A guide book for the traveller on graphs full of blockages

Pierre Bergé

université PARIS-SACLAY

NNT : 2019SACLS480

Thèse de doctorat

UNIVERSITÉ PARIS SUD
Comprendre le monde, construire l'avenir

Algorithmes pour voyager sur un graphe contenant des blocages

Thèse de doctorat de l'Université Paris-Saclay
préparée à Université Paris-Sud

École doctorale n°580 Sciences et Technologies de l'Information et de la Communication (STIC)
Spécialité de doctorat: Informatique

Thèse présentée et soutenue à Gif-sur-Yvette, le 3 décembre 2019, par

M. PIERRE BERGÉ

Composition du Jury :

Cristina Bazgan Professeur, LAMSADE/Université Paris-Dauphine	Président du jury
Bruno Escoffier Professeur, LIP6/Sorbonne Université	Rapporteur
Ignasi Sau Valls Chargé de recherche, LIRMM/Université de Montpellier	Rapporteur
Stephan Westphal Professeur, TU Clausthal	Rapporteur
Olivier Bournez Professeur, LIX/Ecole Polytechnique	Examinateur
Yannis Manoussakis Professeur, LRI/Université Paris-Sud	Examinateur
Joanna Tomasik Professeur, LRI/CentraleSupélec	Directeur de thèse
Arpad Rimmel Professeur assistant, LRI/CentraleSupélec	Encadrant

université PARIS-SACLAY
ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Title : A guide book for the traveller on graphs full of blockages

Keywords : multi-terminal cuts, parameterized complexity, Canadian Traveller Problem, competitive analysis

Abstract : We study graphs with blockages seen as models of networks which are exposed to risk of failures. Among a large variety of graph-theoretic problems dealing with obstacles, we are interested in two particular categories. The problems of the first category consist in seeking the smallest blockage set which meets a certain criterion. Those in the second category aim at minimizing the distance traversed by a traveller on a graph where blockages may occur.

On the one hand, cut problems ask for the minimum set of vertices/edges which splits some elements of the graph. Their solutions, *i.e.* cuts, provide the optimal way to put blockages on a graph in order to produce a certain separation.

On the other hand, the Canadian Traveller Problem (CTP) looks for the optimal (s,t)-path in a weighted graph, where s and t are two distinct vertices, knowing that some edges of the graph may be obstructed. The traveller discovers that an edge is blocked when he visits one of its endpoints. The solutions for the CTP are online algorithms (strategies) which guide the traveller during his trip.

The first part of our work concerns cut problems. We perform a static analysis of the connectivity in undirected graphs. Then, a dynamic study is provided in the second part as we focus on the competitiveness of strategies for the CTP.

We study cut problems via the parameterized complexity framework. The cutset size p is taken as a parameter. Given a set of sources $\{s_1,\dots,s_k\}$ and a target t, we propose an algorithm which builds a small edge cut of size p separating at least r sources from t. This NP-complete problem is called Partial One-Target Cut. It belongs to the family of multiterminal cut problems. Our algorithm is fixed-parameter tractable (FPT) as its execution takes $2^{O(p^2)}n^{O(1)}$, where n is the input size. We also prove that the vertex version of this problem, which imposes cuts to contain vertices instead of edges, is W[1]-hard. Then, we design an FPT algorithm for the counting of minimum (S,T)-cuts, where S and T are two disjoint sets of vertices. It counts the minimum vertex (S,T)-cuts in time $2^{O(p\log p)}n^{O(1)}$. After using a polynomial reduction which transforms edge cuts into vertex ones, it counts minimum edge (S,T)-cuts as well.

We provide numerous results on the competitive ratio of both deterministic and randomized strategies for the CTP. The optimal ratio obtained for the deterministic strategies on general graphs is $2k+1$, where k is a given upper bound on the number of blockages. No randomized strategy performs better for now. We show that randomized strategies which do not use memory cannot improve the bound $2k+1$. In addition, we discuss the tightness of lower bounds on the competitiveness of randomized strategies. To complete the CTP analysis, we study a group of travellers, possibly equipped with telecommunication devices. We compute the distance competitive ratio for the team. Our deterministic strategy with full communication between travellers is optimal. Eventually, we focus on two families of graphs. A strategy dedicated to equal-weight chordal graphs is proposed while another one is built for graphs with small maximum (s,t)-cuts. Both strategies outperform the ratio $2k+1$.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France

université PARIS-SACLAY

université PARIS-SACLAY

ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre : Algorithmes pour voyager sur un graphe contenant des blocages

Keywords : problèmes de coupes, complexité paramétrée, problème du voyageur canadien, algorithmes on-line

Résumé : Nous étudions des problèmes NP-difficiles portant sur les graphes contenant des blocages. L'objectif est d'analyser la résilience des graphes face à de potentielles défaillances impactant un faible nombre d'arêtes ou de nœuds.
Dans un premier temps, nous nous demandons si des terminaux, *i.e.* certains noeuds du graphe fournis en entrée, peuvent être déconnectés avec un nombre d'éléments p. Cette question est formalisée en informatique théorique par des problèmes de coupes que l'on étudie via la complexité paramétrée. Concrètement, on souhaite mettre en places des algorithmes FPT pour ce type de problèmes, c'est-à-dire des algorithmes retournant une solution exacte en temps $f(p)n^{O(1)}$, où f est une fonction arbitraire, donc potentiellement exponentielle. Ce type d'algorithmes offre un temps d'exécution raisonnable lorsque le paramètre p prend de faibles valeurs.
Dans un second temps, nous évaluons l'impact d'éventuels blocages sur le graphe lorsque l'objectif est de relier deux nœuds, une source s et une cible t, avec une distance minimale. Cette problématique est modélisée par le problème du voyageur canadien. Au moment de s'élancer depuis la source s, le voyageur ne connaît pas l'identité des arêtes bloquées. Il les découvre en visitant une de leur extrémité. Ainsi, nous concevons des algorithmes on-line qui s'adaptent à la découverte d'arêtes bloquées lors du trajet.
Nous traitons les problèmes de coupes du point de vue de la complexité paramétrée. La taille p de la coupe est le paramètre. Étant donné un ensemble de sources $\{s_1, \ldots, s_k\}$ et une cible t, nous proposons un algorithme qui construit une coupe de taille au plus p séparant au moins r sources de t. Nous nommons ce problème NP-complet Partial One-Target Cut (POTC).
On propose un algorithme FPT pour ce problème. Il se base sur diverses techniques utilisées dans le contexte de la complexité paramétrée : coupes importantes, codage couleur et dérandomisation. Son temps d'exécution est $2^{O(p^2)}n^{O(1)}$. Nous prouvons également que la variante de POTC, où la coupe est composée de noeuds, est W[1]-difficile. De même, POTC paramétré par r est W[1]-difficile.
Notre seconde contribution est la construction d'un algorithme qui compte les coupes minimums entre deux ensembles S et T en temps $2^{O(p \log p)}n^{O(1)}$. Il améliore nettement le meilleur temps d'exécution connu jusqu'alors, en $2^{O(2^p)}n^{O(1)}$. Une procédure basée sur la programmation dynamique est proposée sur une structure appelée drainage, qui caractérise l'ensemble des coupes minimums entre S et T. L'algorithme compte les coupes minimums composées de nœuds. Cependant, une réduction arête-nœud montre qu'il peut facilement s'adapter au comptage de coupes composées d'arêtes. Nous mettons en évidence une borne inférieure en $2^{o(p)}$ qui montre que cet algorithme est dans le pire des cas très proche du temps d'exécution optimal.
Nous présentons ensuite plusieurs résultats sur le ratio de compétitivité des stratégies

