Large scale urban transport control for enhanced resilience : Analysis via complex networks, artificial intelligence and Big Data processing

Cécile Daniel

N°d'ordre NNT : 2022LYSE1062

THESE de DOCTORAT DE L'UNIVERSITE DE LYON
opérée au sein de
l'Université Claude Bernard Lyon 1

Ecole Doctorale N° 512
Info Maths

Spécialité de doctorat : Informatique
Discipline : Transport, Mobilité

Soutenue publiquement le 21/04/2022, par :
Cécile DANIEL

Large Scale Urban Transport Control for Enhanced Resilience
Analysis via Complex Networks, Artificial Intelligence and Big Data Processing

Devant le jury composé de :

Hamamache KEDDOUCI	Prof. des Universités, Université Lyon 1	Président
Flavien BALBO	Profe., Ecole Nationale des Mines	Rapporteur
Giovanna DI MARZO SERUNGENDO	Prof., Université de Genève (Suisse)	Rapporteure
Zahia GUESSOUM	MC, Université Gustave Eiffel - Bron	Examinatrice
David REY	Prof. Associé, SKEMA Business School – Sophia Antipolis	Examinateur
Salima HASSAS	Prof. des Universités, Université Lyon 1	Directrice de thèse
Nour-Eddin EL FAOUZI	Dir. de Recherches, Université Gustave Eiffel – Bron	Co-directeur de thèse
Angelo FURNO	Chargé de Recherche, Université Gustave Eiffel - Bron	Co-directeur de thèse
Eugenio ZIMEO	Prof. Associé, Université de Sannio	Invité



DISSERTATION OF THE UNIVERSITY OF LYON at :
Université G. Eiffel, ENTPE, Université Claude Bernard Lyon 1

Doctoral School: 512 - Info Maths

Speciality : Info,
Fields : Transport, Mobility

Defended publicly on 21/04/2022, by :
Cécile Daniel

Large Scale Urban Transport Control for Enhanced Resilience

Analysis via Complex Networks, Artificial Intelligence and Big Data Processing

---

Composition of the doctoral committee :

Giovanna DI MARZO SERUGENDO, Professor, Université de Genève	**Reviewer**
Flavien BALBO, Professor, Ecole Nationale des Mines de Saint-Etienne	**Reviewer**
Zahia GUESSOUM, Maître de conférence, Université de Reims	**Examinator**
David REY, Associate Professor, SKEMA Business School	**Examinator**
Hamamache KEDDOUCI, Professor, Université Lyon 1	**Examinator**
Salima HASSAS, Professor, Université Claude Bernard Lyon 1	**Thesis Director**
Nour-Eddin EL-FAOUZI, Research Director, Université G. Eiffel - ENTPE	**Thesis Director**
Angelo FURNO, Researcher, Université G. Eiffel - ENTPE	**Thesis Director**
Eugenio ZIMEO, Associate Professor, University of Sannio	**Guest**

Thesis prepared at:

LICIT

Laboratoire Ingéniérie CIrculation Transports
LICIT UMR T9401
Université de Lyon

Université Gustave Eiffel
COSYS/LICIT
25 Avenue François Miterrand
Case 24 Cité des mobilités
69675 Bron Cedex, France

ENTPE
LICIT
Rue Maurice Audin
69518 Vaulx-en-Velin Cedex, France

And at:

LIRIS

Laboratoire d'Informatique en Image et Systèmes d'information
UMR 5205 CNRS
Université Claude Bernard Lyon 1

Université Claude Bernard Lyon 1
Bâtiment Nautibus
Campus de la Doua
25 avenue Pierre de Coubertin 69622 Villeurbanne Cedex, France

This work has been supported by the French ANR research project PROMENADE
grant number ANR-18-CE22-0008
https://promenade.licit-lyon.eu/

Abstract

The economic and social development of modern cities relies on the efficiency, mobility and resilience of their transportation systems. The latter has thus become a major research challenge involving multiple disciplines, related to urban activities. Old infrastructures and their limited capacity make cities more and more vulnerable to unpredictable events and increasing demand. Congestions are more frequent, as a consequence of the growth of the urban population, vehicle emissions and air pollution create high stress on the infrastructures and increase time waste for travelers.Solutions to improve traffic conditions, in terms of health, security and traffic management are more and more precise, embracing the generalized use of Artificial Intelligence, jointly with Big Data technologies for data collection, storage and computing. Moreover, traffic simulations are now based on various data sources and on more accurate information to better reproduce traffic dynamics and travelers' behaviors. However, analyzing such complex data in a large scale context is still a significant research challenge that requires solutions based on agent-based modelling, distribution and parallelization. Moreover, the characterization and modelling of transport vulnerabilities for improving human mobility is still at early stages of research.

To prevent congestion and identify vulnerable locations, *i.e.* areas or sections where failures would have high cost consequences, two types of vulnerability analyses are most common in the domain of transport: dynamic system-based, and static topological based. They are both studied in this thesis. The first approach is the dynamic system-based representation that simulates travelers, their trips and the infrastructure over a given period of time, supported by the large volume of data now collected. The second approach is a topological analysis based on graph theory, and static topological considerations. To reduce vulnerability inside road networks, we propose in this thesis a control strategy that dynamically protect identified areas and recommends new routes to drivers to avoid creation of congestion in such zones. Our strategy relies on a hierarchical cooperative multi-agent algorithm. Road infrastructures and vehicles are modeled as agents that dynamically react to traffic conditions. This control strategy enables congestion avoidance and a reduction of the congestion duration. We take into consideration drivers behaviours to find a balance between system performance improvement (system optimum) and individual travel choices (individual optimum), as well as privacy constraints that are now necessary for realistic applications. We prove the robustness of our approach by testing it on different demand scenarios and show that identifying and protecting critical spots of the network improves our strategy. To identify such vulnerable spots, our solution integrates the computation of Betweenness Centrality (BC), a metric usually studied with topological approaches. It is indeed quite unusual to include BC in dynamic congestion avoidance approaches whereas the BC is a popular metric in many domains for critical spot identification in the context of static graph analysis. This is due to the high computation time and the difficulty of computing it on large graphs in a context

of real-time applications. This second problem of computation of BC for static vulnerability analysis is addressed in this thesis with a distributed algorithm for the exact and fast computation of BC developed for large graphs. We provide mathematical proofs of our algorithm exactness and show the high scalability of our approach, developed in an optimized framework for parallel computation. Through distributed approaches, we can design a robust solution, based on a combination of control and topological study, to dynamically reduce vulnerability inside cities in a real-time context. The proposed solution for computation of BC on large-scale graphs can be extended for real-time computation of this metric on time-varying weighted graphs and further enhance our control solution for congestion avoidance based on dynamic vulnerability detection of road networks.

Résumé

Le développement économique et social des villes modernes repose sur l'efficacité, la fiabilité et la résilience des systèmes de transport. Les transports sont donc devenus un défi crucial qui touche plusieurs domaines liés à l'activité des villes. L'ancienneté de certaines infrastructures et leur capacité d'accueil limitée rendent les villes de plus en plus vulnérables à des événements imprévisibles et à une demande croissante. Les embouteillages sont ainsi plus fréquents, augmentant les émissions des véhicules et donc la pollution atmosphérique, et augmentant la perte de temps pour les conducteurs. Les embouteillages sont ainsi devenus ces dernières décennies un problème majeur des réseaux urbains. Les solutions améliorant les conditions de transport d'un point de vue santé, sécurité et gestion du trafic sont de plus en plus précises, adoptant en particulier l'usage généralisé de l'Intelligence Artificielle, conjointement au développement des technologies Big Data de stockage, de détection, de communication et de calcul. Les simulations de trafic sont maintenant basées sur des sources de données variées et sur de l'information plus précise pour mieux reproduire les dynamiques de trafic et le comportement des utilisateurs. L'analyse de telles données complexes et volumineuses reste cependant un défi qui requiert notamment des solutions comme le calcul distribué, les modélisations multi-agent et la parallélisation. De plus, l'étude de la caractérisation et de la modélisation des vulnérabilités des réseaux de transport améliorant la mobilité urbaine n'est encore qu'à ses débuts.

Pour prévenir les congestions et identifier les endroits vulnérables du réseau, *i.e.* les zones ou sections où les défaillances (congestions, inaccessibilité) auraient des conséquences importantes, deux types d'analyse de vulnérabilité sont les plus courantes dans le domaine du transport : une analyse dynamique, basée sur la modélisation des systèmes, et une analyse statique basée sur la topologie des réseaux. Ces deux volets de l'analyse de vulnérabilité sont étudiés dans cette thèse. La première approche, que le volume et la qualité des données collectées a encouragée, simule dynamiquement les voyageurs, leurs trajets et les infrastructures de transport sur une période de temps. La deuxième approche est basée sur la théorie des graphes et des considérations statiques de topologie. Dans ce contexte-là, nous proposons dans cette thèse tout d'abord une stratégie de contrôle qui calcule et recommande dynamiquement de nouvelles routes aux conducteurs pour éviter la création de congestions et ainsi réduire la vulnérabilité des réseaux urbains. Notre stratégie repose sur un algorithme de coopération hiérarchique d'un système multi-agent où le réseau routier et les véhicules sont modélisés comme des agents qui réagissent en temps réel aux conditions de trafic. Cette stratégie de contrôle permet une diminution de création de congestion et une réduction de la durée des embouteillages lorsque la congestion ne peut être évitée. Nous prenons en considération le comportement des conducteurs pour trouver un équilibre entre les performances du système de transport et les préférences individuelles, ainsi que les contraintes liées à la protection des données qui sont

nécessaires dans les applications réelles. Nous montrons la robustesse de notre approche en la testant sur différents scénarios de demande de notre modèle et nous montrons aussi que l'identification des noeuds vulnérables du réseau améliore la qualité de notre stratégie. Nous identifions ces endroits vulnérables grâce à la Betweenness Centrality (BC), mesure de résilience qui appartient normalement à l'analyse topologique de la vulnérabilité. Elle est donc rarement incluse dans les approches dynamiques, alors qu'elle est utilisée dans de nombreux domaines pour l'identification de noeuds critiques. Cela s'explique par la difficulté de calculer cette métrique dans un contexte de calcul en temps réel sur des réseaux statiques à large échelle. Ce second problème de calcul de BC pour l'analyse statique de vulnérabilité est étudié dans un deuxième temps avec un algorithme distribué de calcul exact et rapide de la BC sur de grands graphes. Nous apportons les preuves mathématiques de l'exactitude de la BC calculée ainsi et montrons que dans un contexte optimal de calcul distribué, cette approche est scalable.Basée sur des approches distribuées, notre solution de réduction dynamique de vulnérabilité d'un réseau urbain, via du contrôle combiné à une étude topologique, est robuste et fonctionne dans un contexte de temps réel. La solution proposée pour calculer la BC sur de grands graphes peut être étendue pour du calcul en temps réel sur des graphes pondérés et ainsi compléter la solution de réduction de congestion via de la détection dynamique de vulnérabilité.

Remerciements

Tout d'abord je souhaite remercier mes encadrants, qui ont chacun à leur manière contribué à cette thèse et m'ont permis d'avancer et progresser tout au long de ces 3 années:

- Nour-Eddin El Faouzi: merci pour ta patience, le recul et le relief que tu as apportés à cette thèse;
- Salima Hassas: merci pour ton accompagnement dans le milieu multi-agent, pour le temps consacré et pour la multitude d'idées que tu avais, j'espère que celles non exploitées pourront servir pour d'autres travaux;
- Angelo Furno: un grand merci pour pour ta patience (il m'arrive de râler parait-il), pour ta rigueur et ton exigence. Tu m'as souvent forcée à me poser les bonnes questions.

I would like to thank all the members of my PHD committee: Professor Giovanna Di Marzo Serugendo and Professor Flavien Balbo, the reviewers, Zahia Guessoum, David Rey and Hamamache Keddouci, the examinators.

I would also like to thank Professor Eugenio Zimeo and Lorenzo Goglia for their collaboration on the BC work, I'm very glad I was able to spend some time with you in Benevento right before the lock-downs.

Un immense merci à Andrès, pour son accompagnement sur la deuxième moitié de ma thèse, sa patience et sa pédagogie. Travailler avec toi aura été très enrichissant et fort sympathique. J'espère que nos longues conversations continueront encore longtemps.

Merci aussi à Cécile B., qui m'a aidée à dompter SymuPy et qui a patiemment répondu à chacune de mes questions.

Je remercie tous les membres du labo, avec qui j'ai passé de très bons moments, autour de cafés, de gâteaux du lundi, de verres gagnés/perdus en paris.

Un merci tout particulier à Loïc et Elise, mes partenaires de galère du premier instant, nos pauses café, nos footings et nos verres sont des souvenirs précieux.

Un autre merci particulier à Anne-Christine et Loic (encore), qui m'ont généreusement hébergée après mon exode post covid et avec qui la coloc est un régal au sens propre comme au figuré.

Merci à ma famille, qui a vécu presque autant que moi les émotions de la thèse. Merci à mes parents pour leur soutien indéfectible et qui m'ont donné le goût des études, malgré leur insistance parfois un peu oppressante à me faire travailler...

Enfin, last but not least, merci à Alma, la Pénélope de mon Odyssée, ton soutien dans les moments les plus durs, ta patience et tes encouragements m'auront portée jusqu'au bout. Promis je ne bouge plus.

Contents

Chapter 1

Introduction

The urban population has rapidly grown from 751 millions of people in 1950 to 4.2 billion in 2018 and will probably grow with 2.5 billion more people, corresponding to 68% of population[1]. The growing concentration of people translates into increasing mobility demand on the urban transport infrastructures. Hence, transport networks become a backbone of urban systems with new transportation modes and new services, like riding scooters or sharing bikes. The complexity and interdependence of all these modes make the transportation system, especially the road network, more vulnerable to failures other modes. A recent example of strikes in France in 2019 showed that consequences of public transportation strikes created 23% more congestion in Lyon [2] and provoked a rush on bike services in Paris, strongly deteriorating the service [3].

Efficient, sustainable and reliable transport in large urban areas is vital for the economic and social development of modern cities and represents a major research challenge involving multiple disciplines, ranging from information and data science to economics and urban planning. The daily mobility of large masses of people and goods in, from and to urban areas strongly depends on a seamlessly available multimodal transport infrastructure, and on proper knowledge of the mobility demand. A system is considered to be vulnerable if its operations can be significantly reduced by failure (*i.e.* improper functioning or dysfunction of one or more elements). The probability and consequences of failure inside cities thus increase, making them more vulnerable to external events, especially extreme weather disasters that can damage infrastructures and provoke peaks of demand, reducing the performances of transportation systems. The New York subway was for example completely shut down because of Hurricane Sandy in October 2012, among many other consequences, paralyzing the city and obliging authorities to propose other modes of transportation. Especially because of climate change, the intensity and the frequency of climate disasters increase and force the cities and

[1] https://www.un.org/development/desa/en/news/population/2018-revision-of-world-urbanization-prospects.html

[2] https://www.lemonde.fr/economie/article/2019/12/09/greve-des-transports-plus-de-500-km-d-embouteillages-sur-les-routes-franciliennes_6022161_3234.html

[3] https://www.lesechos.fr/industrie-services/tourisme-transport/les-velib-en-surchauffe-pendant-la-greve-des-transports-1156790

their transportation systems to be more robust to unpredictable events. Cities will face threats of different kinds that will have economical and social consequences, as well as pollution and thus health issues, enhanced by the population growth.

Resilient cities is thus a goal for main metropolitan areas, but how can we identify vulnerabilities in large networks? What are the solutions to anticipate or recover from perturbations? With the amount of collected data, is it really possible to consider real-time analysis of large scale networks?

The vulnerability of transportation systems is analyzed through two distinct traditions: *static topological analysis*, that focuses on topological and intrinsic vulnerabilities of the network using graph theory, and *dynamic system-based analysis*, which is a more realistic representation of transport network, dealing with demand and supply issues in a dynamic context. This latter requires more information to be effective, hence more computational power. The system-based vulnerability analysis is enhanced by the development of smart cities, enabling better simulations and offering new types of approaches (Artificial Intelligence in particular). Indeed, cities are becoming "smarter" and collect more and more data in real-time, via connected sensors, devices, people and infrastructures. Such massive data possess the great potential to provide highly valuable insights to improve demand characterization, to pinpoint system vulnerabilities and to help designing solutions increasing resilience of the transportation system. The increasingly available amount of data and computation power enable the development of new data-oriented and realistic solutions, with the constraint of real-time analysis in the context of Big Data. New technologies have made it possible to analyze those large data sets and to consider dynamic solutions, especially in Artificial Intelligence, which would have been impossible to implement due to computation capacity and memory limitation. The concrete applications for those solutions are the Intelligent Transportation Systems (ITS), that are more and more developed, due to the recent advances in Information and Communication Technologies (ICT) and the deployment of sensors inside vehicles and in infrastructures.

The resilience of cities in the future relies on the understanding of vulnerabilities via proper indicators and on finding solutions to face threats of different kinds, using ITS.

Objectives of the thesis

The main objective of this thesis is to propose solutions to reduce the vulnerability of large scale transport networks and enhance network resilience, via artificial intelligence, complex networks and Big Data processing.

The first aspect of vulnerability studied in this thesis is the traffic monitoring and demand vulnerability of the network. The solution we develop is a multi-agent control strategy of rerouting. Embracing the recent technological growth (computational power, data availability, communication), our strategy takes into account network constraints, drivers behaviour and preferences, and topological

bottlenecks of the network. The second contribution is related to vulnerability analysis and is a new algorithm of fast computation of a resilience metric in large static unweighted graphs.

We place ourselves in a large scale context where many districts of an urban area are monitored, and in the context of real-time application. The analysis of large scale road networks in real-time can be done by aggregating information or by processing data with distributed approaches.

This thesis is organized as follows:

- Chapter 2: we introduce the different notions of vulnerability, resilience related to serviceability and reliability of transportation systems. We then present what is done and what data are collected in cities to make transportation systems more resilient.
- Chapter 3: we present general concepts for control inside road networks, the need for distributed approaches and describe our proposed solution for better traffic distribution to address demand fluctuation vulnerability issues.
- Chapter 4: we describe our road network control solution prototype, the settings of the performed simulations and the results obtained.
- Chapter 5: we present our algorithm for fast computation of a resilient metric and propose a performance evaluation for synthetic and real graphs. The control solution presented in Chapter 3 and Chapter 4 is improved by adding this resilient metric in the route choice.
- Chapter 6: we develop the limitations of our work and the different perspectives that can compensate for those limitations.

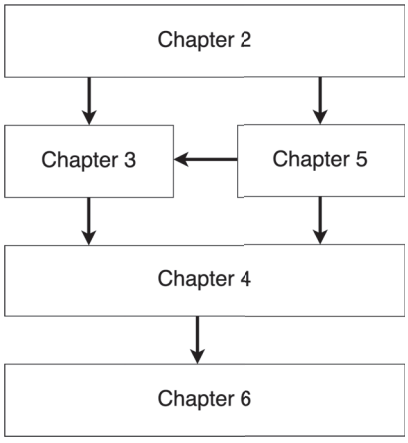


Figure 1.1: Chapter organization

Chapter 2

Vulnerability and Transport Network Management

With the increase of urban population, transportation networks grow as well and become more complex. The dependency of the road network with the other transportation modes and recurrent congestions taking place in most of urban networks make it more vulnerable to disturbance and failure. The consequences of a failure impact more people and more infrastructures than before. Sources of failures are numerous: internal incidents like strikes, technical failures, or road maintenance that are frequent, for example, and external threats, such as terrorist attacks, cyber attacks, climate disasters. The level and duration of disturbance varies depending on the cause of failure and the ability of the transportation system to cope with these events and recover from it.

New technologies enable a better monitoring of environmental, weather or traffic conditions and driver behaviours, offering new possibilities of resilience-oriented applications. However, the amount of collected data represents a challenge and requires distributed approaches to be studied and analyzed.

In this chapter, we first define and introduce the vulnerability, resilience and related concepts inside road networks, we then describe the context of applications enabling a resilient transportation system inside smart cities.

2.1 Vulnerability and Resilience of Transportation Systems

Different concepts are related to the characterization of transport system quality. We present in this section the main notions related to the level of service inside the network, that lead to vulnerability analysis. Our work focuses on a reduction of demand vulnerability and performances are evaluated in view of serviceability and reliability. We also introduce the resilience that encompasses serviceability, reliability and vulnerability.

2.1.1 Serviceability and Reliability

Perturbations inside road networks have consequences on the level of service of transportation systems and many concepts related to transportation system help characterizing the consequences of perturbation, the quality of the network. The performances of a road network are measured in terms of serviceability. The serviceability of a road network is defined by Berdica in [1] as "the possibility to use that link/route/road network during a given time period". An incident is an event that reduces the serviceability of the network, impeding the mobility of some travelers and blocking some parts of the network (road sections, subway station, etc).

The consequences of failures are a decrease of serviceability of the network system as links or nodes become unavailable for a period of time. In road networks, failures result in reduction of accessibility of a road, causing (or caused by) traffic jams, leading to travel times increase. As an example, high demand will produce congestion and will lower performance in terms of speed and travel times, by blocking some road sections, forcing the drivers to find a new route, to wait, to choose another mode or worse, to cancel their trip.

Another related concept of road network analysis is the reliability of the system, that can be measured as the variance of travel cost (travel times *e.g.*) for a given source and destination or the probability of duration lower than a threshold. More globally, reliability of an Origin-Destination pair (OD) is the comparison of the travel duration with a given time. The reliability of a network corresponds to its stability ([2]). Reliability focuses on "maintaining the performance of critical infrastructure elements" (Murray *et al.* [3]). Taylor describes different ways of measuring reliability in [4], related to travel time and capacity of links inside the network. It can be by considering the standard deviation of travel times (Bates *et al.* [5]) or considering percentile values (Lam *et al.* [6] or Tilahun *et al.* [7]). Chen *et al.* ([8]) define it as the probability a network can absorb the demand.

The reliability of the network is affected by two main variables: the demand and the supply. More precisely, Kwon *et al.* ([9]) identify three sources of travel time reliability:

- traffic influencing events, including traffic incidents, work zone activity, weather conditions
- traffic demand, corresponding to fluctuation in day-to-day demand, or special events
- physical road features like traffic control devices (railway crossings for example), inadequate base capacity (bottlenecks)

Typically, after technical failures or extreme weather incidents (flooding, earthquake), infrastructures may be deteriorated and inaccessible, affecting the reliability of the network by forcing the users to find alternative routes. If the failure

happened in a critical spot of the network, like a bridge, the alternative routes may be much longer, thus significantly reducing the reliability. Similarly, when the demand strongly varies inside road networks, the reliability is impacted because the travel times from origin to destination vary as well. This is typically the case when failures like strikes are observed on public transportation system: the demand inside road networks increases because more people will use their car instead of public transportation. The main focus of our work will be on the demand fluctuations.

2.1.2 Vulnerability

Serviceability and reliability are concepts related to the quality of mobility inside the transport system. A more general concept that includes the risks and consequences affecting serviceability and reliability of the network is the *vulnerability*. Many definitions of network vulnerability exist, taking into account probability of incidents, causes and consequences of failures. Berdica ([1], p. 119) defines it as follows: "Vulnerability in the road transportation system is a susceptibility to incidents that can result in considerable reductions in road network serviceability." A network is vulnerable when an unexpected event has high cost consequences and reduces serviceability and reliability of the system. Some nodes or links can for example be identified as vulnerable if on the path of many users. Indeed, consequences would be increased as failure would concern many people. Centrality measures like the Betweenness Centrality help identifying those vulnerable spots.

More generally, the reduction of vulnerability can be done in two ways: *fail-safe* approach, reducing the risk with the protection of vulnerable spots (bridge or tunnel for example) or *safe-fail* approach reducing the consequences of an incident ([1]). The reduction of vulnerability relies on the anticipation of incidents, the reactivity when an incident occurs and the quality of solutions, either to solve the problem or to find alternative ([2]).

2.1.3 Resilience

The resilience is a complementary concept to vulnerability as it is related to the cause and consequences of a disruption in a system but encompasses more parameters of the system as it includes also duration of disturbance, the reach of new equilibrium and is related to the performance of the system. It appeared in biology, when considering the ecosystems ability to recover from disturbance. The concept has now spread in other domains and is more and more studied in transportation systems. US National Academy of Sciences defines the resilience [10] as "the ability to prepare and plan for, absorb, recover from and more successfully adapt to adverse events". Hollnagel [11] proposes a more precise definition that includes the dynamic of the system : the resilience is "the intrinsic ability of a system to adjust its functioning prior to, *during*, or following changes and disturbances, so that it can sustain required operations under both expected and

unexpected conditions". The resilience is thus not only the ability to recover from a disruption but also to anticipate and prepare for unexpected events, and be able to rapidly react. Hollnagel highlights four cornerstones of the resilience, illustrated in Figure 2.1:

- "responding": how to respond to normal or unexpected events by changing the functioning: it can be alerting the people when accident occurs so that they can change plans, or deviation during road maintenance for example;
- "monitoring": monitor the current state of the system: speed, accumulation, air quality, weather conditions for example;
- "learning": from past events, be able to address current situations from the understanding of the past: congestion anticipation with demand prediction typically, knowledge on consequences from previous events like strikes;
- "anticipating": anticipate future events or failures (which falls under the vulnerability analysis). Recent actions are taken, especially for congestion avoidance: traffic speed regulation when high demand is expected (vacation departure day), route recommendation, or during extreme weather events: nearby subway stations can be closed[1].

Vulnerability analysis helps improving the resilience of a system, as it is focused on "what to expect".

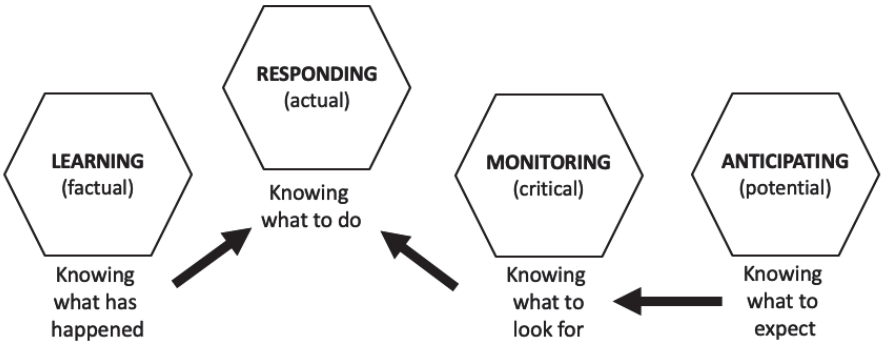


Figure 2.1: The four cornerstones of resilience ([11])

Usually, the resilience is measured when analyzing the performance of the transportation system. In Figure 2.2 from Koren et al. [12], the performance drops after an incident, natural disaster in the case studied by the authors. After the time of response, the system recovers until reaching an equilibrium, that can

[1]https://www.liberation.fr/france/2016/06/03/ratp-une-trentaine-de-stations-au-bord-de-la-fermeture$_1$457194/

be the initial one (A) or a new one (B). If the system is not resilient, it will not be able to recover, like state C, or take more time to recover. The resilience in this case is the area bellow the curve of performance, between the time of disturbance and the time of recovery.

$$R = \int_{t_0}^{t_r} P(t)dt \tag{2.1}$$

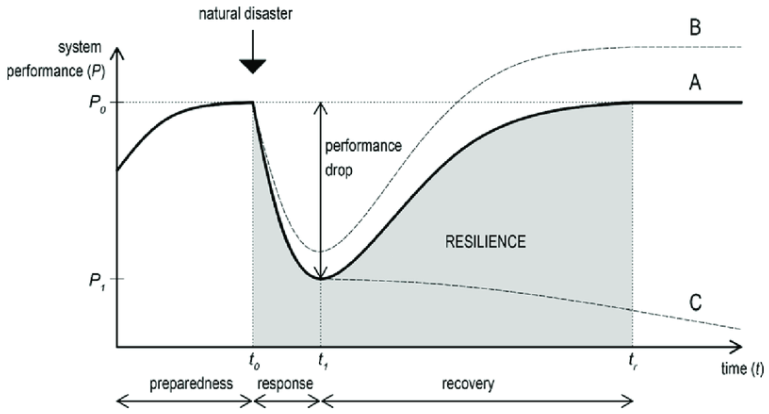


Figure 2.2: Illustrative Example of a Resilience Function (from [12])

The metric for performance assessment depends on the context of the study and area related to the Level Of Service (LOS). It can be the travel time reliability, the speed, the total travel time spent in the network, or any other metric able to provide indications on the performance of the transportation system. It can be related to the whole system or specific components of it. For instance, in the context of an urban large scale transportation network, one could focus on the resilience of specific transport modes or specific regions.

In transportation systems, the main focus is maintaining an acceptable LOS. The increase of resilience is done by limiting the performance drop, reducing the duration of recovery and ideally having a new and better stable state after enhancing the LOS when it is possible.

2.1.4 Vulnerability Analysis and Performance of a Road Network

Cities are more and more vulnerable to external events, due to their size and the different sources of perturbation. Today, global warming in particular is a serious threat for transportation systems as it increases the frequency and the intensity of natural disasters[2], which are the two indicators of vulnerability (probability of happening, and consequences). Extreme weather conditions, like

[2]The different consequences of global warming on network resilience are further developed by Jaroszweski ([13]).

storms or blizzards, may disrupt one or more transport modes, reducing supplies and increasing demand on other transport modes, making the system more vulnerable. Many studies thus focus on the impact of extreme weather incidents, as it is a major threat for infrastructures.

Cities have to be prepared and able to rapidly react otherwise the transportation system would be paralyzed. Their resilience is strongly dependant on the capacity of anticipation, time of reaction and solution found. More globally, two aspects of a transportation system may affect the vulnerability of a road network: variation of demand and variation of supply of transport services. For example, during intense precipitations, drivers behaviours change, the speed is lowered, which decreases the capacity of areas ([14]), and the number of traffic accident increases. In this case, the demand is steady but the supply is lowered, decreasing the performance of the system.

In this thesis, we place ourselves in a more frequent risk context, where the source of vulnerability is the demand variation, as a day-to-day problem to solve.

Vulnerability Analysis: Static vs Dynamic Perspective

Traditionally, vulnerability analysis can be divided in two categories: *topological analysis* and *system-based analysis* ([2]). The topological analysis relies on the representation of transport network as graphs, usually with links representing road sections and nodes road intersections. The advantages of this approach is its simplicity in terms of computation and input data. Graph theory and mathematical considerations enable various topological vulnerability analyses, such as vulnerable nodes identification or topological consequences of node removal. The fast computation algorithms for resilience and vulnerability analysis make it very appealing but the lack of dynamic information such as the demand and the supplies hinders realistic applications. The consequences of disruption from a user perspective are not well evaluated in this case.

The second approach addresses the drawbacks of the first but with the default of complexity. System-based analysis takes into account the demand, the supply and other information about the dynamics of the network. As the demand is a source of vulnerability, it allows testing the robustness of infrastructures with increasing demand or vulnerability measurements. From system-based analysis, consequences on travelers such as travel time delay or mode choice are easily measured. They are also useful to test the performance of the system with a varying demand or capacity. The recent advances in computation technologies reduce the default of system-based analysis and enable large quantities of information to be analyzed.

Those two approaches are rarely combined but in their own way provide lots of insights for vulnerability studies. Recent studies however, focus on the combination of spatio-temporal vulnerability analysis (for example Henry *et al.* [15]). Nowadays, such analyses are enhanced by the quantity and quality of collected

data and new applications in the scope of Intelligent Transportation Systems allow improving the resilience of cities.

2.2 Context of Application for Dynamic Vulnerability and Resilience Analysis

The global notion of resilience of a transport network, we can associate a more concrete field aiming at improving resilience addressing different issues (security, health, congestion,...): the Intelligent Transportation Systems (ITS). As the resilience is knowing what happened, what to look for, what to do and what to expect, technologies of collection, storage and computation of data, in a large scale, are necessary to have a resilient transportation network.

2.2.1 Intelligent Transportation System

Intelligent Transportation Systems (ITS) appeared with the vehicular connectivity and aim at reducing issues related to transportation, via data collection, analysis and communication. They are part of the improvement of resilience inside transportation networks, increasing the knowledge on the transport environment and offering solutions to different kinds of mobility issues. Simple applications appeared decades ago, where traffic information were broadcasted through the radio, or via variable message signs (VMS), signaling traffic perturbations or advising travelers during their trips.

To face the problem of urban mobility, ITS now benefit from the improvement of technologies of data collection and storage, computation power and communication. Recent technological advances enabled indeed the development of applications around mobility inside urban network. Data collection can now be done in real-time and software are more powerful and able to perform computation on large data sets rapidly. ITS are composed of communication and data processing technologies to improve and analyze mobility data. Guerrero-Ibanez *et al.*. ([16], [17]) highlight four principles for the ITS: sustainability, integration, safety, responsiveness and present the main objectives of the ITS: access and mobility, environmental sustainability and economic development. ITS are thus main actors for resilience improvement inside cities, and address the four cornerstones of resilience.

Cooperative ITS, or C-ITS correspond to a new type of ITS where the mobility actors (vehicle or infrastructure) do not act individually but in cooperation. C-ITS are induced by the increasing connectivity and sensor equipment of vehicles and infrastructures. Infrastructures can exchange information with vehicles to alert them on real-time events, affecting their trip. Cooperation between vehicles is also possible to improve general traffic conditions. Adaptive Cruise Control for example helps vehicles to adapt their speed depending on vehicles in front of them, and can regulate traffic speed on portion of roads ([18]).

Applications for ITS are numerous, such as security (lane management, blind spot information), environment (weather prediction, pollution management), assistance (parking spot location, pre-trip information). All these applications help the cities to become more resilient, storing past information, analyzing information in real-time and communicating with users or infrastructures.

2.2.2 Data Collection and Sensors

Data used by the ITS are collected from sensors, that can be *in-road* sensors, also referred as *infrastructure*, or *in-vehicle* sensors. Those sensors enable the monitoring of various indicators, giving information to the traveller or city planners, and more globally help measuring the performance transportation systems.

In-road Sensors

The first and more mature category of sensors is the *in-road* sensors. Measuring outside conditions, they provide information about traffic or also environmental conditions (weather), enabling a better risk evaluation of transportation systems. Guerrero et al. ([17]) distinguish two types of in-road sensors: intrusive (inductive loops, pneumatic tubes, magnetic sensors) and non-intrusive (radar sensors, cameras), depending on their installation on or outside of the road. In both cases, the in-road sensors are accurate and is a well-known technology to rely on. They can measure occupancy, speed, length of a vehicle or flow in a road section. Among the traffic related sensors, the Road Side Units (RSU), measure traffic at fixed location and are able to communicate with vehicles in a close perimeter or other RSU. They give partial information as they are stationary but can provide precious traffic indication such as the flow or the speed of vehicles in a zone.

The main issue of those infrastructures is their high cost, hindering their installation. Traffic State Estimation can reconstruct traffic flow pattern based on observed data ([19]) and can thus compensate for limited deployment of infrastructures. The demand and number of vehicles inside a road network can thus be evaluated with RSU.

In-vehicle Sensors

With the development and spread of self-driving vehicles, more and more sensors are added to new vehicles. They are now able to compute distance to other vehicles for example, have proximity sensors to help parking or monitor tire-pressure.

RAdio Detection And Ranging (RADAR) scan the road and notify drivers when obstacles appear to avoid collision. Self-driving vehicles are equipped with LIght Detection And Ranging (LIDAR) which scans the environment with a 360-degree visibility. Most of the in-vehicle sensors for Advanced Driver Assistance Systems (ADAS) aim at helping the driver and increase its security, alerting the

driver when the risk of collision is high for example or helping him/her to park. Lane management helps the driver stay in its lane, adaptive cruise control manages distance between vehicles and avoid sudden speed change, blind spots are better checked via radars. In the end, all these applications, protecting the driver, reduce the risk of accident and in a way decrease the vulnerability of road network by reducing the probability of incidents.

Nonetheless, the lack of common standards between brands and the difficulties of connecting them with other components are their main drawbacks and are a limitation when designing strategies involving communication with the vehicles.

2.2.3 Communication Systems

In ITS in general, communication can be done in three different ways: infrastructure-to-infrastructure (I2I), vehicle-to-infrastructure (V2I), vehicle-to-vehicle (V2V). Vehicles communicate with other vehicles or infrastructure via vehicular adhoc networks (VANET). V2I and V2V are often used in cooperative control strategies, where vehicles exchange information with other vehicles to adapt their speed for example ([20]), or with infrastructure to send or receive traffic information. V2V approaches require connected vehicles, with the ability to communicate with each other and the same measure system.

V2I communications are more common with the maturity of infrastructure technologies. Moreover, mobile applications (Waze, Google Maps or other navigation systems) are good vectors of communication between vehicles and infrastructures. Those new ways of interactions are though sources of two problems: privacy and security. Particular attention should be paid on the type of information communicated by vehicles and the way they are sent (receiver, data storage, communication protocol, etc). Privacy is a main issue for new applications as information related to mobility are particularly sensitive. Laws protecting individuals privacy emerge, like GDPR in European Union, and force the design of new applications to take into account this constraint before any deployment. Fries *et al.* ([21]) explore the privacy issues that ITS will have or already have to address and the possible solutions (aggregation, masking id, law or third parties).

One of the contributions of this thesis is a TMS on a large scale road network, this is the dimension of ITS we will focus on the rest of the thesis.

