafar Ali, Zhenbin Li, Kalyani Rajaraman and Amit Dhamija. *SRv6 Implementation and Deployment Status.* Internet-Draft draft-matsushima-spring-srv6-deployment-status-15, Internet Engineering Task Force, April 2022. Work in Progress.

[McQuillan & Walden 1977] John M McQuillan and David C Walden. *The ARPA network design decisions.* Computer Networks (1976), vol. 1, no. 5, pages 243–289, 1977.

[McQuillan *et al.* 1979] John M. McQuillan, Ira Richer and Eric C. Rosen. *An Overview of the New Routing Algorithm for the ARPANET.* In Proceedings of the Sixth Symposium on Data Communications, SIGCOMM '79, page 63–68, New York, NY, USA, 1979. Association for Computing Machinery.

[Medina *et al.* 2001] A. Medina, A. Lakhina, I. Matta and J. Byers. *BRITE: an approach to universal topology generation.* In MASCOTS 2001, Proceedings Ninth International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, pages 346–353, 2001.

[Medrano & Church 2015] F. Medrano and Richard Church. *A Parallel Computing Framework for Finding the Supported Solutions to a Biobjective Network Optimization Problem.* Journal of Multi-Criteria Decision Analysis, vol. 22, 04 2015.

[Merindol *et al.* 2018] Pascal Merindol, Pierre David, Jean-Jacques Pansiot, Francois Clad and Stefano Vissicchio. *A fine-grained multi-source measurement platform correlating routing transitions with packet losses.* Computer Communications, vol. 129, page 166–183, 2018.

[Moore 1959] E.F. Moore. The shortest path through a maze. Bell Telephone System. Technical publications. monograph. Bell Telephone System., 1959.

[Mote *et al.* 1991] John Mote, I. Murthy and David L. Olson. *A parametric approach to solving bicriterion shortest path problems.* 1991.

[Moy 1998] John Moy. *OSPF Version 2.* STD 54, RFC Editor, April 1998. http://www.rfc-editor.org/rfc/rfc2328.txt.

[Müller-Hannemann & Weihe 2006] Matthias Müller-Hannemann and Karsten Weihe. *On the cardinality of the Pareto set in bicriteria shortest path problems.* Annals OR, vol. 147, page 269–286, Oct 2006.

[Namorado Climaco & Queirós Vieira Martins 1982] João Carlos Namorado Climaco and Ernesto Queirós Vieira Martins. *A bicriterion shortest path algorithm.* European Journal of Operational Research, vol. 11, no. 4, page 399–404, Dec 1982.

[Nelakuditi *et al.* 2003] Srihari Nelakuditi, Sanghwan Lee, Yinzhe Yu and Zhi-Li Zhang. *Failure Insensitive Routing for Ensuring Service Availability.* In Proceedings of the 11th International Conference on Quality of Service, IWQoS'03, page 287–304, Berlin, Heidelberg, 2003. Springer-Verlag.

[Nordic Gateway for Research and Education 2014] Nordic Gateway for Research and Education. *GÉANT topology map.* https://www.nordu.net/content/g{é}ant, January 2014. (Accessed on 06/21/2021).

[Paixão & Santos 2007] José Manuel Paixão and José Luis Santos. *Labelling methods for the general case of the multi-objective shortest path problem - a computational study.* 2007. Accepted: 2009-09-01T13:06:15Z.

[Papadimitriou & Yannakakis 2000] C.H. Papadimitriou and M. Yannakakis. *On the approximability of trade-offs and optimal access of Web sources.* In Proceedings 41st Annual Symposium on Foundations of Computer Science, page 86–92, Nov 2000.

[Papaefthymiou & Rodrigue 1997] Marios Papaefthymiou and Joseph Rodrigue. *Implementing parallel shortest-paths algorithms.* DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 30, pages 59–68, 1997.

[Pei & Van der Merwe 2006] Dan Pei and Jacobus Van der Merwe. *BGP Convergence in Virtual Private Networks.* In Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement, IMC '06, page 283–288, New York, NY, USA, 2006. Association for Computing Machinery.

[Pei *et al.* 2005] Dan Pei, Matt Azuma, Dan Massey and Lixia Zhang. *BGP-RCN: improving BGP convergence through root cause notification.* Computer Networks, vol. 48, no. 2, pages 175–194, 2005.

[Pelsser *et al.* 2008] C. Pelsser, T. Takeda, E. Oki and K. Shiomoto. *Improving Route Diversity through the Design of iBGP Topologies*. In 2008 IEEE International Conference on Communications, pages 5732–5738, 2008.