déterministes et randomisées pour le problème du voyageur canadien.
Nos premiers résultats concernent les stratégies randomisées. Nous prouvons d'abord que celles n'utilisant pas de mémoire ne peuvent pas améliorer le ratio $2k+1$, qui est optimal pour les stratégies déterministes. Pour cela, nous construisons un ensemble d'instances graphe-blocages sur lesquelles la mémoire est nécessaire si l'on souhaite être plus performant que l'algorithme déterministe optimal.
Nous apportons également des éléments concernant les bornes inférieures de compétitivité de l'ensemble des stratégies randomisées : celles-ci ne peuvent atteindre le ratio $|E_*|+1$ où E_* est l'ensemble des blocages.
Puis, nous étudions la compétitivité en distance d'un groupe de voyageurs avec et sans communication. Nous mettons en évidence l'impact de la communication qui permet d'améliorer le ratio $2k+1$ avec un groupe de voyageurs. Cela montre qu'il vaut mieux utiliser une flotte de voyageurs communiquant tous entre eux, plutôt qu'un unique voyageur.
Enfin, nous nous penchons sur le ratio de compétitivité des stratégies déterministes pour certaines familles de graphes. Deux stratégies, avec un ratio inférieur à $2k+1$ sont proposées: une pour les graphes cordaux avec poids uniformes et l'autre pour les graphes où la taille de la plus grande coupe minimale séparant s et t est au plus k. La seconde a un ratio de compétitivité en $\sqrt{2}k+O(1)$ lorsque la taille de la plus graphe (s,t)-coupe minimale est supposée constante.
Ce dernier résultat apporte une lecture différente sur les bornes inférieures de compétitivité pour les stratégies déterministes. En effet, alors que la borne $2k+1$ suggère que chaque blocage coûte un ratio de 2, notre algorithme prouve que lorsque le nombre de blocages excède la taille des (s,t)-coupes minimales, ce coût diminue à $\sqrt{2}$.

Contents

Symbols

- $\emptyset$: the empty set.
- $\mathbb{N}$: the set of natural numbers.
- $\mathbb{N}^*$: the set of positive natural numbers.
- $|S|$: the cardinality of a finite set S.
- $f(p) = o(g(p))$: $\lim_{p \to +\infty} \frac{f(p)}{g(p)} = 0$.
- $f(p) = O(g(p))$: there are constants $N, C \geq 0$ such that $f(p) \leq Cg(p)$ when $p \geq N$.
- $f(p) = \Omega(g(p))$: there are constants $N, C \geq 0$ such that $g(p) \leq Cf(p)$ when $p \geq N$.
- $f(n, p) = O^*(g(p))$: there is a polynomial function poly such that $f(p) = O\left(\text{poly}(n, p)g(p)\right)$.

(a) Hondius Jocodus, *Freti Magellanici ac novi freti vulgo Le Maire exactissima delineatio*, 1635

(b) Schrämbl Franz Anton, *Karte der Magellanischen Strasse*, 1787

Figure 1: Evolution of the map of Magellan's strait from 1635 to 1787: the gradual discovery of blockages on maritime routes.[a]

[a]These maps are listed in the book of Mateo Martinic, *Cartografiía magallánica*, VIII, 1999. They can be freely retrieved on the website of the library of Princeton University: https://libweb5.princeton.edu.

Chapter 1

Introduction

Graph-theoretic concepts have many applications in different fields, such as telecommunication systems or transportation networks. A graph $G = (V, E)$ is a structure involving vertices V and edges $E \subseteq V \times V$ connecting pairs of vertices. The vertices may represent a set of items while the edges put in evidence a relationship between these items. For example, a graph may model a map where the vertices correspond to different locations. The existence of an edge $(u, v) \in E$ means that a street connects locations u and v. The concept of graph captures the information provided by a road map. Practical situations are characterized thanks to the diversity of graph types. Oriented edges (*directed* graphs) model one-way streets in this case. Weighted edges (*weighted* graphs) associate a distance with any edge/street.

We focus on the handling of blockages in graphs. Our study gathers the connectivity problems which ask for the number of blocking elements we need to put on a graph in order to obtain isolated subgraphs satisfying certain properties. For example, some questions are related to the minimum number of elements in the graph we need to remove to separate certain of its vertices. Such connectivity problems are very important as they allow us to assess the *resilience* of a network, *i.e.* its vulnerability to the blockages or failures that may appear on it. The results related to these problems have consequences on the study of real-world networks robustness.

Another question covered by the handling of blockages is the navigation of a traveller on graphs in which he may run across an obstacle. For example, one can imagine that some edges of a graph are blocked and any of these blockages is discovered when the traveller visits one of its endpoints. In this case, the objective would be then to perform the shortest walk between two different vertices, knowing that blockages can occur.

We treat these two facets in the handling of blockages in graphs. The first facet consists in seeking the smallest sets of edges/vertices to remove from graph G in order to meet a certain criterion on the components produced. From a practical point of view, this is equivalent to looking for the optimal blockage configuration which separates some vertices. In brief, we focus on *cut* problems and study their computational complexity. To do this, we devise exact algorithms and propose hardness proofs.

The second facet is the design of algorithms which bypass the blockages occurring on a weighted graph G. These algorithms are *online*, as they adapt to the discovery of blockages. Concretely, the traveller goes from a source s to a target t and discovers a blocked edge when he meets one of its endpoints. This situation is modeled by the Canadian Traveller Problem (CTP) [70]. More precisely speaking, we study not only the identification of the optimal blockages configurations to separate some vertices in a graph but also the minimization of the distance traversed by a traveller on a weighted graph full of obstacles.

Now, we detail our contributions on these two aspects, beginning with an overview of our work on cut problems. The classical MINIMUM (s, t)-CUT problem is solved in polynomial-time [49]. This is the decision problem asking for the smallest set of edges whose removal

separates vertices s and t. However, a large number of cut problems are NP-complete. As no polynomial-time exact algorithm may exist for such problems, their parameterized complexity stands as a natural question [45]. Indeed, many fixed-parameter tractable (FPT) algorithms have been proposed, *i.e.* exact algorithms with execution time $f(q)n^{O(1)}$, where q is a parameter, n the instance size, and f an arbitrary function.