2.2.4 Traffic Management Systems

One of the fields of ITS applications (Figure 2.3) is Traffic Management Systems (TMS) that aim at improving traffic flow on road network and reducing or avoiding congestion. TMS focus on the improvement of user mobility in general, for public transportation, vehicles or other transportation modes. TMS are composed of sensors to gather information, applications or computation tools to analyze the traffic situation and offer a solution to the vehicles. TMS are articulated with three phases [23]: information gathering via sensors or mobile

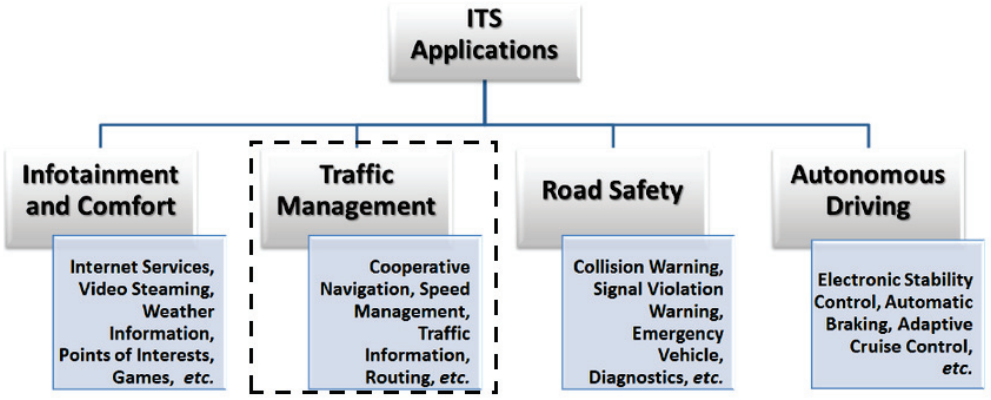


Figure 2.3: ITS Applications (from [22])

applications, data processing to analyze the traffic situation, and service delivery which is the application of the management strategy. Traffic management includes lane management, surveillance, parking management, automatic tolling or intersection management. The common goal of all theses approaches is a better management of traffic flow, mainly of vehicles, in urban areas, addressing demand issues of cities. Depending on the application, the scale of TMS can differ, from a road section to a large urban area. Intersection management, as well as lane management for example, belongs to microscopic approaches, where only portion of roads are observed and vehicles included in the management are only considered for a short period. Larger zones including more roads and more vehicles focus for example on a better distribution of traffic inside the network by making some vehicles change their route in case of congestion, accident for example. Large scale TMS are difficult to deploy because of the large amount of data they can require. De Souza *et al.* ([24]) classify TMS in different categories. The two main categories are *infrastructure-free* and *infrastructure-based*. The infrastructure-free solutions are based on communication between vehicles (V2V) and enables congestion detection or traffic control.

2.3 Toward a Large Scale Management Strategy

Applications for vulnerability analysis are part of ITS and are more and more useful for large cities to improve their resilience. Traffic management systems especially help reducing vulnerability inside the network, avoiding accident or saturation of the network. Their dynamic makes the system more reactive and prepared for unexpected events. ITS enable communication between transportation network actors, large quantity of data and more precise information to be collected for simulation. Analyzing a whole city area increases the size of the data, the duration of observation, the size of the studied population and the size

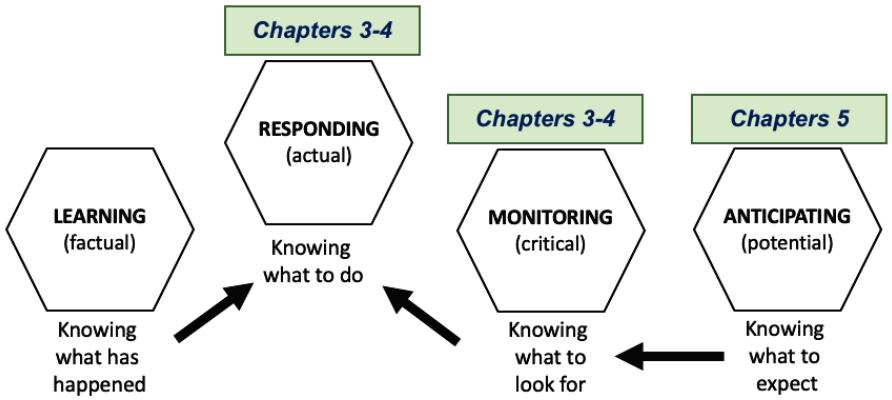


Figure 2.4: Contribution of the thesis on resilience cornerstones

of the network. Performing system-based analysis thus requires to work on a Big Data context, where the computation needs to be distributed.

To address computation issues and be as close as possible to real interactions, decentralized approaches are a solution enabling the distribution of computation on large scale in a dynamic context. In this thesis, we will mainly focus on three aspects of the resilience in a large scale context: responding, monitoring and anticipating. The first two are addressed with a control strategy in a context of TMS, using a distributed multi-agent system in Chapters 3 and 4. Anticipation is addressed in Chapter 5, in a topological vulnerability analysis context where we describe and analyze the performance and scalability of a fast computation algorithm of a resilience indicator that we developed.

Chapter 3

Multi-Agent Control Framework for Vulnerability Mitigation

To design solutions for Traffic Management Systems contributing to a reduction of transportation system vulnerability, the representation of transportation and user interaction need to be defined and contextualized. This modeling is a large part of the dynamic system-based vulnerability analysis, related to the choices of data used, the level of information considered and thus the accuracy of the approach. From simulations, one can evaluate the performance of a traffic management strategy, reproducing part of the behaviours and interactions of travelers. In this chapter, we introduce the context of dynamic analysis we choose, the existing control approaches, and the hypothesis of traffic theory underlying our strategy. Finally, we present our hybrid control strategy.

3.1 Demand as an Enabler of Road Network Vulnerability

As introduced in the previous chapter, two phenomena may affect the reliability and vulnerability of a network: the demand and the supply (*i.e.* network topology, connectivity, etc). In this chapter we focus only on the vulnerability of road networks with respect to the demand.

Berdica ([1]) addresses the demand vulnerability through the reliability on a microscopic level : "Terminal reliability is addressed at link level, and link reliability of connectivity is the probability that demand does not exceed reference capacity on a certain link for given time periods." A congested link is thus considered as a failure because not accessible for the drivers for a period of time. To avoid congested links and at a larger scale, congested areas, different solutions exist to better control traffic, by changing the departure time, the mode or the route of users. The main focus of this work is on the user rerouting during day-to-day perturbation, when the demand is too high (morning peak hours typically).

Our goal is to better distribute vehicles over the network by dynamically anticipate and locate congestions while considering available data and realistic interactions between actors of the rerouting strategy.

3.2 A Brief State of the Art and Motivations

Demand vulnerability in a large scale context is difficult to capture because of the data volume (number of travelers, size of the network, precise traffic information, etc) and the computation time required for simulation. To work on large scale, one common solution is to aggregate the information and the modeling simplification but at the price of lack of realism. We present in the following sections solutions based on aggregated traffic information for large scale network, vehicle-oriented solution based on communication with vehicles and finally the context of multi-agent rerouting strategies.

3.2.1 Traditional Traffic Control

The state estimation and control a network level is a complex task and requires a huge effort in terms of sensor deployment, data processing and forecasting. Reducing model complexity and designing control strategies at this level has become an active research topic and has been one of the main challenges to efficiently cope with traffic congestion in urban cities. Approaches to solve this problem include aggregated directional control (Tumash *et al.* in [25]) or reduction of scalability dependence, (Nikitin *et al.* in [26]) to name a few.

Real-time data-driven solutions based on Intelligent Traffic Systems and connected Vehicles have attracted a lot for deployment effort, as their effectiveness increases with the development of smart city solutions and the growing availability of fine grained data to better monitor traffic conditions. From this perspective, traffic jams can be nowadays more precisely detected and, in some situations, proactively avoided.

Boundary Control

From the perspective of traffic control engineering, infrastructure-based approaches are more suitable as they establish specific boundary actions leaving the routing a decision to the driver. Boundary control has shown to be efficient to limit access to the network and improve its throughput (Boufous *et al.* in [27] for example). In [28], Tumash *et al.* propose a solution of boundary control based on the solution of Hamilton-Jacobi equation on a single road. A subzone boundary control is presented by Yang *et al.* in [29], based on Macroscopic Fundamental Diagram theory (or MFD, later further described). Gao *et al.* ([30]) propose a resilience-oriented boundary control from MFD properties between two regions. They present the concept of traffic-flow resilience and develop a solution to optimize flow between two regions. A hierarchical approach was proposed

by Yildirimoglu *et al.* ([31]) for routing assignment. The upper-level route guidance scheme optimizes network performance based on actuation via regional split ratios, whereas the lower-level path assignment mechanism recommends sub-regional paths for vehicles to follow, satisfying the regional split ratios in order to achieve performance. Through its hierarchical structure, this approach address the problem of translating regional guidance to a more precise route guidance.

Nonetheless, these approaches suffer from low adaptability to sudden changes of the travel demand, variations of the transport supply and evolution of congestion patterns. Moreover, those macroscopic approaches are focused on flow optimization and do not consider individual preferences via microscopic driver behaviour.

Vehicle-Oriented Solutions

Traffic engineers are also focusing the effort of empowering new control schemes by the connectivity between vehicles. In vehicle-oriented approaches, that are infrastructure-free, CoTEC, from Bauza and Gozalvez [32] or CARTIM, from Araujo *et al.* [33] propose a solution for intelligent rerouting to avoid congested areas. They do not require a lot of infrastructures to be effective, only vehicles need to be equipped. Similarly, Pan *et al.* ([34]) propose an infrastructure-free route guidance system based on three different strategies: DSP (Dynamic Shortest Path) which dynamically assigns shortest paths, RkSP (Random k Shortest Path) which randomly chooses a shortest path among the k top and EBkSP (Entropy Based k Shortest Path) which takes into account future positions of vehicles and assigns them to lowest popularity path, enabling a better distribution of vehicles inside the network. The complexity of this last approach face scalability issues. In [35], Wang *et al.* propose a solution of Next Road Rerouting, based on VANET's communications, where infrastructure agents suggest nearby vehicles which road to go next after an unexpected event. Other V2V solutions are based on alerting vehicles when an accident occurs ([36], [37]).

The main goal of those approaches is though to detect congestion rather than avoid them. Another limitation of these applications is that they strongly depend on the number of connected vehicles driving inside the controlled zone. This kind of actions keep traffic safe and try to smooth congestion based on messaging sources but proactive approach to handle congestion requires better and precise advise for users on the road.

3.2.2 Multi-Agent Approaches for Distributed Intelligent Solutions

To compensate the lack of user consideration of boundary control and consider cooperative strategies for a global improvement of the system through anticipation, multi-agent context offers a dynamic, reactive and robust framework, suited

for complex rerouting strategies. We present rapidly in this section the main multi-agent concepts and some multi-agent based control strategies.

Multi-Agent Context

Before motivating the use of multi-agent context, let us first present the main definitions and characteristics of multi-agent systems.

Agent: an agent is an entity able to sense its environment (via sensors) and to act depending the information collected, via effectors. More concretely, the main characteristics associated defined by Wooldridge and Jennings ([38]) to agents are:

- autonomy: the agent performs its action without external intervention;
- social ability: the agent can interact with other agents or external entities;
- responsiveness: the agent is able to sense its environment and respond to changes inside this environment;
- proactiveness: more than responding to the environment variation, the agent acts depending on a given goal, individual or global.

Multi-agent System (MAS): group of agents which evolve in a *environment*, interact and act to achieve an individual or global purpose.

Wooldridge and Jennings ([38]) highlight four types of problem the MAS can solve and that are typically related to traffic assignment:

- openness: inside unpredictable environments that can change quickly, solutions can not be found easily because of the dynamic and various states of the system;
- complexity: decomposing a complex problem, because of its size or its unpredictable state, into simpler sub-problems that can be solved by agents and different types of actions;
- distribution of data, control, expertise or resources: agents representation offers solution when the problem depends on many distributed entities that can interact to find a solution;
- legacy systems: or modularity, the modification of an agent property or behaviour does not require changing the whole system.

MAS offer robustness, simplification of problem solving, reactivity inside a changing environment and a realistic representation of individual behaviours, and is thus our choice for a traffic control strategy.

Multi-agent representation is especially appropriate to address individual microscopic representation as it aims at individualize autonomous entities and behaviours. Moreover, distributing the demand over given supplies is a complex problem of resource allocation, that can be solved in a multi-agent context with

cooperative approaches ([39]). Multi-agent modeling enables problem-solving distribution and thus computational distribution, a dynamic environment and a realistic representation while keeping the simplistic modeling. For these reasons, a multi-agent representation, more and more used in studies at microscopic traffic modeling level, seems to be suited for dynamic analysis on large scale networks. MAS have the advantages to introduce new kinds of interactions either by game theory in cooperative/competitive schemes or by specific communication protocols to reach consensus.

We introduced previously (Chapter 2) different kinds of actors of TMS, vehicles or infrastructures, and their interactions: V2V, V2I or I2I. This can be easily mapped into a multi-agent framework where vehicles and infrastructures are seen as agents. The use of navigation systems makes the travelers even more aware of the environment, receiving information about traffic condition in real-time. The infrastructures such as traffic lights or sensors are also represented in traffic multi-agent modeling as fixed agents.

Multi-Agent Control Strategies

The usual types of multi-agent control strategies are traffic light management and vehicle rerouting, which is the scope of our analysis. Traffic light management is typically a way of applying boundary control and is thus efficient for system improvement but does not really consider traveler preference. Wiering *et al.* [40], and more recently Liu *et al.* [41] developed a multi-agent traffic light management strategy, using reinforcement learning to minimize the waiting time of cars inside urban networks. In [42], Belbachir *et al.* use three infrastructure agents that cooperatively manage intersections in a self-adaptative mechanism but at a microscopic scale. Traffic light management helps smoothing traffic flow inside urban areas but do not face high demand issues that require for the users to find a better route.

For route recommendation, multi-agent frameworks are a way of modeling the autonomy of vehicles and enable cooperation strategies between vehicles and/or infrastructure. Recently Chavhan *et al.* [43] proposed a solution that predict traffic and manage flow distribution in the road network, divided into zones and subzones. The authors use mobile and static agents, corresponding to RSU and vehicles or pedestrians. One type of algorithms explored in multi-agent routing context is the *Ant Colony Optimization* (ACO): introduced in the nineties (Bonabeau *et al.* in [44]) to solve the problem of traveling sales man, ACO algorithms are now widely used to solve complex problems. Inspired by the behaviour of ants searching for food, their functioning is based on the exploration of a topological environment by simple agents, leaving "pheromone" on their way back from a given location (food in the real case of ants). The pheromone later attracts the other agents until it evaporates. The search of shortest paths can thus be done via ACO.

In the context of traffic control solutions, ACO are used to anticipate congestion, evaluating paths and demand on roads with the pheromone, and improve the search of new shortest paths without greedy computations. Cao *et al.*([45]) propose a pheromone based detection of congestion on a microscopic level. Pheromone represents the current traffic state, by using the location of vehicles and another pheromone unveils the future state of congestion from their routes. The pheromone is used to predict potential congestion and reroute vehicles according to this information. An extension includes pheromone for traffic light control. The combination of those two approaches leads to a robust framework. In this case, the size of the network is not very large as every road section is evaluated. An other ant-based solution is developed by Tatomir *et al.* ([46]) where the authors split the network into zones and apply a hierarchical rerouting algorithm. Shortest paths are dynamically updated and computed using local ants (moving only inside zones) and exploration ants (moving between zones). Clustering the network and working with a multi-agent system ensure a robust and scalable solution but in this case, the solution requires a lot of information from the driver. Indeed, the drivers provide traffic state information as well as their route, which could cause some privacy concerns. A hierarchical approach was also developed by Kammoun *et al.*([47]) where ACO is used to reroute vehicles avoiding congestion without creating new congestion.

Based on attraction/repulsion of road sections, these approaches optimize the distribution of vehicles through the network for congestion avoidance but they require a large amount of precise information about the network are though tested on small urban areas because of computation issues and data storing: the traffic conditions are evaluated at road section scale. Our solution aims at reproducing the attractive/repulsive mechanism on a larger scale.

3.2.3 Hierarchical Multi-Agent Control Strategy

The main difficulty in control strategies is to consider both the *user equilibrium*, where the route choices are individually optimized, and the *system optimum*, where the assigned routes are chosen to optimize a performance indicator (total travel time e.g.) in the whole system (Wardrop [48]). Centralized approaches tend to find a balance between those two optimums but require a comprehensive knowledge of the whole network and face scalability issues. On the other hand, user centered approaches commonly found in routing applications focus their attention on providing better individual experience at a cost of degrading the total network performance, and at the end having an impact on user experience. ACO approaches take into consideration the network performance, avoiding concentration of vehicles in the same locations but require a full knowledge of the network and are thus less applicable for large networks.

The combination of aggregated traffic management strategy (on a region level), such as boundary control, and actions of vehicles considered individually has attracted little attention in the recent literature, (Leclercq *et al.* [49]). One of the

main concerns to consider is the possibility to structure an aggregated actuation mechanism that contributes to improve the performance of the system, reducing congestion, without constraining the freedom of drivers to use the network in specific ways. One possible way to perform this is by providing informed routing suggestions, based on attraction or repulsion of monitored zones. Operators can provide an approach where user-centered experience is as relevant as traffic network performance. This combination is proposed and explored within this thesis.

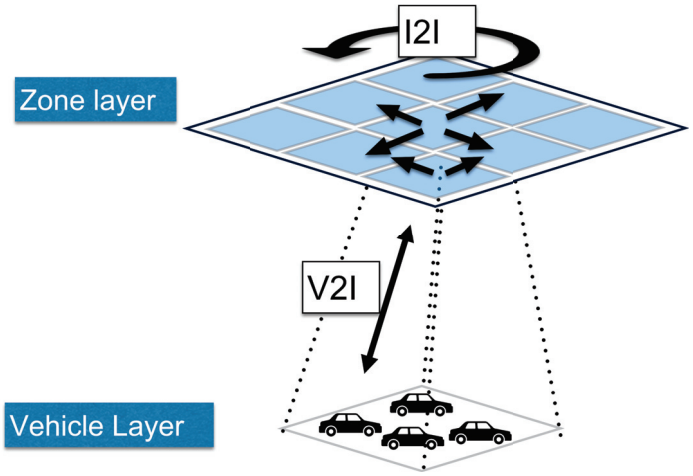


Figure 3.1: Hierarchical Cooperation between Infrastructures and Vehicles

Our solution is to propose a hierarchical dynamic route recommendation algorithm for large scale road networks. It computes and suggests dynamically new routes to vehicles using aggregated congestion proxy measures, computed in a distributed way by infrastructures. In a hierarchical context, with zones and local/global decisions, the traffic network is partitioned into different zones, with homogeneous traffic conditions. Congestion indicators are then locally computed to grade the capability of the zone to accommodate incoming traffic or reroute it via other zones. The aggregated metrics determine zones that may accept vehicles. Vehicles then compute new routes according to the traffic information provided by the zones, with an attraction/repulsion mechanism. The difficulty with this general assumption will be to understand which cooperation caused which effects as vehicles will not always comply with the recommendations of the zones.

Moreover, we designed our strategy to be as realistic as possible, including drivers preferences and privacy constraints as it is a major issue with ITS (see Hahn *et al.* [50]).

This approach overcomes those limitations as it works on large scale networks, through cooperation of two types of agents in a distributed hierarchical framework, with limited vehicle information exchanged with the infrastructures. The combination of dynamic rerouting on microscopic agents via an aggregated control

strategy makes it a new way of improving traffic conditions and hence network vulnerability. From a resilience point of view, by better distributing the traffic flow, our control strategy facilitates the absorption of demand perturbations and reduces the duration of performance drop.

3.3 Multi-Agent Infrastructure and Vehicle Interactions

In this section, we present the general components of our model, and their interactions.

3.3.1 General Architecture of our Framework

When people travel through road networks from one origin to their destination, they go near sensors and produce data during their trip. Traffic state can be sensed by fixed sensors such as inductive loop detectors (ILDs), ultrasonic or radar sensors. Those installations are expensive but mature and accurate enough for traffic management applications ([17]). Those data can be then sent via cellular network to traffic management centers (if we consider a fully distributed approach with one computer per zone for example). While sensors in the infrastructure are expensive, they can provide accurate information at fixed spots. GPS and mobile data provide less accuracy but with better spatial distribution at lower cost, both type of measurements are useful for estimating the traffic state, and can be fused. Once this information is retrieved, it is aggregated and regularly updated on a higher level. The higher level is the abstract representation of reality and is where data processing and analysis are done.

An illustration of those layers is shown in Figure 3.2 where the framework is presented. In our case, the higher level contains distributed agents of two types, representing vehicles and infrastructures. Moreover, we consider two types of agent: the *zones* which corresponds to a group of roads, located in delimited areas corresponding to a given segmentation of a city, and *vehicles* which can actually be seen as an interface to vehicles, such as a navigation system, or an application, capable of interaction with our framework by providing information on the area where the vehicle is located at a given timestamp, and provide route suggestion to the vehicle itself. The road network is represented in the abstract layer as a graph with road intersections as nodes and road sections as links.

The aim of this work is to combine at control level interactions between infrastructures and vehicles.

3.3.2 Zone and Graph Definition

The transcription of the road network inside the control framework is done with two components: the zones, which are the agents and actors of the multi-agent control strategy, and the road graph, which is based on the real network

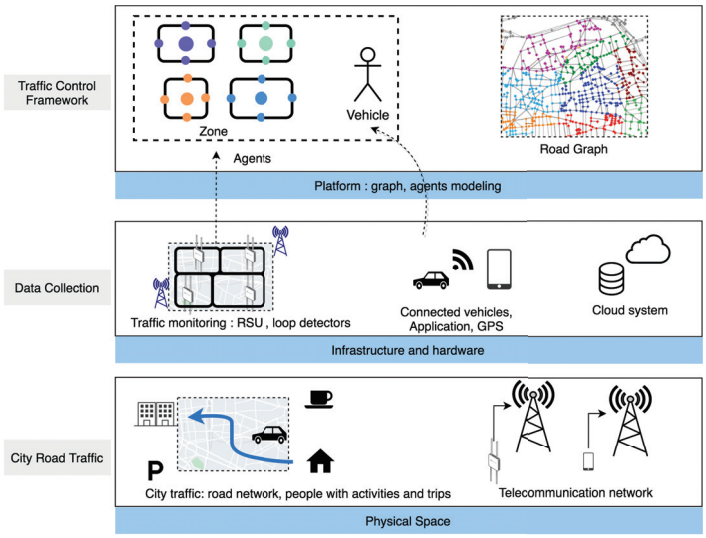


Figure 3.2: Processing of data collected from Road Side Units (RSU) and vehicles

and accessible by both types of agents.

The road network is split into zones. A zone is a delimited area, with sensors fixed at known location. Each sensor is associated to a zone and send information that will be later aggregated by zone. For example the sensors measure the flow inside the zone that will characterize its level of congestion. Zones characteristics, and especially critical performance indicators, are derived from observations.

The graph is an abstract representation of the road network with road sections as links and road intersections as nodes. It is a common resource, broadcasted and updated by all the agents zones, accessible to vehicle agents. In this graph representation, a zone corresponds to a connected sub-network. The weights of the links are the free-flow travel times. The graph is updated with aggregated metrics from sensors data collection. Its goal is to help the vehicles to find a new route from the information communicated by the zones and not to represent the exact network in real-time.

3.3.3 Agent Description

In our model, we consider two types of agents: the vehicle and the infrastructure. We define them so that their communications, their information sharing and their level of traffic understanding are as realistic as possible. Agents are usually characterized by the information they sense, the actions they can perform and the communication with other agents or entities of their environment.

Vehicle Agent:

- *Agent actions*: moves, computes new shortest paths, changes route;

- *Agent information*: id, departure time, origin, destination, route, macro route, arrival time, number of rerouting;
- *Agent interaction*: with its current zone, with its next zone. Mainly receives information from zones;

A vehicle agent has an initial origin, destination, departure time and route. Its route and departure time are considered as an optimal solution for the vehicle (user equilibrium). The vehicle only communicates with its current zone and its next zone. The vehicle precise location is not known by the zone, the zone only knows if the vehicle is in it or not.

The constraint of privacy requires that the vehicle agent computes some information for rerouting on its own. The vehicle agent is able to compute a new route, using the resources it has access to. Here we use local information, such as weighted graph, representing the road network.

It is worth noting that the vehicle information such as speed, acceleration or precise location are not used nor sent to external entities. Moreover, they do not have to be connected as our approach does not require knowledge on the speed or trajectories of the vehicles. Furthermore, our control strategy relies on the cooperation of few vehicles that are willing to change their route, not all vehicles need to have a specific application or to activate their location via GPS to be detected by the zone.

Zone Agent (infrastructure):

- *Agent actions*: monitors traffic inside its perimeter, computes its own indicators of congestion, auto-evaluates itself;
- *Agent information*: id, graph of the road network, critical accumulation, critical spatial speed;
- *Agent interaction*: with its neighbor zones, with vehicles driving in its perimeter;

The zones aggregate local information such as the accumulation to evaluate their state of congestion. They can communicate with their neighbor zones, which are zones that are close to them. They also communicate with the vehicles inside them or driving toward them to help them find a better route. The zones update and share common variables such as the graph.

3.3.4 Communication Between Zones

As defined previously, a zone agent can communicate with its neighbors. To have a fully distributed multi-agent system, we use gossip algorithm to compute global and local aggregated metrics such as mean, min, max or else, via communication between the agents. The purpose of gossip algorithms is to propagate

information among a large network of agents, via simple communications between each agent and part of their neighbors. Gossip algorithms can be implemented to propagate information or to compute a global metric in a network of agents in a distributed way. By communicating their local state and the information received from some of their neighbors, agents are able to converge to the exact value of a global metric. Agents compute an aggregate indicator and share the same information, computed in a distributed way, and achieve consensus or agreement. Through distributed computation, gossip algorithms reduce the dependency on a central system and increase the robustness of our solution. If an agent fails, the others are still able to perceive the surrounding environment.

Kempe *et al.*, in [51] describe the mechanism of gossip implemented with *Push/Sum* algorithm. This algorithm enables the computation of sum, mean, min, max of a metric inside a network of agents with iterative interactions.

Let consider the sum x of the metric values x_i of all the agents:

$$x = \sum_i x_i$$

.

We want x to be computed and known by all the agents in a distributed way. Each agent i will send and update two variables written s_i and w_i. Those two variables will be sent to neighbors and updated iteratively. At the first iteration ($t = 0$), $s_{0,i}$ is initialized with x_i and $w_{0,i}$ is initialized with zero for all the agents except for one, where $w_{0,i} = 1$ (in the case of "mean" computation, all the weights w are initialized at 1). The pseudo-code of Kempe *et al.* is presented in Algorithm 1. In the paper, x is vector, we thus simplify the pseudo-code to a one-dimension variable.

Algorithm 1 Pseudo-code of a Push/Sum iteration (from [51])

1: Let $\{(\hat{s_r}, \hat{w_r})\}$ be all pairs sent to a_i in round $t-1$
2: Let $s_{t,i} = \sum_r \hat{s}_r$, $w_{t,i} = \sum_r \hat{w}_r$
3: Choose shares $\alpha_{t,i,j}$ for each j
4: Send $(\alpha_{t,i,j} \cdot s_{t,i}, \alpha_{t,i,j} \cdot w_{t,i})$ to each j
5: $\hat{x}_{t,i} = \frac{s_{t,i}}{w_{t,i}}$

Each iteration, an agent broadcasts to k random neighbors and itself the following information: $\{\alpha_{t,i,j} \cdot s_{t,i}, \alpha_{t,i,j}$, where $\sum_j \alpha_{t,i,j} = 1$. It will receive from part of its neighbors the information $\{s_{t,r}, w_{t,r}\}$, with r a neighbor of i. From the received information, agent will update its local variable $s_{t+1,i}$ and $w_{t+1,i}$.

$$s_{t+1,i} = \sum_r s_{t,r}$$

$$w_{t+1,i} = \sum_r w_{t,r}$$

where $s_r(t)$ and $w_r(t)$ are the information sent by neighbor r.

At each iteration, the estimated sum $\hat{x}(t)$ computed by agent i is:

$$\hat{x}_{t,i} = \frac{s_{t,i}}{w_{t,i}}$$

.

After few iterations, for each agent i, the ratio $\frac{s_i}{w_i}$ converges to x, enabling every agent to have an estimation of the sum x.

In this case, the agents communicating through gossip algorithms are the zones. They exchange information about their state of congestion and the global state of congestion of the whole network, computed via Push/Sum. Through this distributed algorithm, we remove a dependency of zones on a centralized component and ensure a robust system of information sharing and aggregated computation. The zones can work on a complete decentralized environment.

3.4 Traffic Conditions Analysis and Network Partitioning

Our goal is to homogenize traffic to avoid congestion in a spatial environment and reduce zone vulnerability to high-level demand. By controlling the distribution of vehicles inside the network, between zones, we aim at reducing the duration of congestion and their intensity, and thus the total travel time spent inside the network by the vehicles. This section introduces how the individual choice of users impact the system optimum. We then describe the traffic model that will characterize the zone state and will be a corner stone of our model and induces the partitioning of the road network.

3.4.1 User Equilibrium and Aggregated Effects

Users naturally tend to choose the route that minimize their own time spent (e.g.) in the network. When all the users chose their best route, this situation is referred to as *user equilibrium* situation. The problem of this equilibrium is that it does not guarantee the optimization of transport system which is desirable from Traffic Management perspective. The congestion though worsens and increases the travel time of other users. A situation where the sum of travel times among all the users is minimized is a *system optimum*. Our objective is to lower the congestion index in zones. The idea is still to provide optimal routes to the drivers, subject to a minimum degradation of system optimum. The equilibrium assumptions are often done on a macro-scale and take rarely into account microscopic behaviors. In our work, we combined macroscopic theories with microscopic actions to improve traffic conditions.

3.4.2 Macrocospic Fundamental Diagram: Congestion Proxy

To protect zones from congestion, we need to characterize the congestion state with the available data. Sensors, as previously introduced, enable to measure the out-flow and the accumulation inside zones. We choose the Macroscopic Fundamental Diagrams (MFD) first introduced theoretically by Godfrey *et al.* ([52]) in 1969 as they connect the congestion state of a zone with its accumulation, spatial speed and out-flow. The existence of MFD was later proved under dynamic homogeneous conditions by Geroliminis and Daganzo ([53]) in urban areas with homogeneous distribution of congestion and MFD are now widely used for traffic flow optimization, and thus boundary control.

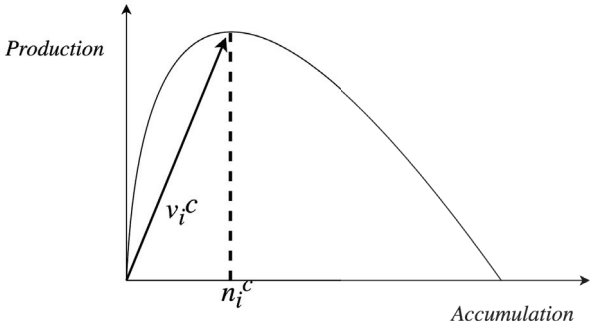


Figure 3.3: MFD

The MFD distinguish two states inside the studied region: the free-flow regime and the unstable state, or congestion state.

The "free flow regime" corresponds to the accumulation lower than a critical value n_i^c. When the accumulation is greater than n_i^c, the zone becomes unstable and congestion appears: the out-flow, or production, decreases, fewer vehicles leave the zone. When the accumulation reaches its maximum, no vehicle can leave the zone, the zone is fully congested.

The maximum of production is reached at $n_i = n_i^c$. The critical speed corresponds to the slope of the line between $n_i = 0$ and $n_i = n_i^c$.

The MFD is very convenient when working on delimited regions as it links the state of congestion of each region with their accumulation and thus characterize congestion state with few and easily measurable data. In our case, we will base our cooperative rerouting strategy with the observed accumulation, in comparison with the critical accumulation value inside each region. The speed is later measured as an indicator of performance but is not an input to characterize congestion in our model. The MFD of an area and the critical values of the indicators can be obtained from real observations ([54], [55]) or simulations ([31], [56]).

The MFDs help identify state of congestion in regions of the network and are thus a good indicator for vulnerability analysis: we want to prevent zones to be in an unstable state.

Detecting the congestion and preventing it with the MFD plays a capital role in the partitioning of the road network. The ellaboration of MFD depends on the partitioning of the network into homogeneous zones ([57]).

3.5 Control Strategy for Vehicle Rerouting

Now that we have introduced the functioning and interactions of the different components, and characterized the congestion identification, we present in this section our hierarchical control algorithm. First we describe the cooperation between zones and then we present how the rerouting is applied from the recommendations of zones.

3.5.1 Infrastructure Cooperation Mechanism

By controlling the distribution of vehicles inside the network, we aim at reducing the duration of congestion and their intensity, and thus the total travel time spent on the network. This problem does not only apply to traffic engineering. The general idea targeting a metric inside a network with a redistribution among agents has been applied in other domains. An illustrative example is the work of Lequay *et al.* ([58]), which describes how multi-agent system enables the reduction of energy consumption. The flexibility of a home (agent) is defined as the reduction of consumption the agent is willing to do without reducing its comfort. The agents have a common goal: the total consumption of electricity can not exceed a certain quantity. By exchanging information, and recalculating the common effort, agents converge to a consensus respecting the initial goal. The whole mechanism and definitions are described in [58]. At each round, agents will engage a certain value of flexibility, regarding the common goal and knowing the total engagement from gossip communications. It will adapt its flexibility depending on the engagement of the others. Comparing real value of flexibility and the engaged initial value, the agent computes its error from which it gets a grade. This grade will later be included in the computation of the new engaged flexibility.

Likewise, our multi-agent system aims at reproducing this mechanism of negotiation and redistribution of vehicles, in order to respect a global constraint and auto-evaluation. We keep the concept of flexibility, engagement and grades, from the work of Lequay *et al.*, by adapting them to our case.

The main differences between our approach and one of Lequay *et al.* are: first of all, the topology constraint of our situation: the redistribution of vehicles is from one zone agent to a close zone agent. Receiving more or less vehicles than expected inside a zone will also affect traffic conditions in the neighbor zones. Moreover, the actions of the agents do not immediately impact the whole network and can thus not be as accurately evaluated as in the context of energy reduction. Another main difference is that we consider two types of agents and the success of our strategy depends on the ability of the vehicles to fulfill the engagement of the

zones. The vehicles do not have the same goal as the zones, we thus used common resources to both types of agents to link them and enable a better cooperation. These common resources are for example a graph, updated by the zones with their congestion state, used later by the vehicles to compute new shortest paths. Finally, we do not consider an aggregated metric such as the sum of flexibilities as an order but rather the reach of a better distributed traffic. Traffic flow will be redistributed based on the accumulation of zones compared with the critical accumulation (from the MFD theory).

3.5.2 Notations and Definitions

The zone agents are denoted as a_i and vehicle agents are denoted as ve. The set of neighbor zone agents of a_i is written $\mathcal{N}_i$. Time-dependant variables are sub-scripted with (t) and are updated periodically with an interval of time equal to τ. The road network is represented as the graph $\mathbf{G}$, with $\mathbf{V}$ the set of nodes (intersections) and $\mathbf{E}$ the set of edges (road sections). $\mathbf{G}$ is weighted with the free-flow travel times and partitioned into to connected sub-networks (the *zones*).

In our strategy, we characterize the zone agents with the following indicators:

- *total flexibility*: their global state of congestion
- *flexibility*: the evolution of their traffic conditions
- *grade*: their ability to cooperate.

Those three aspects and related concepts are described in the dedicated sections.

Table 3.1: Notations

Notation	Description
Graph related notations	
$\mathbf{G}$, $\mathbf{G_M}$	graph, macro graph of road network
$\mathbf{V}$, $\mathbf{V_M}$	set of nodes of $\mathbf{G}$, $\mathbf{G_M}$
$\mathbf{E}$, $\mathbf{E_M}$	set of edges of $\mathbf{G}$, $\mathbf{G_M}$
$tt(l)$	free-flow travel time of link $l \in \mathbf{E}$
$tt_w(l)$	weighted free-flow travel time of link $l \in \mathbf{E}$
$tt_{BC-w}(l)$	BC-weighted free-flow travel time of link $l \in \mathbf{E}$
$BC(l)$	Betweenness Centrality of link $l \in \mathbf{E}$
Zone agent related notations	
a_i	a zone agent
$\mathcal{N}_i$	set of neighbors zones of a_i
n_i^c	critical accumulation inside zone a_i
$n_i(t)$	accumulation inside a_i at time t
$f_i^{tot}(t)$	total flexibility of a_i
$f_i(t),\ \hat{f}_i(t)$	flexibility, engaged flexibility of a_i
$e_{min\|max}(t)$	minimum or maximum moving average error among all zones
$g_i(t)$	grade of a_i
$f_i^{base}(t)$	base flexibility of a_i
Vehicle agent related notations	
ve	a vehicle agent
$t_r(ve)$	timeout for rerouting decision of vehicle ve
$r(ve)$	number of reroutings already done by vehicle ve
Hyperparameters	
τ	time interval for the communications between zones and vehicle
β	hyper-parameter of the link weight function
T_{base}	base time window for the rerouting decision timeout
Indicators	
$TTT_{nocontrol}$	Total Travel Time spent in the network during the simulation without control
$TTT_{control}$	Total Travel Time spent in the network during the simulation with control
$\%TTT$	percentage difference between $TTT_{nocontrol}$ and $TTT_{control}$
Γ_r	rerouting gain

3.5.3 Vehicle Distribution and Flexibility

From the MFD of a zone, we can estimate the maximum number of vehicles that can be inside it before the emergence of congestion, the critical accumulation n_i^c. We thus define the *total flexibility* of a zone i as follows: where $n_i(t)$ is the occupation (number of vehicles) of zone agent a_i at time t. When the total flexibility of a zone is negative, it means that its current occupation is greater than the critical value and a congestion will occur or is already happening.

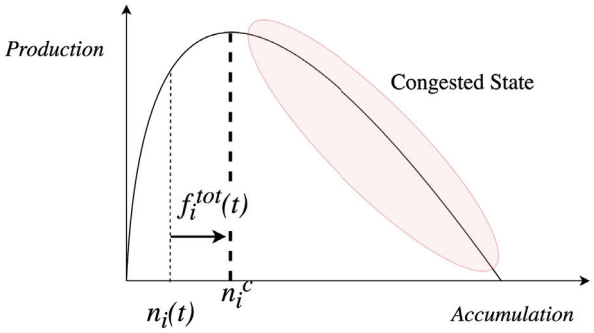


Figure 3.4: Total flexibility compared with the production and accumulation

At each time step, zones will *engage* part of their flexibility, $\hat{f}_i(t)$, depending on their current accumulation $n_i(t)$, their critical capacity $n_i^c(t)$ and the congestion state of their neighbors. The engaged flexibility is the number of vehicles a zone is willing to accept or evacuate until the next time interval.