[Pelsser *et al.* 2011] Cristel Pelsser, Olaf Maennel, Pradosh Mohapatra, Randy Bush and Keyur Patel. *Route Flap Damping Made Usable*. In Neil Spring and George F. Riley, editors, Passive and Active Measurement, pages 143–152, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[Pelsser 2006] Cristel Pelsser. *Interdomain traffic engineering with MPLS*. PhD thesis, Catholic University of Louvain, Louvain-la-Neuve, Belgium, 2006.

[Poon & Dryja 2016] Joseph Poon and Thaddeus Dryja. *The bitcoin lightning network*, 2016. Accessed: 2016-07-07.

[Programme 2020] GSMA Future Networks Programme. *Network Slicing : Use Case Requirements*. https://www.gsma.com/futurenetworks/wp-content/uploads/2020/01/2.0_Network-Slicing-Use-Case-Requirements-1.pdf, 2020. (Accessed on 05/26/2021).

[Psenak *et al.* 2020] Peter Psenak, Shraddha Hegde, Clarence Filsfils, Ketan Talaulikar and Arkadiy Gulko. *IGP Flexible Algorithm*. Internet-Draft draft-ietf-lsr-flex-algo-07, IETF Secretariat, April 2020. http://www.ietf.org/internet-drafts/draft-ietf-lsr-flex-algo-07.txt.

[Quoitin *et al.* 2009] Bruno Quoitin, Virginie Van den Schrieck, Pierre Francois and Olivier Bonaventure. *IGen: Generation of router-level Internet topologies through network design heuristics*. In 2009 21st International Teletraffic Congress, pages 1–8, 2009.

[Raith & Ehrgott 2009] Andrea Raith and Matthias Ehrgott. *A comparison of solution strategies for biobjective shortest path problems*. Computers & Operations Research, vol. 36, no. 4, page 1299–1331, Apr 2009.

[Raj & Ibe 2007] Alex Raj and Oliver C. Ibe. *A survey of IP and multiprotocol label switching fast reroute schemes*. Computer Networks, vol. 51, no. 8, pages 1882–1907, 2007.

[Raszuk *et al.* 2021] Robert Raszuk, Bruno Decraene, Christian Cassar, Erik Aman and Kevin Wang. *BGP Optimal Route Reflection (BGP ORR)*. RFC 9107, August 2021.

[Reeves & Salama 2000] D.S. Reeves and H.F. Salama. *A distributed algorithm for delay-constrained unicast routing*. IEEE/ACM Transactions on Networking, vol. 8, no. 2, page 239–250, Apr 2000.

[Rekhter *et al.* 2006] Yakov Rekhter, Susan Hares and Tony Li. *A Border Gateway Protocol 4 (BGP-4)*. RFC 4271, January 2006.

[Rekhter 1991] Yakov Rekhter. *BGP Protocol Analysis*. RFC 1265, October 1991.

[rfc 1982a] *DARPA Internet gateway*. RFC 823, September 1982.

[rfc 1982b] *Exterior Gateway Protocol (EGP)*. RFC 827, October 1982.

[rfc 1990] *OSI IS-IS Intra-domain Routing Protocol.* RFC 1142, February 1990.

[rfc 1991] *Experience with the OSPF Protocol.* RFC 1246, July 1991.

[Sahni 1977] Sartaj Sahni. *General Techniques for Combinatorial Approximation.* Operations Research, vol. 25, no. 6, page 920–936, 1977.

[Sahoo *et al.* 2006] A. Sahoo, K. Kant and P. Mohapatra. *Improving BGP Convergence Delay for Large-Scale Failures.* In International Conference on Dependable Systems and Networks (DSN'06), pages 323–332, 2006.

[Savage *et al.* 1999] Stefan Savage, Andy Collins, Eric Hoffman, John Snell and Thomas Anderson. *The End-to-End Effects of Internet Path Selection.* SIGCOMM Comput. Commun. Rev., vol. 29, no. 4, pages 289–299, August 1999.

[Schoute *et al.* 2016] Eddie Schoute, Laura Mancinska, Tanvirul Islam, Iordanis Kerenidis and Stephanie Wehner. *Shortcuts to quantum network routing.* arXiv preprint arXiv:1610.05238, 2016.

[Schrijver 2012] Alexander Schrijver. *On the History of the Shortest Path Problem.* Documenta Mathematica, page 14, 2012.