The parameterized complexity of cut problems is treated in Chapter 2 which we start by reminding the state of the art on this topic (Section 2.1) before presenting our complexity results for two problems, introduced below.

The fixed-parameter tractability of multi-terminal cut problems has been intensively investigated over the years. The MULTICUT problem asks for the smallest set of edges which separates k pairs of terminals, *i.e.* (s_1, t_1), (s_2, t_2),...,(s_k, t_k). It has been proven FPT parameterized by the solution size [27, 63]. In this study, we analyze the complexity of one of its natural variants, PARTIAL ONE-TARGET CUT (POTC) which consists in finding the smallest set of edges separating a number r of sources among set $\{s_1, \ldots, s_k\} \subseteq V$ from a single target t. The parameterized complexity of POTC for several parameters combinations is given in Section 2.2. Our main contribution to this problem is an FPT algorithm executed in time $O^*\left(2^{O(p^2)}\right)$, where p is the solution size [17]. It mixes different techniques: the sampling of *important cuts* and color-coding techniques associated with *edge passes*, a concept we devised.

Then, we worked on the counting version of MINIMUM (s,t)-CUT, *i.e.* computing the number of minimum (s,t)-cuts in a graph G. Our objective was to propose an efficient FPT algorithm for this problem, parameterized by the size p of the minimum (s,t)-cuts, as it is #P-complete. First, we built a $O^*\left(2^{O(p^2)}\right)$-time algorithm for the counting of minimum edge (s,t)-cuts [16]. Second, we proposed another algorithm extending and outperforming the first one [12], as it not only counts both minimum edge and vertex (s,t)-cuts but also runs in time $O^*\left(2^{O(p \log p)}\right)$. Both algorithms use the concepts we define, such as the *drainage* and the *dry instances* which characterize minimum (s,t)-cuts. We believe they can be reused for other problems involving minimum (s,t)-cuts. The vertex cut problem required the development of the *local drainage* concept. Incidentally, this concept made the algorithm complexity decrease. The details of these contributions are provided in Section 2.3.

Chapter 3 of this study deals with the design of online algorithms for the CTP. The CTP is PSPACE-complete [70]. The objective is to make a traveller walk from s to t on an undirected weighted graph $G = (V, E, \omega)$ in the most efficient way despite the existence of blocked edges $E_* \subsetneq E$. Edge weights are given by the function $\omega : E \to \mathbb{Q}^+$. There is a parameterized version k-CTP, where k is an upper bound of the number of blocked edges: $|E_*| \leq k$. The traveller does not know which edges are blocked when he begins his walk. He discovers a blocked edge when he visits one of its endpoints. A solution to the k-CTP is an online algorithm, called a *strategy*. The quality of any online algorithm is usually assessed via the competitive analysis [2]. Indeed, the competitive ratio of a strategy is the quotient between its performance and the performance of an offline optimal algorithm. For the k-CTP, it is known from the literature that no deterministic strategy achieves a competitive ratio better than $2k + 1$ for general graphs [75].

We begin Chapter 3 with the state of the art on the CTP (Section 3.1). The competitive ratio of strategies for general graphs is first studied and, second, we propose competitive strategies for certain families of graphs. We present our proposition to divide the results on the competitive ratio for the CTP in two parts (Section 3.2). A distinction is made between *global competitiveness* (Section 3.3), where the competitive ratio on general graphs is discussed, and *local competitiveness* (Section 3.4), where we study certain families of graphs.

As the global competitiveness of deterministic strategies has been completely treated in the literature, we focus on the competitive ratio of randomized strategies for general graphs.

Our main result is that no randomized memoryless strategy can defeat the optimal ratio $2k+1$ of deterministic strategies for the k-CTP [14]. A memoryless strategy makes decisions which do not depend on its previous trip. A major open question is whether a randomized strategy can go below ratio $2k+1$ for general graphs, a consequence of this result is that such a strategy would not be memoryless. Our second contribution concerns the initial CTP version without parameter k given. It is known that no randomized strategy has a competitive ratio less than $|E_*|+1$. We prove that this bound cannot be reached [15], using linear inequality systems and Farkas' lemma.

Eventually, we focus on the global distance competitiveness of strategies guiding multiple travellers on the graph [13]. The objective is to measure the benefits from communication between several Canadians. As the travellers may send and receive messages from their teammates, we treat different communication levels. An optimal deterministic strategy, MULTI-ALTERNATING, is proposed when the travellers are authorized to share information about the blocked edges discovered at any moment of the trip.

Then, we devise two deterministic strategies for the k-CTP dedicated to families of graphs. The first one, DETOUR, is customized for graphs, where value k is greater than the size μ_{max} of the largest minimal (s,t)-cut [19]. The competitive ratio of DETOUR, when $\mu_{\mathsf{max}} < k$, is $2\mu_{\mathsf{max}} + \sqrt{2}(k-\mu_{\mathsf{max}})+1$, strictly less than $2k+1$. We also prove that the competitive ratio of the best existing strategies, REPOSITION and COMPARISON, is $2k+1$ in this situation. Furthermore, the competitive ratio of DETOUR when $\mu_{\mathsf{max}} \geq k$ is $2k+1$. Therefore, DETOUR offers better guarantees than the deterministic strategies known up to now.

The second one, CHORD-REP [18], has a competitive ratio $\frac{2k+\omega_{\mathsf{opt}}}{\omega_{\mathsf{opt}}}$ on equal-weight chordal graphs, where ω_{opt} is the optimal offline cost. All edges are weighted with one on equal-weight graphs. The case $\omega_{\mathsf{opt}} = 1$ being trivial, the competitive ratio of CHORD-REP goes below $k+1$ when $\omega_{\mathsf{opt}} \geq 2$. It becomes a constant ratio, $O(1)$, for values $\omega_{\mathsf{opt}} = \Omega(k)$.

After the summary of our contributions (Chapter 4), possible lines for future research are discussed in Chapter 5. The advances presented in this study for both the parameterized complexity of cut problems and the CTP are just waiting to be improved or extended.

Many questions emerge from our FPT results on cut problems: for example, the existence of a polynomial kernel for POTC or an FPT algorithm for the more general problem PARTIAL MULTICUT.

One of our objectives for the CTP has not been met: we do not know whether a randomized strategy can outperform the competitive ratio $2k+1$ for general graphs. We explain how local assessments could help us to answer this question in the future.