The actual flexibility at time t depends instead on both the current and the previous state of the transport network, and is defined as follows:

$$f_i(t) = n_i(t) - n_i\left(t - \tau\right) \tag{3.1}$$

In other words, $f_i(t)$ corresponds to the number of vehicles that were actually accepted inside the zone during τ, *i.e.,* the time interval between two engagements of flexibility. The objective of each zone is to keep $n_i(t) < n_i^c$.

3.5.4 Zone Performance and Engagement

Once the actual flexibility is retrieved for each zone (via sensors), we compare it to the one engaged by the zone in the previous time interval[1]. The relative *error* is computed comparing those two values:

$$e_i(t) = \frac{\left(\hat{f}_i(t) - f_i(t)\right)^2}{\hat{f}_i(t)^2} \tag{3.2}$$

[1]We note that $\hat{f}_i(t)$ is computed by each zone at time $t - \tau$.

Based on this error, each zone can grade itself by comparing itself to the others, as follows:

$$g_i(t) = \frac{e_{max}(t) - e_i(t)}{e_{max}(t) - e_{min}(t)} \tag{3.3}$$

where $g_i(t)$ is the grade of zone a_i and $e_{max}(t)$ and $e_{min}(t)$ are the maximum and minimum of errors from all zones. These information are computed in a fully decentralized way via gossiping. Grades range from 0 to 1.

As introduced earlier, the engagement is the number of vehicles a zone is willing to accept. We want the zones to cooperate so that a global optimum in terms of travel time spent in the network could be reached. Hence the engagement of a zone is designed to depend on its own total flexibility, its ability to respect the engagement (measured via the zone's grade) and the total flexibilities and grades of its neighbors. The engaged flexibility is thus defined as follows:

$$\begin{aligned}\hat{f}_i(t) =& \frac{g_i(t) \cdot f_i^{tot}(t) + (1 - g_i(t)) \cdot f_i^{base}(t)}{||\mathcal{N}_i|| + 1} \\ &+ \frac{\sum_{\gamma \in \mathcal{N}_i} \left(g_\gamma(t) \cdot f_\gamma^{tot}(t) + (1 - g_\gamma(t)) \cdot f_\gamma^{base}(t)\right)}{||\mathcal{N}_i|| + 1}\end{aligned} \tag{3.4}$$

Specifically, the engagement of a zone corresponds to the weighted average of the total engagements of its neighbors including itself. The second term of Equation 3.4 corresponds to the cooperative mechanism. The weights are here the grades of each zone.

In this formula, we consider f_i^{base} which is the base flexibility of the zone, computed based on the minimum and maximum flexibilities of all the zones.

$$f_i^{base}(t) = f_{min}(t) \cdot \frac{f_{max}(t) - f_i^{tot}(t)}{f_{max}(t) - f_{min}(t)} \tag{3.5}$$

The use of f_i^{base} forces non-reliable agent to still engage part of their flexibility. The minimum engagement f_i^{base} depends on how the importance of the agent's flexibility $f_i^{tot}(t)$ compares to the minimum and maximum flexibility among the population, respectively f_{min} and f_{max} (known via decentralized aggregation)

f_{min} and f_{max} are also computed in a distributed way, via gossiping.

3.5.5 Routing Mechanism

Initial routes are computed based on a user equilibrium situation, with predefined paths and with times corresponding to a free-flow state. When traffic conditions evolve, in particular in the presence of congestion, those times are altered.

To help the zones achieve their engagement, vehicles are rerouted following the principle of attraction/repulsion. The vehicles are meant to more attracted by zones without congestion, but the access to congested zones must not be fully

restricted. This is achieved by two mechanisms, the *timeout for rerouting decision* and a *new path finding*.

Timeout for Rerouting Decision

When a vehicle enters a zone, it will calculate a timeout that sets when the vehicle will look for an alternative shortest path, allowing to avoid congestion. This timeout will depend on the congestion state of its next zone.

Denoted as $t_r(ve)$ where ve is a vehicle, it is computed as follows:

$$t_r(ve) = t + T_{base} \cdot \left(g_j(t) + max\left(0, \frac{f_j^{tot}(t)}{n_j^c}\right)\right) \tag{3.6}$$

where T_{base} is a time, $g_j(t)$ is the grade of the next zone a_j of vehicle ve, and $\frac{f_j^{tot}(t)}{n_j^c}$ corresponds to the level of congestion of a_j.

The rationale behind (3.6) is that when the zone is congested and currently exhibits a low grade, the timeout is low, and the vehicle should look for a new route almost instantaneously. On the contrary, if its next zone has no or low congestion, and accordingly a high grade, the driver might wait for conditions to evolve before searching for an alternative path. Potentially, the vehicle will enter the next zone before $t_r(ve)$ and thus will not change its route. Vehicles moving through non-congested zones have a low probability to be rerouted. It is worth noting that $f_j^{tot}(t)$ can be negative when the zone is congested, but (3.6) is designed to avoid a negative timeout.

T_{base}, defined in Equation 3.6 represents the reactivity of the driver to find a new route. The timeout for rerouting computed at t is comprised between t and $t + 2 * T_{base}$. The higher T_{base} is and the more chances a vehicle has to be in a new zone. On the contrary, with a low T_{base}, a vehicle will very likely still be in the same zone and will have to compute a new route. The number of potential reroutings should thus increase when the time window decreases. T_{base} represents the anticipation time window of our strategy. Low T_{base} will tend to produce a high number of rerouting decisions.

New Path Finding

At time t_r, if the vehicle has not reached a new zone, the vehicle will look for an alternative route as illustrated in Figure 3.5. To compute its shortest path, the vehicle will request the updated graph to its zone. The new path is computed on $\mathbf{G}$, with weighted links, depending on the aggregated traffic metrics of their zone. A nearly congested zone will have links with higher weights than initially, whereas links of non congested zones would have decreased weights. The vehicles are thus more attracted by zones without congestion but the access to congested zones is not fully restricted. We simulate a mechanism of attraction/repulsion depending on the aggregated level of congestion of the zones. The choice of the weight

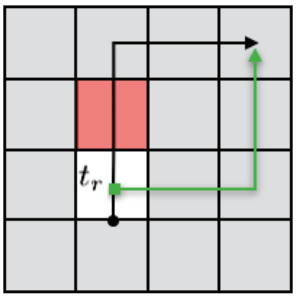


Figure 3.5: New route computation at t_r

function and its parameter is later described in the implementation chapter. The new routes are computed using Dijkstra, with a complexity of $\mathcal{O}(|\mathbf{E}| \cdot log(|\mathbf{V}|))$. In this computation, we need to weight the time spent to cross zones to reflect their congestion state.

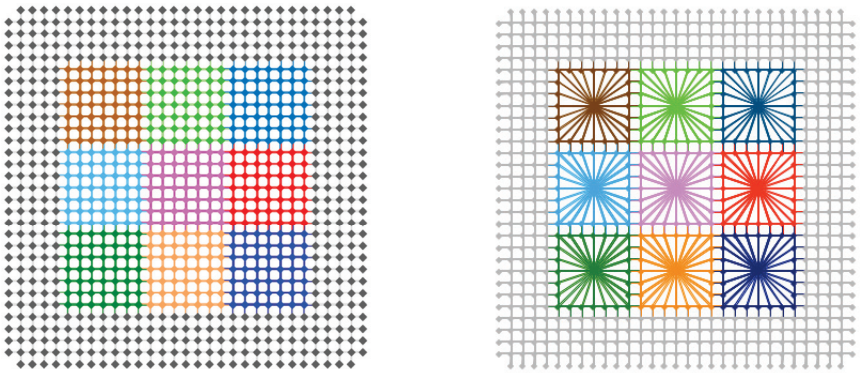
3.5.6 Graph and Macro Graph: Common Resources for all the Agents

To lower the complexity, we simplify the shortest path computation by introducing a *macro graph*. The macro graph enables us to compute a *macro path* which is the succession of zones a vehicle will go through. When a vehicle is about to reroute, it will compute its macro path. The new path will be the shortest path computed on $\mathbf{G}$ filtered on the zones of the macro path.

The macro graph is built from the initial graph and the zone composition. Each zone contains border nodes, that are nodes with neighbors from other zones. A node is chosen as centroid when it is the closest to the barycenter of all the nodes of the zone. From $\mathbf{G}$, we define the macro graph $\mathbf{G_M}$, containing the border nodes of the zones and their centroid. The centroid is connected to all the border nodes of its zone. The weight of the link between the centroid and a border node is computed as the total travel time in free-flow conditions between those two nodes when considering the shortest path inside the real graph $\mathbf{G}$.This needs to be computed only once, and offline. We keep the existing links between border nodes.

The weights of $\mathbf{G}$ and $\mathbf{G_M}$ are updated depending on the total flexibility of the zone agent of the links. At each time interval, the zones update only their links with their local values of total flexibility. The weighted travel time of links belonging to a congested zone will increase with the level of congestion whereas the weight will decrease if the total flexibility is high.

When a vehicle needs to be rerouted, its current zone will send the last updated version of $\mathbf{G}$ and $\mathbf{G_M}$. The graph and macro graph are the common resources to all the agents and enable vehicles to cooperate with zone agents.



Graph of Manhattan 3X3 Macro graph of Manhattan 3X3

Figure 3.6: Graph and macro graph representation of a Manhattan road network partitioned in 9 zones

3.5.7 Updating the Travel-Time Computation

The weights of $\mathbf{G}$ and $\mathbf{G_M}$ are updated depending on the total flexibility of the corresponding zone agent. At each time interval, the zones update only their links with their local values of total flexibility. The weighted travel time of links belonging to a congested zone will increase with the level of congestion whereas the weight will decrease if the total flexibility is high.

When a vehicle needs to be rerouted, its current zone will send the last updated version of $\mathbf{G}$ and $\mathbf{G_M}$. The graph and macro graph are common resources to all the agents.

At t_r, if the vehicle has not reached a new zone, it first computes a macro path from the macro graph $\mathbf{G_M}$, corresponding to the succession of zones it will cross, and then a new route from the graph $\mathbf{G}$, filtered with the zones of the macro path. The macro paths and the new shortest paths are computed on $\mathbf{G}$ and $\mathbf{G_M}$ with weighted links. To attract/repulse vehicles from free-flow state/congested zones, the weight function is defined to reduce/increase link travel time. The weights of the links depend on the initial free flow travel time and the total flexibility of the zones.

We chose a sigmoid function, defined as follows:

$$tt_w(l) = tt(l) \cdot \left(\alpha - \frac{1}{1 + e^{-\beta \cdot f_i^{tot}(t)}} \right) \tag{3.7}$$

where l is a link inside zone agent a_i, $tt(l)$ is its free-flow travel time and $tt_w(l)$ is travel time weighted by zone-related congestion information.

The sigmoid enables the attraction/repulsion mechanism to be more or less smooth depending on its parameter β. Low values of β correspond to a weight function close to affine, whereas high values correspond to a binary step function

(see Fig. 3.7). If the weight applied to a zone is too high, the zone will not be reachable at all. Conversely, if the weight is too low, it will attract all the vehicles.

In our case, the limits of the weights are set with a minimum of $0.5 * tt(l)$ and a maximum of $1.5 * tt(l)$, corresponding to $\alpha = 1.5$.

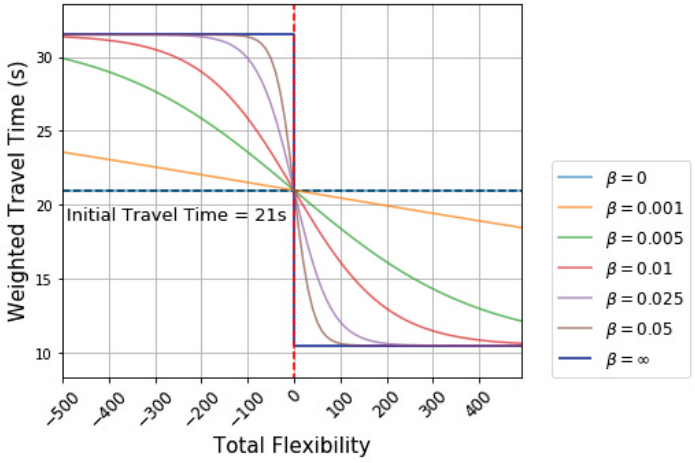


Figure 3.7: Sigmoid weight function to update links of $\mathbf{G}$ and $\mathbf{G_M}$, with $tt(l) = 21s$, corresponding to the free-flow travel time of most of the links of Manhattan network

Gridlocks may appear at simulation level due to the circular paths existing in specific traffic networks, and the nature of the dynamic traffic assignment. In order to avoid them, we assign a random shortest path among the k-top shortest paths (like in [34]) with $k = 5$. Otherwise, some vehicles would choose the same paths and create even bigger congestion than without control action.

3.5.8 Stochastic Acceptance of Rerouting

As the initial routes correspond to a user equilibrium, we consider that few drivers actually accept to change their routes many times. The probability they do should quickly decrease when they have already done one rerouting. We can thus consider their rerouting acceptance with a decreasing probability of happening:

$$p(accept, ve) = e^{-r(ve)-0.1} \tag{3.8}$$

with $r(ve)$ the number of rerouting of the vehicle ve. If $r(ve) = 0$, vehicles have 90% chances of accepting the rerouting, the confiance rate is hence set to 0.9. With $r(ve) = 1$, the probability drops to 0.33, as shown in Figure 3.8.

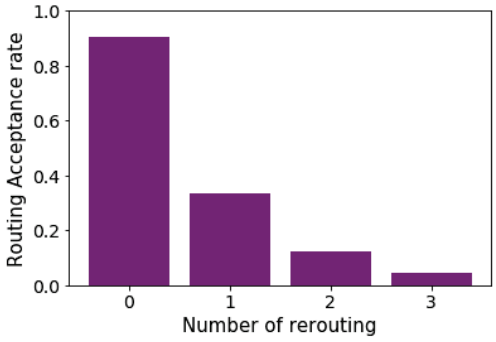


Figure 3.8: Routing acceptance

3.6 Conclusion

This approach embraces the new possibilities offered by recent technologies that enable communications between infrastructures and vehicles and a better monitoring of traffic conditions in a large scale road network. The large scale issue is addressed using a hierarchical traffic control, with aggregated information used for microscopic rerouting. Zones of the network are characterized using the MFD and compare themselves with their neighbors. Through cooperation and auto-evaluation, they provide information to vehicles that help them find the accurate moment to compute new routes, based on the graph updated with traffic state of the zones. Vehicles do not share particularly sensitive information with the zones but are still able to take part to the cooperative control strategy. From this theoretical presentation of our strategy, we present in the next chapter its implementation and performance evaluation in a simulation context.

Chapter 4

Implementation and Use Cases

In this chapter, we propose an implementation of the control strategy described in the previous chapter. After presenting the prototype of our framework and the main functions of agents, we analyze the results obtained after a sensibility analysis, on synthetic and real networks. Finally, we include a topological resilience metric as part of our control strategy and show that the former enables better reroutings and thus fewer driver solicitations.

4.1 Prototype

To test our strategy, we needed to implement a simulation on a large scale road network for a duration of at least 3 hours, corresponding to rush hours inside cities. We present in this section the required components of our prototype and the choices made to run the simulation. We also present the traffic simulator and the data we used for our strategy.

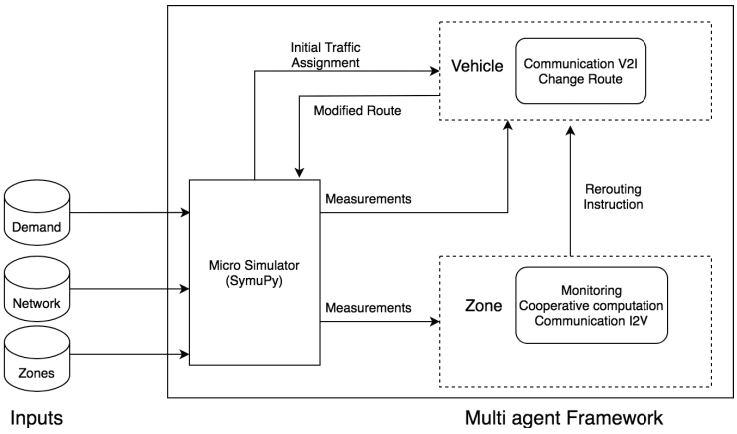


Figure 4.1: Framework of simulation

The traffic simulator is **SymuPy**[1], an open source simulation tool for microscopic transport simulation. The car-following law is based on the Lagrangian

[1]https://github.com/licit-lab/symuvia

resolution (Leclercq et al. [59]) of the LWR model ([60], [61]). SymuPy is not designed to be multi-agent, we thus created in Python the vehicle agents and zone agents outside of the simulator as shown in Figure 4.1. The agents update their information from the simulator, process the information and once a new route is assigned, it is transmitted to the simulator. Our prototype, implemented in Python, wraps SymuPy in a multi-agent framework. The latter, in the end, becomes a multi-agent framework that integrates a powerful tool for realistic traffic simulations. Realistic simulations are generally a limitation of concurrent multi-agent solutions from the state of the art (like MATSim or GAMA).

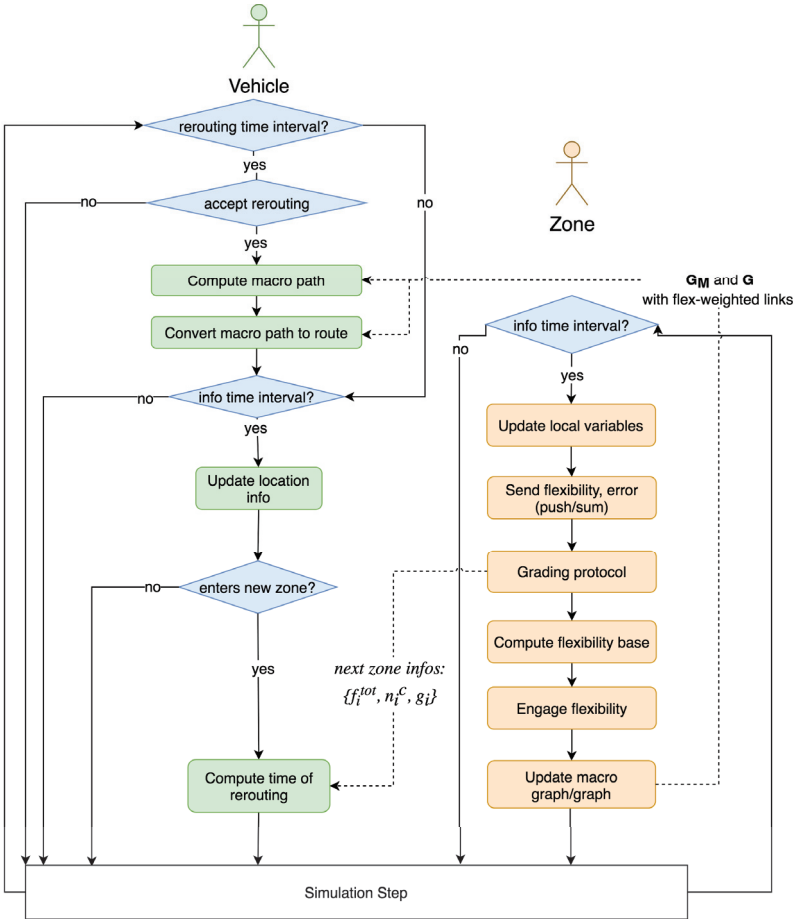


Figure 4.2: Zone-Vehicle Interactions

Agents are thus Python objects and shortest paths are computed with Dijsktra algorithm on the library Networkx. The actions of zone agents and vehicle agents are presented in Figure 4.2, that includes the adjustments we have done for prototyping purpose, described in the following sections.

To improve our strategy and understand the impacts on performance, we

modify some hyper-parameters and evaluate the gain or loss compared with the situation without control. The two main hyper-parameters we focus on are T_{base}, the time window determining the time of rerouting, and β, a parameter of the link-weight function of the graph. We define in this section the hyper-parameters and the evaluation of our strategy for the system and the vehicles. We then present the results, with hyper-parameter modifications and on a single case, and then test our solution on unknown demand scenarios. Finally we present the results for a real case scenario on a subnetwork of Lyon road network.

4.2 Hyper-Parameter Sensitivity Analysis

To improve our strategy and understand the impacts on performance, we evaluate the gain or loss compared with the situation without control, within a set of selected values for two hyper parameters. Those hyper-parameters were chosen as they affect the quantity and the nature of reroutings.

The first hyper-parameters on which we focused is T_{base}, the time window determining the timeout for rerouting decision, which is presented in section **??**. T_{base} can be seen as the reactivity of the vehicles and has a major impact on the number of reroutings, as shown in our experiments. The values of T_{base} that we test are expressed in seconds.

The second is β, a parameter relative to the sigmoid function applied to weight the links in the graph, in function of the congestion level (section 3.5.7). A high β will distinguish zones as "congested" / "non congested", close to a binary function ($\beta = +\infty$), while a low β will smooth the weights of the links depending on the total flexibility (see Figure 3.7) up to the point when weights are left unchanged ($\beta = 0$). We tested the extreme cases as well as values empirically chosen between 0.001 and 0.05. New paths thus strongly depends on β as it affects the attraction/repulsion of zones.

A last hyperparameter is τ, which corresponds to the time step used to update agent state variables. Experimentally, we set it at 60 seconds.

4.3 Performance Evaluation

The indicator of congestion reduction we consider to evaluate our strategy and compare the results is the reduction of Total Travel Time (TTT) in percentage.

$$\%TTT = \frac{TTT_{nocontrol} - TTT_{control}}{TTT_{nocontrol}} \tag{4.1}$$

The best parameters are those with high $\%TTT$ (Equation 4.1) and a limited impact on user equilibrium. To quantify the user disturbance, we consider the percentage of rerouted vehicles and the total number of reroutings.

The variation of T_{base} makes the anticipation of congestion avoidance change (hence the number of reroutings) while β, as it changes the weights on the graph and macro graph, makes the choice of the new shortest paths vary.

4.4 Case Study: the Manhattan Grid

As a baseline, we consider a grid Manhattan network to test our algorithm, with 3712 nodes and 10324 edges, as it represents many city centers (see the work of Boeing *et al.* [62] on US cities). This grid is split into 9 (3X3) and 25 (5X5) zones, representing a zone in the center, surrounded by one circle of zones (3X3 zones) or two circles (5X5 zones). The two sizes of zones enable a comparison of our control strategy performance depending on two clusterings. The critical values of spatial speed and accumulation of the zones are computed in advance, from previous simulations. A *neutral zone* surrounds the set of zones, containing the entry and exit point of the demand. The nodes belonging to the neutral zone are included to the graph and macro graph but as the neutral zone is not an agent, it does not take part to the cooperative strategy and does not communicate with zones nor vehicles. Vehicles driving inside the neutral zone do not compute new route nor receive information from their next zone and the weights of links inside the neutral zone do not change during the simulation. It is worth noting that the global performance of the system, however, includes the traffic indicators of the neutral zone.

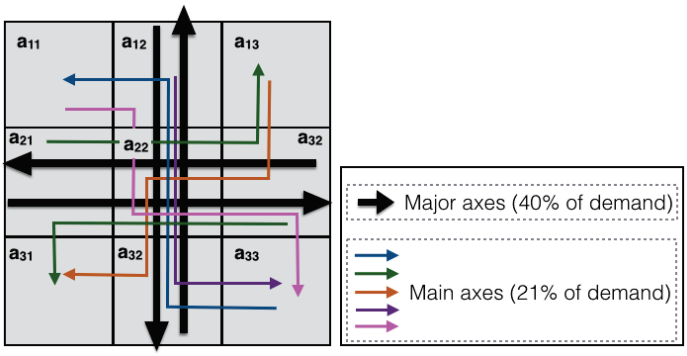


Figure 4.3: Main axes of demand in Manhattan 3X3

The vehicles demand is predefined and corresponds to a user equilibrium situation. Every driver agent starts the simulation with a given route and a given departure time. We consider in our base case simulation 16743 vehicles, in an approximately $4h$ simulation. Their origin and destination points are in the neutral zone. Most of the trips goes through the center, the main trip axes are shown in Figure 4.3.

We run our solution with varying hyper-parameters that modify the reactivity of our strategy and the penalization of congested zones, on one reference demand scenario and later test it on different demand scenarios.

4.4.1 Hyper-Parameter Choice

The exploration of hyper-parameters allows us to first notice that clustering with 25 zones gives more often good results than with 9 zones. The more zones we have and the more precise is the congestion detection and the rerouting which explains the stability and robustness of the rerouting strategy when working with 25 zones.

We can observe that the variation of one parameter when fixing the other does not induce a predictable $\%TTT$ reduction. There is no monotonic variation neither an optimum that could lead to global optimal hyper-parameters.

Logically, we can observe that when T_{base} increases, the number of reroutings decreases. The outliers, corresponding to blank squares in Figure 4.4 and Figure 4.5 are simulations where a grid lock has appeared, creating more congestions and thus more reroutings than expected.

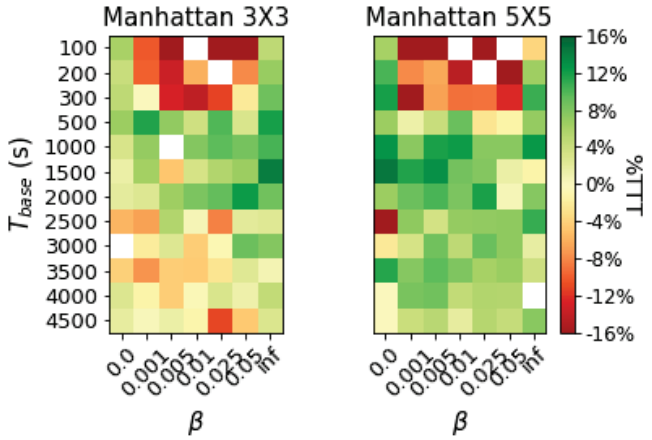


Figure 4.4: %TTT depending on β and T_{base} (expressed in seconds)

In Manhattan 3X3, we can observe that we have better results when β is high, which corresponds to a strict filtering of congested zones. On the contrary, low values of β improve the simulations in the case of 5X5. This can be explained by the fact that in Manhattan 3X3, due to the low number of zones, if the filtering is not binary, vehicles would still go to the congested zone. A clustering with few zones offers less alternative to congestion making the congested location not that repulsive. On the other hand, in the case of Manhattan 5X5, low values of β offer a smooth filtering, more precise as the values of the links strongly depend on the total flexibility of their zone. In the extreme case where $\beta = 0$, corresponding to

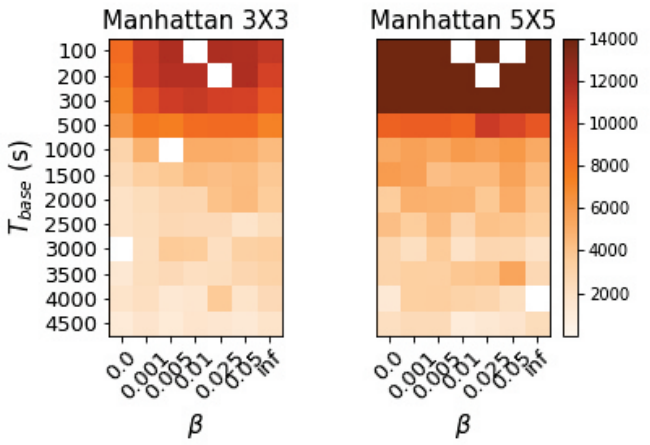


Figure 4.5: Number of reroutings depending on β and T_{base}

$\mathbf{G}$ and $\mathbf{G_M}$ never updated, results show an improvement for most of T_{base} values in Manhattan 5X5 but unstable: when $T_{base} = 2500s$, the performance dropped. Results for Manhattan 3X3 are always lower than 5% with $\beta = 0$ showing that with fewer zones, the penalization of congested zones is more important to improve the results whereas a more precise clustering of the network does not require a special value of β (and weight function) to be efficient.

Moreover, higher values of T_{base} reduce the $\%TTT$ improvement or worse, deteriorate the initial situation. This is explained by the reactivity of vehicles: large T_{base} will provoke late reroutings if the vehicle is still in the zone. This means that when a congestion appears, vehicles entering a nearby zone will reduce their chance of avoiding the congestion by computing a new route.

In both cases, T_{base} has to be higher than 500 to ensure $\%TTT$ reduction, except for extreme values of β.

In Figure 4.6, we can notice that the efficiency of our strategy depends on the number of reroutings. Especially, with too few reroutings (≤ 3000), most of the results are worse with control than without. A certain number of reroutings, around 3000, ensures an improvement. On the contrary, too many reroutings (more than 10000) worsen the initial situation and provoke sometime more congestions than initially, which is the case when T_{base} is lower than $500s$. From the colors representing T_{base}, the relationship between the rerouting time window and the number of reroutings is confirmed: the lower T_{base} is and the greater the number of rerouting is.

Many sets of $\{T_{base}, \beta\}$ enable a reduction of $\%TTT$ greater than 10% but with a varying number of reroutings. We consider rerouting action as a cost for the driver and thus the best solution is not only the one maximizing the $\%TTT$ reduction but also having the less impact on vehicle rerouting. From Table 4.1, one perfect hyper-parameter calibration does not appear. The highest $\%TTT$

Figure 4.6: $\%TTT$ compared with the number of rerouting and T_{base}

Table 4.1: Results for set of parameters giving the the highest $\%TTT$ reduction

size	T_{base}	β	$\%TTT$	**%rerouted vehicles**	**rerouting per vehicles**
3X3	500	$+\infty$	11.9 %	40 %	0.43
	1500	$+\infty$	14.2 %	22 %	0.23
	2000	0.05	12.4 %	25 %	0.27
5X5	1000	0	12.8 %	29 %	0.31
		0.01	12.2 %	33 %	0.36
		$+\infty$	12.4 %	30 %	0.31
	1500	0	14.6 %	33 %	0.36
		0.005	13.1 %	25 %	0.25
	2500	0.025	13.0 %	28 %	0.29

reduction (14.6%) provoked reroutings for 33% of the vehicles, which is quite high. We can not assume that a third of the vehicles are willing to change their route.

More globally, best results are obtained for a T_{base} between 1000s and 2500s.

4.4.2 Single Case Comparison With and Without Control

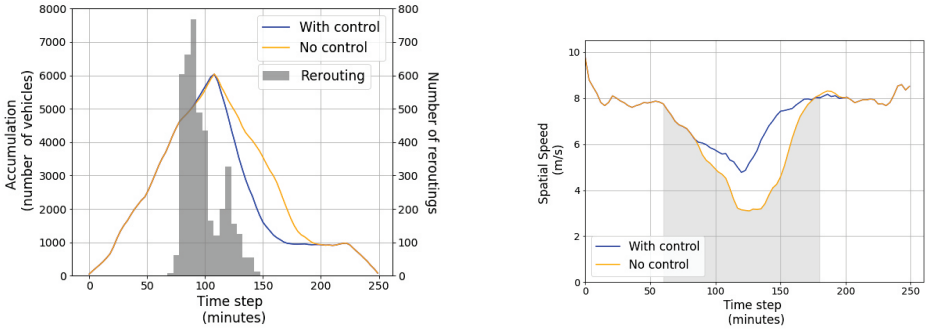
To better understand the impact of our control strategy on the agents and further analyze the dynamic between zones and vehicles, we focus on a single case, with a high $\%TTT$. We choose Manhattan 5X5 for this single case analysis and generalization as this size of clustering gives more stable and robust results. From the previous results, the hyper-parameters maximizing the total travel time reduction without generating too many reroutings are T_{base} around 1500s and

low β values. We here consider the case of $\beta = 0.005$ and $T_{base} = 1500s$ for a clustering of 25 zones.

In this optimized scenario, we compare different indicators, such as spatial speeds or total travel time, to better understand the impacts of the hybrid cooperation of zone agents and vehicles on the traffic and on the trips.

Considering the speed as the performance indicator for resilience characterization, we can see in Figure 4.7b that the drop of performance inside the whole network is reduced. The minimum speed is around 5 $m \cdot s^{-1}$ whereas without control it dropped bellow 3 $m \cdot s^{-1}$. The recovery is also shorter as without control, the stable state is around time step $180min$ when it is recovered at $160min$ with control. Considering the quantification of resilience as in Equation 2.1 where the speed is the performance indicator, we have the following:

$$R = \int_{T_0}^{T_r} v_s(t)dt \tag{4.2}$$



Accumulation Spatial Speed

Figure 4.7: Global indicators comparison for Manhattan 5X5

where v_s is the spatial speed, T_0 corresponds to the beginning of perturbation, T_r to the end of the recovery. We consider T_0 and T_r from the initial simulation, without control and compare $R_{noControl}$ with $R_{control}$. From Figure 4.7b, $T_0 = 60min$ and $T_r = 180min$ and the increase of resilience is 21%.

Globally, when comparing the accumulation of vehicles in the whole network during the simulation, we can see in Figure 4.7a that the accumulation of the simulation with control decreases faster than the initial situation, although the peak of accumulation is not reduced. This means that vehicles arrive sooner to their destination. It is confirmed by the global total travel time reduction equal to 13% compared with the simulation without control. The reroutings start around time step 60, right before the congestions of zones like zone a_{22} or zone a_{33} (see Figure 4.9), with a peak of reroutings corresponding to the beginning of actual congestions. This shows the reactivity of the system as actions are produced right when monitored indicators are reaching a critical value. The effect of reroutings

are seen with a certain delay which explains why not all speed of zones remain higher than the critical speed. The rerouting strategy increased the resilience of the network in day-to-day demand perturbations with high reactivity and accurate solutions.

Figure 4.8 depicts the accumulation per zone and the number of reroutings induced by the zone (when vehicles are rerouted when they should have gone to the zone). When the accumulation of a zone is close to its critical points, we can observe that many reroutings are triggered. In some cases, it reduces the accumulation (a_{24}) or maintains the accumulation around the critical value (a_{32}, a_{34}). The central zone, where most of the vehicles intend to go through, can not avoid congestion and higher accumulation than required even though a_{22} triggered a lot of reroutings. Those new assignments are effective later, thus reducing the duration of congestion. This is a limitation due to the fact that vehicles can change their path only with respect the next zone. One could consider zones that are two or three hops away in the macro path to improve the strategy. The zones that were not on the path of many vehicles have absorbed part of the reroutings and we can see that they only accept vehicles if their accumulation is not too high. In zones a_{31}, a_{44}, a_{53}, the reach of critical accumulation induces a small number of reroutings, effective almost immediately.

The consequences on the speed inside the zones is visible in Figure 4.9. We can see that even though congestions were not totally avoided, we managed to reduce the duration of congestion (e.g. a_{22}, a_{44}). Zones that were not congested at all have a reduction of their spatial speed but without congestion. Indeed the spatial speed remains close to the critical speed and barely lower. In zones such as zone a_{23}, zone a_{34} or zone a_{42} the congestion is totally avoided.

From Figure 4.10a, we can observe that the number of longer of trips is reduced as only few vehicles spent more than 60 minutes inside the network. The control strategy has thus reduced the number of long trips and increased the number of shorter trips. In particular, the variance of the trip duration has decreased.

Rerouting can be a constraint for drivers with no actual benefits. From Figure 4.10b, we can see first of all that most of the drivers are not rerouted (75.4%), which means that our solution does not require the cooperation of all the vehicles to be efficient. Furthermore, vehicles that have been rerouted once have reduced their trip duration with a median gain of 5 minutes. The gain for vehicles having been rerouted twice is even higher but concerns fewer people (because of the probability of rerouting acceptance). Some outliers have increased their travel time and in further developments the rerouting acceptance could also depend on the detour induced by the potential new route. Globally, 86% of vehicles have shorter or equal travel trip duration and the mean gain is 13 min whereas the median increase of travel time is 5 min.