[Serafini 1987] Paolo Serafini. *Some Considerations about Computational Complexity for Multi Objective Combinatorial Problems.* In Johannes Jahn and Werner Krabs, editors, Recent Advances and Historical Development of Vector Optimization, pages 222–232, Berlin, Heidelberg, 1987. Springer Berlin Heidelberg.

[Skriver & Andersen 2000] A.J.V. Skriver and K.A. Andersen. *A label correcting approach for solving bicriterion shortest-path problems.* Computers & Operations Research, vol. 27, no. 6, page 507–524, May 2000.

[Skriver 2000] Anders Skriver. *A classification of Bicriterion Shortest Path (BSP) algorithms.* Asia-Pacific Journal of Operational Research, vol. 17, 11 2000.

[Sobrinho 2002] J.L. Sobrinho. *Algebra and algorithms for QoS path computation and hop-by-hop routing in the Internet.* IEEE/ACM Transactions on Networking, vol. 10, no. 4, pages 541–550, 2002.

[Song & Sahni 2006] Meongchul Song and S. Sahni. *Approximation algorithms for multiconstrained quality-of-service routing.* IEEE Transactions on Computers, vol. 55, no. 5, pages 603–617, 2006.

[Sriram *et al.* 1998] R Sriram, G Manimaran and C. Siva Ram Murthy. *Preferred link based delay-constrained least-cost routing in wide area networks.* Computer Communications, vol. 21, no. 18, page 1655–1669, Dec 1998.

[Stephens *et al.* 2013] Brent Stephens, Alan L. Cox and Scott Rixner. *Plinko: Building Provably Resilient Forwarding Tables.* In Proceedings of the Twelfth ACM Workshop on Hot Topics in Networks, HotNets-XII, New York, NY, USA, 2013. Association for Computing Machinery.

[Subramanian *et al.* 2005a] Lakshminarayanan Subramanian, Matthew Caesar, Cheng Tien Ee, Mark Handley, Morley Mao, Scott Shenker and Ion Stoica. *HLP: A next Generation Inter-Domain Routing Protocol.* In Proceedings of the 2005 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, SIGCOMM '05, page 13–24, New York, NY, USA, 2005. Association for Computing Machinery.

[Subramanian *et al.* 2005b] Lakshminarayanan Subramanian, Matthew Caesar, Cheng Tien Ee, Mark Handley, Morley Mao, Scott Shenker and Ion Stoica. *HLP: A next Generation Inter-Domain Routing Protocol.* SIGCOMM Comput. Commun. Rev., vol. 35, no. 4, page 13–24, aug 2005.

[Taillon *et al.* 2020] Mike Taillon, Tarek Saad, Rakesh Gandhi, Abhishek Deshmukh, Markus Jork and Vishnu Pavan Beeram. *RSVP-TE Summary Fast Reroute Extensions for Label Switched Path (LSP) Tunnels.* RFC 8796, July 2020.

[Takaoka 2003] Tadao Takaoka. *Theory of 2–3 Heaps.* Discrete Applied Mathematics, vol. 126, no. 1, pages 115–128, 2003. 5th Annual International Computing and combinatorics Conference.

[Teixeira & Rexford 2006] R. Teixeira and J. Rexford. *Managing Routing Disruptions in Internet Service Provider Networks.* Comm. Mag., vol. 44, no. 3, pages 160–165, March 2006.

[Teixeira *et al.* 2004] Renata Cruz Teixeira, Aman Shaikh, Timothy G. Griffin and Jennifer Rexford. *Dynamics of hot-potato routing in IP networks.* In SIGMETRICS '04/Performance '04, 2004.

[Theiss & Lysne 2003] I. Theiss and Olav Lysne. *FROOTS - Fault Handling in Up*/Down* Routed Networks With Multiple Roots.* Springer-Verlag, 2003.

[Thulasiraman *et al.* 2016] Krishnaiyan Thulasiraman, Subramanian Arumugam, Andreas Brandstädt and Takao Nishizeki. *Handbook of graph theory, combinatorial optimization, and algorithms.* 2016.

[Tsaggouris & Zaroliagis 2009] George Tsaggouris and Christos Zaroliagis. *Multiobjective Optimization: Improved FPTAS for Shortest Paths and Non-Linear Objectives with Applications.* Theory of Computing Systems, vol. 45, no. 1, page 162–186, Jun 2009.