Chapter 2

Identifying small blockage sets in graphs

This chapter introduces our contributions on NP-hard cut problems. Our objective is to identify the smallest sets of either edges or vertices which separate *terminals* in the graph. Terminals are certain vertices in the input of these problems. Assuming P$\neq$NP, no polynomial-time exact algorithm can be designed for such problems. This is why we focus on their parameterized complexity and design FPT algorithms. Concretely, we assume that the size of the cuts which we are seeking is small, so that the running time of these FPT algorithms is reasonable, even for instances of large size.
In Section 2.1, we introduce the parameterized complexity of decision and counting cut problems. All problems we study are defined there. An overview of the results reported in the literature for these problems is also provided. In Section 2.2, we introduce the PARTIAL ONE-TARGET CUT problem, which asks for one of the smallest set of edges separating certain sources from a single target [17]. We give our contribution to this problem: its complexity for different parametrizations is analyzed. In particular, we propose an FPT algorithm, parameterized by the cutset size. In Section 2.3, we work on the counting of both minimum edge and vertex (S, T)-cuts. In particular, we propose an FPT algorithm parameterized by the size of the minimum (S, T)-cuts improving the running time obtained up to now [12, 16].

2.1 State of the art

The parameterized complexity framework is summarized in Section 2.1.1. The definitions of the complexity classes FPT, W[1]-hard, and XP are reminded and commented for both decision and counting problems. Then, we highlight parameterized results on cut problems in Section 2.1.2 which are related to our contributions. Eventually, in Section 2.1.3, we present the crucial concepts associated with the notions of cuts and connectivity: Menger's and min-cut max-flow theorems, source and target sides, and important separators.

2.1.1 Parameterized complexity

Downey and Fellows [45] formulated the foundations of the parameterized complexity in 1999. Our summary below is also based on the more recent books of Cygan *et al.* [37] and Niedermeier [68]. We distinguish decision and counting problems, as their respective parameterized complexity classes do not have necessarily the same name and the same definition.
The idea behind the parameterized framework is to refine the classical complexity analysis as it introduces a hierarchy of the NP-complete problems in function of *parameters*. For

example, the parameter can be the maximum degree of the input graph, or the solution size, etc. Given a parameter p, some problems are FPT parameterized by p, while some of them are W[1]-hard, *i.e.* unlikely to be FPT.

Decision problems

Under the hypothesis P≠NP, NP-hard problems cannot be solved in time expressed as a polynomial function of the instance size. Nevertheless, if some parameters are associated with problem instances and their values are small, efficient algorithms solving these problems may be designed.

A *fixed parameter tractable* (FPT) algorithm solves a parameterized decision problem in time $O\left(f(p)n^{O(1)}\right) = O^*(f(p))$, where n is the instance size, p is a parameter of the problem, and f is an arbitrary function. A problem is FPT if an exact solution can be found with an FPT algorithm. A problem may be studied for different parameters $p_1, p_2, \ldots, p_\ell$. To disambiguate notations, a problem is FPT$\langle p_1 \rangle$ if it can be solved with running time $O^*\left(f(p_1)\right)$. A problem may be FPT for one parameter, but not FPT for another one. For example, there is an FPT algorithm solving CLIQUE parameterized by the maximum degree of the input graph [68]. Nevertheless, CLIQUE is unlikely FPT parameterized by the solution size, *i.e.* the size of the clique we are looking for [68].

From the parameterized point of view, a problem Q_1 with parameter p_1 can be reduced to problem Q_2 with parameter p_2 if and only if (iff) there is an FPT$\langle p_1 \rangle$ algorithm g such that:

- for any instance I_1 of Q_1, algorithm g builds an instance $(I_2, p_2) = g(I_1, p_1)$ of Q_2,
- instance (I_2, p_2) verifies $|I_2| \leq f\left(p_1\right) |I_1|^{O(1)}$ and $p_2 \leq \widehat{f}(p_1)$, with arbitrary f and $\widehat{f}$,
- pair (I_1, p_1) is a yes-instance for Q_1 iff (I_2, p_2) is a yes-instance for Q_2.

The fact that Q_1 reduces to Q_2 in the parameterized way is written $Q_1 \leq_{\text{fpt}} Q_2$. If Q_2 has been proven FPT, then Q_1 is also FPT.

As FPT is in a certain sense the parameterized version of class P, the equivalent of NP is denoted W[1]. Formally, class W[1] contains all problems that can be reduced to WEIGHTED 2-CNF SAT (Definition 2.1 below), which is a parameterized variant of SAT.

Definition 2.1 (Weighted 2-CNF-SAT [68])**.**
Input: *Boolean formula F in CNF with at most two variables per clause, parameter k.*
Question: *Is there a truth assignment with exactly k variables that are set* true*?*

Any FPT problem is necessarily in W[1]. The equation FPT≠W[1] is generally admitted, as its opposite FPT=W[1] would imply that 3-SAT is solvable in time $2^{o(n)}$ and would contradict the Exponential Time Hypothesis (ETH) [68]. A problem Q is W[1]-hard if:

$$\text{WEIGHTED 2-CNF SAT} \leq_{\text{fpt}} Q.$$

Under the hypothesis FPT≠W[1], a problem Q is not FPT if we reduce a W[1]-hard problem to it. CLIQUE and INDEPENDENT SET are two well-known W[1]-hard problems parameterized by the solution size [37, 68]. Given a NP-hard problem and a parameter, a natural question is whether either an FPT algorithm exists or a W[1]-hardness reduction is identified.

A problem is XP if it can be solved in time $O(n^{f(p)})$ with parameter p. Obviously, any FPT problem is XP. Generally, problems parameterized by the solution size admit an XP algorithm, as BRUTE FORCE enumerates at most $\binom{n}{p} = O(n^p)$ solutions.

Counting problems

The counting version of a decision problem aims to determine the number of solutions. For example, VERTEX COVER asks whether there is a vertex cover of size k. Its counting version asks how many vertex covers of size k exist.
Class #P contains the counting problems such that their decision version is in NP [74]. No #P-complete problem can be solved in polynomial time unless P=NP. The complexity of counting problems is also studied via the parameterized framework [36, 47]. A relevant question about a #P-complete problem is whether an FPT algorithm counts all its solutions. For example, with graphs G and H as input, FPT algorithms counting the number of occurrences of H as a subgraph of G have been devised [4, 55, 76].
The complexity class #W[1] points out the parameter intractability of a counting problem. A problem is #W[1]-hard if there is a parameterized counting reduction (a parameterized reduction preserving the number of solutions) from a known #W[1]-hard problem to it, *i.e.* a parameterized reduction preserving the number of solutions. As we suppose FPT≠W[1], #W[1]-hard problems are unlikely solvable in FPT time. Even if a decision problem is FPT, its counting version may be #W[1]-hard [36]. For example, counting the number of matchings of size p in an undirected graph is #W[1]-hard [35].