To conclude, from all the results shown, the cooperation between zone agents, at an aggregated level, and vehicles, at microscopic level is successful. The rerouting instructions are well transmitted from zones to vehicles and have a rapid effect on the global traffic state. This results to an increased level of performance inside

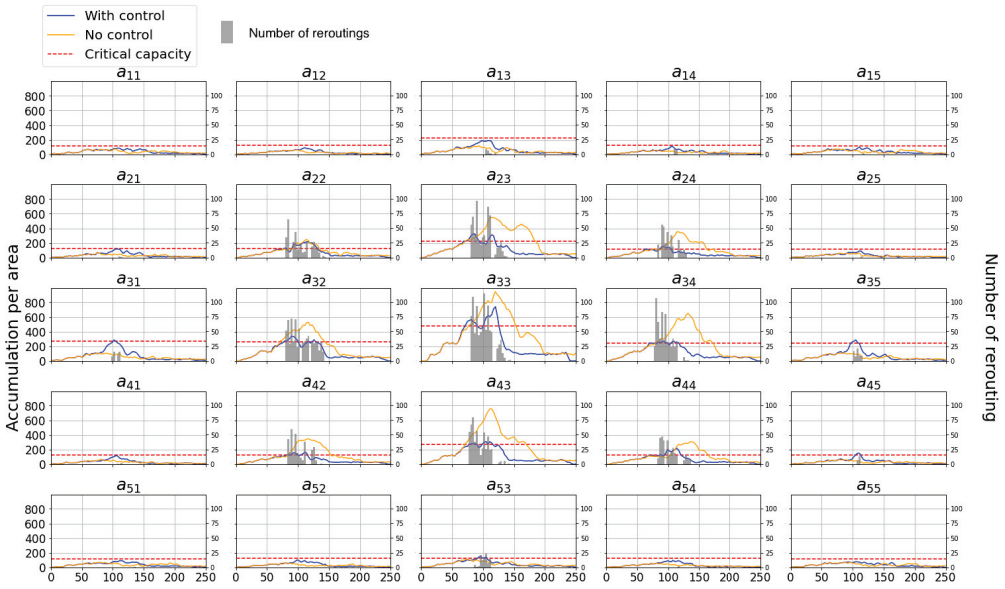


Figure 4.8: Accumulation (vehicles) and reroutings produced per zone per simulation step (min)

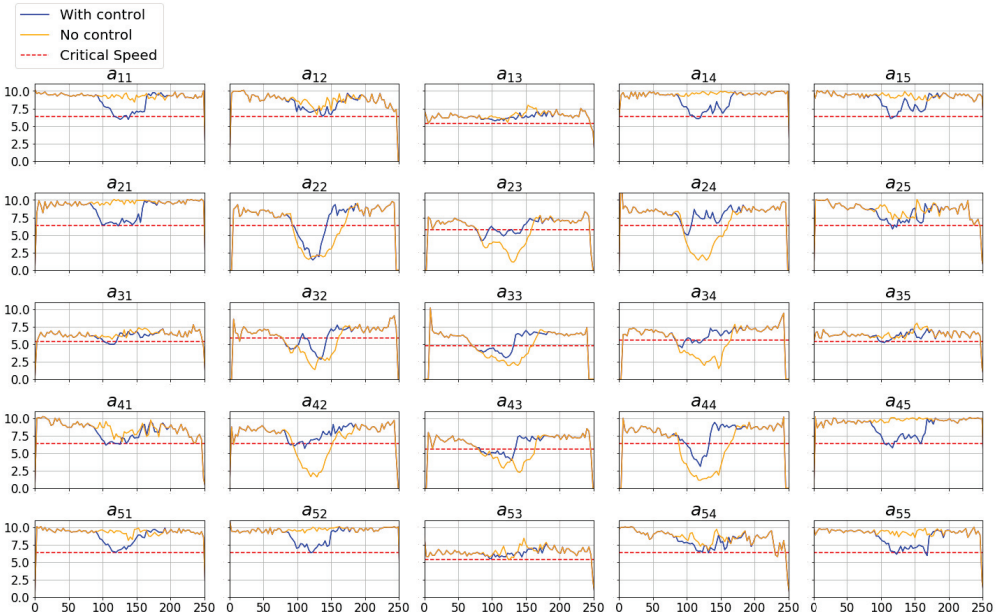
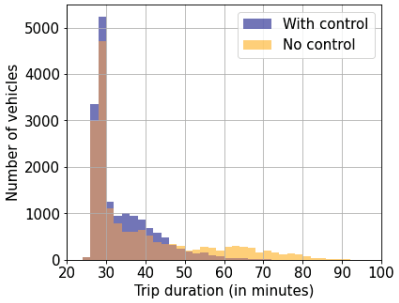
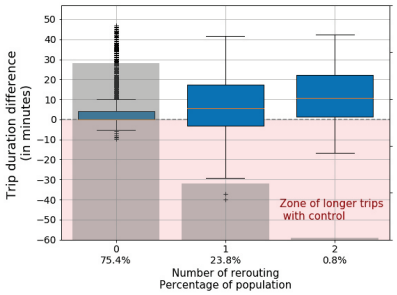


Figure 4.9: Spatial Speed per zone ($m \cdot s^{-1}$) per simulation step (min) with and without control on Manhattan 5X5 network

each zone and in general in the whole network, with the speed higher than critical



Trip duration histogram



Boxplots of trip duration difference per number of rerouting

Figure 4.10: Results on trip duration for Manhattan 5X5

for a longer duration. The system is thus more resilient to demand perturbations.

4.4.3 Robustness Tests on Different Demand Scenarios

Transport networks are usually sized in order to accommodate a recurrent travel demand, which is traditionally known by transport operators with different estimation methods with some uncertainty. Even a regular travel demand can produce relevant performance drops, due to the way travellers distribute over the network over time. In the previous analysis, we focused on such recurrent demand scenario and showed the effectiveness of our approach. However, to study the capability of our solution to sustain the resilience of transport network, we focus in the following on more extreme configurations of the travel demand, that would stress the network and expose its vulnerability without a proper control action. We calibrated our control strategy on a reference case, we now test it with unknown demand scenarios. This allows us to have a global performance evaluation of our model, similarly to a train/test evaluation of machine learning models.

Five different scenarios are proposed: scenario A is the one used for the calibration of the model (reference case in the previous sections), scenarios B, C, D, and E with varying number of vehicles (see Figure 4.11) and peaks of congestion.

The worst case scenario is B, with more vehicles (around 18k) and is close to creation of grid locks, its peak of accumulation is 2.5% higher than that of scenario A. Scenario C corresponds to the case of few vehicles, and thus fewer congestions, with a lower peak of accumulation of -7.9%. Scenario D contains a similar amount of vehicles than scenario A but has a lower peak of accumulation (-2.5%). Scenario E is a scenario with no congestion.

The main results are presented in Table 4.2. For the scenario B, we obtain a reduction of total travel time of 17.9% with a third of rerouted vehicles. Our strategy and the hyper-parameters are thus suited for a high-demand scenario with

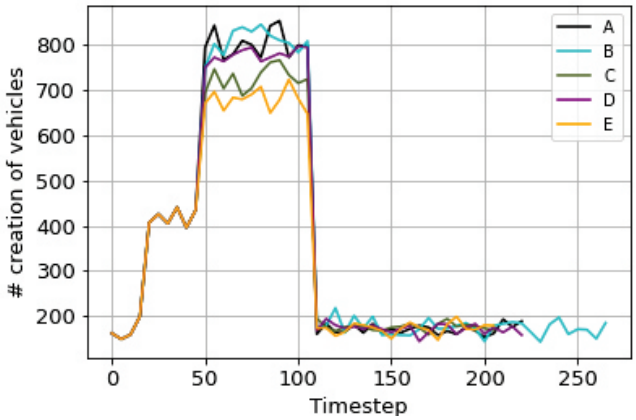


Figure 4.11: Demand scenarios

a bigger disturbance on the global performance of the road network. The effort required to the drivers is bigger than for scenario A but has also more positive consequences as 11% of trips are longer with control and 47% are shorter.

For scenarios with less congestion, the increase of performance is not as good but the performance is still significantly increased for C and D, more than 10% compared with the simulation without rerouting. However, cost of rerouting in the case of demand E is too high (14.6% of rerouted vehicles) compared to the benefits for the system and the users. More globally, the TTT of all the scenarios have decreased, and the more congested the network was and the higher was the reduction of $\%TTT$. The accumulation and spatial speed during the simulation for every scenario are shown in Figure 4.12. In all cases, the lower peak of spatial speed is reduced and the accumulation decreases faster with control.

With the chosen set of hyper-parameters, we obtain a reduction of TTT, keeping the global performance of the system higher without requiring too much changes on driver behaviours. The main problem is however when there is no need for control: if our strategy does not deteriorate the initial situation, the benefits are too low compared with user cost. In conclusion, our strategy performs better when strong demand perturbations occur than when the network is not congested. Our approach is robust and can deal with unknown demand negative perturbations. To face its lack of efficiency for low-demand scenario, the proposed strategy should be activated only once certain conditions of congestion or travel demand increase are met, with respect to a reference scenario (recurrent travel demand).

Table 4.2: Results for other scenarios, with $T_{base} = 1500s$, $\beta = 0.005$

scenario	vehicles	%TTT	%rerouted vehicles	rerouting per vehicles	% longer - shorter trips vehicles	resilience increase
A	16743	13.1 %	24.6 %	0.25	13.3% - 43.5%	21%
B	18475	17.9 %	30.8 %	0.33	11.2% - 46.8%	37%
C	15426	7.8 %	20.7 %	0.21	14.3% - 37.0%	10%
D	16470	9.3 %	27.4 %	0.27	17.0% - 40.6%	14%
E	14803	1.1 %	14.6 %	0.15	18.4% - 25.2%	1%

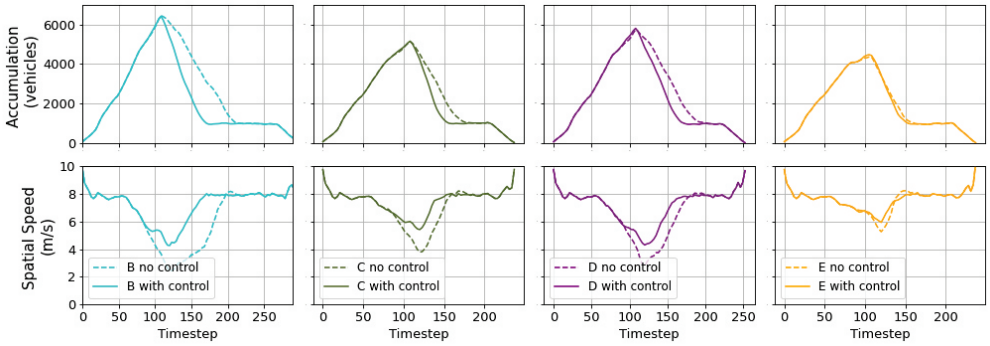


Figure 4.12: Results on other demand scenarios

4.5 Application on a Real Case Scenario

To further validate our approach, we worked with Lyon northern parts as real case scenario. Lyon is the main city of an urban area of more than one million inhabitants in France. This network, illustrated on Figure 4.13, contains 1883 nodes, 3383 links and is divided in 17 zones. The number of trips for the simulation is 68573, between 7 : 30 and 10 : 30, corresponding to the morning peak hours. Origins and destinations are located in the perimeter and inside the network. The demand has been estimated based on real data, collected via surveys and loop detectors. The zones are delimited depending on the existence of a MFD and based on our knowledge of the area. A zone can contain only one critical intersection which is often congested, and only one main arterial. Three zones are composed of a single arterial segment because their alternative paths are close to them and they need to be in another zone to be chosen.

There is a neutral zone, corresponding to a highway where vehicles can go to but which is not an agent and thus does not trigger reroutings.

Similarly to the Manhattan case, we performed a hyper-parameter sensibility analysis on T_{base} and β.

Figure 4.13: Network of Lyon (3rd, 6th district and Villeurbanne). Every zone is represented by a distinct color.

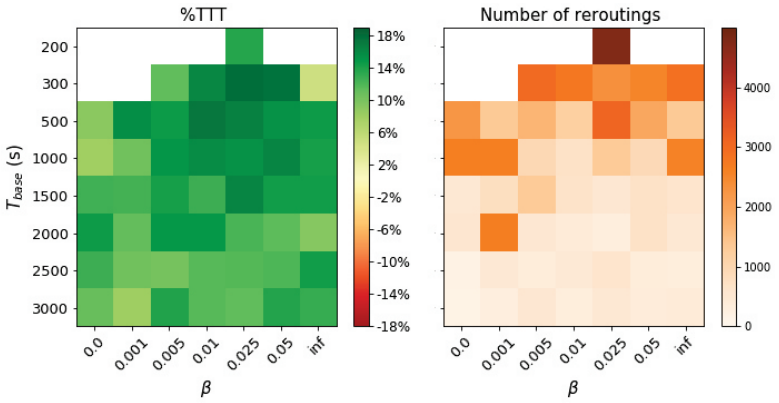


Figure 4.14: Hyper-parameter exploration: reduction of TTT and number of rerouting for Lyon subnetwork

Figure 4.14 shows that for almost all the parameters, when no gridlock appears, the control strategy improves the initial strategy. This can be explained by the fact that we work on real datasets, which does not imply an reached optimum in the initial situation. The initial simulation can thus be easily improved. An optimal area of $\%TTT$ is observed for $T_{base} \in [300s, 1500s]$ and $\beta \in [0.01, 0.05]$. The optimum is reached with $\beta = 0.025$ and $T_{base} \leq 1000$. The maximal $\%TTT$ is equal to 18%.

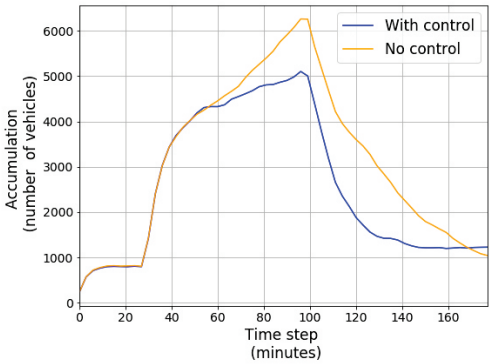


Figure 4.15: Accumulation with $\beta = 0.025$ and $T_{base} = 300s$ inside Lyon network

Very few reroutings are necessary to improve the initial situation. The maximum percentage of rerouted vehicles is 6.7%, corresponding to 4700 rerouted vehicles. This is partially due to the fact that 46% of vehicles do not go through a zone agent and only drive in the neutral zone. Those vehicles can never be rerouted. Moreover, the mean number of zones in the vehicle paths is 4 whereas for Manhattan network split into 25 zones, all vehicles drive through at least 11 zones. The low number of crossed zones reduces the possibility of rerouting for the vehicles.

In Figure 4.16, we can see that for the case of $T_{base} = 300s$ and $\beta = 0.025$, the drop of spatial speed inside the network is reduced and the time of recovery is lower with control. The resilience is thus increased by 22%. This performance improvement is due to the accumulation reduction showed in Figure 4.15, where the peak of accumulation is reduce and decreases much faster with contorl than without.

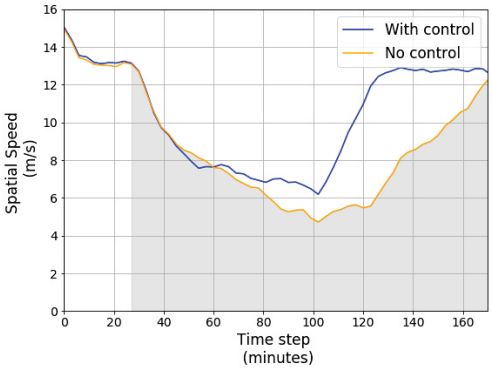


Figure 4.16: Spatial Speed comparison with $\beta = 0.025$ and $T_{base} = 300s$ inside Lyon network

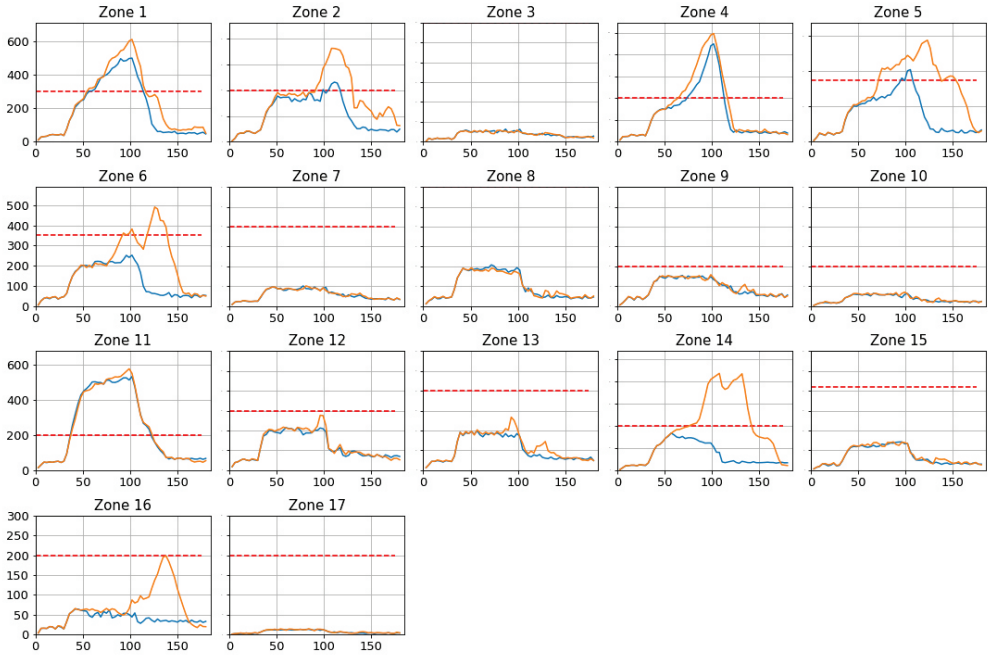


Figure 4.17: Accumulation comparison with $\beta = 0.025$ and $T_{base} = 300s$ inside Lyon network for each zone

Figure 4.17 shows the accumulation per zone and we can notice that 7 zones where highly congested initially. The control strategy enabled the reduction of accumulation for zones 2, 5, 6, 14 bellow the critical capacity. On the contrary, we can observe in the zones 1, 4, and 11 which were highly congested, no real improvement. This is explained by the fact they are on the south border of the network (see Figure 4.13) and the alternative paths are thus more limited for the drivers.

The consequences on the performance of each zone in terms of spatial speed are shown in Figure 4.18. The control strategy reduced the time of recovery for many zones but the zones where accumulation was not reduced have the same drop of spatial speed than without control. The improvements per zone are thus not as obvious than for Manhattan network. This is due to the topological differences: Manhattan network offers more alternative zones for rerouting than Lyon and is more homogeneous. Congestions are more easily avoided and vehicles are more often attracted by new paths than in Lyon network.

These results show that the reroutings were efficient to globally reduce the total travel time spent in the network but the control strategy could not prevent some zones to be congested. Some further developments could help improving these results by changing other hyper-parameters like α that sets the limit of the weight function in $\mathbf{G}$.

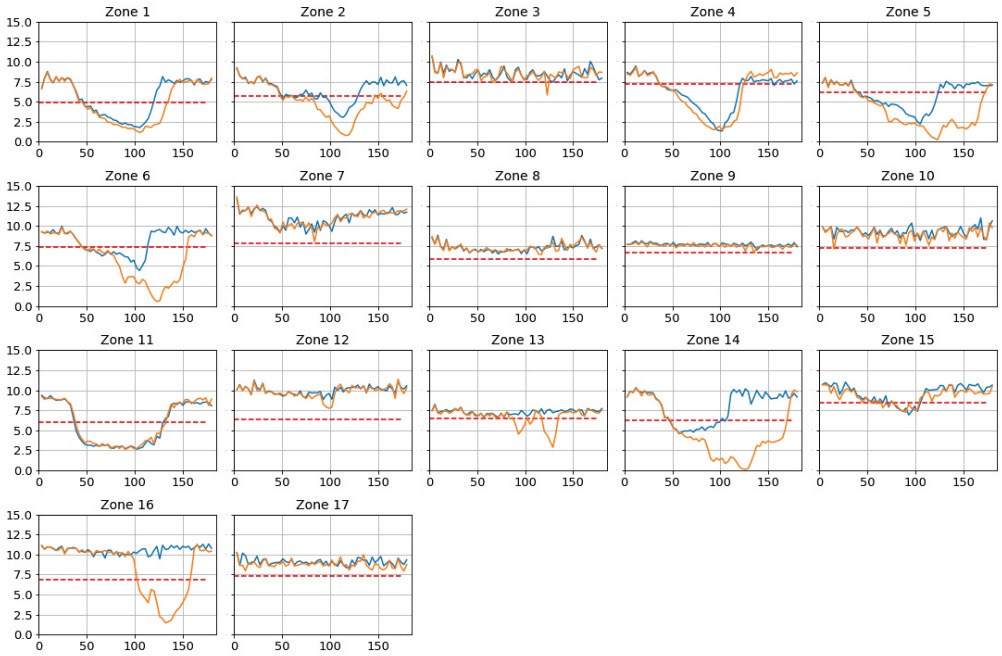


Figure 4.18: Speed comparison with $\beta = 0.025$ and $T_{base} = 300s$ inside Lyon network for each zone

4.6 Preliminary Conclusions

The results showed great robustness and ability of anticipation of our strategy for both real and synthetic networks. We managed to reduce or avoid congestion, especially for high-demand scenario in the synthetic network. This is particularly interesting because a high demand scenario is harder to manage, and the need of flow redistribution is greater. Furthermore, it is easier to detect low-demand scenario and not perform control than the opposite, where the solution is less efficient when it is more needed.

We modeled the acceptance of rerouting as dependant on the number of rerouting. In reality, drivers accept to change their route depending on the new suggested route, which could be added to the acceptance function in further work. Another perspective would be to distinguish the type of drivers depending on their equipment: not all the drivers activate a GPS device or navigation application and it would be interesting to compare the performance of our strategy depending on the proportion of equipped vehicles.

4.7 Resilience Inside the Controlled Area

The results of our approach in a context of simulation show that we were able to maintain a certain level of performance by modifying traffic flow distribution inside the network. By cooperating, zone agents were able to reduce the duration and intensity of congestion. The traffic is better absorbed by the system and the calibrated version of the algorithm showed robustness toward scenarios with travel demand higher than usual. The algorithm thus reduces the road network vulnerability and offers a robust solution in case of exceptional high demand.

Nonetheless, some locations inside the network are more vulnerable than others. Their failures have more consequences or impact more people than that of other parts of the network. Reducing vulnerability inside road network can be done by identifying vulnerable nodes or links and protecting them. Those nodes or links can be identified with resilience metric, the most popular one being the Betweenness Centrality, used in topological analysis of various networks (not only road networks).

4.7.1 Betweenness Centrality as a Metric of Resilience

BC is widely used, with both directed and undirected graphs, to identify opinion leaders or influential people in social network analysis [63], critical intersections in transportation networks [64]–[68], vulnerabilities in computer networks [69], threats from terrorist networks [70]. However, in spite of the great potential, the computation time of BC often represents a barrier to the application of this metric in large-scale contexts, especially with dynamic graphs.

In graph theory, the BC of a node or a link is a topological metric that quantifies the proportion of shortest paths crossing the node or link. The number of shortest paths between a source node s and a destination node t is denoted by $\sigma_{s,t}$, whereas the number of shortest paths between s and t that cross a generic link $l \in \mathbf{E}$ is denoted by $\sigma_{s,t}(l)$.

The *Betweenness Centrality* (BC) of a link $l \in \mathbf{E}$ is defined as follows:

$$BC(l) = \sum_{s \neq, t \in \mathbf{V}} \frac{\sigma_{s,t}(l)}{\sigma_{s,t}} \tag{4.3}$$

A high BC on a link means that a high proportion of shortest paths go through this link, making it a vulnerability for the system.

High BC links tend indeed to have higher probabilities of being chosen for new shortest paths and have thus more chances of being congested. Congestion on high BC links create a higher drop of performance as it impacts more people and more shortest paths. In our case, rerouting vehicles on links with high BC would create a congestion in the new routes which could be worse than the first congestion. Those links need to be protected to prevent propagation of congestion in the network.

The computation time of BC on all the links or nodes of large graph is high due to the high number of explorations across the whole graph needed for shortest path computation. A main focus on the computation of BC is done in the next chapter, in a more global context than that of transport.

4.7.2 BC in the Control Strategy

Links with high BC are likely to attract vehicles as they are on many shortest paths. To maintain robustness inside the network and avoid creating new congestions, we modify the weight function of the graph by adding the BC to protect the vulnerable links. They should remain attractive when no congestion appears but we choose to make them much more unattractive when their zone is congested. This way, we do not prevent vehicles to go through them in a regular situation but start limiting their access in unstable state.

The BC is thus used to help cars finding new shortest paths without creating new congestion in vulnerable locations of the network. As the BC values depend on the size of the graph, we normalized the values with a division by the maximum. This way, all the BC values are comprised between 0 and 1.

The weight function, defined in Equation 3.7 and illustrated in Figure 3.7 in the previous section is a sigmoid, with the parameter $\alpha = 1.5$, defined as follows:

$$tt_w(l) = tt(l) \cdot \left(\alpha - \frac{1}{1 + e^{-\beta \cdot f_i^{tot}(t)}} \right)$$

With $\alpha = 1.5$, the limits of the weights are $0.5 * tt(l)$ and $1.5 * tt(l)$.

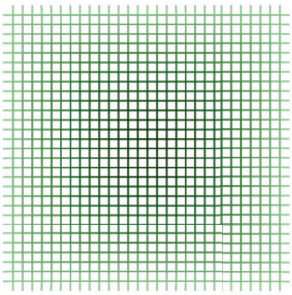
We include the BC normalized, written $\hat{BC}$ in the weighted travel time changing the formula of Equation 3.7 as follows:

$$tt_{BC-w}(l) = tt(l) \cdot \left(\frac{e^{\hat{BC}(l)}}{1 + e^{\beta \cdot f_i^{tot}(t)}} + \alpha' \right) \tag{4.4}$$

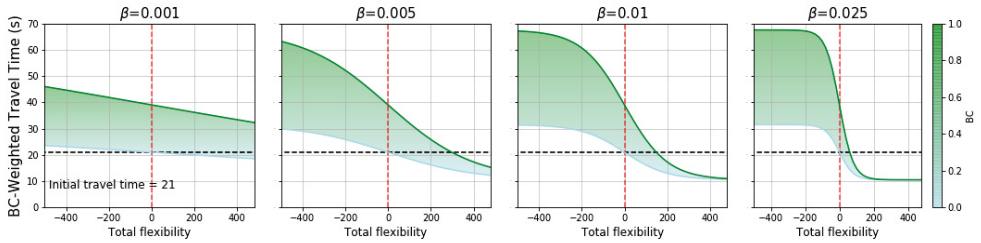
The normalized BC of each link inside Manhattan network is shown as an example in Figure 4.19a. Logically, the highest-BC links are those in the center of the graph.

As a link with a high BC is a vulnerable link if congested, we increase the higher bound to protect it, making it more repulsive than links with lower BC. On the contrary, if the traffic is fluid, the weight will decrease, but reaching the same lower bound for all links.

To maintain a higher bound to $1.5 \cdot tt(l)$, the value of α' is set to 0.5 and the coefficient formula is slightly different from Equation 3.7 in terms of signs. Figure 4.19b illustrates the new weight BC function depending on the total flexibility of the link zone for a set of values of β. The upper curve corresponds to weight function for link with a normalized BC equal to 1 whereas the lower curve is for links with normalized BC equal to 0. This latter case gives the same function than Equation 3.7 (with $\alpha = 1.5$ and $\alpha' = 0.5$). The lower bound is not modified as the



Normalized BC of Manhattan network



Weight function depending on total flexibility and BC for different values of β

Figure 4.19: BC inside Manhattan network and BC-weight function

goal is not to particularly attract vehicles to high BC links but rather to prevent new congestion. For this reason, only the upper bound varies depending on the BC of the links and the flexibility of their zone. Compared to the initial weight function, links with high BC would have higher travel time weights than simulations without considering the BC. Nonetheless, we can notice from Figure 4.19b that with lower values of β, the decrease of the weight is slower and thus for high values of total flexibility we still have higher values of weighted travel time of high BC links.

For computation's sake, we only consider a static topological BC, computed on the graph with free-flow travel time weights.

4.7.3 Use of the BC for Traffic Control (results)

We test the integration of the BC in case of Manhattan 5X5 and demand scenario A. As the BC plays a role in the choice of new shortest paths, the goal is not only to improve the total travel time reduction but to increase the gain of each rerouting. This means that with less reroutings we would be able to still have the same total travel time reduction.

First we perform the simulation on multiple sets of hyper-parameters as in section 4.2. We reduced the number of hyper-parameters based on the results obtained without BC. We thus select higher values of T_{base}, greater than $1000s$

and $\beta \in \{0.001, 0.005, 0.025\}$. We take 0.025 as the maximum because higher values of β in the case of Manhattan 5X5 did not give good results in general. When $\beta = 0$, the weights without considering the BC are constant and equal to the free-flow travel times. We thus do not evaluate the use of BC for this case.

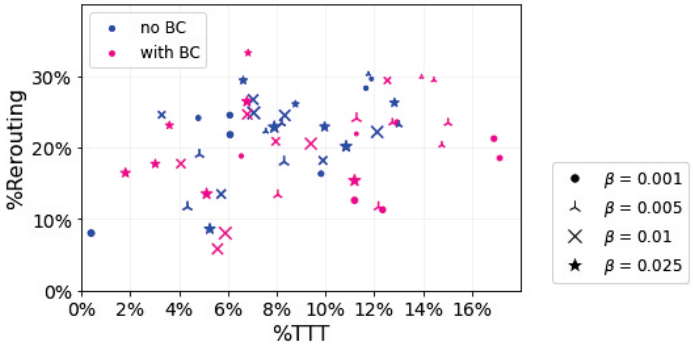


Figure 4.20: Results comparison with and without use of BC for different β and T_{base}

Visualizing the percentage of rerouted vehicles in comparison with the percentage of total travel time, shown in Figure 4.20, we can see that if in general the performances are quite similar but the best results are obtained using the BC. Indeed, only using the BC were we able to obtain a $\%TTT$ reduction greater than 14%, with low values of β keeping a proportion of rerouted vehicles close to 20%. Moreover, for a similar travel time reduction, simulations with the BC require less reroutings (except for some outliers).

This shows that the BC helps finding new shortest paths that have a better impact in terms of resilience when computed taking into acccount the values of BC. We characterize this new route improvement with the *rerouting gain* written Γ_r:

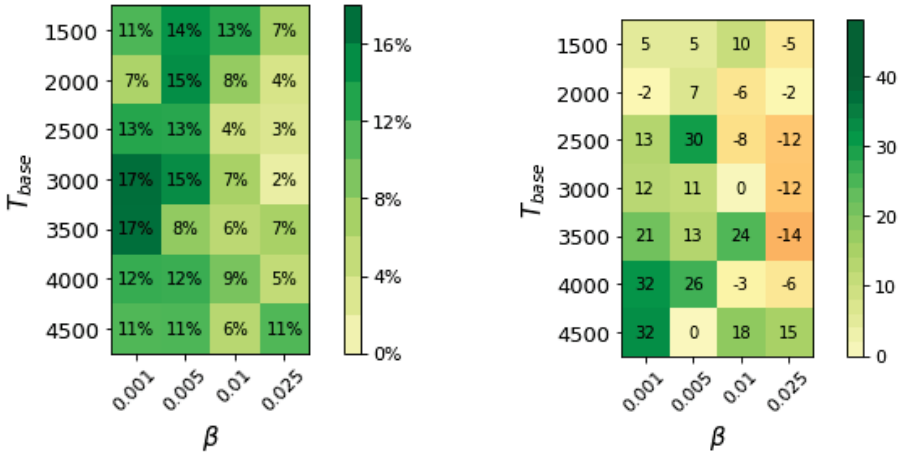
$$\Gamma_r = \frac{TTT_{nocontrol} - TTT_{control}}{number\ of\ reroutings} \tag{4.5}$$

The rerouting gain represents the total travel time saved per rerouting. The higher it is and the more positive the impact generated by one rerouting is. We compare the gain per rerouting of simulations with and without BC by computing the difference, expressed in minutes.

From Figure 4.21, we can see that the best performances in terms of $\%TTT$ and increase of gain per rerouting are observed for low values of β, when the weight function is close to an affine function (Figure 4.19b). As expected, for low values of T_{base}, the improvement is not really high because a lot of reroutings are done with and without BC which reduces the gain per rerouting. The number of reroutings is indeed the main source of $\%TTT$ reduction. What is actually interesting and useful is the improvement observed for high T_{base}: by increasing

the gain per rerouting, it enables good performance for the system with fewer requests to the vehicles.

Too much penalization of high-BC congested links is also not that efficient: with β too high, performances are lower than without considering the BC. The highest increase of performance is found for $\beta = 0.001$ and $T_{base} = \{4000s, 4500s\}$ but the overall $\%TTT$ is around 12% which is low compared to the reduction of TTT of simulations with different values of $\{T_{base}, \beta\}$.



%TTT reduction using the BC

Difference of rerouting Gain $(\Gamma_r(BC) - \Gamma_r(noBC))$

Figure 4.21: Increase of performance via BC adding

In conclusion, using the BC boosted the performance of our control strategy by fine-tuning the new path computation. The new choice of paths increased the travel time gain per rerouting, making the drivers efforts more valuable. Even if the BC was static and thus not dynamically updated with real-time traffic information, the improvements are significant, showing that combining static topological vulnerability analysis and a dynamic system-based approach gives interesting results and should be further studied.

4.8 Conclusion

To conclude this section and the global work on dynamic rerouting, our control strategy performs well after some parameter optimizations. We are able to reduce the duration of congestion and for some zones to totally avoid it without creating new congestion, with a low impact on vehicles. In a fully distributed way, we manage to make zones cooperate and vehicles help them achieve their goals. The hierarchical control is efficient at an aggregated level (zones) and microscopic level (vehicles). The reactivity of both types of agents reduced quickly the drop

of performance measured by the speed inside the whole network. It also reduced the duration of congestion, thus increasing overall resilience inside the network.

The use of BC has enabled a better stability of results, and ensures a minimum of performance. The combination of the topological properties of the road network and the dynamic rerouting solution enabled a higher reduction of vulnerability. The topological understanding of the network is thus useful for dynamic system-based vulnerability analysis while still quite uncommon.

In our case, the BC is computed in a static network, not considering the evolution of traffic conditions. We did not grasp the dynamics of the network into the BC computation. To improve even more the performance of control strategies using the BC, update values just before the rerouting process is a solution. Unfortunately, computing the BC means exploring the whole graph from all the nodes and with the state-of-the-art algorithms, the computation time is high. This explains why we only considered the BC computed before, on a free-flow travel time network.

To enable a real-time computation of BC, we developed an algorithm able to reduce computation time by decomposing the BC and dividing the network into clusters (similarly to zones). This algorithm is further described in the next chapter.

Chapter 5

Cluster-based Algorithm for Fast and Exact Computation of Betweenness Centrality

This chapter is dedicated to a contribution on the fast and exact computation of the Betweenness Centrality published in two papers ([71], [72]). As the BC is a common resilience metric in traffic network, we developed an algorithm able to perform its exact computation in a short amount of time. The BC is not only used in traffic but in many other domains such as social networks, or internet networks, we thus present our work in a more general context than transportation.

Introduction

In recent years, the Floyd method [73], which requires $O(n^3)$ computation time, has been overcome by the well known Brandes' algorithm [74]. Given a graph $G(V, E)$, it exhibits $O(n + m)$ space complexity, $O(nm)$ time complexity for unweighted graphs and $O(nm + n^2 log(n))$ for weighted ones, where $n = |V|$ is the number of nodes and $m = |E|$ the number of edges. However, the polynomial complexity of Brandes' algorithm, which is almost quadratic for very sparse graphs, is still an obstacle for analyzing very large networks. Such problem becomes even more evident and limiting if centrality is used for real-time analysis of dynamic networks.

In the last decade, many researchers have therefore worked with the aim of improving the performance of Brandes' algorithm.

In this chapter, we propose an algorithm based on clustering, inspired by previous work on approximated BC computation [67], [75], which makes possible the exact computation of BC on large, undirected graphs with an impressive speedup when compared to Brandes' algorithm and a significant improvement over recent variants of Brandes' algorithm based on clustering [76].

The algorithm leverages structural properties of graphs to find classes of equivalent nodes: by selecting one representative node for each class, we are able to compute BC by significantly reducing the number of single-source shortest path

explorations required by Brandes' algorithm. We formally prove the graph properties that we exploit to define and implement two versions of the algorithm based on Scala for both sequential and parallel map-reduce executions. The experimental analysis has been conducted by testing both versions of the algorithm on synthetic and real-world graphs. The algorithm we propose is able to work with undirected, weighted or unweighted graphs. In this chapter, we focus on unweighted graphs while its extension to weighted ones can be easily obtained by substituting the breadth-first search (BFS) with Dijkstra algorithm.

Undirected graphs are very common in real-world systems; examples are social networks, communication networks, protein interactions graphs, people interaction graphs, finite element meshes, *etc.* Among these graphs, scale-free and Barabási-Albert graphs [77] represent an important target of our analysis, since they model many real-world systems, such as the World Wide Web, the Internet and other computer networks, citation networks, social networks, airline networks, financial networks, *etc.*

The main contributions of this work are:

- Introduction of the general concept of equivalence class for reducing BC computation time.
- Formal proof about the existence of topological properties to identify an equivalence class with respect to clusters' border nodes in undirected graphs.
- Two variants of Brandes' back propagation technique to avoid the direct computation of the dependency score on cluster nodes due to pivots.
- Scala-based implementations of the proposed algorithm for both sequential and parallel map-reduce executions.
- Extensive evaluation of the proposed algorithm with both synthetic and real large-scale graphs.

The rest of the chapter is organized as follows. Section 5.1 positions our work with reference to the main results from the literature. Section 5.2 introduces the notation we use in the rest of the chapter, as well as the background concepts and algorithms that are at the basis of the proposed solution. Section 5.3 presents the main properties we exploit to define the algorithm when clustering is used to identify classes of equivalent nodes. Section 5.4 illustrates the rationale of the specific implementation of the algorithm and the main steps that characterize it, by presenting also the constituent sub-algorithms exploited for the implementation. Section 5.5 reports on the experimental results obtained by running the proposed algorithm on several synthetic and real graphs characterized by different sizes and topological properties. Section 5.6 is dedicated to the formal proofs of the theorem and claims used in our algorithm for fast BC computation. Finally, Section 5.8 summarizes the results and discusses the limits of the proposed solution by highlighting possible future improvements.

5.1 Related Work

Brandes' algorithm is a fast and robust solution to compute BC, but it is not adequate for real-time processing of large graphs, since BC computation time increases very rapidly with the graph size, even in sparsely-connected configurations.

Several approaches, either exact or approximated, have been developed to reduce BC computation time by improving Brandes' algorithm. The proposed solutions can be classified into five main categories: (a) exploiting and increasing parallelism; (b) updating the BC of specific nodes in dynamically evolving graphs; (c) estimating BC via a partial exploration of the graph in terms of nodes or edges; (d) exploiting structural properties of some kinds of graphs to compress them; (e) reducing complexity by decomposing graphs in clusters. It is worth noting that some proposals may belong to multiple categories since in many cases, techniques falling in a category can be complemented with other ones to further reducing computation time.