[Tulumello *et al.* 2020] Angelo Tulumello, Andrea Mayer, Marco Bonola, Paolo Lungaroni, Carmine Scarpitta, Stefano Salsano, Ahmed Abdelsalam, Pablo Camarillo, Darren Dukes, Francoid Clad and Clarence Filsfils. *Micro SIDs: a solution for Efficient Representation of Segment IDs in SRv6 Networks*, 2020.

[Uhlig & Tandel 2006] Steve Uhlig and Sébastien Tandel. *Quantifying the BGP Routes Diversity Inside a Tier-1 Network.* In Fernando Boavida, Thomas Plagemann, Burkhard Stiller, Cedric Westphal and Edmundo Monteiro, editors, NETWORKING 2006. Networking Technologies, Services, and Protocols; Performance of Computer and Communication Networks; Mobile and Wireless Communications Systems, pages 1002–1013, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[Ulungu & Teghem 1995] Ekunda Lukata Ulungu and Jacques Teghem. *The two phases method: An efficient procedure to solve bi-objective combinatorial optimization problems.* Foundations of Computing and Decision Sciences, vol. 20, no. 2, pages 149–165, 1995.

[Uttaro *et al.* 2016a] Jim Uttaro, Pierre Francois, Keyur Patel, Jeffrey Haas, Adam Simpson and Roberto Fragassi. *Best Practices for Advertisement of Multiple Paths in IBGP.* Internet-Draft draft-ietf-idr-add-paths-guidelines-08, Internet Engineering Task Force, April 2016. Work in Progress.

[Uttaro *et al.* 2016b] Jim Uttaro, Pierre Francois, Keyur Patel, Jeffrey Haas, Adam Simpson and Roberto Fragassi. *Best Practices for Advertisement of Multiple Paths in IBGP.* Internet-Draft draft-ietf-idr-add-paths-guidelines-08, IETF Secretariat, April 2016.

[Van den Schrieck *et al.* 2010] Virginie Van den Schrieck, Pierre Francois and Olivier Bonaventure. *BGP Add-Paths: The Scaling/Performance Tradeoffs.* IEEE Journal on Selected Areas in Communications, vol. 28, no. 8, pages 1299–1307, 2010.

[Van Mieghem & Kuipers 2003] P. Van Mieghem and F.A. Kuipers. *On the complexity of QoS routing.* Computer Communications, vol. 26, no. 4, page 376–387, Mar 2003.

[Vanaubel *et al.* 2019] Yves Vanaubel, Jean-Romain Luttringer, Pascal Mérindol, Jean-Jacques Pansiot and Benoit Donnet. *TNT, Watch me Explode: A Light in the Dark for Revealing MPLS Tunnels.* In 2019 Network Traffic Measurement and Analysis Conference (TMA), pages 65–72, 2019.

[VanMieghem & Kuipers 2004] P. VanMieghem and F.A. Kuipers. *Concepts of Exact QoS Routing Algorithms.* IEEE/ACM Transactions on Networking, vol. 12, no. 5, page 851–864, Oct 2004.

[Ventre *et al.* 2020] Pier Luigi Ventre, Stefano Salsano, Marco Polverini, Antonio Cianfrani, Ahmed Abdelsalam, Clarence Filsfils, Pablo Camarillo and Francois Clad. *Segment Routing: a Comprehensive Survey of Research Activities, Standardization Efforts and Implementation Results.* arXiv:1904.03471 [cs], Jul 2020. arXiv: 1904.03471.

[Villamizar *et al.* 1998] Curtis Villamizar, Ravi Chandra and Dr. Ramesh Govindan. *BGP Route Flap Damping.* RFC 2439, November 1998.

[Vincke 1974] P Vincke. *Problèmes multicritères.* Cahiers du Centre d'Etudes de Recherche Opérationelle, vol. 16, pages 425–439, 1974.

[Vissicchio *et al.* 2012] Stefano Vissicchio, Luca Cittadini, Laurent Vanbever and Olivier Bonaventure. *iBGP deceptions: More sessions, fewer routes.* In 2012 Proceedings IEEE INFOCOM, pages 2122–2130, 2012.

[Vissicchio *et al.* 2013] Stefano Vissicchio, Laurent Vanbever, Cristel Pelsser, Luca Cittadini, Pierre Francois and Olivier Bonaventure. *Improving Network Agility with Seamless BGP Reconfigurations.* IEEE/ACM Trans. Netw., vol. 21, no. 3, page 990–1002, jun 2013.