2.1.2 Results on cut problems

Many results dealing with the classical and the parameterized complexity of cut problems are listed. We remind the definition of the well-known problem MIN-(S,T)-CUT, asking for the smallest set of edges separating sets S and T. We also present NP-complete multi-terminal cut problems, which are more general. We specify some problems for which questions on their parameterized complexity are still open. As announced, our contributions in Section 2.2 concern one of these problems, PARTIAL ONE-TARGET CUT [17].
Then, we outline the state of the art for the counting of minimum (S,T)-cuts. We list the parameters for which the complexity of this counting problem has been determined. We focus on parameter p, the size of the minimum (S,T)-cuts. Two algorithms are examined: the natural brute force algorithm which is XP for parameter p and an FPT$\langle p\rangle$ one deduced from the following works [20, 62]. Our contribution, presented in Section 2.3, is the design of an algorithm improving the running time of these two methods [12].

Multi-terminal cut problems

Cuts in graphs have been fervently studied since Menger's theorem [65], reminded in Section 2.1.3, and Ford-Fulkerson's algorithm [49]. Problem MIN-(S,T)-CUT consists in determining the minimum number of edges needed to separate sets S and T (Definition 2.2 below). Min-cut max-flow theorem states that the minimum (S,T)-cut size and the maximum flow of a network are equal. The computation of a maximum flow with the polynomial-time Ford-Fulkerson's algorithm uncovers one of the minimum (S,T)-cuts, regardless of whether the graph is directed or not, weighted or not.

Definition 2.2 (Min-(S,T)-cut [49])**.**
Input: *Graph $G=(V,E)$, sets $S,T\subseteq V$, $S\cap T=\emptyset$, positive integer p.*
Question: *Is there a cutset $X\subseteq E$, $|X|\leq p$ such that no path connects S and T in graph G deprived of edges X?*

Many generalizations of MIN-(S,T)-CUT have been proposed and proven NP-complete. One of them is MULTICUT [40] which aims to separate k pairs of terminals, $(s_1,t_1),\ldots,(s_k,t_k)$, in an undirected graph with a minimum cut. There are two versions of this problem: EDGE MULTICUT when the cut is made of edges and VERTEX MULTICUT when it is made of vertices. The edge version is given in the following definition.

Definition 2.3 (Edge Multicut [40])**.**
Input: *Graph $G = (V, E)$, pairs of terminals $(s_1, t_1), \ldots, (s_k, t_k)$, positive integer p.*
Question: *Is there a cutset $X \subseteq E$, $|X| \leq p$, which disconnects all pairs of terminals (s_i, t_i), $i \in \{1, \ldots, k\}$?*

Both versions have been studied with the parameterized complexity tools [45]. Marx and Razgon [63] and also Bousquet *et al.* [27] proved that VERTEX MULTICUT is FPT parameterized by the cutset size, *i.e.* FPT$\langle p \rangle$. EDGE MULTICUT is also FPT$\langle p \rangle$, as it can be reduced to VERTEX MULTICUT [63]. On directed graphs, MULTICUT is unlikely to be FPT parameterized by the cutset size [63], even when the number of pairs k is fixed and greater than four [71]. For two terminal pairs ($k = 2$), MULTICUT is FPT$\langle p \rangle$ on directed graphs [34]. The case of three terminal pairs remains open.
A generalization of MULTICUT is called PARTIAL MULTICUT [59]. Its objective is to find one of the smallest sets of edges/vertices separating at least r pairs of terminals among k pairs. We distinguish EDGE PARTIAL MULTICUT (Definition 2.4 below) and VERTEX PARTIAL MULTICUT. The edge version can be reduced to the vertex one, as for MULTICUT.

Definition 2.4 (Edge Partial Multicut [59])**.**
Input: *Graph $G = (V, E)$, pairs of terminals $(s_1, t_1), \ldots, (s_k, t_k)$, positive integers p and r, $r \leq k$.*
Question: *Is there a cutset $X \subseteq E$, $|X| \leq p$, which disconnects at least r pairs of terminals?*

PARTIAL MULTICUT is naturally NP-complete as it becomes MULTICUT for $r = k$. It was defined in [59] whose authors proposed an algorithm on trees with approximation factor $\frac{8}{3} + \varepsilon$ for any $\varepsilon > 0$. For general graphs, an algorithm with factor $O(\log^2(n) \log \log n)$ was designed [54]. An open question is whether PARTIAL MULTICUT is FPT, for parametrizations involving p, r, and k.
PARTIAL MULTICUT models a transportation problem. Each vehicle of a fleet counting k of them, initially located in s_i, heads off to its target t_i. The fleet operator wants to know whether p blocked edges (or vertices) may prevent r among these vehicles from reaching their targets, where $0 \leq p$ and $0 \leq r \leq k$. We define (as we did in [17]) a special case of this problem, called PARTIAL ONE-TARGET CUT (POTC), where all vehicles have the same target, *i.e.* $t_i = t$. Its edge version is described below.

Definition 2.5 (Edge Partial One-Target Cut [17])**.**
Input: *Graph $G = (V, E)$, sources $S = \{s_1, \ldots, s_k\}$, target t, positive integers p and $r \leq k$.*
Question: *Is there a cutset $X \subseteq E$, $|X| \leq p$, which disconnects at least r sources from target t?*

When $r = k$, POTC is solvable in polynomial time as a minimum (S, t)-cut of size less than p is a solution. In Section 2.2, we prove the NP-completeness of this problem in general and study its parameterized complexity for all combinations involving p, r, and k.

Counting minimum (S, T)-cuts

The issue of counting minimum cuts in graphs has several practical applications. Indeed, the number of minimum cuts is an important factor for the network reliability analysis [5, 6, 8, 66]. Thereby, the probability that a stochastic graph is connected is related to the number of minimum cuts [6]. Furthermore, cuts on planar graphs are used for image segmentation [28]. An image is seen as a planar graph where vertices represent pixels and edges connect two neighboring pixels if they are similar. Counting minimum cuts provides an estimation of the number of segmentations.
We focus on the problem of counting minimum edge (S, T)-cuts in undirected graphs $G = (V, E)$, $S, T \subseteq V$. We call it COUNTING MIN-(S, T)-CUTS (Definition 2.6 below) as it

is the counting variant of MIN-(S,T)-CUT. Ball and Provan showed in [7] that COUNTING MIN-(S,T)-CUTS is unlikely solvable in polynomial time as it is #P-complete. They also devised a polynomial-time algorithm for COUNTING MIN-(S,T)-CUTS on planar graphs [6]. Bezáková and Friedlander [21] generalized it with an $O(n\mu + n\log n)$-time algorithm on weighted planar graphs, where μ is the length of the shortest (s,t)-paths. For general graphs, some upper bounds on the number of minimum cuts have been given [31] in function of parameters such as the radius, the maximum degree, etc. Two fixed-parameter tractable (FPT) algorithms have been proposed for COUNTING MIN-(S,T)-CUTS. Bezáková *et al.* [20] built an algorithm for both directed and undirected graphs with small treewidth λ; its time complexity is $O(2^{3\lambda}\lambda n)$. Moreover, Chambers *et al.* [29] designed an algorithm for directed graphs embedded on orientable surfaces of genus g: its execution time is $O(2^g n^2)$.