Exploiting parallelism. Brandes' algorithm is extremely parallelizable due to the possibility of performing n independent breadth first searches (BFS) or Dijkstra explorations on a shared graph structure. In [78], an efficient parallel implementation of BC computation is provided. The solution leverages fine-grained multi-level parallelism by concurrently traversing the neighbors of a given node via a shared data structure with granular locking in order to increase concurrency. The improved version of the previous approach, proposed in [79], removes the need for locking in the dependency accumulation stage of Brandes' algorithm through the adoption of a successor list instead of a predecessor list for each node. In [80], the authors propose a MPI-based parallel implementation of the adaptive sampling algorithm KADABRA [81]. Other efforts in this direction try to exploit the large amount of cores available on GPUs to better exploit parallelism [82].

Incremental computation. These (stream-based) approaches try to avoid recomputing the BC values of all the nodes of a graph when they are known for a previous configuration, by performing computation over only a small portion of the graph that is impacted by some changes. Recently, an efficient algorithm for streamed BC computation [83] of evolving graphs has been proposed based on edges addition or removal. However, the algorithm is efficient only when the new graph changes in only one edge if compared with the old one. Continuous BC processing of large graphs to handle streamed changes of a significant number of edges is therefore inefficient. *MR-IBC* [84] is a MapReduce-based incremental algorithm for BC computation in large-scale dynamic networks that supports edge addition or removal. The paper exploits distributed computing to achieve scalability and reduce computing time. Even in this case, the focus is on changes related to one single edge. The solution proposed in [85], instead, handles batches of updates in parallel. In particular, it exploits a bi-connected component decomposition technique along with some structural properties to improve performance.

Approximated computation. These algorithms aim at achieving low computation time by calculating approximated BC values. Brandes and Pich proposed in [86] an approximated algorithm for faster BC calculation by choosing only $k \ll n$ nodes, called pivots, as sources for the single-source shortest path (SSSP) algorithm through different strategies, showing that random selection of pivots can achieve accuracy levels comparable to other heuristics. The approach has been further improved by other authors [87]. The goal of these algorithms is to calculate the BC only for selected nodes called pivot nodes. The selection of these nodes depends on the problem to solve and may limit the use of BC.
KADABRA [81] is an adaptive sampling algorithm to approximate betweenness centrality. In particular, it adopts a probabilistic approach: BC of a node v is seen as the probability that, given two randomly selected nodes s and t and a randomly selected shortest path p between them, v belongs to p. The algorithm allows to specify the maximum absolute error and the probability that error is guaranteed.
A similar approach is followed in ABRA [88], a suite of algorithms to compute high-quality approximations of the betweenness centrality of all nodes (or edges) of both static and fully dynamic graphs by using progressive random sampling.

Topology manipulation. Some algorithms exploit topological properties of graphs to accelerate BC computation. Puzis et al. in [89] propose two heuristics to simplify BC computation: (a) identification of structural equivalent nodes, *i.e.,* nodes that have the same centrality index and contribute equally to the centrality of other nodes; (b) partitioning a large graph in smaller bi-connected sub-graphs. Computation time on the graph partitions is significantly lower due to the quadratic to cubic complexity of Brandes' algorithm. The authors also combine the two techniques to improve the speedup when compared with Brandes' algorithm. In [90], the authors use both compression and splitting techniques, including the ones developed in [89], to reduce the size of the input graph and its largest connected component since these are the main parameters that affects the computation time. In particular, they split the input graph by using bridges and articulation vertices and compress it by removing degree-1, identical and side vertices. Bridges and articulation vertices are edges and nodes, respectively, whose removal from a graph leads to a new graph with a greater number of connected components; degree-1 vertices are leaf nodes which, considered as source and targets, contribute equally to the computation of BC of crossed nodes; identical vertices are the ones characterized by the same neighbors and, consequently, by the same BC values; side vertices are nodes such that the graphs induced by their neighbors are cliques and they are not crossed by shortest paths. By using all these techniques, the authors achieve significant speedup with different kinds of graphs. The authors in [91] propose a variant of Brandes' algorithm based on topological characteristics of social networks where nodes belonging to particular tree structures are not considered for Brandes' SSSP explorations; their contribution is simply computed by counting. Topology manipulation and graph

compression are very useful techniques with some types of graphs and are complementary to other solutions from the literature, including the one proposed in this chapter.

Reducing complexity by decomposing graphs in clusters. A way to compute BC is to cluster a large graph into smaller sub-graphs, calculate the BC inside these small graphs, and then compute the BC on the remaining part of the graph. A first paper based on this approach was proposed in [75]. This technique exploits a fast clustering method [92] to identify clusters inside a graph. The border nodes of the clusters are then used as reference nodes to discover, for each cluster, classes of nodes that contribute the same way to the dependency score of the nodes outside the clusters. For each class, a pivot node is selected as representative node for the computation of the dependency scores from the class nodes to the other graph nodes by exploiting the well-known SSSP exploration of the graph. Hence, the dependency score is multiplied by the cardinality of the class the source node belongs to and summed up to the local contribution of BC, computed by considering only nodes belonging to the clusters, to obtain the final approximated values of BC. This technique can be also classified among the ones based on pivots, typically used for computing approximated BC values, even if the strategy adopted to identify pivots is based on clustering. The authors in [76] propose a technique based on clustering to reduce the complexity of BC computation. They prove that with a decomposition of graphs into hierarchical sub networks (HSNs), time complexity can be reduced to $O(n^2)$ for unweighted graphs under the hypotheis that the number of clusters $c \gg k/2$. In that case, the speedup, compared with Brandes' algorithm, is in the order of one half of the graph's average degree k, since the number of edges $m = k \cdot n/2$. This means that if the considered graph has a number of edges $m \sim n$, then $k \sim 2$ and the speedup is 1, that is, the algorithm is not able to improve Brandes' algorithm.

A very similar solution has been proposed in [93]. Differently from [76], the authors propose to build a simplified hierarchical representation of the graph after clustering (named *Skeleton*) by substituting each cluster with a weighted clique connecting the cluster border nodes. This way, they reduce the number of nodes in the Skeleton but need the more computationally expensive Dijkstra algorithm for computing the shortest paths over the weighted graph. Moreover, the proposed solution computes exact BC values of nodes only with respect to a subset of nodes of a graph, named target set. When the target set includes all the nodes of a given graph, the solution converges towards Brandes' algorithm, but with the additional overhead due to the creation and exploitation of the skeleton graph.

Very recently, a Graph Neural Network (GNN) based model to approximate betweenness and closeness centrality has been proposed [94]. This work, among other similar ones [95], demonstrates that the efficient computation of the BC is a topic of great interest even in the field of deep learning and, particularly, graph neural networks.

In this chapter, we propose a technique to reduce the time needed for computing exact values of BC in undirected graphs by: *i)* computing BC as the sum of two main contributions, *local* for each cluster and *global* among clusters (category e), *ii)* reducing the SSSP explorations for the global phase through the identification of pivot nodes (category c), and *ii)* considering HSN-based corrections on local contributions and the properties of undirected graphs to completely remove errors during computation (which affected the first proposal in [75]).

This chapter extends our previous proposal in [71], by significantly improving the algorithm and its implementations that now have been tested also with real graphs. Moreover, we formally prove the correctness of the proposed technique.

It is worth noting that our algorithm could be complemented by algorithms from different aforementioned categories, such as finer-grained parallelism from the first category or compression-based techniques exploiting graphs topological properties as the ones falling within the second category. Conversely, incremental and approximated computations are approaches for specific classes of applications that regard slowly changing graphs or rank-based exploitation of BC, respectively, which we consider out of the scope of this chapter.

5.2 Background

In this section, we first introduce the notation used throughout the chapter, then we briefly describe Brandes' algorithm. Finally, we present the concept of *equivalence class*, which constitutes the basis of our algorithm.

Let $\mathbf{G}(\mathbf{V}, \mathbf{E})$ be an undirected unweighted graph with $\mathbf{V}$ representing the set of n vertices (or nodes) and $\mathbf{E}$ the set of m edges (or links). Let $s, t \in \mathbf{V}$ be two generic nodes of $\mathbf{G}$. We denote by $e_{s,t}$ the edge connecting s and t. The *neighbors* of a vertex s are all vertices u such that $e_{s,u} \in \mathbf{E}$. The *distance* between s and t, denoted by $d_{\mathbf{G}}(s,t)$, is the length of the shortest path(s) connecting them in $\mathbf{G}$. The *number* of shortest paths between s and t is denoted by $\sigma_{s,t}$, whereas the number of shortest paths between s and t that cross a generic node $v \in \mathbf{V}$ is denoted by $\sigma_{s,t}(v)$. It is worth noting that since the graph is undirected, $d_{\mathbf{G}}$ and σ are symmetric functions, thus $d_{\mathbf{G}}(s,t) = d_{\mathbf{G}}(t,s)$, $\sigma_{s,t} = \sigma_{t,s}$ and $\sigma_{s,t}(v) = \sigma_{t,s}(v)$. Given a generic node $w \in \mathbf{V}$, $\mathbf{P}_s(w) = \{u \in \mathbf{V} : e_{u,w} \in \mathbf{E}, d_{\mathbf{G}}(s,w) = d_{\mathbf{G}}(s,u)+1\}$ is the set of direct *predecessors* of vertex w on shortest paths from s.

The *Betweenness Centrality* (BC) of a vertex $v \in \mathbf{V}$ is defined as follows:

$$BC(v) = \sum_{s \neq v \neq t \in \mathbf{V}} \frac{\sigma_{s,t}(v)}{\sigma_{s,t}} \tag{5.1}$$

$BC(v)$ thus represents the fraction of shortest paths containing v among all the shortest paths in the graph between any generic pair of nodes s and t, summed over all possible pairs s and t with $s \neq v$, $s \neq t$ and $v \neq t$.

We refer to Table 5.1 for a summary of the notation used in this chapter.

Notation	Description
$\mathbf{G}$	undirected unweighted input graph
$\hat{\mathbf{G}}$	a connected sub-graph of $\mathbf{G}$
$\mathbf{V}$	set of vertices of $\mathbf{G}$ ($\lvert\mathbf{V}\rvert = n$)
$\mathbf{V}_{\hat{\mathbf{G}}}$	set of vertices of $\mathbf{G}$ inducing $\hat{\mathbf{G}}$ (set of vertices of $\hat{\mathbf{G}}$)
$\overline{\mathbf{V}_{\hat{\mathbf{G}}}}$	set of vertices in $\mathbf{V} \setminus \mathbf{V}_{\hat{\mathbf{G}}}$
$\mathbf{V}_{HSN}$	set of vertices of HSN
$\mathbf{E}$	set of edges of $\mathbf{G}$ ($\lvert\mathbf{E}\rvert = m$)
$e_{s,t}$	edge connecting vertices s and t
$d_{\mathbf{G}}(s,t)$	distance between vertices s and t in $\mathbf{G}$
$\hat{d}_{\mathbf{G}}(s,t)$	normalized distance between vertices s and t in $\mathbf{G}$
$\sigma_{s,t}$	number of shortest paths between vertices s and t
$\sigma_{s,t}(v)$	number of shortest paths between vertices s and t which cross vertex v
$\hat{\sigma}_{s,t}$	normalized number of shortest paths between vertices s and t
$\mathbf{P}_s(v)$	set of direct predecessors of vertex v on shortest paths from vertex s
$\mathbf{P}_s(\mathbf{V})$	set of direct predecessors of vertices in $\mathbf{V}$ on shortest paths from vertex s
$BC(v)$	betweenness centrality of vertex v
$\delta_{s,t}(v)$	pair-dependency of pair of vertices (s,t) on the intermediary vertex v
$\delta_{s,\bullet}(v)$	dependency score of vertex s on vertex v due to all destination vertices
$\delta_{s,\mathbf{V}_{\hat{\mathbf{G}}}}(v)$	dependency score of vertex s on vertex v due to all destination vertices in $\mathbf{V}_{\hat{\mathbf{G}}}$
$\mathbf{C}$	set of clusters of $\mathbf{G}$
$\mathbf{C}_i$	a generic cluster in $\mathbf{C}$
$\mathbf{C}(v)$	the cluster vertex v belongs to
$\mathbf{C}^*$	set of extended clusters in $\mathbf{G}$
$\mathbf{C}_i^*$	a generic extended cluster in $\mathbf{C}^*$
$\mathbf{K}$	set of all the equivalence classes
$\mathbf{K}_i$	an equivalence class
$\mathbf{K}_{\mathbf{C}_i}$	set of equivalence classes of cluster $\mathbf{C}_i$
$\mathbf{P}$	set of all the pivots
k_i	pivot node of the equivalence class $\mathbf{K}_i$
$\mathbf{EN}$	set of all the external nodes
$\mathbf{EN}_{\mathbf{C}_i}$	set of external nodes of cluster $\mathbf{C}_i$
$\mathbf{BN}$	set of all the border nodes
$\mathbf{BN}_{\mathbf{C}_i}$	set of border nodes of cluster $\mathbf{C}_i$
$\mathbf{BN}_{\mathbf{C}_i}(s,t)$	set of border nodes of cluster $\mathbf{C}_i$ on shortest paths from $s \in \mathbf{V}_{\mathbf{C}_i}$ to $t \in \overline{\mathbf{V}_{\mathbf{C}_i}}$
b_i	a generic border node in $\mathbf{BN}$
$\delta^{\gamma}_{s,\bullet}(v)$	global dependency score of s on v due to all $t \in \overline{\mathbf{V}_{\mathbf{C}(s)}}$ (same as $\delta^{\gamma}_{s,\overline{\mathbf{V}_{\mathbf{C}(s)}}}(v)$)
$\delta^{\gamma}_{s,\mathbf{V}_{\mathbf{C}(v)}}(v)$	global dependency score of s on v due to all $t \in (\overline{\mathbf{V}_{\mathbf{C}(s)}} \cap \mathbf{V}_{\mathbf{C}(v)})$

$\delta^{\gamma}_{s,\mathbf{V}_{\mathbf{C}(v)}}(\mathbf{V}_{\mathbf{C}(v)})$	global dependency score of s on vertices in $\mathbf{V}_{\mathbf{C}(v)}$ due to all $t \in (\overline{\mathbf{V}_{\mathbf{C}(s)}} \cap \mathbf{V}_{\mathbf{C}(v)})$
$\delta^{\gamma}(v)$	sum of all the global dependency scores (global BC) on v
$\delta^{\gamma}(\mathbf{V})$	sum of all the global dependency scores (global BC) on vertices in $\mathbf{V}$
$\delta^{\lambda}_{s,\bullet}(v)$	local dependency score of s on v due to all $t \in \mathbf{V}_{\mathbf{C}(s)} = \mathbf{V}_{\mathbf{C}(v)}$
$\delta^{\lambda}_{s,\bullet}(\mathbf{V})$	local dependency score of s on vertices in $\mathbf{V}$ due to all $t \in \mathbf{V}_{\mathbf{C}(s)} = \mathbf{V}_{\mathbf{C}(v)}$
$\delta^{\lambda}(v)$	sum of all the local dependency scores (local BC) on v
$\delta^{\lambda}(\mathbf{V})$	sum of all the local dependency scores (local BC) on vertices in $\mathbf{V}$
$\delta^{\epsilon}_{s,\bullet}(v)$	dependency score of s on v, as external node, due to all $t \in \mathbf{V}_{\mathbf{C}(s)}$
$\delta^{\epsilon}_{s,\bullet}(\mathbf{EN})$	dependency score of s on external nodes $\mathbf{EN}$ due to all $t \in \mathbf{V}_{\mathbf{C}(s)}$
$\delta^{\epsilon}(v)$	sum of all the dependency scores on v as external node
$\delta^{\epsilon}(\mathbf{EN}_{\mathbf{C}(s)})$	sum of all the dependency scores on external nodes of cluster $\mathbf{C}(s)$

Table 5.1: Notation.

5.2.1 Brandes' Algorithm

Brandes' algorithm is the fastest known general-purpose sequential algorithm for computing BC. It is based on the notions of *pair-dependency* and *dependency score*. Let us consider two generic nodes $s, t \in \mathbf{V}$. Given shortest paths counts $\sigma_{s,t}(v)$ and $\sigma_{s,t}$, the pair-dependency $\delta_{s,t}(v)$ of a pair s, t on an intermediary node $v \in \mathbf{V}$ is defined as follows:

$$\delta_{s,t}(v) = \frac{\sigma_{s,t}(v)}{\sigma_{s,t}} \tag{5.2}$$

The pair-dependency represents the fraction of shortest paths between s and t crossing v. The dependency score $\delta_{s,\bullet}(v)$ of a vertex s on a vertex $v \in \mathbf{V}$ is then defined as follows:

$$\delta_{s,\bullet}(v) = \sum_{t \in \mathbf{V}} \delta_{s,t}(v) \tag{5.3}$$

BC can thus be redefined in terms of dependency score:

$$BC(v) = \sum_{s \neq v \neq t \in \mathbf{V}} \frac{\sigma_{s,t}(v)}{\sigma_{s,t}} = \sum_{s \neq v \neq t \in \mathbf{V}} \delta_{s,t}(v) = \sum_{s \in \mathbf{V}} \delta_{s,\bullet}(v) \tag{5.4}$$

The key observation of Brandes' algorithm is that the dependency score obeys a recursive formula. In particular, for each $s \in \mathbf{V}$ we have:

$$\delta_{s,\bullet}(v) = \sum_{w : v \in \mathbf{P}_s(w)} \frac{\sigma_{s,v}}{\sigma_{s,w}} \cdot (1 + \delta_{s,\bullet}(w)) \tag{5.5}$$

Brandes' algorithm runs in two phases, exploiting equation 5.5. For each (source) node $s \in \mathbf{V}$, in the first phase, a single-source shortest-paths (SSSP) algorithm, based on breadth-first search (BFS), is executed on $\mathbf{G}$ to find all the

shortest paths rooted in s. In the second phase, dependency scores are accumulated by backtracking along the discovered shortest paths using the recursive relation in Equation 5.5. In backtracking, nodes are visited in descending order of distance from the source. During these two phases, for each node $v \in \mathbf{V}$ the algorithm builds and exploits the following data structures: the set of direct predecessors $\mathbf{P}_{\boldsymbol{s}}(v)$ on shortest paths from the source, the distance $d_{\mathbf{G}}(s, v)$ from the source, the number of shortest paths $\sigma_{s,v}$ from the source and the dependency score $\delta_{s,\bullet}(v)$ that accumulates the contribution of the source on node v due to all destinations during the back-propagation step.

5.2.2 Equivalence Class

To reduce the number of explorations and thus lower the BC computation time, we exploit the concept of *equivalence class*. Formally, given a connected sub-graph $\hat{\mathbf{G}}$ of $\mathbf{G}$ induced by the set of nodes $\mathbf{V}_{\hat{\mathbf{G}}} \subset \mathbf{V}$, we define an equivalence class $\mathbf{K}_i$ as any subset of nodes in $\mathbf{V}_{\hat{\mathbf{G}}}$ that produce the same dependency score on all nodes - and for destinations - outside sub-graph $\hat{\mathbf{G}}$ when used as sources for SSSP explorations.

By choosing only one representative node (called pivot) for each class, the correct dependency scores of nodes can be computed by multiplying the scores computed via the SSSP rooted in the pivot by the cardinality of the class, *i.e.,* let k_i be a pivot of $\mathbf{K}_i$ and $v \notin \mathbf{V}_{\hat{\mathbf{G}}}$, a node outside sub-graph $\hat{\mathbf{G}}$, we have:

$$\sum_{s \in \mathbf{K}_i} \sum_{t \notin \mathbf{V}_{\hat{\mathbf{G}}}} \delta_{s,t}(v) = |\mathbf{K}_i| \cdot \sum_{t \notin \mathbf{V}_{\hat{\mathbf{G}}}} \delta_{k_i,t}(v)$$

which, according to our notation, can be re-written as:

$$\sum_{s \in \mathbf{K}_i} \delta_{s,\overline{\mathbf{V}_{\hat{\mathbf{G}}}}}(v) = |\mathbf{K}_i| \cdot \delta_{k_i,\overline{\mathbf{V}_{\hat{\mathbf{G}}}}}(v) \tag{5.6}$$

Equation 5.6 clearly shows that a low number of classes significantly reduces the computation time, by allowing to skip a high number of SSSP explorations.

5.3 Clustering and BC Computation

A possible technique to identify equivalence classes is to consider reference nodes. Given a generic sub-graph $\hat{\mathbf{G}}$, the reference nodes in $\mathbf{V}_{\hat{\mathbf{G}}}$ are those that need to be traversed to reach, via shortest paths from nodes in $\mathbf{V}_{\hat{\mathbf{G}}}$, any other node in $\overline{\mathbf{V}_{\hat{G}}}$.

In this chapter, to easily identify reference nodes, we use clustering, and to increase the chances of identifying a low number of equivalence classes, we consider a clustering technique based on modularity, which allows reducing the amount

of connections among groups of nodes belonging to different clusters, and, consequently, lowers the number of reference nodes to be considered for discovering equivalence classes.

The proposed approach relies on a set of mathematical properties that, for the sake of readability, are introduced and used in the following subsections, but proved in Sec. 5.6, at the end of the chapter.

5.3.1 Equivalence Class with Clustering

Let us assume a given graph $\mathbf{G}$ is split into a set of clusters $\mathbf{C}$, where a single cluster $\mathbf{C}_i$ is a connected sub-graph of $\mathbf{G}$ induced by a set of nodes $\mathbf{V}_{\mathbf{C}_i} \subset \mathbf{V}$.

For each cluster $\mathbf{C}_i \in \mathbf{C}$, it is possible to identify a set of *border nodes* $\mathbf{BN}_{\mathbf{C}_i}$. A border node $b_i \in \mathbf{BN}_{\mathbf{C}_i}$ is a node belonging to $\mathbf{C}_i$ and having at least one neighbor belonging to another cluster, as graphically presented in Figure 5.1 (circled nodes are border nodes).

To discover equivalence classes, for each cluster $\mathbf{C}_i$, we group nodes based on their distance and number of shortest paths to the border nodes. To this end, we can leverage the following theorem (see Section 5.6, Theorem 5.6.1, for formal proof).

Let $k \in \mathbb{R}^+$ and $l \in \mathbb{R}$, let $\mathbf{C}_i$ be a generic cluster of graph $\mathbf{G}$ with border nodes $\mathbf{BN}_{\mathbf{C}_i}$, and $s, p \in \mathbf{V}_{\mathbf{C}_i}$. If $\forall\ b_j \in \mathbf{BN}_{\mathbf{C}_i}\ \sigma_{s,b_j} = k \cdot \sigma_{p,b_j}$ and $d_{\mathbf{G}}(s, b_j) = d_{\mathbf{G}}(p, b_j) + l$, then $\delta_{s,\overline{\mathbf{V}_{\mathbf{C}_i}}}(v) = \delta_{p,\overline{\mathbf{V}_{\mathbf{C}_i}}}(v)$, $\forall v \in \overline{\mathbf{V}_{\mathbf{C}_i}}$.

In other words, any given pair of nodes s, p belonging to the sub-graph induced by nodes in cluster $\mathbf{C}_i$ (*i.e.,* $s, p \in \mathbf{V}_{\mathbf{C}_i}$), produces the same dependency score on all nodes $v \in \overline{\mathbf{V}_{\mathbf{C}_i}}$ for destinations $t \in \overline{\mathbf{V}_{\mathbf{C}_i}}$ if the distances and the number of shortest paths from s and p to every border node of $\mathbf{C}_i$ are the same, except for an additive or multiplicative factor, respectively.

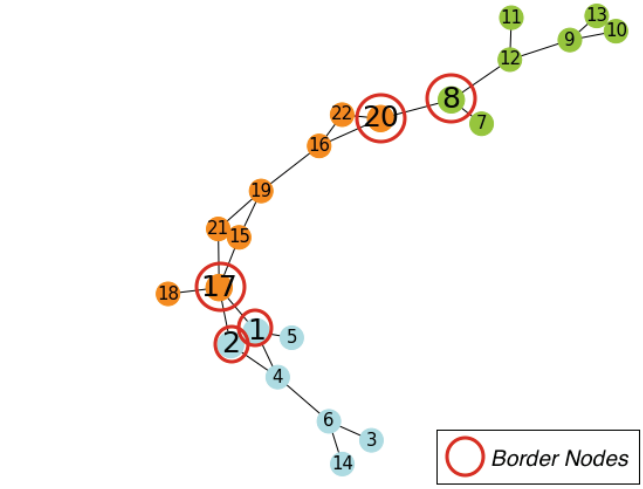
From the previous theorem, we can derive the following corollary (formally proved in Sec. 5.6 as Corollary 5.6.1):
if $\forall\ b_j \in \mathbf{BN}_{\mathbf{C}_i}$, $\hat{\sigma}_{s,b_j} = \hat{\sigma}_{p,b_j}$ *and* $\hat{d_{\mathbf{G}}}(s, b_j) = \hat{d_{\mathbf{G}}}(p, b_j)$, *then* $\delta_{s,\overline{\mathbf{V}_{\mathbf{C}_i}}}(v) = \delta_{p,\overline{\mathbf{V}_{\mathbf{C}_i}}}(v)$, $\forall v \in \overline{\mathbf{V}_{\mathbf{C}_i}}$,
where $\hat{d}_{\mathbf{G}}(s, b_j)$ represents the normalized distance of the generic node s to the generic border node b_j, defined as follows:

$$\hat{d}_{\mathbf{G}}(s, b_j) = d_{\mathbf{G}}(s, b_j) - min_{b_k \in \mathbf{BN}_{\mathbf{C}_i}} d_{\mathbf{G}}(s, b_k)$$

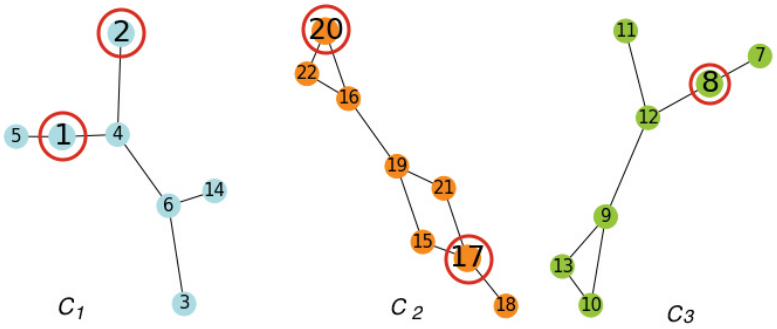
and $\hat{\sigma}_{s,b_j}$ represents the normalized number of shortest paths from the generic node s to the generic border node b_j, and is defined as:

$$\hat{\sigma}_{s,b_j} = \sigma_{s,b_j} / min_{b_k \in \mathbf{BN}_{\mathbf{C}_i}} \sigma_{s,b_k}$$

Normalized distance and normalized shortest paths simplify the identification of classes since they are characterized by the same vector of normalized distances



Example of clustered graph



Three sub-graphs originated by clustering the graph

Figure 5.1: Example of clustering.

and shortest paths as explained below with an example and the support of a graphical representation.

Let **G** be the simple graph reported in Figure 5.1a, decomposed in three clusters, each separately shown in Figure 5.1b. We focus on the blue cluster, referred as $\mathbf{C}_1$, in order to illustrate the concept of equivalence class (see Table 5.2 and Figure 5.2).

In $\mathbf{C}_1$, nodes **1** and **2** are border nodes (*i.e.,* b_1 and b_2 in Table 5.2), while the remaining nodes of $\mathbf{C}_1$ are related to b_1 and b_2 according to the properties detailed in Table 5.2: for each node the normalized distances and normalized number of shortest paths to the border nodes are reported. According to our previous definitions, nodes **3**, **4**, **6**, **14** and **5**, **1** can be grouped in two classes respectively, whereas node **2** is assigned to a singleton class. Nodes **1**, **2** and **14**

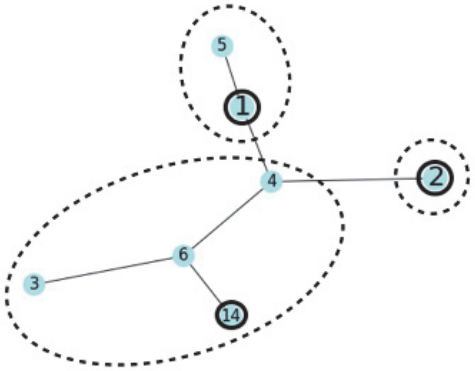


Figure 5.2: Classes of equivalent nodes in the blue cluster C_1

node v	$\hat{d}_{C_1}(v, b_1)$	$\hat{d}_{C_1}(v, b_2)$	$\hat{\sigma}_{vb_1}$	$\hat{\sigma}_{vb_2}$
1	0	2	1	2
2	2	0	2	1
3	0	0	1	1
4	0	0	1	1
5	0	2	1	2
6	0	0	1	1
14	0	0	1	1

Table 5.2: Normalized distances and normalized number of shortest paths for the blue cluster C_1

are the pivots[1].

5.3.2 Cluster-based Exact BC Computation

The equivalence classes allow us to compute the dependency score on nodes - and for destinations - that do not belong to the same cluster of the source node, which means that the contributions computed via this approach are only *partial*.

[1]In our previous version of the algorithm, the pivots were chosen to minimize the error. Here, as we will explain later, any node in a class can be a pivot.

To obtain the total BC, we rewrite Equation 5.4 as follows:

$$
\begin{aligned}
BC(v) &= \sum_{s \in \mathbf{V}} \delta_{s,\bullet}(v) \\
&= \sum_{s \in \mathbf{V}} \sum_{t \in \mathbf{V}_{\mathbf{C}(s)}} \delta_{s,t}(v) + \sum_{s \in \mathbf{V}} \sum_{t \notin \mathbf{V}_{\mathbf{C}(s)}} \delta_{s,t}(v) \\
&= \underbrace{\sum_{s \in \mathbf{V}_{\mathbf{C}(v)}} \sum_{t \in \mathbf{V}_{\mathbf{C}(v)}} \delta_{s,t}(v)}_{\text{sum of local dependency scores} = \delta^{\lambda}(v)} + \underbrace{\sum_{s \in \mathbf{V}} \sum_{t \notin \mathbf{V}_{\mathbf{C}(s)}} \delta_{s,t}(v)}_{\text{sum of global dependency scores} = \delta^{\gamma}(v)} \\
&+ \underbrace{\sum_{s \notin \mathbf{V}_{\mathbf{C}(v)}} \sum_{t \in \mathbf{V}_{\mathbf{C}(s)}} \delta_{s,t}(v)}_{\text{sum of dependency scores on external nodes} = \delta^{\epsilon}(v)}
\end{aligned}
\tag{5.7}
$$

As a result, we can distinguish two main terms, *local* and *global* dependency scores. The additional term is necessary to properly take into account the possible existence of shortest paths connecting nodes of the same cluster via nodes belonging to one or more different clusters, *i.e., external nodes*.

We define the *local* dependency score of a node s on a node v, $\delta^{\lambda}_{s,\bullet}(v)$, as the sum of pair dependency scores for which source s, the destinations and node v belong all to the same cluster. We define the *local BC* of a node v, $\delta^{\lambda}(v)$, as the BC of v computed on the sub-graph $\mathbf{C}(v)$.

Local BC is computed using Brandes' algorithm inside each cluster[2], which generates, as a by-product, additional information (*i.e.,* the number of shortest paths and distances to border nodes). This information is later used to group nodes into equivalence classes and to fasten the computation of global dependency scores, as further discussed (see Subsec. 5.4.2).

The *global* dependency score of a node s on a node v, $\delta^{\gamma}_{s,\bullet}(v)$, is the sum of all the pair dependency scores for which destinations do not belong to the same cluster of source node s. The *global BC* of the generic node v, $\delta^{\gamma}(v)$, is thus the sum of the global dependency scores for source node s ranging over the whole set of nodes $\mathbf{V}$.

The dependency score of a node s on an external node v, *i.e.* $\mathbf{C}(v) \neq \mathbf{C}(s)$, noted as $\delta^{\epsilon}_{s,\bullet}(v)$, is the sum of all the pair dependency scores for which destinations belong to the same cluster of the source node s. We denote by $\delta^{\epsilon}(v)$ the sum of all the dependency scores on v, when v is an external node and the sources and destinations are in the same cluster, different from $\mathbf{C}(v)$.

This last term $\delta^{\epsilon}(v)$ is equal to zero when the clustering is ideal, *i.e.* when all the shortest paths between any pair of nodes of a cluster only contain nodes from that same cluster. When this condition is not fulfilled, multiple side effects

[2]As explained later, in the special case where there are external shortest paths in the cluster, the local BC is actually computed inside the *extended* cluster.

due to the presence of external nodes have to be taken into account, as discussed below.

External Nodes/Shortest Paths

Given a cluster $\mathbf{C}_i$, two nodes $s, t \in \mathbf{C}_i$ and two border nodes $b_1, b_2 \in \mathbf{C}_i$, there may exist shortest paths between s and t which exit $\mathbf{C}_i$ through b_1, cross a certain number of nodes belonging to other clusters and then re-enter $\mathbf{C}_i$ through b_2. We call these shortest paths *external* shortest paths and the nodes lying on them which do not belong to $\mathbf{C}_i$, $\mathbf{EN}_{\mathbf{C}_i}$, *external* nodes of $\mathbf{C}_i$. If the existence of such external shortest paths is neglected, BC computation will be affected by an error due to incorrect values of the distances and the counts of shortest paths between pairs of nodes inside the same cluster. Consequently, an error in the computation of the local BC, δ^{λ}, and in the identification of equivalence classes will be introduced. This was one of the approximation errors affected the previous version of our algorithm [75]. To remove this intra-cluster error, we join the idea proposed by the authors in [76]. After clustering, we build a Hierarchical Sub-Network (HSN), *i.e.,* a sub-graph of $\mathbf{G}$ induced by the border nodes of all the clusters and nodes lying on the intra-cluster shortest paths between pairs of border nodes of the same cluster.

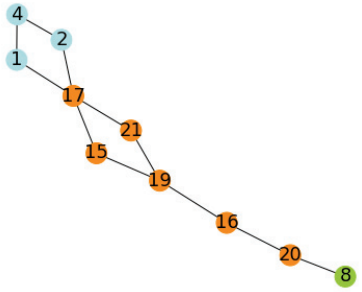
By retrieving all the shortest paths between pairs of border nodes of the same cluster via the HSN, we are able to identify possible external nodes for that cluster. Afterwards, we can extend each cluster with the related external nodes and use the extended clusters as sub-graphs to identify equivalence classes and pivots. Thus, local BC δ^{λ} can be correctly computed inside these extended clusters instead of the initial ones.

Formally, an extended cluster $\mathbf{C}_i^*$ of a cluster $\mathbf{C}_i \in \mathbf{C}$ is defined as a connected sub-graph induced by nodes $\mathbf{V}_{\mathbf{C}_i^*} = \mathbf{V}_{\mathbf{C}_i} \cup \mathbf{EN}_{\mathbf{C}_i}$.

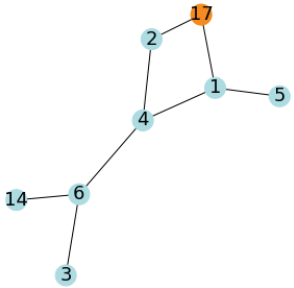
To better understand how the HSN is built and how it is used to form the extended clusters, we provide an illustrative example. Let us consider again the clustered graph from Figure 5.1. In cluster $\mathbf{C}_1$, nodes **1** and **2** are border nodes, while node **4** lies on the only intra-cluster shortest path between them. In cluster $\mathbf{C}_2$, nodes **17** and **20** are border nodes and nodes **15**, **21**, **19** and **16** lie on the intra-cluster shortest paths between them. Finally, in cluster $\mathbf{C}_3$, there is only border node **8**. All the aforementioned nodes build up the HSN (see Figure 5.3a). If we now consider the shortest paths between border nodes **1** and **2** via the HSN, we notice that node **17** lies on a shortest path connecting the two former nodes. Consequently, it represents an external node of $\mathbf{C}_1$ (see Figure 5.3b).

Dependency Score of Pivots

From the equivalence class relationship described in Subsec. 5.2.2, a pivot of such a class is representative only for the dependency scores on nodes v - and destinations t - which do not belong to its own cluster. In fact, given a cluster



Hierarchical Sub-Network from clusters in Figure 5.1b



Cluster $\mathbf{C}_1$, extended with the external node **17**

Figure 5.3: Example of external node found through the HSN.

$\mathbf{C}_i \in \mathbf{C}$ and all its equivalence classes $\mathbf{K}_{\mathbf{C}_i}$, from Equation 5.6, we have:

$$\sum_{s \in \mathbf{K_i}} \delta_{s,\overline{\mathbf{V}_{\mathbf{C}_i}}}(v) = |\mathbf{K}_i| \cdot \delta_{k_i,\overline{\mathbf{V}_{\mathbf{C}_i}}}(v) \quad \forall v \in \overline{V_{\mathbf{C}_i}}, \mathbf{K}_i \in \mathbf{K}_{\mathbf{C}_i}. \tag{5.8}$$

This equation can be exploited to speed up computation of BC building on Brandes' algorithm and SSSP explorations, but only holds if $v \in \overline{\mathbf{V}_{C_i}}$. Thus, it cannot be directly applied to correctly compute values of global BC when v is in the same cluster of the source. Therefore, the algorithm requires a more elaborated approach to properly and efficiently calculate the contribution from the pivot of $\mathbf{K}_{\mathbf{C}_i}$ to the BC of nodes $v \in \mathbf{V}_{C_i}$ [3].

First of all, let us decompose the global dependency scores from Equation 5.7 based on the cluster of node v as follows:

$$\delta^{\gamma}(v) = \sum_{s \notin \mathbf{V}_{\mathbf{C}(v)}} \sum_{t \notin \left(\mathbf{V}_{\mathbf{C}(v)} \cup \mathbf{V}_{\mathbf{C}(s)}\right)} \delta_{s,t}(v) + \sum_{s \notin \mathbf{V}_{\mathbf{C}(v)}} \sum_{t \in \mathbf{V}_{\mathbf{C}(v)}} \delta_{s,t}(v) + \sum_{s \in \mathbf{V}_{\mathbf{C}(v)}} \sum_{t \notin \mathbf{V}_{\mathbf{C}(v)}} \delta_{s,t}(v) \tag{5.9}$$

[3] In our previous version of the algorithm, we used Equation 5.8 without taking into account the cluster of v and the pivots were chosen to minimize the error. Here, we avoid such an error during the computation of global dependency scores by exploiting the properties of undirected graphs, as explained later.