[Viswanathan *et al.* 2001] Arun Viswanathan, Eric C. Rosen and Ross Callon. *Multiprotocol Label Switching Architecture.* RFC 3031, January 2001.

[Walton *et al.* 2016] Daniel Walton, Alvaro Retana, Enke Chen and John Scudder. *Advertisement of Multiple Paths in BGP.* RFC 7911, July 2016.

[Wang & Crowcroft 1996] Zheng Wang and J. Crowcroft. *Quality-of-service routing for supporting multimedia applications.* IEEE Journal on Selected Areas in Communications, vol. 14, no. 7, page 1228–1234, Sep 1996.

[Wang 1999] Zheng Wang. *On the complexity of quality of service routing.* Information Processing Letters, vol. 69, no. 3, pages 111–114, 1999.

[Warburton 1987] Arthur Warburton. *Approximation of Pareto Optima in Multiple-Objective, Shortest-Path Problems.* Operations Research, vol. 35, no. 1, page 70–79, 1987.

[Widyono 1994] Ron Widyono. *The Design and Evaluation of Routing Algorithms for Real-time Channels.* page 37, 1994.

[Wood *et al.* 2001] Lloyd Wood, Antoine Clerget, Ilias Andrikopoulos, George Pavlou and Walid Dabbous. *IP routing issues in satellite constellation networks.* International Journal of Satellite Communications, vol. 19, no. 1, pages 69–92, 2001.

[Wu & Cui 2022] Duo Wu and Lin Cui. *A comprehensive survey on Segment Routing Traffic Engineering.* Digital Communications and Networks, 2022.

[Yuan & Liu 2001] Xin Yuan and Xingming Liu. *Heuristic Algorithms for Multi–Constrained Quality of Service Routing.* IEEE INFOCOM, page 10, 2001.

[Zhou 1998] Jianzhong Zhou. *A new distributed routing algorithm for supporting delay-sensitive applications.* In ICCT'98. 1998 International Conference on Communication Technology. Proceedings (IEEE Cat. No.98EX243), volume 2, pages 7 pp. vol.2–, Oct 1998.

Jean-Romain LUTTRINGER

Calcul de Chemins pour Réseaux IP : Routage de la patate Chaude et Froide lors de Pannes & Chemins Multi-contraints pour Segment Routing

Résumé

Les travaux présentés dans cette thèse se décomposent en deux parties centrées autour du routage.

Nous nous intéressons d'abord aux calculs de chemins multicritères, notamment utiles pour router du trafic exigeant une latence faible. Le problème NP-Difficile étudié, appelé DCLC, devient radicalement plus complexe lorsque l'on considère les contraintes opérationnelles rajoutées par la technologie utilisée pour déployer ces chemins, Segment Routing. Nous proposons différents méthodes et algorithmes afin de résoudre DCLC dans un tel contexte opérationnel, et montrons l'efficacité de nos solutions via une évaluation sur des réseaux large-échelle.

Nous nous concentrons ensuite sur les effets néfastes induits par les interactions inter-protocolaires. Les interactions entre BGP (le protocole de routage utilisé dans l'Internet) et l'IGP (utilisé au sein d'un réseau) provoquent un temps de convergence long lors de changements topologiques. Nous retravaillons ces interactions et proposons OPTIC, ramenant ce temps de convergence à une durée marginale. Nous montrons la faisabilité d'OPTIC via évaluation théorique basée sur des données réelles.

Mot-clés : Routage, Calcul de chemins, DCLC, Segment Routing, BGP, IGP, Qualité de Service, Ingénierie de Trafic

Résumé en anglais

The work presented in this thesis is divided into two parts centered around routing.

First, we focus on multi-criteria path computations, which are particularly useful for routing traffic requiring low latency. The NP-hard problem studied, called DCLC, becomes radically more complex when we consider the operational constraints added by the technology used to deploy these paths, Segment Routing. We propose different methods and algorithms to solve DCLC in such an operational context, and show the efficiency of our solutions via an evaluation on large-scale networks.

We then focus on the adverse effects induced by inter-protocol interactions. Interactions between BGP (the routing protocol used in the Internet) and the IGP (used within a network) cause long convergence times during topological changes. We rework these interactions and propose OPTIC, reducing this convergence time to a marginal duration. We show the feasibility of OPTIC via theoretical evaluation based on real data.

Keywords: Routing, Path Computation, DCLC, Segment Routing, BGP, IGP, Quality of Service, Traffic Engineering