Definition 2.6 (Counting min-(S,T)-cuts)**.**
Input: *Undirected graph $G=(V,E)$, sets of vertices $S,T \subsetneq V$, $S\cap T=\emptyset$.*
Output: *The number of minimum edge (S,T)-cuts.*

Now let us discuss the complexity of COUNTING MIN-(S,T)-CUTS, parameterized by the size p of minimum (S,T)-cuts. A trivial brute force XP algorithm computes the number $C(\mathcal{I})$ of minimum (S,T)-cuts in time $n^{O(p)}$ by enumerating all edge sets of size p and picking up those which are (S,T)-cuts. More efficient exponential algorithms exist, as the one of Nagamochi *et al.* [66], in time $O\left(pn^2 + pnC(\mathcal{I})\right)$.
An FPT$\langle p\rangle$ algorithm can be deduced from the results in two articles [20, 62] and its execution time is $O^*\left(2^{H(p)}\right)$, where $H(p)=\Omega\left(\frac{2^p}{\sqrt{p}}\right)$. The treewidth reduction theorem established by Marx *et al.* in [62] says that there is a linear-time reduction transforming graph G into another graph G' which preserves the (s,t)-cuts of size p and its treewidth $\tau(G')$ verifies $\tau(G')=2^{O(p)}$. After this transformation, the number of minimum (S,T)-cuts of G' is obtained thanks to the algorithm given in [20]. The overall time taken with this method is $O^*\left(2^{2^{O(p)}}\right)$. In Section 2.3, we design FPT$\langle p\rangle$ algorithms for COUNTING MIN-(S,T)-CUTS which improve this exponential factor.

2.1.3 Methods to solve cut problems

We introduce combinatorial concepts commonly used to treat cuts, as we refer to them in the following sections.
First, we put in evidence notation and definitions. Second, we remind Menger's theorem and its consequences. We also explain why the *important cuts* are a powerful tool to tackle cut problems. Eventually, we show how to reduce in polynomial time an edge cut problem to its vertex version.

Cuts and paths

Our work mainly concerns undirected graphs $G=(V,E)$, where $n=|V|$ and $m=|E|$. For any set of vertices $U\subseteq V$, we denote by $E[U]$ the set of edges of G with two endpoints in U. Let $G[U]$ be the subgraph of G induced by U, $G[U]=(U,E[U])$. We denote by $G\backslash U$ the graph deprived of vertices in U: $G\backslash U = G[V\backslash U]$. Similarly, for any set of edges $E'\subseteq E$, the graph G deprived of E' is denoted by $G\backslash E'=(V,E\backslash E')$.
A *simple path* is a sequence of pairwise different vertices $v_1\cdot v_2\cdot v_3\cdots v_i\cdot v_{i+1}\cdots v_N$, where two successive vertices (v_i,v_{i+1}) are adjacent in G: $(v_i,v_{i+1})\in E$. To improve readability, we abuse notations: $v_1\in Q$ and $(v_1,v_2)\in Q$ mean that vertex v_1 and edge (v_1,v_2) are on path Q, respectively. Any (S,T)-path $Q=v_1\cdot v_2\cdots v_i\cdot v_{i+1}\cdots v_N$, $v_1\in S$ and $v_N\in T$, has a natural orientation from S to T. Therefore, we say that the *ancestors* of v_i are the vertices $v_1,\ldots,v_{i-1}$ and its *descendants* are $v_{i+1},\ldots,v_N$. Its *predecessor* is v_{i-1} and its *successor* is v_{i+1}. These notions are also naturally defined for edges: the ancestors of (v_i,v_{i+1}) are

all the edges between v_{i+1} and t, following the orientation of the (S,T)-path. Two paths P and Q are *vertex-disjoint* if there is no vertex v such that $v \in P$ and $v \in Q$.
Cut problems usually consist in finding the smallest set of edges/vertices X which separates vertices of the graph $G\backslash X$ in a certain way. Given a set of sources S and targets T, set $X \subseteq V$ is a vertex (S,T)-cut if no path connects a vertex from S with a vertex from T in $G\backslash X$. An (S,T)-cut X is said *minimal* if there is no (S,T)-cut $X' \subsetneq X$. It is *minimum* if there is no (S,T)-cut X' such that $|X'| < |X|$. Any minimum (S,T)-cut is minimal. From now on, we focus only on the (S,T)-cuts which are minimal. So, the minimality of (S,T)-cuts is implicit later on.
For $U \subseteq V$, we denote by $\delta(U)$ the set of edges with exactly one endpoint in U. For any (S,T)-cut X, let $R(X,S)$ contains the *source side* of X, *i.e.* the vertices reachable from S in $G\backslash X$. Its *target side* $R(X,T)$ contains the vertices reachable from T in $G\backslash X$. One observes that $\delta(R(X,S)) = X = \delta(R(X,T))$. We also fix $R^+(X,S) = R(X,S) \cup X$ and $R^+(X,T) = R(X,T) \cup X$, the augmented source and target sides of X.

Menger's theorem and its consequences

Menger's theorem [65] states that the size of the minimum edge (S,T)-cuts in an undirected graph is equal to the cardinality of the largest set containing edge-disjoint (S,T)-paths.

Theorem 2.1 (Menger's theorem for edge (S,T)-cuts [65])**.** *If p denotes the size of the minimum edge (S,T)-cuts in G, then the largest set of edge-disjoint simple (S,T)-paths of G contains p paths.*

The vertex version of Theorem 2.1 says that the sizes of the minimum vertex (S,T)-cut and of the largest set of vertex-disjoint (S,T)-paths are equal. However, we pursue this introductory paragraph by considering the edge version only, as our overview can be extended in a straightforward manner to the vertex version.
Menger's theorem is generalized by the min-cut max-flow theorem [49]. For this reason, a set of edge-disjoint (S,T)-paths of size p is computed in time $O(mp)$ with p rounds of Ford-Fulkerson's algorithm. Let $\mathcal{Q} = \{Q_1, Q_2, ..., Q_p\}$ denote an arbitrary collection of edge-disjoint (S,T)-paths. Paths Q_i, $i \in \{1, \ldots, p\}$, are called *Menger's paths*.
We present helpful consequences of Menger's theorem. These observations will be used to design our algorithms. The first one is stated in the following lemma. We prove that an edge of any minimum (S,T)-cut belongs to a single Menger's path.