The previous equation can be further simplified by considering the following claim which is proved in Sec. 5.6 as Claim 5.6.1. In undirected graphs:

$$\sum_{s\in\mathbf{V}_{\mathbf{C}(v)}}\sum_{t\notin\mathbf{V}_{\mathbf{C}(v)}}\delta_{s,t}(v)=\sum_{s\notin\mathbf{V}_{\mathbf{C}(v)}}\sum_{t\in\mathbf{V}_{\mathbf{C}(v)}}\delta_{s,t}(v) \tag{5.10}$$

By relying on Equation 5.10, it becomes possible to replace with zero the sum of the pair-dependencies $\delta_{s,t}(v)$ for which $s\in\mathbf{V}_{\mathbf{C}(v)}$ and $t\in\overline{\mathbf{V}_{\mathbf{C}(v)}}$ in Equation 5.9 and compensate later the lack of this term by doubling the sum of the pair-dependencies $\delta_{s,t}(v)$ for which $s\in\overline{\mathbf{V}_{\mathbf{C}(v)}}$ and $t\in\mathbf{V}_{\mathbf{C}(v)}$.

Global dependency scores in Equation 5.9 are therefore redefined as follows:

$$\delta^{\gamma}(v)=\sum_{s\notin\mathbf{V}_{\mathbf{C}(v)}}\sum_{t\notin\left(\mathbf{V}_{\mathbf{C}(v)}\cup\mathbf{V}_{\mathbf{C}(s)}\right)}\delta_{s,t}(v)+2\cdot\sum_{s\notin\mathbf{V}_{\mathbf{C}(v)}}\sum_{t\in\mathbf{V}_{\mathbf{C}(v)}}\delta_{s,t}(v) \tag{5.11}$$

With this further step, we can now use pivots to efficiently compute the exact global BC. In particular, let $\delta^{\gamma}_{s,\mathbf{V}_{\mathbf{C}(v)}}(v)$ and $\delta^{\gamma}_{s,\overline{\mathbf{V}_{\mathbf{C}(v)}}}(v)$ be the global dependency scores from node s on node v for destinations not belonging to $\mathbf{C}(s)$, but belonging to $\mathbf{C}(v)$, and the global dependency score from node s on node v for destinations not belonging to $\mathbf{C}(s)$ and $\mathbf{C}(v)$. Equation 5.11 can be rewritten as follows:

$$\delta^{\gamma}(v)=\sum_{s\notin\mathbf{V}_{\mathbf{C}(v)}}\left[2\cdot\delta^{\gamma}_{s,\mathbf{V}_{\mathbf{C}(v)}}(v)+\delta^{\gamma}_{s,\overline{\mathbf{V}_{\mathbf{C}(v)}}}(v)\right)] \tag{5.12}$$

Therefore, given a cluster $\mathbf{C}_i\in\mathbf{C}$ and all its equivalence classes $\mathbf{K}_{\mathbf{C}_i}$, we have:

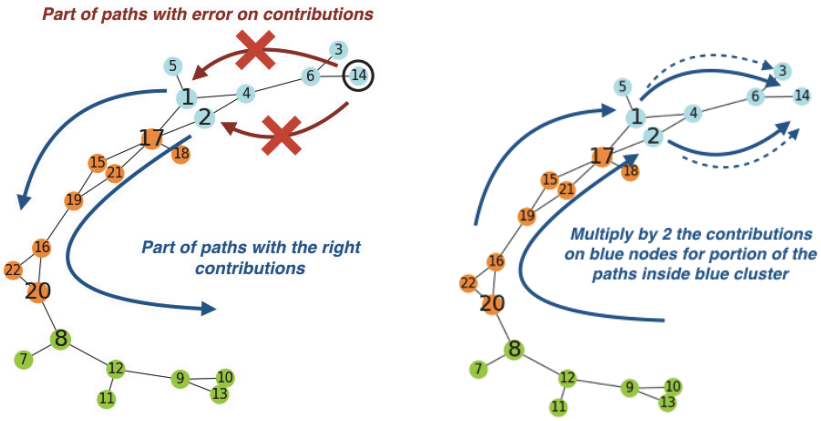
$$\begin{aligned}&\forall v\notin\mathbf{V}_{\mathbf{C}_i},\mathbf{K}_i\in\mathbf{K}_{\mathbf{C}_i},\\&\sum_{s\in\mathbf{K}_i}\left(2\cdot\delta^{\gamma}_{s,\mathbf{V}_{\mathbf{C}(v)}}(v)+\delta^{\gamma}_{s,\overline{\mathbf{V}_{\mathbf{C}(v)}}}(v)\right)=|\mathbf{K}_i|\cdot\left(2\cdot\delta^{\gamma}_{k_i,\mathbf{V}_{\mathbf{C}(v)}}(v)+\delta^{\gamma}_{k_i,\overline{\mathbf{V}_{\mathbf{C}(v)}}}(v)\right)\end{aligned} \tag{5.13}$$

Equation 5.13 means that, during the back propagation phase, we should distinguish between contributions due to destinations inside the same cluster of v and contributions due to destinations outside the cluster of v.

For a better understanding of the formulas above, let us consider an illustrative example by leveraging again the clustered graph from Figure 5.1 and the equivalence classes of cluster $\mathbf{C}_1$ from Figure 5.2.

The pivot node of the equivalence class composed of nodes {**3, 4, 6, 14**} is node **14**. According to the proposed approach, we calculate the dependency scores from node **14** on all nodes of clusters $\mathbf{C}_2$ and $\mathbf{C}_3$ and multiply them by 4, avoiding to calculate the dependencies scores from nodes **3, 4** and **6**. This way, the computation time is divided by 4. However, while it is correct to multiply by 4 the dependency scores for nodes in $\mathbf{C}_2$ and $\mathbf{C}_3$, it is not for nodes belonging to the same cluster of the pivot (see Figure 5.4a) since nodes **14, 3, 4, 6** of the class are equivalent only with reference to border nodes of cluster $\mathbf{C}_1$ (nodes **1**,

2). Therefore, we can not multiply by 4 the dependency scores on nodes **1, 2, 3, 4, 5, 6** since these scores are not the same when computed, for instance, from node **14** or node **4**. To avoid the problem, we put these dependency scores to 0 and we later compensate during SSSP explorations from a pivot node in $\mathbf{C}_2$ and $\mathbf{C}_3$ (see Figure 5.4b).



Problem with nodes belonging to clusters of pivots

Solution for undirected graphs

Figure 5.4: Global SSSP explorations from pivots

Back-propagation. Differently from Brandes' algorithm, it is not possible to directly express the global dependency score[4] of a node v, $\delta^{\gamma}_{s,\bullet}(v)$, in terms of the global dependency scores of w, $\delta^{\gamma}_{s,\bullet}(w)$, where $v \in \mathbf{P}_s(w)$. Indeed, when $\mathbf{C}(v) \neq \mathbf{C}(w)$ (*i.e.,* when crossing a cluster), the set of destinations of w which do not belong to $\mathbf{C}(w)$ can be composed of both destinations belonging to $\mathbf{C}(v)$ and destinations not belonging to $\mathbf{C}(v)$: for the former, the pair-dependencies have to be multiplied by 2, whereas for the latter no further operation is needed (see Equation 5.13).

To overcome this problem, we apply the classic recursive formula of Brandes' algorithm (Equation 5.5) on a *vector* of contributions, propagating the global dependency scores $\delta^{\gamma}_{s,\bullet}(v)$. The dimensions of this vector of contributions correspond to the number of clusters, so that the contribution due to a destination t is assigned to $\delta^{\gamma}_{s,\mathbf{V}_{\mathbf{C}(t)}}(v)$. Formally, we have the following recursive formula:

$$\forall \mathbf{C}_i \in \mathbf{C} \setminus \mathbf{C}(s) : \delta^{\gamma}_{s,\mathbf{V}_{\mathbf{C}_i}}(v) = \sum_{w:v\in\mathbf{P}_s(w)} \frac{\sigma_{s,v}}{\sigma_{s,w}} * (\mathbb{1}_{w\in\mathbf{C}_i} + \delta^{\gamma}_{s,\mathbf{V}_{\mathbf{C}_i}}(w)), \qquad (5.14)$$

where $\mathbb{1}_{w\in\mathbf{C}_i}$ represents a boolean variable equal to 1 if $w \in \mathbf{C}_i$, 0 otherwise[5]. At the end of the back-propagation phase, we put the dependency scores of nodes

[4] $\delta^{\gamma}_{s,\bullet}(v)$ is equivalent to $\delta_{s,\overline{\mathbf{V}_{\mathbf{C}(s)}}}(v)$

[5] This is the part of contribution due to w as a destination

v belonging to the same cluster of the (pivot) source node to 0, whereas the dependency scores of nodes belonging to the other clusters are computed using the following formula:

$$\delta^{\gamma}_{s,\overline{\mathbf{V}_{\mathbf{C}(s)}}}(v) = 2 \cdot \delta^{\gamma}_{s,\mathbf{V}_{\mathbf{C}(v)}}(v) + \sum_{\mathbf{C}_i \neq \mathbf{C}(v)} \delta^{\gamma}_{s,\mathbf{V}_{\mathbf{C}_i}}(v) \tag{5.15}$$

Finally, according to Equation 5.13, $\delta^{\gamma}_{s,\bullet}(v)$ is multiplied by the cardinality of the equivalence class s belongs to.

5.4 *E1C-FastBC Algorithm*

In this section, we describe the *E1C-FastBC* algorithm, the implementation of the cluster-based solution introduced in the previous section. We also discuss a parallel version based on MapReduce.

5.4.1 Louvain Clustering

To group nodes in clusters and minimize the number of border nodes, $|\mathbf{BN}|$, and consequently $|\mathbf{BN}_{\mathbf{C}_i}|$ for each cluster $\mathbf{C}_i \in \mathbf{C}$, we exploit a modularity-based clustering algorithm. *Modularity* is a scalar metric, defined in the range -1 and 1, which measures the density of links inside clusters as compared to links between them: the higher its value, the lower the number of inter-clusters links. Consequently, maximizing the modularity score reduces the number of border nodes in the clusters. This allows not only to keep low the complexity of the algorithm by reducing the size of the HSN and the number of nodes against which topological properties (normalized distances and normalized number of shortest paths) have to be computed, but also to maximise the chances of having few equivalence classes, each with many nodes, since smaller vectors (those storing the topological properties) increase the probability of having linear dependency among them and consequently a smaller number of classes. This is highly beneficial from the perspective of reducing SSSP explorations.

The Louvain method [92] is an example of modularity-based clustering technique. Its time complexity of $O(nlog^2n)$ is very good compared to that of Brandes' algorithm. The Louvain algorithm runs in two phases which are iteratively repeated. In the first phase, each node is initially assigned to its own cluster and moved in the cluster of the neighbor which ensures the maximum increase of modularity, with respect to the previous configuration. This phase terminates when all nodes have been explored and no further modularity improvement is possible. In the second phase, a new graph is generated by considering the identified clusters as nodes, and the loops inside them as self-loops. Phase one is then repeated using the graph generated by the second phase. The two phases are iterated until a maximum of modularity is reached and a hierarchical structure of clusters has been formed. The output of the algorithm, and consequently the modularity of

the identified clusters, may be affected by the order nodes are evaluated with. This order can also influence the computation time. To improve solutions that are sub-optimal in terms of modularity, multiple runs of the algorithm can be performed over the same network, each associated to a different order for the analysis of the nodes.

5.4.2 Algorithm Implementation

Algorithm 2 reports the pseudo-code of the *E1C-FastBC* algorithm taking as input an undirected unweighted graph $\mathbf{G}$ and producing in output the exact values of BC for every node in $\mathbf{V}$. The algorithm is composed of several phases. We provide a detailed description for all the intermediate phases, while the associated pseudo-code is provided for the most relevant ones in Algorithm 2.

Algorithm 2 Pseudo-code of the E1C-FastBC algorithm

```
1: function E1CFastBC(G)
2:     C ← modularityBasedClustering(G)
3:     BN ← findBorderNodes(G, C)
4:     V_HSN ← buildHSN(BN, C, G)
5:     EN ← findExternalNodes(V_HSN, C, BN, G)
6:     C* ← updateClusters(C, EN)
7:     δ^λ, δ^ϵ, σ̂, d̂ ← computeLocalδ(C*, C, EN, BN, G)
8:     K ← findClasses(σ̂, d̂)
9:     δ^γ ← computeGlobalδ(K, G, C)
10:    for v ← 1, V do                                   ▷ For all nodes
11:        BC(v) ← δ^λ(v) + δ^γ(v) + δ^ϵ(v)
12:    end for
13:    return BC
14: end function
```

At line 2, the Louvain, modularity-based clustering algorithm is exploited for splitting graph $\mathbf{G}$ into a set of clusters $\mathbf{C}$ (see Subsection 5.4.1). These clusters do not need to be explicitly stored in a dedicated data structure as they represent a view of the starting graph, filtered through a membership information stored in every node.

At line 3, we identify the set of border nodes $\mathbf{BN}$ by checking, for each node $v \in \mathbf{V}$, the existence of at least one neighbor belonging to a different cluster.

At line 4, the nodes building up the HSN, referred as $\mathbf{V}_{HSN}$, are retrieved. As detailed in the pseudo-code of Algorithm 3, to build the HSN we first execute $|\mathbf{BN}|$ *local* BFS, each rooted in a border node used as source, *i.e.,* $s \in \mathbf{BN}$. The term local here refers to the fact that only nodes belonging to the same cluster of s, *i.e.,* $\mathbf{V}_{\mathbf{C}(s)}$, are crossed during the explorations. Each BFS returns the set of direct predecessors $\mathbf{P}_s(\mathbf{V}_{\mathbf{C}(s)})$ of every node in $\mathbf{V}_{\mathbf{C}(s)}$ on shortest paths from s.

These sets are later used at line 6 of Algorithm 3 to cross the discovered shortest paths backwards starting from destinations t.

Each traversal returns the set of nodes lying on the shortest paths between a pair of border nodes of the same cluster: these nodes, together with the source and destination border nodes themselves, belong to the HSN and are therefore added to the set of all its nodes, *i.e.,* $\mathbf{V}_{HSN}$.

Algorithm 3 Pseudo-code of building HSN algorithm

```
1: function buildHSN(BN, C, G)
2:     for s ← BN do
3:         P_s(V_C(s)) ← BFS(s, G, V_C(s))
4:     end for
5:     for s, t ← BN where C(s) == C(t) do
6:         V_HSN ← V_HSN ⋃ crossSPBackwards(t, P_s(V_C(s)))
7:     end for
8:     return V_HSN
9: end function
```

At line 5, we identify external nodes **EN** as detailed in Algorithm 4. First, a BFS is executed from each source $s \in \mathbf{BN}$. In these explorations, only nodes of the HSN, $\mathbf{V}_{HSN}$, are considered. Each BFS returns the set of direct predecessors of every node in $\mathbf{V}_{HSN}$ on shortest paths from s, *i.e.,* $\mathbf{P}_s(\mathbf{V}_{HSN})$. Similarly to the previous step, shortest paths are crossed backwards from destination t to source s using the sets of predecessors and every crossed node not belonging to the cluster of s and t is added to the set of external node **EN**.

Algorithm 4 Pseudo-code of external nodes identification algorithm

```
1: function findExternalNodes(V_HSN, C, BN, G)
2:     for s ← BN do
3:         P_s(V_HSN) ← BFS(s, G, V_HSN)
4:     end for
5:     for s, t ← BN where C(s) == C(t) do
6:         EN ← EN ⋃ crossSPBackwards(t, P_s(V_HSN))
7:     end for
8:     return EN
9: end function
```

The extended clusters $\mathbf{C}^*$ are generated at line 6 of Algorithm 2 from the original clusters, updated with the external nodes.

At line 7, we compute in each (extended) cluster, *i)* the local BC on every node, *i.e.,* δ^{λ}, *ii)* BC contributions δ^{ϵ} on external nodes, and *iii)* the topological properties of every node, *i.e.,* the normalized distances $\hat{d}$ and the normalized numbers of shortest paths $\hat{\sigma}$, computed with respect to the set of border nodes

belonging to the cluster of the node. These topological properties are subsequently used at line 8 to find equivalence classes (see Subsec. 5.3.1). A modified version of Brandes' algorithm enables the computation of all these metrics, as described in Algorithm 5.

The only difference compared to the canonical implementation of Brandes' algorithm is related to the back-propagation phase: in our case, the contributions due to the external nodes (as destinations) are not propagated, since they represent non-local destinations.

Algorithm 5 Pseudo-code of local BC computation

1: **function** COMPUTELOCAL$\delta(\mathbf{C}^*, \mathbf{C}, \mathbf{EN}, \mathbf{BN}, \mathbf{G})$
2: **for** $s \leftarrow \mathbf{V}$ **do**
3: $\delta^{\lambda}_{s\bullet}(\mathbf{V}_{\mathbf{C}(s)}), \delta^{\epsilon}_{s\bullet}(\mathbf{EN}_{\mathbf{C}(s)}), \hat{\sigma}_{s,\mathbf{BN}_{\mathbf{C}(s)}}, \hat{d}_{\mathbf{G}}(s, \mathbf{BN}_{\mathbf{C}(s)})) \leftarrow BrandesModifiedV1($
$s, \mathbf{C}^*(s), \mathbf{C}(s), \mathbf{BN}, \mathbf{G})$
4: $\delta^{\lambda}(\mathbf{V}_{\mathbf{C}(s)}) \leftarrow \delta^{\lambda}(\mathbf{V}_{\mathbf{C}(s)}) + \delta^{\lambda}_{s\bullet}(\mathbf{V}_{\mathbf{C}(s)})$
5: $\delta^{\epsilon}(\mathbf{EN}_{\mathbf{C}(s)}) \leftarrow \delta^{\epsilon}(\mathbf{EN}_{\mathbf{C}(s)}) + \delta^{\epsilon}_{s\bullet}(\mathbf{EN}_{\mathbf{C}(s)})$
6: **end for**
7: **return** $\delta^{\lambda}, \delta^{\epsilon}, \hat{\sigma}, \hat{d}$
8: **end function**

At line 9, we compute the global BC, δ^{γ}. As shown in Algorithm 6, a second modified version of Brandes' algorithm, which exploits Equation 5.13, Equation 5.14 and Equation 5.15, is run from one pivot k_i, randomly selected from each class $K_i \in \mathbf{K}$. From the explorations rooted at the class pivots, the global dependency scores, $\delta^{\gamma}_{k_i,\overline{\mathbf{V}_{\mathbf{C}(k_i)}}}(\mathbf{V})$ are computed on every other node v of the graph. The global BC of every node v is then obtained by summing the global dependency scores deriving from all the pivots.

Finally, at lines 10-12 of Algorithm 2, all the previously computed partial terms are aggregated via a sum operation to obtain the exact BC values for each node.

Algorithm 6 Pseudo-code of global BC computation

1: **function** COMPUTEGLOBAL$\delta(\mathbf{K}, \mathbf{G}, \mathbf{C})$
2: **for** $K_i \leftarrow \mathbf{K}$ **do** ▷ For all classes
3: $k_i \leftarrow random(K_i)$
4: $\delta^{\gamma}_{k_i,\overline{\mathbf{V}_{\mathbf{C}(k_i)}}}(\mathbf{V}) \leftarrow BrandesModifiedV2(k_i, \mathbf{G}, \mathbf{C})$
5: $\delta^{\gamma}(\overline{\mathbf{V}_{\mathbf{C}(k_i)}}) \leftarrow \delta^{\gamma}(\overline{\mathbf{V}_{\mathbf{C}(k_i)}}) + \delta^{\gamma}_{k_i,\overline{\mathbf{V}_{\mathbf{C}(k_i)}}}(\overline{\mathbf{V}_{\mathbf{C}(k_i)}}) \cdot |K_i|$
6: **end for**
7: **return** δ^{γ}
8: **end function**

5.4.3 Parallel Implementation with Map/Reduce

The proposed *E1C-FastBC* algorithm can be parallelized since the execution of its sub-algorithms is highly parallelizable, with the only exception of the selected clustering algorithm. We can exploit *data parallelism* by performing the same operations on different partitions of a given graph leveraging the MapReduce paradigm since most of the computations are applied to each node of the graph.

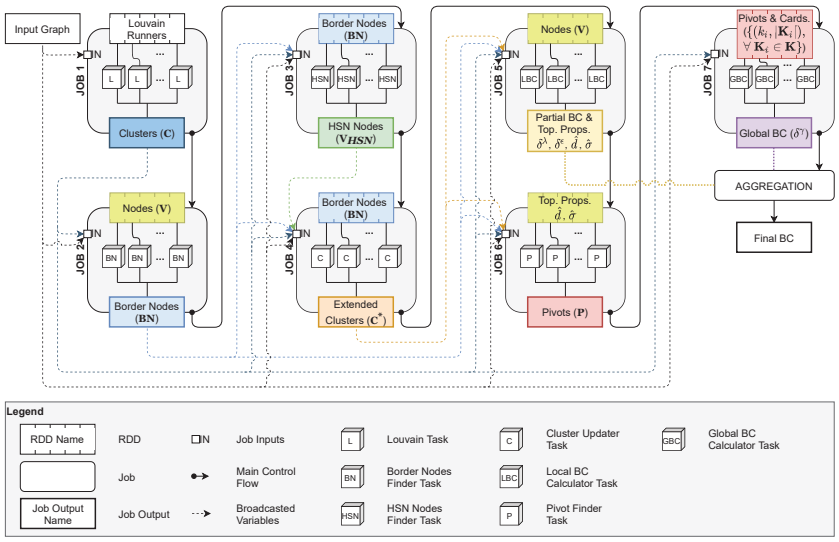


Figure 5.5: Map-reduce, Spark-based description of E1C-FastBC.

The main execution flow is represented by solid arrows, while dashed ones represent broadcast variables. Jobs and tasks are depicted as rounded rectangles and cubes, respectively. RDDs are shown as non-rounded rectangles.

Figure 5.5 reports the representation of the parallel version of *E1C-FastBC*, built using some key concepts introduced in Apache Spark, a popular big data processing engine that we use to run the tests reported in the experimental evaluation. Spark applications are generally defined in terms of *transformations* and *actions* that are applied to *Resilient Distributed Datasets* (RDDs). RDDs are immutable collections of data partitioned across the worker nodes of a cluster. Transformations and actions can be processed in parallel on such partitions. In particular, transformations are functions that produce new RDDs starting from the ones they are invoked on, whereas actions are functions that return the result of a computation performed over an RDD.

A Spark application is a collection of *jobs*, each created to perform an action, and executed by one or more *executors* deployed on the worker nodes of the cluster by running, in parallel, *tasks* over the partitions of an RDD. The tasks of the jobs encapsulate all the transformations that have to be applied to RDDs. The latter are then collected at the end of the jobs by the master node of a cluster: such node hosts the *driver*, which is the process responsible for running the application.

To process RDDs, jobs may also require other inputs that can possibly be shared across executors through the so-called *broadcast variables*: they are lazily copied between worker nodes during execution.

The parallel version of *E1C-FastBC* is a sequence of jobs, each implementing one or more sub-algorithms of the algorithm, as detailed in Figure 5.5. The main execution flow is represented with solid arrows, whereas, with dashed arrows, we present data that are copied via broadcast among all Spark workers and needed to carry out the jobs. Each job executes a specific type of task, as illustrated in Figure 5.5, and receives two classes of inputs: *i)* RDDs, which are used to guide parallelism, *i.e.,* the number of tasks, and *ii)* broadcast variables, which are used by every single task to process its own partition. In the following, we describe each job in terms tasks behaviors and needed inputs.

- *Job 1* organizes the graph into clusters (Algorithm 2, line 2) by performing parallel executions of the Louvain method, using different configurations with the aim of selecting the one that produces the clustering with the best modularity score.

  The job takes as input the graph, passed as broadcast variable, and outputs the clusters. The starting RDD, which does not contain data, only enables parallel executions of multiple runs of the Louvain method.

- *Job 2* identifies border nodes (Algorithm 2, line 3), by checking for each node the existence of at least one neighbor belonging to a different cluster. It requires as input the set of clusters and the graph, both passed as broadcast variables. The starting RDD contains all the nodes to analyze, and it is built from the whole set of graph vertices

- *Job 3* retrieves the HSN nodes (Algorithm 2, line 4), by performing, for each border node, a constrained (intra-cluster) BFS. It needs the border nodes, the clusters and the graph as its inputs. A broadcast variable is used for all of them, but the set of border nodes is also used to build the starting RDD. Nevertheless, each execution requires the availability of the whole set of border nodes *i)* to avoid leaving clusters while performing BFSs and *ii)* to check whether a destination is a border node.

- *Job 4* discovers the external nodes (Algorithm 2, lines 5-6) through BFSs bound to nodes belonging to the HSN. Consequently, compared to *Job 3*, it requires HSN nodes as additional input, passed in the form of broadcast variable, while the starting RDD is the same as that of *Job 3*. At the end, the job outputs the clusters extended with external nodes. .

- *Job 5* computes the local BC, the BC on external nodes, the normalized distances and normalized numbers of shortest paths (Algorithm 2, line 7). The job receives the graph, clusters, the extended clusters and the border

nodes as inputs, all transferred as broadcast variables[6]. The starting RDD of this job contains all the nodes of the graph.

- *Job 6* identifies the equivalence classes and their pivots (Algorithm 2, line 8). The starting RDD contains the topological properties (normalized distances and normalized number of shortest paths) per node, while the inputs passed as broadcast variables are the same as the previous job.
- *Job 7* computes the global BC (Algorithm 2, line 9) by using a starting RDD containing pairs composed of a pivot and the cardinality of its equivalence class. The only inputs passed via broadcast variables are the graph and the clusters.

Final BC values are obtained by aggregating all the previously calculated values. This step is performed entirely on the driver in a sequential manner. In all cases, except for *Job 1*, we use a node-level grain: all functions encapsulated in the various tasks are defined to work starting from a single node (simple node, border node or pivot).

Figure 5.6 reports a detailed description of Job 2 to exemplify how a job is performed. Solid arrows represent elaboration phases, while the dashed ones represent data transfer phases. The dotted box shows the set of transformations applied by the tasks hosted on the executors over the different partitions. The first dashed box reports the source code related to the job, whereas the second reports the source code of the first map task in the pipeline. The job is triggered by the *collect* action. The driver builds the initial RDD by executing the *parallelize* method (1). The number of partitions is equal to the number of executors, *i.e.,* each executor works on a single partition. Partitions are sent to executors along with the operations to be performed on them (2). These operations are encapsulated in a task. In particular, the first *map* operation (3a) generates an intermediate RDD of key-value pairs where the second element is the unique node identifier and the first element is a boolean value (true/false) that depends on whether the node is a border node or not. Then, the *filter* operation creates a new intermediate RDD containing only the pairs with the key equal to true (3b). Finally, the second *map* operation produces an RDD by extracting only the second element of the previous pairs (3c). Hence, the border nodes are collected on the driver (4) and stored in a set (5).

5.5 Experimental Evaluation

In this section, we report the main results of the experimental evaluation we conducted by testing the algorithm with both sequential and parallel executions on different types of graphs, and with different graphs and sizes.

[6]We do not explicitly pass the set of external nodes as it is trivial to recognize them by leveraging the extended clusters.

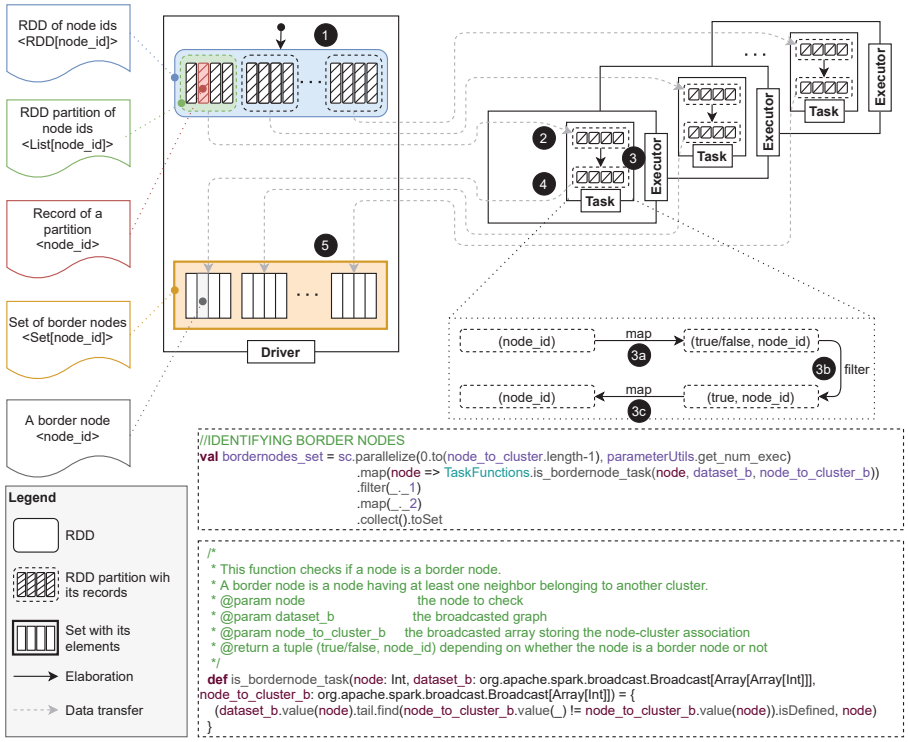


Figure 5.6: Map-reduce, detailed description of *Job2*

Includes elaboration and data transfer phases (solid and dashed arrows), the set of transformations applied by the tasks (dotted box) and the related source code (dashed boxes).

We compare the execution times obtained with our algorithm to those obtained with other algorithms by using the *Algorithmic Speedup* (AS). Given two algorithms, a_1 and a_2, the algorithmic speedup of a_1 over a_2 with p cores, noted as $AS_p^{a_1/a_2}$, is defined as $T_p^{a_2}/T_p^{a_1}$, where $T_p^{a_2}$ and $T_p^{a_1}$ are the computation times obtained with p cores using algorithms a_2 and a_1, respectively. Hence, the larger the value of AS, the faster a_1 is compared to a_2 with the same computing resources. For example, $AS = 2$ means that the time taken by a_1 is half the time taken by a_2 and therefore that a_1 is two times faster than a_2.

In particular, we will compare the *E1C-FastBC* algorithm, labelled with $\mathcal{E}$, with Brandes' algorithm, labelled as $\mathcal{B}$ and with the solution proposed in [76], labelled with $\mathcal{H}$. We chose this algorithm for comparison because it belongs to the same category as ours (cluster-based computation) and it addresses the problem of exact BC computation.

However, due to the unavailability of source/executable code for $\mathcal{H}$, we only consider the AS metric in sequential mode, by relying on the indications provided by the authors in this chapter for its computation (see Equation 7 in [76]).

To further explore the performance of our solution, we also analyze the efficiency of the *E1C-FastBC* algorithm, based on the canonical definition of speedup. Specifically, the speedup obtained with p cores is defined as $S_p = T_s/T_p$, where T_s is the computation time in sequential mode and T_p is the computation time with p cores. The efficiency with p cores, noted as E_p, is then defined as S_p/p.

Efficiency E_p may influence the AS metric as demonstrated by the following analysis. From the definition of speedup we can write that $T_p^{a_2} = T_s^{a_2}/S_p^{a_2}$, where $T_s^{a_2}$ and $S_p^{a_2}$ are the execution time in sequential mode and the speedup obtained with p cores of algorithm a_2. Similarly, we can write that $T_p^{a_1} = T_s^{a_1}/S_p^{a_1}$. By using these equations in the definition of the AS metric, we have that $AS_p^{a_1/a_2} = (T_s^{a_2}/T_s^{a_1}) \cdot (S_p^{a_1}/S_p^{a_2})$. Since efficiency, for a given number of cores p, is $S_p^{a_1}/S_p^{a_2} = E_p^{a_1}/E_p^{a_2}$, we have: $AS_p^{a_1/a_2} = (T_s^{a_2}/T_s^{a_1}) \cdot (E_p^{a_1}/E_p^{a_2})$.

The relationship between AS and E_p suggests that if a_1 and a_2 have comparable efficiency with p cores, then the AS only depends on the ratio of the execution times of the two algorithms in sequential mode, thus providing interesting insights to comparatively analyze the two different solutions.

Finally, we also provide a breakdown analysis of the computation time of our solution, useful to investigate the contribution of each composing sub-algorithms. In all reported tests, we checked the accuracy of our solution by always observing zero error on BC values.

5.5.1 Datasets

In our tests, we consider both synthetic and real graphs.

For the first category, we focus on scale-free graphs generated using the implementation of the Barabási-Albert model provided by the Python library NetworkX. According to that model, a graph of n nodes is grown by attaching new nodes, one at a time, each with m' edges that are preferentially attached to existing nodes with high degree. In our case, m', which is called the preferential attachment coefficient, is equal to 1. This way, we have graphs with $m = n - 1$ edges and an average degree approximately equal to 2, *i.e.,* double the preferential attachment coefficient. This choice is motivated by the features of the current implementation of our algorithm that benefits of high modularity. In other words, this class of dataset is considered as best-case scenario. However, as mentioned in the introduction, this does not limit the applicability of our solution because many real-world systems can be represented with the Barabási-Albert model. In particular, to analyze the algorithm in terms of performance and scalability, we generate graphs with different sizes (see Table 5.3).

For the second category, we focus on some real graphs[8] available in public datasets. Table 5.3 reports all the graphs we use, together with some relevant

[7]This dataset was supplied by the French National Institute of Geographic Information (IGN). https://www.ign.fr

[8]For each graph, we extract the largest connected component. Then, the latter is converted in an unweighted undirected graph.

	Graph	n	m	d_{avg}	d_{max}	cc_{avg}
Synthetic	barabási-albert	6,250	6,249	1.999	126	0.000
	barabási-albert	12,500	12,499	”	225	”
	barabási-albert	25,000	24,999	”	344	”
	barabási-albert	50,000	49,999	”	463	”
	barabási-albert	100,000	99,999	”	1,138	”
	barabási-albert	200,000	199,999	”	676	”
	barabási-albert	400,000	399,999	”	1,142	”
	barabási-albert	800,000	799,999	”	1,587	”
Real	web-webbase-2001[96]	16,062	25,593	3.187	1,679	0.224
	ego-twitter[97]	22,322	31,823	2.851	238	0.072
	internet[96]	124,651	193,620	3.107	151	0.062
	lyon-road-network[7]	156,102	178,845	2.291	8	0.017
	email-euAll[98]	224,832	339,925	3.024	7,636	0.079

Table 5.3: Topological information of synthetic & real graphs.

The names of the graphs are given in the first column, whereas the number of nodes and edges are given in the second and third columns. d_{avg} and d_{max} are the average and max degree, respectively. cc_{avg} is the average clustering coefficient.

properties. In particular, for each graph we consider the average degree (d_{avg}), the max degree (d_{max}) and the average clustering coefficient (cc_{avg}).

All the datasets, except the one related to the Lyon road network, are scale-free graphs.

5.5.2 Experimentation Testbed

The platform for our experiments is a server machine with 128 GB of RAM and 2 sockets Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40GHz, with 14 physical cores and 2 threads per core for a total of 28 logical cores per socket and 56 virtual cores in hyper threading, running Linux Debian as operating system.

Both Brandes' algorithm and *E1C-FastBC* are implemented in Scala and executed using Apache Spark 2.2.0 in standalone mode. In particular, we deploy a spark cluster composed of the master node and one worker node holding all available resources (56 cores and approximately 125GB of memory). Tests are performed employing a driver with one core and 30GB of memory, and a variable number of executors having a variable amount of memory and computing resources. Specifically, except for the case with one core (sequential execution) where there is only one executor that holds all the resources (*i.e.,* the single core and 90GB of memory), we fix the total number of cores for the experimentation and instantiate a number of executors such that each of them has 5 cores.

The amount of memory is divided evenly among executors. For instance, with 5 cores we only deploy one executor with 90GB of memory, while with 10 cores two executors, each with 45GB of memory, are deployed.

The RDDs are decomposed in a number of partitions equal to the total number of cores.

5.5.3 Synthetic Graphs Analysis

Figure 5.7 shows the algorithmic speedup of *E1C-FastBC* over Brandes' algorithm, $AS_p^{\mathcal{E}/\mathcal{B}}$, obtained on the synthetic graphs in both sequential and parallel modes. In particular, we double the number of nodes from 25,000 to 800,000, and we consider a number of cores p equals to 1, 5, 10, 15, 20 and 25. We estimate by log-log regression the computation times with Brandes' algorithm for the graphs with 400,000 and 800,000 nodes, since executions would require weeks to complete, whereas our algorithm ends in maximum 31.5 minutes and 1.64 hours, respectively (sequential mode).

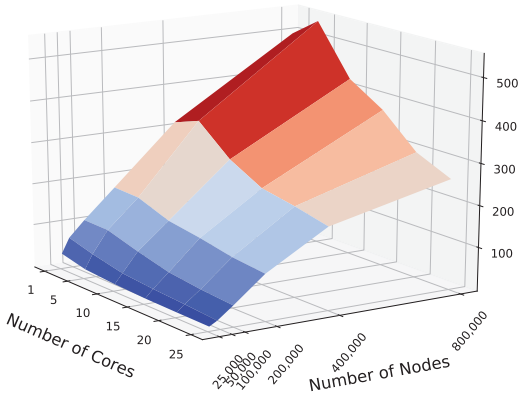


Figure 5.7: Comparison with Brandes' algorithm. Algorithmic speedup analysis - $AS_{p=[1,5,10,15,20,25]}^{\mathcal{E}/\mathcal{B}}$

As highlighted by Figure 5.7, $AS_p^{\mathcal{E}/\mathcal{B}}$ increases with the size of the graph, meaning that *E1C-FastBC* is not only faster than Brandes' algorithm but its speedup grows with larger graphs. This is due to the fact that the computation of our algorithm is strongly dependent on the number of border nodes ($|\mathbf{BN}|$), pivots ($|\mathbf{K}|$) and external nodes ($|\mathbf{EN}|$), in addition to the number of nodes (n) and edges (m). The first two variables increase slowly compared to the number

of nodes and edges, while the third is almost always zero (only in one case it was equal to 2). The drawback is that $AS_p^{\mathcal{E}/\mathcal{B}}$ decreases as the number of cores increases. This behaviour is due to the fact that the Brandes' algorithm is more efficient than *E1C-FastBC* (see Figure 5.8). This means that the ratio $E_p^{a_1}/E_p^{a_2}$ in the relationship between the AS metric and the efficiency is lower than 1. Consequently, the AS value in the sequential case is not preserved as the number of cores increases. However, as the following efficiency analysis will further clarify, this does not mean that *E1C-FastBC* is less scalable than Brandes' algorithm, but rather that it needs very large graphs to better exploit the available computing resources. This statement is also confirmed by Figure 5.7, which clearly shows that when the graph size is 400,000, a higher number of cores performs even better than a smaller one: in particular, we have that the $AS_p^{\mathcal{E}/\mathcal{B}}$ is better with 5 cores than with 1 core. To have a similar behavior even for a number of cores greater than 5, we should consider larger graphs.

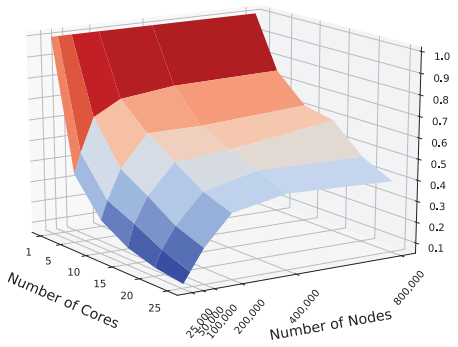
To better understand the performance of *E1C-FastBC*, we investigate its efficiency with respect to that of Brandes' algorithm. Figure 5.8 reports the results of the efficiency analysis performed for the two algorithms. In both cases, it is possible to observe that: *i)* the efficiency decreases as the number of cores increases and *ii)* for a given number of cores, it increases as the number of nodes increases.

However, it is worth to highlight that in the efficiency analysis, we use different but overlapping ranges of values for the number of nodes. In particular, for our solution we select larger graphs since we aim at showing that our algorithm scale well especially with very large graphs. In fact, the efficiency trend is almost the same in the two cases reported in Figure 5.8. Moreover, given the maximum values of the number of nodes for the two algorithms (800,000 for ours, 200,000 for Brandes'), efficiency values are approximately the same with 5 cores (*i.e.,* the first considered parallel configuration) but significantly diverge as the number of cores increases. In particular, efficiency of *E1C-FastBC* decreases with a higher rate.

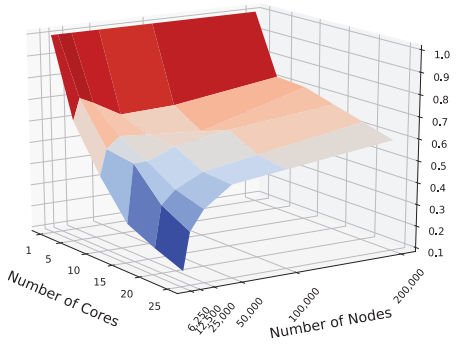
The reason for this behaviour lies in the reduced amount of computation required by our solution. Indeed, pivots allow to significantly decrease the number of (modified) Brandes' SSSP explorations performed on the whole graph, which represent the heaviest part of the whole computation (see Figs. 5.13 and 5.14), thus reducing the workload of each core.

Our solution also introduces another benefit: it allows to mitigate the variability of the computation times due to the different topological characteristics of the graphs and to the partitioning of data performed by Spark during executions. Indeed, there may exist some partitions of the RDDs characterized by a high concentration of nodes that generate the most complex shortest path trees.

The time required to process these partitions directly impacts the time required to process the whole RDD, since partitions are processed in parallel. However, Spark tasks process each single partition sequentially. This aspect, combined



E1C-FastBC algorithm



Brandes' algorithm

Figure 5.8: Comparison with Brandes' algorithm. Efficiency analysis - $E_{p=[1,5,10,15,20,25]}$

with the fact that the number of partitions of an RDD is always equal to the number of cores and the default partitioning scheme of Spark distributes data evenly across the partitions, explains the punctual efficiency drops that can be observed in the plot related to Brandes' algorithm, when using graphs with 50,000 and 100,000 nodes and a low number of cores (see Figure 5.8b).

Figure 5.9 reports the algorithmic speedup of *E1C-FastBC* over Brandes' algorithm, $AS^{\mathcal{E}/\mathcal{B}}_{p=1}$, alongside with the algorithmic speedup of the approach in [76] over Brandes, $AS^{\mathcal{H}/\mathcal{B}}_{p=1}$, on synthetic graphs and in sequential settings. $AS^{\mathcal{H}/\mathcal{B}}_{p=1}$ is

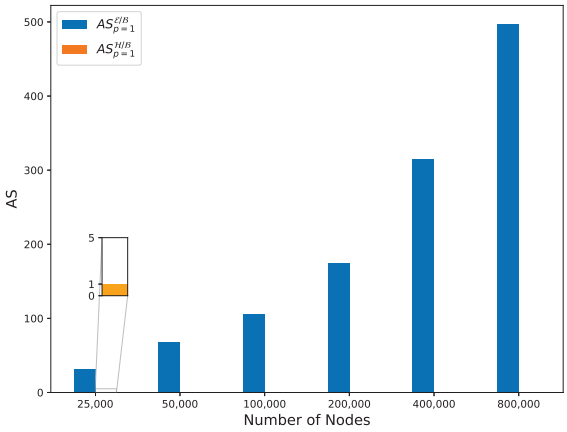


Figure 5.9: Comparison with the solution proposed in [76]. Algorithmic speedup analysis with synthetic graphs in sequential mode - $AS_{p=1}^{\mathcal{E}/\mathcal{B}}$ and $AS_{p=1}^{\mathcal{H}/\mathcal{B}}$. $AS_{p=1}^{\mathcal{H}/\mathcal{B}}$ is always equal to 1 for synthetic graphs (see Subsection 5.5.3). We therefore add only one single zoomed bar in the figure just to make it visible to the reader.

analytically computed based on Equation 7 provided in [76].

Using such equation, it is possible to observe that: *i)* $AS_{p=1}^{\mathcal{H}/\mathcal{B}}$ depends on the number of clusters ($|\mathbf{C}|$) and the average degree (d_{avg}), and *ii)* when $|\mathbf{C}| + 2 \gg d_{avg}/2$, it can be approximated with $d_{avg}/2$. Therefore, since for synthetic graphs the average degree is constant and the number of clusters increases with the number of nodes, $AS_{p=1}^{\mathcal{H}/\mathcal{B}}$ is always approximately equal to 1 (the average degree is 2). In particular, the higher the number of clusters, the closer to 1 the $AS_{p=1}^{\mathcal{H}/\mathcal{B}}$. This means that the algorithm proposed in [76] is not able to improve that of Brandes. Conversely, ours is able to do it by a large multiplicative factor. We can thus conclude that our solution always outperforms the one in [76] with synthetic graphs.

5.5.4 Real Graphs Analysis

Figure 5.10 reports the results of the analysis of $AS_{p=1}^{\mathcal{E}/\mathcal{B}}$ and $AS_{p=1}^{\mathcal{H}/\mathcal{B}}$ carried out on real graphs. $AS_{p=1}^{\mathcal{H}/\mathcal{B}}$ is computed again using Equation 7 provided in [76]. In all cases, our solution outperforms the one in [76].

To further confirm the considerations on the scalability of our solution, reported in the previous section, we analyze in the following both the algorithmic speedup and efficiency values of *E1C-FastBC* on the *lyon-road-network* graph, for which we observed a very high number of pivots (about 60% of the number

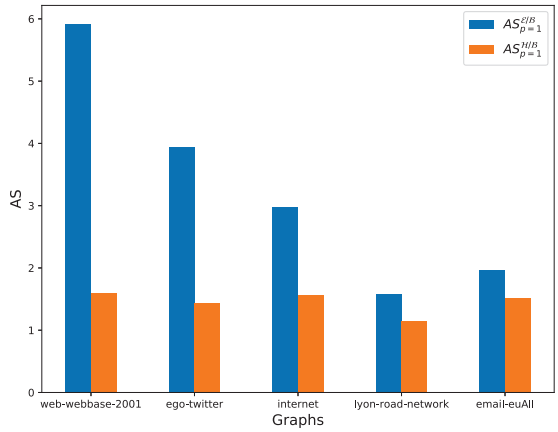


Figure 5.10: Comparison with the solution proposed in [76]. Algorithmic speedup analysis with real graphs in sequential mode - $AS^{\mathcal{E}/\mathcal{B}}_{p=1}$ and $AS^{\mathcal{H}/\mathcal{B}}_{p=1}$

of nodes) and the lowest algorithmic speedup factor. As shown in Figure 5.11, the *AS* is always greater than 1, thus confirming the usefulness of our solution, although the reported values are not comparable to those obtained on synthetic graphs (see Figure 5.7) with a similar number of nodes (100,000 and 200,000). In spite of this, the algorithm becomes more scalable and efficient than in the case of synthetic graphs with 100,000 and 200,000 nodes due to the increased amount of computation resulting from the higher number of border nodes, pivots and external nodes (see Figure 5.12). Also in this case, by considering the two figures (Figure 5.11 and Figure 5.12), it is possible to note the dependency relationship between the *AS* metric and the efficiency. In particular, going from 15 to 20 cores the difference between efficiency values for the Brandes' algorithm and of *E1C-FastBC* decreases while *AS* increases.

5.5.5 Breakdown of Computation Time

In this section, we analyze the contributions of the different component sub-algorithms to the overall computation time of E1C-FastBC. The goal of this analysis is to find bottlenecks that limit scalability and, consequently, room for further improvements.

We split the algorithm in seven parts (see Algorithm 2): *Dataset Reading*, *Louvain Clustering* (line 2), *HSN & External Nodes* (lines 3-6), *Local BC & Top. Props.* (line 7), *Classes & Pivots* (line 8), *Global BC* (line 9) and *Final Accumulation* (lines 10-12). Figures 5.13 and 5.14 report the time taken by each of the

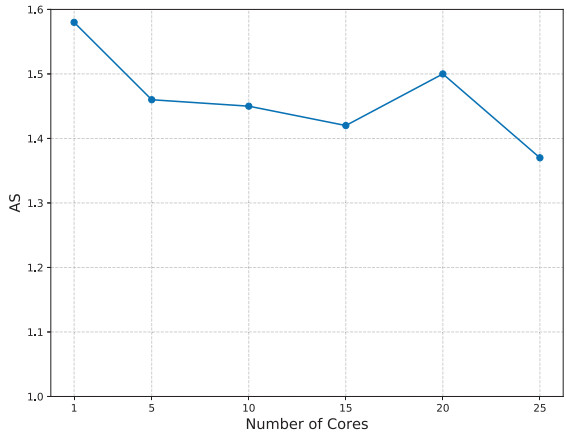


Figure 5.11: Algorithmic speedup analysis for lyon-road-network graph in parallel mode - $AS^{\mathcal{E}/\mathcal{B}}_{p=[1,5,10,15,20,25]}$

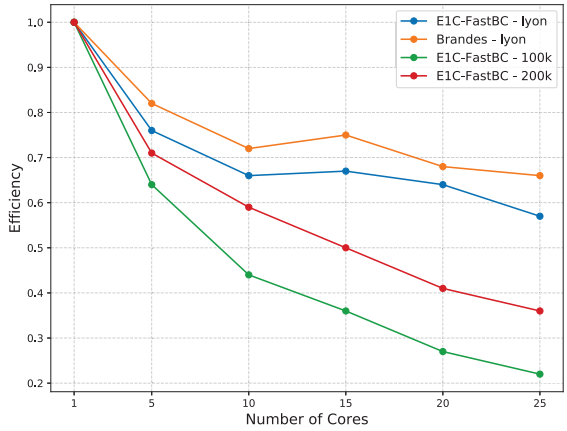


Figure 5.12: Efficiency analysis for lyon-road-network graph - $E_{p=[1,5,10,15,20,25]}$. Comparison with synthetic graphs having similar size (100k and 200k nodes) and with Brandes' algorithm on the same graph.

parts above. Results have been obtained by running our algorithm on the synthetic graphs with 100,000 and 200,000 nodes in sequential and parallel modes. The level of parallelism (*i.e.*, total number of exploited cores) is indicated on the x-axis of the figures.

The parts exhibiting the highest computation time are *Global BC* and *Local*

BC & Top. Props.. They both show a very good scalability as increasing resources up to the maximum limit (25 cores) translates into a reduction in time.

The third heaviest part is *Louvain Clustering*. Its computation time does not keep decreasing when augmenting parallelism beyond 5 cores. This is due to the fact that we have chosen to launch 10 parallel executions of the Louvain method in different configurations with the aim of selecting the one that produces the clustering with the best modularity score. This aspect highlights a potential bottleneck, since the computation time of *Louvain Clustering* at high levels of parallelism becomes comparable with those related to *Global BC* and *Local BC & Top. Props.*.

As already discussed in Subsec. 5.5.3, the number of external nodes for our synthetic graphs is almost always equal to zero. Therefore, the contribution of *HSN & External Nodes* is not relevant as well as those of *Classes & Pivots* and *Final Accumulation*. In particular, the latter is a sequential step entirely performed on the driver. Therefore it does not vary with the number of cores.

For *Dataset Reading*, computation time slowly increases with the number of cores due to the overhead introduced for creating the initial RDD, by reading data from the file system, with a number of partitions equal to the number of cores. Even for this step, the computation time becomes comparable with those obtained for *Local BC & Top. Props.* and *Global BC* when the parallelism increases.

It is worth to note that synthetic graphs represent a sort of best-case scenario: the very low number of border nodes and pivots, together with the almost complete absence of external nodes, allows for excellent performance. A more realistic scenario is analyzed in the following, by focusing on a real graph.

Figure 5.15 reports the time taken by each part of the algorithm on the ego-twitter graph[9], in sequential and parallel modes. In his case, *HSN & External Nodes* becomes the second heaviest contribution with values comparable to those of *Local BC & Top. Props.*. This is mainly due to the increased number of external nodes and confirms the importance of achieving ideal clustering. Similar considerations as those made for synthetic graphs apply as well to the remaining contributions from the breakdown analysis related to the ego-twitter graph.

5.6 Mathematical Foundations

As discussed in Sec. 5.3, our algorithm relies on multiple mathematical properties to allow for the exact fast computation of BC. In the following, we provide the mathematical proofs of the mathematical foundation (theorem, corollary and claim) at the basis our algorithm. In theorem 5.6.1, we prove that two nodes of the same cluster that satisfy some properties produce the same dependency score on nodes outside the cluster for destinations that are outside the cluster.

[9]We do not use lyon-road-network graph as in Subsection 5.5.4 because of the Out of Memory error that arise when running the algorithm in profiling mode.

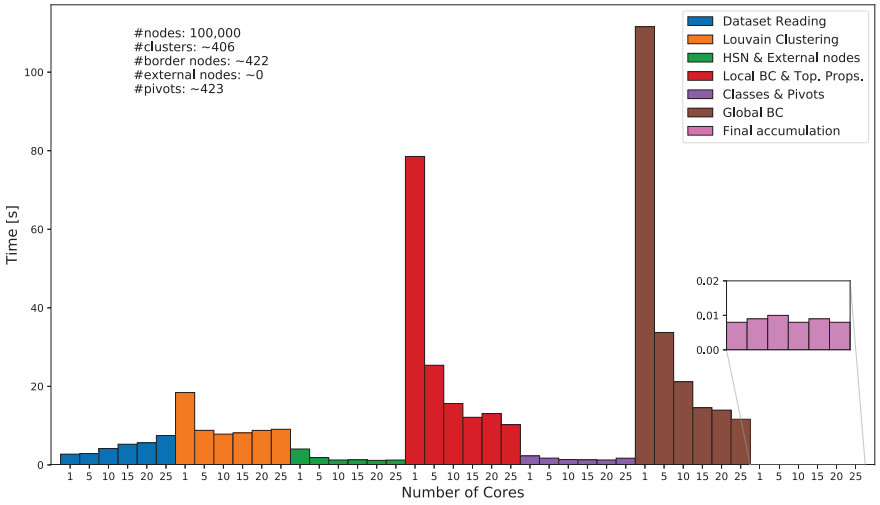


Figure 5.13: Breakdown analysis of computation time on synthetic graph with 100,000 nodes

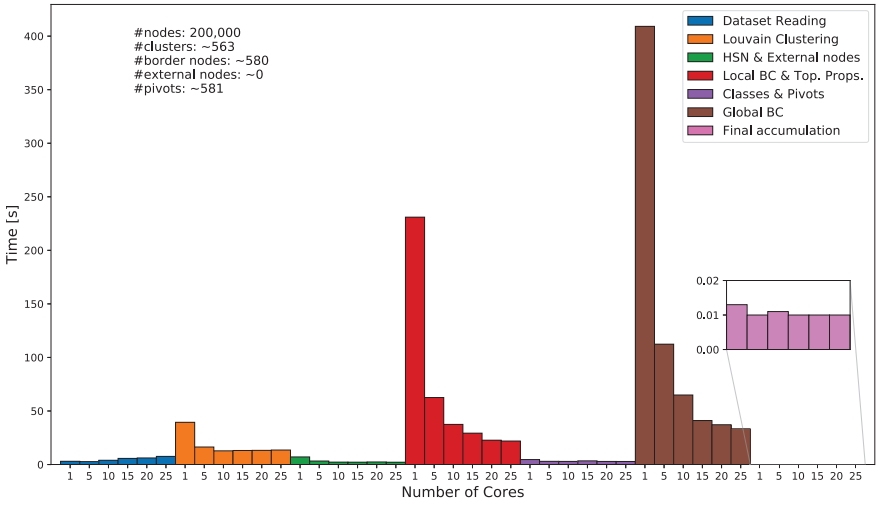


Figure 5.14: Breakdown analysis of computation time on synthetic graph with 200,000 nodes

Theorem 5.6.1. *Let $k \in \mathbb{R}^+$ and $l \in \mathbb{R}$, let $\mathbf{C}_i$ be a generic cluster of graph $\mathbf{G}$ with border nodes $\mathbf{BN}_{\mathbf{C}_i}$ and $s, p \in \mathbf{V}_{\mathbf{C}_i}$. If $\forall\ b_j \in \mathbf{BN}_{\mathbf{C}_i}\ \sigma_{s,b_j} = k \cdot \sigma_{p,b_j}$ and*

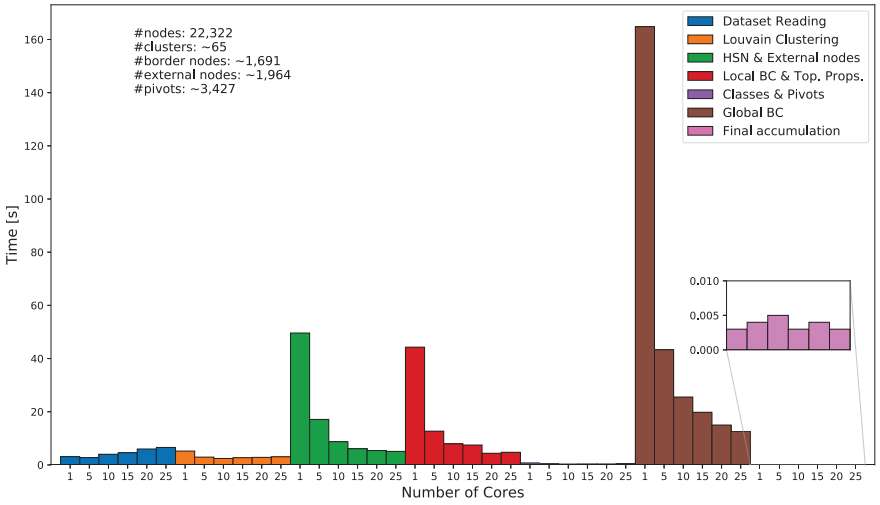


Figure 5.15: Breakdown analysis of computation time on real graph ego-twitter

$d_{\mathbf{G}}(s, b_j) = d_{\mathbf{G}}(p, b_j) + l$, *then* $\delta_{s,\overline{\mathbf{V}_{\mathbf{C}_i}}}(v) = \delta_{p,\overline{\mathbf{V}_{\mathbf{C}_i}}}(v)$, $\forall v \in \overline{\mathbf{V}_{\mathbf{C}_i}}$.

Proof. By rewriting the statement of the theorem as follows:

$$\delta_{s,\overline{\mathbf{V}_{\mathbf{C}_i}}}(v) = \delta_{p,\overline{\mathbf{V}_{\mathbf{C}_i}}}(v) \quad \forall v \in \overline{\mathbf{V}_{\mathbf{C}_i}} \iff \sum_{t \notin \mathbf{V}_{\mathbf{C}_i}} \frac{\sigma_{s,t}(v)}{\sigma_{s,t}} = \sum_{t \notin \mathbf{V}_{\mathbf{C}_i}} \frac{\sigma_{p,t}(v)}{\sigma_{p,t}} \qquad \forall v \in \overline{\mathbf{V}_{\mathbf{C}_i}} \tag{5.16}$$

we can prove it by proving that the two following conditions:

$$\sigma_{s,t}(v) = k \cdot \sigma_{p,t}(v) \qquad \forall v, t \in \overline{\mathbf{V}_{\mathbf{C}_i}} \tag{5.17}$$

and

$$\sigma_{s,t} = k \cdot \sigma_{p,t} \qquad \forall t \in \overline{\mathbf{V}_{\mathbf{C}_i}} \tag{5.18}$$

hold under the hypotheses of the theorem.

Let us first prove the following Lemma 5.6.1, which permits to express the relationship on the distances to cluster border nodes (*i.e.*, $d_{\mathbf{G}}(s, b_j) = d_{\mathbf{G}}(p, b_j) + l$) as an equivalence of the sets of border nodes traversed from s and p to reach nodes t outside the given cluster.

Lemma 5.6.1. *Let* $\mathbf{BN}_{\mathbf{C}_i}(u, t) \subseteq \mathbf{BN}_{\mathbf{C}_i}$ *denote the set of border nodes of cluster* $\mathbf{C}_i$ *on the shortest paths from* $u \in \mathbf{V}_{\mathbf{C}_i}$ *to* $t \in \overline{\mathbf{V}_{\mathbf{C}_i}}$. *Given a constant* $l \in \mathbb{R}$ *and two nodes* $s, p \in \mathbf{V}_{\mathbf{C}_i}$, *if* $d_{\mathbf{G}}(s, b_j) = d_{\mathbf{G}}(p, b_j) + l \;\; \forall \; b_j \in \mathbf{BN}_{\mathbf{C}_i}$ *then* $\mathbf{BN}_{\mathbf{C}_i}(s, t) = \mathbf{BN}_{\mathbf{C}_i}(p, t)$.

Proof. Let us consider two border nodes $b_j, b_k \in \mathbf{BN}_{\mathbf{C}_i}$ with $b_j \in \mathbf{BN}_{\mathbf{C}_i}(s,t)$ and $b_k \notin \mathbf{BN}_{\mathbf{C}_i}(s,t)$. By definition of shortest path between two nodes, we have:

$$d_{\mathbf{G}}(s,b_k) + d_{\mathbf{G}}(b_k,t) > d_{\mathbf{G}}(s,t) \tag{5.19}$$

Given that $b_j \in \mathbf{BN}_{\mathbf{C}_i}(s,t)$ by hypothesis, Equation 5.19 can be easily re-written as follows:

$$\begin{aligned} d_{\mathbf{G}}(s,b_k) + d_{\mathbf{G}}(b_k,t) > d_{\mathbf{G}}(s,t) \iff \quad & d_{\mathbf{G}}(s,b_k) + d_{\mathbf{G}}(b_k,t) > d_{\mathbf{G}}(s,b_j) \\ & + d_{\mathbf{G}}(b_j,t) \end{aligned} \tag{5.20}$$

Now, by relying on the hypothesis of the lemma, we exploit the relationships holding between the distances of generic nodes s and p to each border node in $\mathbf{BN}_{\mathbf{C}_i}$, thus obtaining:

$$\begin{aligned} d_{\mathbf{G}}(s,b_k) + d_{\mathbf{G}}(b_k,t) > d_{\mathbf{G}}(s,t) \iff \quad & d_{\mathbf{G}}(p,b_k) + l + d_{\mathbf{G}}(b_k,t) > d_{\mathbf{G}}(p,b_j) \\ & + l + d_{\mathbf{G}}(b_j,t) \\ \iff \quad & d_{\mathbf{G}}(p,b_k) + d_{\mathbf{G}}(b_k,t) > d_{\mathbf{G}}(p,b_j) \\ & + d_{\mathbf{G}}(b_j,t) \end{aligned} \tag{5.21}$$

From Equation 5.21, we can derive that b_k does not belong to any shortest path between p and t, *i.e.*:

$$d_{\mathbf{G}}(s,b_k) + d_{\mathbf{G}}(b_k,t) > d_{\mathbf{G}}(s,t) \iff \quad b_k \notin \mathbf{BN}_{\mathbf{C}_i}(p,t)$$

As the relation above holds for any node $b_k \in \mathbf{BN}_{\mathbf{C}_i}$ which does not belong to any shortest path between s and t, we can conclude that:

$$\mathbf{BN}_{\mathbf{C}_i}(p,t) \subseteq \mathbf{BN}_{\mathbf{C}_i}(s,t) \tag{5.22}$$

Likewise, it is possible to prove that if $b_j \in \mathbf{BN}_{\mathbf{C}_i}(p,t)$ and $b_k \notin \mathbf{BN}_{\mathbf{C}_i}(p,t)$, we have:

$$d_{\mathbf{G}}(p,b_k) + d_{\mathbf{G}}(b_k,t) > d_{\mathbf{G}}(p,t) \iff \quad \mathbf{BN}_{\mathbf{C}_i}(s,t) \subseteq \mathbf{BN}_{\mathbf{C}_i}(p,t) \tag{5.23}$$

Therefore, from Equation 5.22 and Equation 5.23, we can conclude that the following relationship holds:

$$\begin{aligned} & \mathbf{BN}_{\mathbf{C}_i}(s,t) \subseteq \mathbf{BN}_{\mathbf{C}_i}(p,t) \text{ AND } \mathbf{BN}_{\mathbf{C}_i}(p,t) \subseteq \mathbf{BN}_{\mathbf{C}_i}(s,t) \\ & \iff \\ & \mathbf{BN}_{\mathbf{C}_i}(s,t) = \mathbf{BN}_{\mathbf{C}_i}(p,t) \end{aligned} \tag{5.24}$$

which proves the lemma. □

To complete the proof of Theorem 5.6.1, we need now to prove Equation 5.17 and Equation 5.18. To that purpose, we consider the following lemma.

Lemma 5.6.2. *Let s be a node of cluster $\mathbf{C}_i$, and t any node in $\overline{\mathbf{V}_{\mathbf{C}_i}}$. $\mathbf{BN}_{\mathbf{C}_i}(s,t)$ is the set of border nodes of cluster $\mathbf{C}_i$ that belong to the shortest paths between s and t. If $\mathbf{BN}_{\mathbf{C}_i}(s,t) = \mathbf{BN}_{\mathbf{C}_i}(p,t)$, then, $\forall t \in \overline{\mathbf{V}_{\mathbf{C}_i}}$, $\sigma_{s,t} = k \cdot \sigma_{p,t}$ and $\sigma_{s,t}(v) = k \cdot \sigma_{p,t}(v)$.*

Proof. By leveraging Bellman's criterion:

$$\sigma_{s,t} = \sum_{b_j \in \mathbf{BN}_{\mathbf{C}_i}(s,t)} \sigma_{s,b_j} \cdot \sigma_{b_j,t}. \tag{5.25}$$

From the hypothesis of Theorem 5.6.1, we know that $\sigma_{s,b_j} = k \cdot \sigma_{p,b_j}$ $\forall b_j \in \mathbf{BN}_{\mathbf{C}_i}$ and equivalently $\forall b_j \in \mathbf{BN}(s,t)$, as $\mathbf{BN}(s,t) \subseteq \mathbf{BN}_{\mathbf{C}_i}$. Therefore, Equation 5.25 becomes:

$$\sigma_{s,t} = \sum_{b_j \in \mathbf{BN}_{\mathbf{C}_i(s,t)}} k \cdot \sigma_{p,b_j} \cdot \sigma_{b_j,t} \tag{5.26}$$

By the hypotheses of this lemma, we also know that $\mathbf{BN}_{\mathbf{C}_i}(s,t) = \mathbf{BN}_{\mathbf{C}_i}(p,t)$. Thus, we have:

$$\begin{aligned} \sum_{b_j \in \mathbf{BN}_{\mathbf{C}_i(s,t)}} k \cdot \sigma_{p,b_j} \cdot \sigma_{b_j,t} &= k \cdot \sum_{b_j \in \mathbf{BN}_{\mathbf{C}_i}(p,t)} \sigma_{p,b_j} \cdot \sigma_{b_j,t} \\ &= k \cdot \sigma_{p,t} \end{aligned} \tag{5.27}$$

With the same reasoning, it is also evident to prove the following:

$$\sigma_{s,t}(v) = k \cdot \sigma_{p,t}(v) \tag{5.28}$$

□

Equation 5.28 and Equation 5.27 from Lemma 5.6.2 prove, via Lemma 5.6.1, Equation 5.17 and Equation 5.18 respectively. Therefore Theorem 5.6.1 is proved. □

We now prove that the normalized distances and number of shortest paths fulfill the conditions of Theorem 5.6.1 and hence can be used to group nodes into classes of equivalence.

Corollary 5.6.1. *If $\forall\, b_j \in \mathbf{BN}_{\mathbf{C}_i}$: $\hat{\sigma}_{s,b_j} = \hat{\sigma}_{p,b_j}$ and $\hat{d_{\mathbf{G}}}(s,b_j) = \hat{d_{\mathbf{G}}}(p,b_j)$, then $\delta_{s,\overline{\mathbf{V}_{\mathbf{C}_i}}}(v) = \delta_{p,\overline{\mathbf{V}_{\mathbf{C}_i}}}(v)$, $\forall v \in \overline{\mathbf{V}_{\mathbf{C}_i}}$.*

Proof. To prove the corollary, we only need to prove that the following two equations hold:

$$\hat{d_{\mathbf{G}}}(s,b_j) = \hat{d_{\mathbf{G}}}(p,b_j) \;\; \forall b_j \in \mathbf{BN_{C}}_i \Longrightarrow d_{\mathbf{G}}(s,b_j) = d_{\mathbf{G}}(p,b_j) + l \;\; \forall b_j \in \mathbf{BN_{C}}_i \tag{5.29}$$

and:

$$\hat{\sigma}_{s,b_j} = \hat{\sigma}_{p,b_j} \;\; \forall b_j \in \mathbf{BN_{C}}_i \Longrightarrow \sigma_{s,b_j} = k \cdot \sigma_{p,b_j} \;\; \forall b_j \in \mathbf{BN_{C}}_i \tag{5.30}$$

Let us consider any two generic pair of nodes s and p belonging to cluster $\mathbf{C}_i$ such that:

$$\begin{aligned} &\forall b_j \in \mathbf{BN_{C}}_i, \\ &\quad d_{\mathbf{G}}(s,b_j) - min_{b_k \in \mathbf{BN_{C}}_i} d_{\mathbf{G}}(s,b_k) = d_{\mathbf{G}}(p,b_j) - min_{b_k \in \mathbf{BN_{C}}_i} d_{\mathbf{G}}(p,b_k) \quad (5.31) \\ &\quad \text{AND} \\ &\quad \frac{\sigma_{s,b_j}}{min_{b_k \in \mathbf{BN_{C}}_i} \sigma_{s,b_k}} = \frac{\sigma_{p,b_j}}{min_{b_k \in \mathbf{BN_{C}}_i} \sigma_{p,b_k}} \quad (5.32) \end{aligned}$$

By definition, Equation 5.31 can be easily re-written as follows:

$$\begin{aligned} &\forall b_j \in \mathbf{BN_{C}}_i, \\ &d_{\mathbf{G}}(s,b_j) - min_{b_k \in \mathbf{BN_{C}}_i} d_{\mathbf{G}}(s,b_k) = d_{\mathbf{G}}(p,b_j) - min_{b_k \in \mathbf{BN_{C}}_i} d_{\mathbf{G}}(p,b_k) \\ &\iff \quad d_{\mathbf{G}}(s,b_j) = d_{\mathbf{G}}(p,b_j) \underbrace{- min_{b_k \in \mathbf{BN_{C}}_i} d_{\mathbf{G}}(p,b_k) + min_{b_k \in \mathbf{BN_{C}}_i} d_{\mathbf{G}}(s,b_k)}_{\text{constant value}} \\ &\iff \quad d_{\mathbf{G}}(s,b_j) = d_{\mathbf{G}}(p,b_j) + l \;\; \text{with } l \in \mathbb{R} \end{aligned}$$

which corresponds to Equation 5.29.

Likewise, Equation 5.32 can be re-written as:

$$\begin{aligned} &\forall b_j \in \mathbf{BN_{C}}_i, \\ &\frac{\sigma_{s,b_j}}{min_{b_k \in \mathbf{BN_{C}}_i} \sigma_{s,b_k}} = \frac{\sigma_{p,b_j}}{min_{b_k \in \mathbf{BN_{C}}_i} \sigma_{p,b_k}} \\ &\iff \quad \sigma_{s,b_j} = \sigma_{p,b_j} \cdot \underbrace{\frac{min_{b_k \in \mathbf{BN_{C}}_i} \sigma_{s,b_k}}{min_{b_k \in \mathbf{BN_{C}}_i} \sigma_{p,b_k}}}_{\text{constant ratio}} \\ &\iff \quad \sigma_{s,b_j} = \sigma_{p,b_j} \cdot k \;\; \text{with } k \in \mathbb{R}^+ \end{aligned}$$

which corresponds to Equation 5.30. As the two equations (Equation 5.29 and Equation 5.30) are jointly satisfied, the corollary is proved from Theorem 5.6.1. □

Claim 5.6.1. *In undirected graphs:*

$$\sum_{s \in \mathbf{V}_{\mathbf{C}(v)}} \sum_{t \notin \mathbf{V}_{\mathbf{C}(v)}} \delta_{s,t}(v) = \sum_{s \notin \mathbf{V}_{\mathbf{C}(v)}} \sum_{t \in \mathbf{V}_{\mathbf{C}(v)}} \delta_{s,t}(v) \tag{5.33}$$

The proof of the claim above, used in Subsec. 5.3.2, is entirely based on the property of undirection. We remove part of the estimation errors we had in the previous implementations ([75]) by changing the computation of global dependency score using this claim.

Proof. Thanks to the undirected nature of the graph, we have:

$$\begin{aligned}
\sum_{s \in \mathbf{V}_{\mathbf{C}(v)}} \sum_{t \notin \mathbf{V}_{\mathbf{C}(v)}} \delta_{s,t}(v) &= \sum_{s \in \mathbf{V}_{\mathbf{C}(v)}} \sum_{t \notin \mathbf{V}_{\mathbf{C}(v)}} \frac{\sigma_{s,t}(v)}{\sigma_{s,t}} \\
&= \sum_{s \in \mathbf{V}_{\mathbf{C}(v)}} \sum_{t \notin \mathbf{V}_{\mathbf{C}(v)}} \frac{\sigma_{t,s}(v)}{\sigma_{t,s}} \\
&= \sum_{t \notin \mathbf{V}_{\mathbf{C}(v)}} \sum_{s \in \mathbf{V}_{\mathbf{C}(v)}} \frac{\sigma_{t,s}(v)}{\sigma_{t,s}}
\end{aligned}$$

Now, by changing the name of variables s and t:

$$\begin{aligned}
\sum_{s \in \mathbf{V}_{\mathbf{C}(v)}} \sum_{t \notin \mathbf{V}_{\mathbf{C}(v)}} \delta_{s,t}(v) &= \sum_{t \notin \mathbf{V}_{\mathbf{C}(v)}} \sum_{s \in \mathbf{V}_{\mathbf{C}(v)}} \frac{\sigma_{t,s}(v)}{\sigma_{t,s}} \\
&= \sum_{s \notin \mathbf{V}_{\mathbf{C}(v)}} \sum_{t \in \mathbf{V}_{\mathbf{C}(v)}} \frac{\sigma_{s,t}(v)}{\sigma_{s,t}} \\
&= \sum_{s \notin \mathbf{V}_{\mathbf{C}(v)}} \sum_{t \in \mathbf{V}_{\mathbf{C}(v)}} \delta_{s,t}(v)
\end{aligned}$$

□

5.7 Complementary Information on the Implementation of E1CFBC

In our work, we implemented two alternative approaches to back-propagate contributions during the computation of the global BC. The first one, detailed in Subsec. 5.3.2, is based on using one vector of contributions per node, with one entry per cluster of destinations. The main limitation of this approach is related to high memory consumption as it implies working with n vectors (n = number of nodes), each including $|\mathbf{C}|$ contributions. To optimise our solution with respect to this limitation, we developed another approach based on a vector of two elements only per node. This method is described below.