Lemma 2.1. *Let X be a minimum (S,T)-cut. For any edge $x \in X$, there is a unique path $Q_j \in \mathcal{Q}$ such that $x \in Q_j$. Conversely, each Menger's path contains exactly one edge from X. All ancestors of x on Q_j belong to $R(X,S)$ while its descendants belong to $R(X,T)$.*

Proof. We suppose *ad absurdum* that X is a minimum (S,T)-cut and one of its edges $x \in X$ does not belong to any Menger's path. As $|\mathcal{Q}| = p$, at least one Menger's path is not interrupted in $G\backslash X$ and connects S to T. So, X is not an (S,T)-cut, which is a contradiction. Now, we suppose that X contains two edges x_1 and x_2 which are on the same Menger's path, say Q_j. As $|X| = |\mathcal{Q}|$, this means that another path Q_i in $\mathcal{Q}$ has no element of X. The same contradiction occurs.
As path Q_j only contains $x \in X$, none of the edge ancestors of x can be in cut X. Therefore, all vertices from S to x are reachable from S in $G\backslash X$. For the same reason, the vertex descendants of x are reachable from T in $G\backslash X$. □

For a minimum (S,T)-cut X and one of its edges $x \in X$, we denote by $\sigma(x)$ the *signature* of x, *i.e.* the path in $\mathcal{Q}$ which contains x. Lemma 2.1 ensures that this path exists and is unique. We abuse notations: for any $B \subseteq X$, notation $\sigma(B)$ refers to the set of Menger's paths which contain edges in B, $\sigma(B) = \{Q_i \in \mathcal{Q} : B \cap Q_i \neq \emptyset\}$.

An arbitrary collection of Menger's paths $\mathcal{Q}$ can also be determined in time $O(mp)$ for the minimum vertex (S,T)-cuts. Lemma 2.1 is valid for vertex cuts X' too, so the signature $\sigma(x')$ refers to the Menger's path containing $x' \in X'$.

Important cuts

In 2006, Marx [61] put in evidence a certain set of (S,T)-cuts called *important cuts*. For two (S,T)-cuts X and X', we say that X' *dominates* X if $|X'| \leq |X|$ and $R(X,S) \subsetneq R(X',S)$. Conversely, cut X' is *closer* than X if $|X'| \leq |X|$ and $R(X',S) \subsetneq R(X,S)$. An (S,T)-cut X is *important* if there is no other (S,T)-cut X' which dominates X.

Definition 2.7. *An (S,T)-cut X is important if there is no other (S,T)-cut X' such that $|X'| \leq |X|$ and $R(X,S) \subsetneq R(X',S)$.*

Intuitively, an important (S,T)-cut is such that there is no other cut smaller in size which is closer to T. Letter Y is generally used to denote important cuts.
For any S,T, the important (S,T)-cuts satisfy a powerful property: the number of important (S,T)-cuts of size at most p depends only on p, not on the input size n [33]. Logically, this concept sounds as an efficient tool for identifying FPT algorithms on cut problems.

Lemma 2.2 (Important cuts enumeration [33])**.** *For disjoint sets of vertices S and T, there are at most 4^p important cuts Y with $|Y| \leq p$ and they can all be listed in time $4^p n^{O(1)}$.*

Closest (S,T)-cuts refer to the reverse definition of important (S,T)-cuts. As mentioned in [58], an (S,T)-cut X is *closest* if there is no other (S,T)-cut X' closer than X. Letter Z is generally used to denote closest cuts. On undirected graphs, a closest (S,T)-cut is also an important (T,S)-cut. Both the minimum important (S,T)-cut and the minimum closest (S,T)-cut are unique. Figure 2.1 gives an example of graph G with two (S,T)-cuts X_1 and X_2, where $S = \{s_1, s_2\}$ and $T = \{t\}$. Cut X_1 is not closest as the edges incident to S form a cut Z_1 smaller than X_1 and $R(Z_1,S) \subseteq R(X_1,S)$. Cut X_2 is closest because there is no cut with at most three edges whose reachable set of vertices is contained in $R(X_2,S)$.

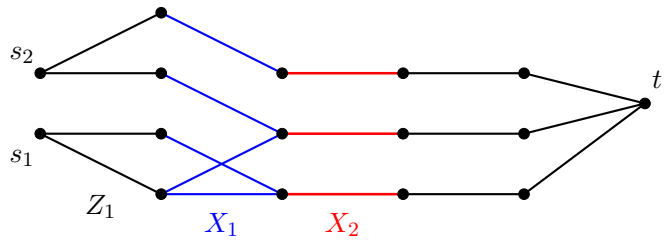


Figure 2.1: Illustration of closest (S,T)-cuts: X_2 is closest whereas X_1 is not.

Lemma 2.3 (Unicity of minimum important and closest cuts [58, 61])**.** *There is a unique minimum important (S,T)-cut Y and a unique minimum closest (S,T)-cut Z.*

Lemmas 2.2 and 2.3 are valid for both edge and vertex (S,T)-cuts. The computation of both the minimum important (S,T)-cut and the minimum closest (S,T)-cut consists in executing p rounds of Ford-Fulkerson's algorithm [34]. Indeed, they are obtained from the residual graph. It follows:

Lemma 2.4 (Computation time of minimum important/closest cuts [34])**.** *The minimum important (S,T)-cut and the minimum closest (S,T)-cut are obtained in $O(mp)$, where p is the size of minimum (S,T)-cuts.*

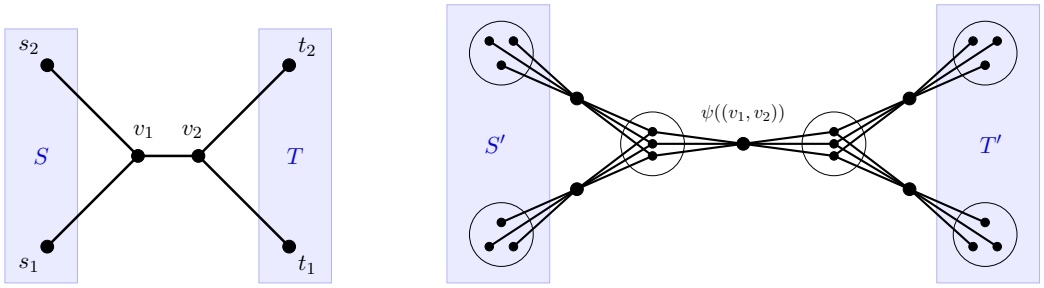
The minimum important and closest (S,T)-cuts are essential tools of our FPT algorithms dedicated to cut problems.