We can consider a vector of two elements instead of $|\mathbf{C}|$ elements, so that the contribution due to a destination t is assigned to $\delta^{\gamma}_{s,\mathbf{V}_{\mathbf{C}(v)}}(v)$ or $\delta^{\gamma}_{s,\overline{\mathbf{V}_{\mathbf{C}(v)}}}(v)$ depending on whether t is in the same cluster as v, $\mathbf{C}(v)$, or not. During the back-propagation, the two contributions are propagated independently as follows:

$$\delta^{\gamma}_{s,\mathbf{V}_{\mathbf{C}(v)}}(v) = \begin{cases} \frac{\sigma_{s,v}}{\sigma_{s,w}} \cdot (1 + \delta^{\gamma}_{s,\mathbf{V}_{\mathbf{C}(w)}}(w)) & \text{if } \mathbf{C}(v) = \mathbf{C}(w), \\ 0 & \text{otherwise} \end{cases}$$

$$\delta^{\gamma}_{s,\overline{\mathbf{V}_{\mathbf{C}(v)}}}(v) = \begin{cases} \frac{\sigma_{s,v}}{\sigma_{s,w}} \cdot \delta^{\gamma}_{s,\overline{\mathbf{V}_{\mathbf{C}(w)}}}(w) & \text{if } \mathbf{C}(v) = \mathbf{C}(w), \\ \frac{\sigma_{s,v}}{\sigma_{s,w}} \cdot (1 + \delta^{\gamma}_{s,\mathbf{V}_{\mathbf{C}(w)}}(w) + \delta^{\gamma}_{s,\overline{\mathbf{V}_{\mathbf{C}(w)}}}(w)) & \text{otherwise} \end{cases} \tag{5.34}$$

At the end of back-propagation, we put the dependency scores of nodes v belonging to the same cluster of the (pivot) source node to 0, whereas the dependency scores of nodes not belonging to the same cluster of the source node are computed using the following formula:

$$\delta^{\gamma}_{s,\overline{\mathbf{V}_{\mathbf{C}(s)}}}(v) = 2 \cdot \delta^{\gamma}_{s,\mathbf{V}_{\mathbf{C}(v)}}(v) + \delta^{\gamma}_{s,\overline{\mathbf{V}_{\mathbf{C}(v)}}}(v) \tag{5.35}$$

Finally, according to Equation 5.13, $\delta^{\gamma}_{s,\bullet}(v)$ is multiplied by the cardinality of the equivalence class s belongs to. However, this solution introduces an error. In fact, in Equation 5.34, $\delta^{\gamma}_{s,\overline{\mathbf{V}_{\mathbf{C}(w)}}}(w)$ may contain contributions of destination nodes belonging to $\mathbf{V}_{\mathbf{C}(v)}$, which should be moved from $\delta^{\gamma}_{s,\overline{\mathbf{V}_{\mathbf{C}(v)}}}(v)$ to $\delta^{\gamma}_{s,\mathbf{V}_{\mathbf{C}(v)}}(v)$, so that they can be correctly multiplied by 2 as required by Equation 5.35. This situation is a consequence of the presence of external shortest paths that leave and then re-enter the clusters.

The correction above has to be performed during the back-propagation phase for each border node such that *i)* there is at least one external shortest path starting from that border node, and *ii)* all the contributions of the border node have been computed, *i.e.,* when the border node is the current w. Consequently, for each pair of border nodes b_1, b_2 belonging to the same cluster $\mathbf{C}_i$, we need to compute the distance $d_{\mathbf{G}}(b_1, b_2)$ and the number of external shortest paths $\sigma^{ext}_{b_1,b_2}$ between them. We perform this operation when searching for external nodes.

5.8 Conclusion

In this chapter, we presented a very fast algorithm for performing the exact computation of BC in undirected graphs. The algorithm exploits clustering and structural properties of graphs to reduce the computing time. In particular, the algorithm exhibits an impressive speedup (compared with Brandes' algorithm and the one labelled with $\mathcal{H}$, especially for very large scale-free graphs with an attachment coefficient $m = 1$. A significant speedup is achieved also with other

kinds of graphs, as demonstrated by the results obtained with real graphs. The reduction of the computation time is mainly due to the adoption of pivots, *i.e.,* nodes that contribute equally to the dependency score of other graph nodes.

This chapter described both a sequential and a map-reduce parallel version of the algorithm implemented in Scala over Spark. The experimental analysis, performed with reference to the number of cores exploited for computation, revealed that the efficiency is slightly lower than Brandes' algorithm but it increases with graph size. In fact the granularity per Spark-task of the SSSP computations is small when graphs are not very large due to the relative low number of pivots.

The speedup of E1C-FastBC strongly depends on the number of pivots; thus clustering and modularity play a key role for the computation time of the algorithm. As future work, we aim to study other clustering methods for more effectively identifying border nodes in (synthetic and real) graphs with different topologies. Finally, we will investigate a better mapping of the algorithm on distributed resources, when data-parallelism is exploited, by improving locality especially when different Spark executors are used.

Chapter 6

Limitations, Conclusion and Prospects

We studied in this thesis two types of approaches for vulnerability analysis inside road networks to make cities more resilient to incidents. The first dynamical approach was a solution of traffic management for large scale road networks. This solution offered a reduction of congestion in critical zones. It reduced the drop of performance induced by a high peak of demand. Our approach showed robustness as it also performed well for extreme scenarios, especially a highly congested one. The dynamic and cooperative multi-agent based decision process made our approach reactive to unpredictable events, thus increasing the global resilience of the network. Not only did we reduce the drop of performance measured by the mean speed but the duration of disturbance was reduced as well. The hierarchical cooperation between infrastructures and vehicles enable a reduction of travel times for most of the users. This solution has been later improved with the use of the resilience metric Betweenness Centrality, widely used to identify vulnerable nodes inside a graph. We added a stronger penalization to nodes with high BC when their zone was congested to better protect vulnerable nodes. Better choices of shortest path, taking into account the topological vulnerability of road sections, improved the total travel time reduction per rerouting. It thus improved the overall performance of our strategy, showing that the combination of topological and system-based vulnerability analysis is possible and beneficial for resilience increase. Studying the BC was the main focus of the second type of vulnerability analysis done during this thesis. Computing the BC in real-time, to include it in a dynamical analysis, is a challenge because of its computation time. We thus developed an algorithm for distributed computation of BC in large scale network, based on graph theory and paralleling of tasks. The algorithm showed great scalability and performed well for sparse graphs, in different domains other than transport.

6.1 Resilience Scope

The main scope our work was travel demand fluctuations over a road network. We tested our control strategy on different demand scenarios in order to evaluate the capability of our approach to handle perturbations that strongly increase demand on road networks such as failures in the transport systems. Further work could be focused on failures on the supplies, *i.e.* when strong weather incidents occur, blocking parts of the network. In this case, many nodes and links of the graph become unreachable, areas can even become totally inaccessible. The threat would not come from the demand but rather the supply.

Moreover, only static BC, computed on the free-flow travel time weighted graph, was included into our strategy. One of the first possible improvements would be to update the BC according to traffic conditions and to evaluate scenarios of supply failures, making an entire zone inaccessible. The dynamic computation of BC would be even more necessary for this kind of scenarios, where major changes occur on the graph that would undergo many link or node removal.

The computation of BC over a dynamic graph would thus be a first step for control strategy improvement and necessary for new scenario of supply failure.

6.2 Framework Modeling and Other Alternatives

We chose the context of multi-agent system to develop our strategy but on a framework (SymuPy) that is not multi-agent oriented. Traditional transport models like SymuPy are based on aggregated flows between zones. The demand is represented with an OD matrix (Origin-Destination). When dealing with a complex population (age, activity, time of day), the demand becomes more complicated to represent and requires many OD matrices. Modeling the demand with agents allows more realistic and complex simulations ([99], [100]).

For few decades, multi-agent simulation frameworks have been developed that better model driver's choice, preferences and reaction to its environment in a large scale context. They are derivated from the *activity-based* modeling of demand, where the population has a succession of activities during one day and travel from one to another. It is opposed to the *trip-based* modeling, which our approach was closer to. Multi-agent tools enable a better representation of driver interactions and their travel preferences, and modeling solutions for traffic management. The unpredictable behaviours are better represented and large-scale simulations can be run modeling each traveler. These simulators can perform one-day simulations using the succession of trips and activities. We could thus consider zone agents that learn from previous days of simulation and demand to adapt their strategy with past information.

Among multi-agent activity-based simulators, the main ones are TRANSIMS (TRansportation Analysis and SIMulation System), developed by the Los Alamos

National Laboratory for the U.S. Department of Transportation, the environment-oriented ILUTE (Integrated Land Use, Transportation, Environment), maintained and contributed by Canadian universities or OpenAMOS (Open Activity-Mobility Simulator). Most of the multi-agent transportation simulator tools are activity-based, which means that the demand is not only a succession of trips but rather of scheduled activities per person, with trips to reach them.

One of the most popular activity-based mesoscopic simulators is MATSim (MultiAgent Transport Simulation), an open-source tool, used for dynamic traffic assignment. Its main advantage compared with the other simulators is its parallelization of computation enabling large scale simulation of millions of agents.

MATSim is an open-source project, started with Kai Nagel at ETH Zürich, when TRANSIMS was not open-sourced yet. MATSim is developed in Java and module-based, enabling the adding and plugging of own developments and thus a useful customizability. MATSim performs a microscopic modeling of traffic, using microscopic modelling of demand and a co-evolutionary algorithm to find a stochastic user equilibrium. The main reason we did not choose MATSim for our control strategy is that it omits the car-following and lane-changing behaviour to reduce computation time. This misrepresentation is due to the queue-based implementation where roadways are FIFO queues. With this representation, parallelization is feasible and allows short computation times. Such a misrepresentation of traffic dynamics could have though distorted the evaluation of our control strategy. Now that we proved its performance, further developments could be done on MATSim. Especially, MATSim offers two solutions to compensate the problem of the incorrect modeling of backward wave. The first solution is to use JDEQSim as mobility simulation or "mobsim" (see Figure 6.1), loosing part of the parallelization efficiency. The other option is to tune the QSim as suggested in [101].

In MATSim, the demand is activity-based (and not trip-based): the persons, agents, have a succession of activities, and their trips are no longer the goal but the mean. At the end a simulation, each agent has a score, computed using a scoring function which depends on the time spent in trips, the delays and the time spent during the activities. Basically, a plan is optimal when the agent spent a minimal time in transportation which left him/her a maximal time for his/her activities. To optimize plans, MATSim runs several simulations with a co-evolutionary algorithm and modifies some plans, by changing the transportation mode or the departure time of some activities or the routes. These parameters can be tuned, the user can choose not to change the transportation mode for example. At the end of the simulation, we reached a user equilibrium, where the people have the best plan they can hope for.

To run simulations from large data set (population, network, duration), MATSim is multi-threaded. The consequences is that the computation time is not as high as other simulation tools but the congestion propagation is not perfectly simulated, as explained earlier. As an example, by converting SymuPy demand into MATSim inputs, it took 2 minutes for MATSim to run the simulation for the

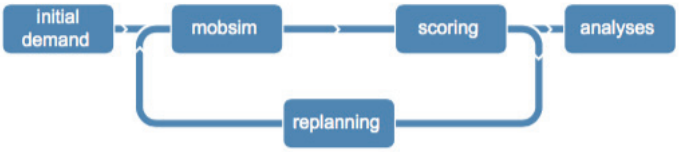


Figure 6.1: MATSim loop ([101])

same scenario instead of the 4 hours of SimuPy. As we worked with a given sets of trips, only one iteration was necessary, the equilibrium being already found.

With such computation improvement, we could consider more complete scenarios. With a shortened computation time, more complex simulations can be chosen, especially for multi-modal vulnerability analysis. The interdependence of transportation modes makes some parts of the network more vulnerable such as road sections near big subway hubs. Moreover, bus or taxi suffer from jams and can be distinguished from other vehicles as their constraints and objectives are completely different from other travelers.

The modularity of MATSim makes it also easy to plug the fast BC computation algorithm, E1CFastBC, to a simulation.

6.3 Future work

We were able to reproduce multi-agent behaviour and interactions but the possibilities offered by multi-agent systems were not fully explored. The use of object oriented solution makes it easier to be close to a multi-agent context but does not capture driver behaviours. Our solution did not require a fully multi-agent oriented framework and the first objective was to prove that our approach was pertinent to reduce vulnerability. For further developments though, the use of multi-agent frameworks will be relevant. Multi-agent frameworks make it easier to model the diversity of population (connected vehicles or not) and the diversity of behaviours. For example, in our case, we consider that all vehicles can be rerouted. In a more detailed context, we could differentiate the kind of drivers, the connectivity of the vehicle, the availability of the driver to accept recommendation.

Moreover, the cornerstone of resilience not addressed in this thesis is the "learning" stage, which corresponds to the ability of the system to store past information and reuse it adequately in the present. The zone agents could for example act depending on the effects of their past actions, in a context of Reinforcement Learning. The demand or the level of congestion can also be predicted from past data. We saw in Chapter 4 that the performance of our control was higher when the demand was high. By predicting the global demand on the network, we could have a level of control dependant on the prediction: with low predicted demand, no rerouting would be suggested whereas a high predicted demand would trigger our rerouting strategy. Many studies already focus on demand prediction: Raj *et al.* ([102]) using k-NN and neural network to predict

demand from sensor data, Polson *et al.* ([103]) using deep learning for short-term flow prediction during two special events (sport game and storm) for example.

The tool and scope of our analysis allowed to prove the robustness and efficiency of our approaches in a realistic traffic flow dynamic context. Nonetheless, for further developments that would include an enlarged scope, more suited multi-agent frameworks should be considered that are more appropriate and efficient. MATSim would enable larger urban zone to be studied with a more diversified population. It would for example help to better characterize resilience and vulnerability from different points of view and better evaluate the performance of transportation systems (including pollution for example). The use of activity-based modeling via multi-agent frameworks is a good perspective to continue the vulnerability analysis of road networks. As the combination of topological and dynamic vulnerability analysis is possible and benefits the transportation system, our algorithm for BC fast computation should be integrated to the control framework and more critical scenarios should be tested, with further developments on consequences analysis. Multi-modal representation of transportation network is also interesting for further development to better grasp the dependence between transportation modes. The analysis of the multi-modal transport network is interesting for both the dynamic and static vulnerability approaches.

Bibliography

[1] Katja Berdica. "An introduction to road vulnerability: what has been done, is done and should be done". In: *Transport policy* 9.2 (2002), pp. 117–127.

[2] Lars-Göran Mattsson and Erik Jenelius. "Vulnerability and resilience of transport systems–A discussion of recent research". In: *Transportation Research Part A: Policy and Practice* 81 (2015), pp. 16–34.

[3] Alan T Murray and Tony H Grubesic. "Overview of reliability and vulnerability in critical infrastructure". In: *Critical infrastructure*. Springer, 2007, pp. 1–8.

[4] Michael AP Taylor. "Travel through time: the story of research on travel time reliability". In: *Transportmetrica B: transport dynamics* 1.3 (2013), pp. 174–194.

[5] John Bates, John Polak, Peter Jones, et al. "The valuation of reliability for personal travel". In: *Transportation Research Part E: Logistics and Transportation Review* 37.2-3 (2001), pp. 191–229.

[6] Terence C Lam and Kenneth A Small. "The value of time and reliability: measurement from a value pricing experiment". In: *Transportation Research Part E: Logistics and Transportation Review* 37.2-3 (2001), pp. 231–251.

[7] Nebiyou Y Tilahun and David M Levinson. "A moment of time: reliability in route choice using stated preference". In: *Journal of Intelligent Transportation Systems* 14.3 (2010), pp. 179–187.

[8] Yuh-Wen Chen and Gwo-Hshiung Tzeng. "A fuzzy multi-objective model for reconstructing the post-quake road-network by genetic algorithm". In: *International Journal of Fuzzy Systems* 1.2 (1999), pp. 85–95.

[9] Jaimyoung Kwon, Tiffany Barkley, Rob Hranac, et al. "Decomposition of travel time reliability into various sources: incidents, weather, work zones, special events, and base capacity". In: *Transportation research record* 2229.1 (2011), pp. 28–33.

[10] Susan L Cutter, Joseph A Ahearn, Bernard Amadei, et al. "Disaster resilience: A national imperative". In: *Environment: Science and Policy for Sustainable Development* 55.2 (2013), pp. 25–29.

[11] Erik Hollnagel. *Resilience engineering in practice: A guidebook*. Ashgate Publishing, Ltd., 2013.

[12] David Koren, Vojko Kilar, and Katarina Rus. "Proposal for holistic assessment of urban system resilience to natural disasters". In: *IOP Conference Series: Materials Science and Engineering*. Vol. 245. 6. IOP Publishing. 2017, p. 062011.

[13] David Jaroszweski, Elizabeth Hooper, and Lee Chapman. "The impact of climate change on urban transport resilience in a changing world". In: *Progress in Physical Geography* 38.4 (2014), pp. 448–463.

[14] Derrick Hambly, Jean Andrey, Brian Mills, et al. "Projected implications of climate change for road safety in Greater Vancouver, Canada". In: *Climatic Change* 116.3 (2013), pp. 613–629.

[15] Elise Henry, Angelo Furno, and Nour-Eddin El Faouzi. "Approach to quantify the impact of disruptions on traffic conditions using dynamic weighted resilience metrics of transport networks". In: *Transportation research record* 2675.4 (2021), pp. 61–78.

[16] Juan Antonio Guerrero-Ibanez, Sherali Zeadally, and Juan Contreras-Castillo. "Integration challenges of intelligent transportation systems with connected vehicle, cloud computing, and internet of things technologies". In: *IEEE Wireless Communications* 22.6 (2015), pp. 122–128.

[17] Juan Guerrero-Ibáñez, Sherali Zeadally, and Juan Contreras-Castillo. "Sensor technologies for intelligent transportation systems". In: *Sensors* 18.4 (2018), p. 1212.

[18] Greg Marsden, Mike McDonald, and Mark Brackstone. "Towards an understanding of adaptive cruise control". In: *Transportation Research Part C: Emerging Technologies* 9.1 (2001), pp. 33–51.

[19] Toru Seo, Alexandre M Bayen, Takahiko Kusakabe, et al. "Traffic state estimation on highway: A comprehensive survey". In: *Annual reviews in control* 43 (2017), pp. 128–151.

[20] Maxime Guériau, Romain Billot, Nour-Eddin El Faouzi, et al. "How to assess the benefits of connected vehicles? A simulation framework for the design of cooperative traffic management strategies". In: *Transportation research part C: emerging technologies* 67 (2016), pp. 266–279.

[21] Ryan N Fries, Mostafa Reisi Gahrooei, Mashrur Chowdhury, et al. "Meeting privacy challenges while advancing intelligent transportation systems". In: *Transportation Research Part C: Emerging Technologies* 25 (2012), pp. 34–45.

[22] Elyes Ben Hamida, Hassan Noura, and Wassim Znaidi. "Security of cooperative intelligent transport systems: Standards, threats analysis and cryptographic countermeasures". In: *Electronics* 4.3 (2015), pp. 380–423.

[23] Allan M de Souza, Celso ARL Brennand, Roberto S Yokoyama, et al. "Traffic management systems: A classification, review, challenges, and future perspectives". In: *International Journal of Distributed Sensor Networks* 13.4 (2017), p. 1550147716683612. DOI: 10.1177/1550147716683612. eprint: https://doi.org/10.1177/1550147716683612. URL: https://doi.org/10.1177/1550147716683612.

[24] Allan M De Souza, Celso ARL Brennand, Roberto S Yokoyama, et al. "Traffic management systems: A classification, review, challenges, and future perspectives". In: *International Journal of Distributed Sensor Networks* 13.4 (2017), p. 1550147716683612.

[25] Liudmila Tumash, Carlos Canudas-De-Wit, and Maria Laura Delle Monache. "Boundary and VSL Control for Large-Scale Urban Traffic Networks". In: *IEEE, Transactions on Automatic Control* (2021).

[26] Denis Nikitin, Carlos Canudas-De-Wit, and Paolo Frasca. "Scale-free boundary control of multiple aggregates in large-scale networks". In: *MTNS 2020-24th International Symposium on Mathematical Theory of Networks and Systems*. 2021.

[27] Omar Boufous, Claudio Roncoli, and Themistoklis Charalambous. "Centralized and Distributed Multi-Region Traffic Flow Control". In: *2020 European Control Conference (ECC)*. IEEE. 2020, pp. 420–427.

[28] Liudmila Tumash, Carlos Canudas de Wit, and Maria Laura Delle Monache. "Boundary Control Design for Traffic with Nonlinear Dynamics". In: *IEEE Transactions on Automatic Control* (2021).

[29] Xin Yang, Juncheng Chen, Mantun Yan, et al. "Regional Boundary Control of Traffic Network Based on MFD and FR-PID". In: *Journal of Advanced Transportation* 2021 (2021).

[30] Shengling Gao, Daqing Li, Nan Zheng, et al. "Resilient perimeter control for hyper-congested two-region networks with MFD dynamics". In: *Transportation Research Part B: Methodological* 156 (2022), pp. 50–75.

[31] Mehmet Yildirimoglu, Isik Ilber Sirmatel, and Nikolas Geroliminis. "Hierarchical control of heterogeneous large-scale urban road networks via path assignment and regional route guidance". In: *Transportation Research Part B: Methodological* 118 (2018), pp. 106–123.

[32] Ramon Bauza and Javier Gozálvez. "Traffic congestion detection in large-scale scenarios using vehicle-to-vehicle communications". In: *Journal of Network and Computer Applications* 36.5 (2013), pp. 1295–1307.

[33] Guilherme B Araujo, Matheus M Queiroz, Fatima de LP Duarte-Figueiredo, et al. "Cartim: A proposal toward identification and minimization of vehicular traffic congestion for vanet". In: *2014 IEEE symposium on computers and communications (ISCC)*. IEEE. 2014, pp. 1–6.

[34] Juan Pan, Mohammad A Khan, Iulian Sandu Popa, et al. "Proactive vehicle re-routing strategies for congestion avoidance". In: *2012 IEEE 8th International Conference on Distributed Computing in Sensor Systems*. IEEE. 2012, pp. 265–272.

[35] Shen Wang, Soufiene Djahel, and Jennifer McManis. "An adaptive and vanets-based next road re-routing system for unexpected urban traffic congestion avoidance". In: *2015 IEEE vehicular networking conference (VNC)*. IEEE. 2015, pp. 196–203.

[36] Allan Mariano de Souza, Roberto Sadao Yokoyama, Nelson Luis Saldanha da Fonseca, et al. "Garuda: a new geographical accident aware solution to reduce urban congestion". In: *2015 IEEE international conference on computer and information technology; ubiquitous computing and communications; dependable, autonomic and secure computing; pervasive intelligence and computing*. IEEE. 2015, pp. 596–602.

[37] Allan Mariano de Souza, Azzedine Boukerche, Guilherme Maia, et al. "Decreasing greenhouse emissions through an intelligent traffic information system based on inter-vehicle communication". In: *Proceedings of the 12th ACM international symposium on Mobility management and wireless access*. 2014, pp. 91–98.

[38] Michael Wooldridge and Nicholas R Jennings. "Intelligent agents: Theory and practice". In: *The knowledge engineering review* 10.2 (1995), pp. 115–152.

[39] Edmund H Durfee and Jeffrey S Rosenschein. "Distributed problem solving and multi-agent systems: Comparisons and examples". In: *Proceedings of the Thirteenth International Distributed Artificial Intelligence Workshop*. 1994, pp. 94–104.

[40] Marco A Wiering. "Multi-agent reinforcement learning for traffic light control". In: *Machine Learning: Proceedings of the Seventeenth International Conference (ICML'2000)*. 2000, pp. 1151–1158.

[41] Ying Liu, Lei Liu, and Wei-Peng Chen. "Intelligent traffic light control using distributed multi-agent Q learning". In: *2017 IEEE 20th international conference on intelligent transportation systems (ITSC)*. IEEE. 2017, pp. 1–8.

[42] Assia Belbachir, Amal El Fallah-Seghrouchni, Arthur Casals, et al. "Smart mobility using multi-agent system". In: *Procedia Computer Science* 151 (2019), pp. 447–454.

[43] Suresh Chavhan and Pallapa Venkataram. "Prediction based traffic management in a metropolitan area". In: *Journal of traffic and transportation engineering (English edition)* 7.4 (2020), pp. 447–466.

[44] Eric Bonabeau, Guy Theraulaz, Marco Dorigo, et al. *Swarm intelligence: from natural to artificial systems*. 1. Oxford university press, 1999.

[45] Zhiguang Cao, Siwei Jiang, Jie Zhang, et al. "A Unified Framework for Vehicle Rerouting and Traffic Light Control to Reduce Traffic Congestion". In: *IEEE Transactions on Intelligent Transportation Systems* 18.7 (2017), pp. 1958–1973. DOI: 10.1109/TITS.2016.2613997.

[46] Bogdan Tatomir and Leon Rothkrantz. "Hierarchical routing in traffic using swarm-intelligence". In: *2006 IEEE intelligent transportation systems conference*. IEEE. 2006, pp. 230–235.

[47] Habib M Kammoun, Ilhem Kallel, Jorge Casillas, et al. "Adapt-Traf: An adaptive multiagent road traffic management system based on hybrid ant-hierarchical fuzzy model". In: *Transportation Research Part C: Emerging Technologies* 42 (2014), pp. 147–167.

[48] J G WARDROP. "ROAD PAPER. SOME THEORETICAL ASPECTS OF ROAD TRAFFIC RESEARCH." In: *Proceedings of the Institution of Civil Engineers* 1.3 (1952), pp. 325–362. DOI: 10.1680/ipeds.1952.11259. URL: https://doi.org/10.1680/ipeds.1952.11259.

[49] Ludovic Leclercq, Andres Ladino, and Cecile Becarie. "Enforcing optimal routing through dynamic avoidance maps." In: *Transportation Research Part B: Methodological* 149 (2021), pp. 118–137.

[50] Dalton Hahn, Arslan Munir, and Vahid Behzadan. "Security and privacy issues in intelligent transportation systems: Classification and challenges". In: *IEEE Intelligent Transportation Systems Magazine* 13.1 (2019), pp. 181–196.

[51] David Kempe, Alin Dobra, and Johannes Gehrke. "Gossip-based computation of aggregate information". In: *44th Annual IEEE Symposium on Foundations of Computer Science, 2003. Proceedings.* IEEE. 2003, pp. 482–491.

[52] JW Godfrey. "The mechanism of a road network". In: *Traffic Engineering & Control* 8.8 (1969).

[53] Nikolas Geroliminis and Carlos F Daganzo. "Existence of urban-scale macroscopic fundamental diagrams: Some experimental findings". In: *Transportation Research Part B: Methodological* 42.9 (2008), pp. 759–770.

[54] Igor Dakic and Monica Menendez. "On the use of Lagrangian observations from public transport and probe vehicles to estimate car space-mean speeds in bi-modal urban networks". In: *Transportation Research Part C: Emerging Technologies* 91 (2018), pp. 317–334.

[55] Takahiro Tsubota, Ashish Bhaskar, Alfredo Nantes, et al. "Comparative analysis of traffic state estimation: cumulative counts-based and trajectory-based methods". In: *Transportation Research Record* 2491.1 (2015), pp. 43–52.

[56] Reza Mohajerpoor, Meead Saberi, Hai L Vu, et al. "H robust perimeter flow control in urban networks with partial information feedback". In: *Transportation Research Part B: Methodological* 137 (2020), pp. 47–73.

[57] Meead Saberi, Hani S Mahmassani, Tian Hou, et al. "Estimating network fundamental diagram using three-dimensional vehicle trajectories: extending edie's definitions of traffic flow variables to networks". In: *Transportation Research Record* 2422.1 (2014), pp. 12–20.

[58] Victor Lequay, Mathieu Lefort, Saber Mansour, et al. "Ajustement diffus et adaptatif de la consommation électrique résidentielle par un système multi-agent auto-adaptatif". In: *Revue des Sciences et Technologies de l'Information-Série RIA: Revue d'Intelligence Artificielle* 31.4/2017 (2017), pp. 427–447.

[59] Ludovic Leclercq, Jorge Andres Laval, and Estelle Chevallier. "The Lagrangian coordinates and what it means for first order traffic flow models". In: *Transportation and Traffic Theory 2007. Papers Selected for Presentation at ISTTT17Engineering and Physical Sciences Research Council (Great Britain) Rees Jeffreys Road FundTransport Research FoundationTMS ConsultancyOve Arup and Partners, Hong KongTransportation Planning (International) PTV AG*. 2007.

[60] Michael James Lighthill and Gerald Beresford Whitham. "On kinematic waves II. A theory of traffic flow on long crowded roads". In: *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences* 229.1178 (1955), pp. 317–345.

[61] Paul I Richards. "Shock waves on the highway". In: *Operations research* 4.1 (1956), pp. 42–51.

[62] Geoff Boeing. "A Multi-Scale Analysis of 27, 000 Urban Street Networks". In: *CoRR* abs/1705.02198 (2017). arXiv: 1705.02198. URL: http://arxiv.org/abs/1705.02198.

[63] Stephen P Borgatti, Ajay Mehra, Daniel J Brass, et al. "Network analysis in the social sciences". In: *Science* 323.5916 (2009), pp. 892–895.

[64] David King and Amer Shalaby. "Performance Metrics and Analysis of Transit Network Resilience in Toronto". In: *Transportation Research Record* (2016).

[65] Yuanyuan Zhang, Xuesong Wang, Peng Zeng, et al. "Centrality Characteristics of Road Network Patterns of Traffic Analysis Zones". In: *Transportation Research Record: Journal of the Transportation Research Board* 2256 (2011), pp. 16–24.

[66] Bertrand Berche, Christian von Ferber, Taras Holovatch, et al. "Resilience of public transport networks against attacks". In: *The European Physical Journal B* 71.1 (2009), pp. 125–137.

[67] Angelo Furno, Nour-Eddin El Faouzi, Rajesh Sharma, et al. "Two-level clustering fast betweenness centrality computation for requirement-driven approximation". In: *2017 IEEE International Conference on Big Data (Big Data)*. IEEE. 2017, pp. 1289–1294.

[68] Angelo Furno, Nour-Eddin El Faouzi, Rajesh Sharma, et al. "A Graph-Based Framework for Real-Time Vulnerability Assessment of Road Networks". In: *2018 IEEE International Conference on Smart Computing, SMARTCOMP 2018, Taormina, Sicily, Italy, June 18-20, 2018*. 2018, pp. 234–241. DOI: 10.1109/SMARTCOMP.2018.00096.

[69] Petter Holme, Beom Jun Kim, Chang No Yoon, et al. "Attack vulnerability of complex networks". In: *Physical Review E* 65.5 (2002), p. 056109.

[70] Tami Carpenter, George Karakostas, and David Shallcross. "Practical Issues and Algorithms for Analyzing Terrorist Networks". In: *Proceedings of the Western Simulation MultiConference* (2002).

[71] Cecile Daniel, Angelo Furno, and Eugenio Zimeo. "Cluster-based Computation of Exact Betweenness Centrality in Large Undirected Graphs". In: *2019 IEEE International Conference on Big Data (Big Data)*. IEEE. 2019, pp. 603–608.

[72] Cecile Daniel, Angelo Furno, Lorenzo Goglia, et al. "Fast cluster-based computation of exact betweenness centrality in large graphs". In: *Journal of Big Data* 8.1 (2021), pp. 1–39.

[73] Robert W. Floyd. "Algorithm 97: Shortest path". In: *Commun. ACM* 5.6 (1962), p. 345. DOI: 10.1145/367766.368168.

[74] Ulrik Brandes. "A faster algorithm for betweenness centrality". In: *Journal of Mathematical Sociology* 25.163 (2001).

[75] P. Suppa and E. Zimeo. "A Clustered Approach for Fast Computation of Betweenness Centrality in Social Networks". In: *2015 IEEE International Congress on Big Data*. June 2015, pp. 47–54. DOI: 10.1109/BigDataCongress.2015.17.

[76] Yong Li, Wenguo Li, Yi Tan, et al. "Hierarchical Decomposition for Betweenness Centrality Measure of Complex Networks". In: *Sci. Rep. 7*. 2017.

[77] Albert-László Barabási and Réka Albert. "Emergence of Scaling in Random Networks". In: *Science* 286.5439 (Oct. 1999), pp. 509–512. ISSN: 1095-9203. DOI: 10.1126/science.286.5439.509.

[78] David A Bader and Kamesh Madduri. "Parallel algorithms for evaluating centrality indices in real-world networks". In: *Parallel Processing, 2006. ICPP 2006. International Conference on*. IEEE. 2006, pp. 539–550.

[79] Kamesh Madduri, David Ediger, Karl Jiang, et al. "A faster parallel algorithm and efficient multithreaded implementations for evaluating betweenness centrality on massive datasets". In: *Parallel & Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on*. IEEE. 2009, pp. 1–8.

[80] Alexander van der Grinten and Henning Meyerhenke. "Scaling Betweenness Approximation to Billions of Edges by MPI-based Adaptive Sampling". In: May 2020, pp. 527–535. DOI: 10.1109/IPDPS47924.2020.00061.

[81] Michele Borassi and Emanuele Natale. "KADABRA is an ADaptive Algorithm for Betweenness via Random Approximation". In: *Journal of Experimental Algorithmics* 24 (Apr. 2016). DOI: 10.1145/3284359.

[82] Zhiao Shi and Bing Zhang. "Fast network centrality analysis using GPUs". In: *BMC Bioinformatics* 12.1 (2011), p. 149. ISSN: 1471-2105. DOI: 10.1186/1471-2105-12-149.

[83] Nicolas Kourtellis, Gianmarco De Francisci Morales, and Francesco Bonchi. "Scalable online betweenness centrality in evolving graphs". In: *2016 IEEE 32nd International Conference on Data Engineering (ICDE)* 00 (2016), pp. 1580–1581. DOI: doi.ieeecomputersociety.org/10.1109/ICDE.2016.7498421.

[84] Ranjan Behera, Debadatta Naik, Dharavath Ramesh, et al. "MR-IBC: MapReduce-based incremental betweenness centrality in large-scale complex networks". In: *Social Network Analysis and Mining* 10 (Dec. 2020). DOI: 10.1007/s13278-020-00636-9.

[85] Kshitij Shukla, Sai Regunta, Sai Harsh, et al. "Efficient parallel algorithms for betweenness- and closeness-centrality in dynamic graphs". In: June 2020, pp. 1–12. DOI: 10.1145/3392717.3392743.

[86] Ulrik Brandes and Christian Pich. "Centrality estimation in large networks". In: *International Journal of Bifurcation and Chaos* 17.07 (2007), pp. 2303–2318.

[87] Robert Geisberger, Peter Sanders, and Dominik Schultes. "Better Approximation of Betweenness Centrality." In: *ALENEX*. SIAM. 2008, pp. 90–100.

[88] Matteo Riondato and Eli Upfal. "ABRA: Approximating Betweenness Centrality in Static and Dynamic Graphs with Rademacher Averages". In: 12.5 (2018). ISSN: 1556-4681. DOI: 10.1145/3208351. URL: https://doi.org/10.1145/3208351.

[89] R. Puzis, Y. Elovici, P. Zilberman, et al. "Topology manipulations for speeding betweenness centrality computation". In: *Journal of Complex Networks* 3.1 (Apr. 2014), pp. 84–112. ISSN: 2051-1329. DOI: 10.1093/comnet/cnu015.

[90] Ahmet Erdem Sariyüce, Kamer Kaya, Erik Saule, et al. "Graph manipulations for fast centrality computation". In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 11.3 (2017), p. 26.

[91] Miriam Baglioni, Filippo Geraci, Marco Pellegrini, et al. "Fast Exact Computation of betweenness Centrality in Social Networks". In: *International Conference on Advances in Social Networks Analysis and Mining, ASONAM 2012, Istanbul, Turkey, 26-29 August 2012*. 2012, pp. 450–456. DOI: 10.1109/ASONAM.2012.79.

[92] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, et al. "Fast unfolding of communities in large networks". In: *Journal of Statistical Mechanics: Theory and Experiment* 2008.10 (Oct. 2008), P10008. DOI: 10.1088/1742-5468/2008/10/p10008.

[93] Dóra Erdős, Vatche Ishakian, Azer Bestavros, et al. "A Divide-and-Conquer Algorithm for Betweenness Centrality". In: (2014). DOI: 10.1137/1.9781611974010.49.

[94] Sunil Kumar Maurya, Xin Liu, and Tsuyoshi Murata. "Graph Neural Networks for Fast Node Ranking Approximation". In: *ACM Trans. Knowl. Discov. Data* 15.5 (May 2021). ISSN: 1556-4681. DOI: 10.1145/3446217. URL: https://doi.org/10.1145/3446217.

[95] Changjun Fan, Li Zeng, Yuhui Ding, et al. "Learning to Identify High Betweenness Centrality Nodes from Scratch: A Novel Graph Neural Network Approach". In: CIKM '19. Beijing, China: Association for Computing Machinery, 2019, pp. 559–568. ISBN: 9781450369763. DOI: 10.1145/3357384.3357979. URL: https://doi.org/10.1145/3357384.3357979.

[96] Ryan A. Rossi and Nesreen K. Ahmed. "The Network Data Repository with Interactive Graph Analytics and Visualization". In: *AAAI*. 2015. URL: http://networkrepository.com.

[97] Jim Mcauley and Jure Leskovec. "Learning to discover social circles in ego networks". In: *NIPS* 1 (Jan. 2012), pp. 539–547.

[98] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. "Graph Evolution: Densification and Shrinking Diameters". In: *ACM Trans Knowledge Discov Data* 1 (Apr. 2006).

[99] Atousa Tajaddini, Geoffrey Rose, Kara M Kockelman, et al. "Recent Progress in Activity-Based Travel Demand Modeling: Rising Data and Applicability". In: *Models and Technologies for Smart, Sustainable and Safe Transportation Systems* (2020).

[100] Hong Zheng, Young-Jun Son, Yi-Chang Chiu, et al. "A primer for agent-based simulation and modeling in transportation applications". In: (2013).

[101] Kay W Axhausen, Andreas Horni, and Kai Nagel. *The multi-agent transport simulation MATSim*. Ubiquity Press, 2016.

[102] Jithin Raj, Hareesh Bahuleyan, and Lelitha Devi Vanajakshi. "Application of data mining techniques for traffic density estimation and prediction". In: *Transportation Research Procedia* 17 (2016), pp. 321–330.

[103] Nicholas G Polson and Vadim O Sokolov. "Deep learning for short-term traffic flow prediction". In: *Transportation Research Part C: Emerging Technologies* 79 (2017), pp. 1–17.