Relationship between edge and vertex (S,T)-cuts

We develop an approach to reduce in polynomial time edge cut problems to their vertex versions, both of them parameterized by the cutset size. Indeed, there is a transformation $\psi : (G,S,T) \to (G',S',T')$ such that all edge (S,T)-cuts in G become vertex (S,T)-cuts of the same size in G', and conversely. Then, a vertex cut problem may be harder than its edge version. If an FPT algorithm exists for a vertex cut problem, then its edge version is FPT as well.
Function $\psi : (G,S,T) \to (G',S',T')$ is clarified below. Given a size $q \geq 1$, it transforms all edge (S,T)-cuts X, $|X| \leq q$ into vertex (S',T')-cuts X' of size $|X'| = |X|$. We begin with the graph transformation:

- for each vertex $u \in V$, we add $q+1$ vertices called the *duplicates* of u in V': $u'_1, u'_2, ..., u'_{p+1} \in V'$,
- for each edge $e = (u,v) \in E$, we define a corresponding vertex $v'_e = \psi(e)$ in V' and next we add edges between it and the duplicates of u and v: $(u'_1, v'_e), \ldots, (u'_{p+1}, v'_e), (v'_1, v'_e), \ldots, (v'_{p+1}, v'_e) \in E'$.

Set of sources S' contains the duplicates of sources from S. Similarly, set of targets T' is composed of the duplicates of targets from T. Abusing notations, for any edge (S,T)-cut X of G, set $\psi(X)$ is defined as $\psi(X) = \{v'_e \in V' : e \in X\}$. We have $|X| = |\psi(X)|$ by definition. Figure 2.2a provides a graph G while its image $G' = \psi(G)$ is represented in Figure 2.2b. Sets of duplicates in $\mathcal{I}' = (G',S',T')$ are encircled. Edge (v_1,v_2) is the minimum (S,T)-cut of G and vertex $\psi((v_1,v_2))$ is the minimum vertex (S,T)-cut of G'.



(a) Instance $\mathcal{I} = (G,S,T)$ (b) Instance $\mathcal{I}' = (G',S',T')$

Figure 2.2: Reduction ψ: instance $\mathcal{I}$ transformed into instance $\mathcal{I}'$

Let $\mathcal{M}$ and $\mathcal{M}'$ be the sets of (S,T)-cuts for the triplet (G,S,T) and (S',T')-cuts for (G',S',T') of size q, respectively. It follows:

Theorem 2.2. *Function $\psi : \mathcal{M} \to \mathcal{M}'$ is bijective.*

Proof. Let us suppose that $\psi(X)$ is not a vertex (S',T')-cut for a certain edge (S,T)-cut X. There is a path P' from S' to T' which bypasses $\psi(X)$. Path P' passes through vertices v'_e on G', *i.e.* vertices v' such that $\psi^{-1}(v') \in E$. By collecting edges $e \in E$ such that $v'_e \in P'$, we form a simple (S,T)-path in G which bypasses edges in X. This contradicts that X is an (S,T)-cut. Therefore, its image, $\psi(X)$, is a vertex (S',T')-cut.
Conversely, let X' be a vertex (S',T')-cut of G' of size at most q. Cut X' cannot contain duplicates. Indeed, given a vertex $u \in V$, if X' contains at most q duplicates of u, this does not compromise the connectivity in G', so X' is not minimal. Furthermore, the entire set of duplicates of u cannot be in X', otherwise $|X'| > q$. Thus, cut X' contains vertices $v' \in V'$ only, where $\psi^{-1}(v') \in E$ and $\psi^{-1}(X') \subseteq E$. Set $\psi^{-1}(X')$ forms naturally an (S,T)-cut, according to the argument used previously: an (S,T)-path bypassing $\psi^{-1}(X')$ in G implies that there is an (S',T')-path avoiding X' in G'. In a nutshell, set $\psi^{-1}(X')$ is an edge (S,T)-cut of size $|X'| = |X|$. □

In accordance with Theorem 2.2, for any edge (S,T)-cut in G, there is a "twin" vertex (S',T')-cut $\psi(X)$ in G', both of the same size. As a consequence, a problem as EDGE POTC can be reduced to VERTEX POTC as any of its solution in G becomes a vertex solution in G'. The property EDGE POTC $\leq_{\text{fpt}}$VERTEX POTC is used in Section 2.2. In Section 2.3, this reduction allows us to affirm that an algorithm solving the counting of minimum vertex (S,T)-cuts also computes the number of minimum edge (S,T)-cuts. Indeed, the number of minimum (S,T)-cuts in G is equal to the number of minimum (S',T')-cuts in G'.

2.2 Separating certain sources from a single target

After having set the background in the introductory section, we present our contributions concerning cut problems.
The results described in this section were published in our article [17]. We study the parameterized complexity of POTC: the objective is to know whether a cut of size p separates r sources of the set $S = \{s_1, \ldots, s_k\}$ from a single target t. We prove the fixed-parameter tractability of EDGE POTC parameterized by p in Section 2.2.1. Then, we provide two hardness proofs in Section 2.2.2: one showing that EDGE POTC is W[1]-hard parameterized by r and another one showing that VERTEX POTC is W[1]-hard for all parametrizations involving p and r. The design of FPT algorithms for EDGE POTC is motivated by the fact that this problem is necessarily NP-complete, as it is W[1]-hard for parameter r.

2.2.1 Cut size p as a parameter

We characterize the solutions of EDGE POTC with important cuts. We show that at least one of them is the union of important $(S^{(i)},t)$-cuts, where all sets $S^{(i)} \subseteq S$. Using this property and a result on the parameterized problem PARTIAL SET COVER [22], we obtain that EDGE POTC is FPT$\langle r,p\rangle$. To achieve our initial goal, *i.e.* the construction of an FPT$\langle p\rangle$ algorithm, we devise an efficient sampling of important cuts, also based on our characterization. We define the concept of *edge passes*, which are certain edges of the graph defined in function of our solution made up of important cuts. Color-coding and derandomization techniques finally allow us to generate a solution in FPT$\langle p\rangle$ time.

Relationship between important cuts and solutions of Edge POTC

We focus on important (S',t)-cuts Y with $S' \subseteq S$ such that vertices in S' stay pairwise connected despite cut Y, in other words $G\left[R(Y,S')\right]$ remains connected. Such important cuts are said *single-component*. The following lemma is an intermediary result for the proof that a solution of EDGE POTC is the union of single-component important cuts.

Lemma 2.5. *Let Y be a single-component important (S',t)-cut. For any $S'' \subseteq S'$, Y is also an important (S'',t)-cut.*

Proof. As Y is an (S',t)-cut, it necessarily separates $S'' \subseteq S'$ from t. Any vertex reachable from S'' in $G\backslash Y$ is also reachable from S': $R(Y,S'') \subseteq R(Y,S')$. Conversely, let vertex v be reachable from S' in $G\backslash Y$. Any source in S'' is connected to sources in $S'\backslash S''$ as $R(Y,S')$ is connected. Consequently, vertex v is reachable from S'': $R(Y,S'') = R(Y,S')$. Then, Y is a minimal (S'',t)-cut otherwise it would not be a minimal (S',t)-cut.
If Y is not an important (S'',t)-cut, there is an (S'',t)-cut $\widehat{Y}$ with $\left|\widehat{Y}\right| \leq |Y|$ and $R(Y,S'') \subsetneq R(\widehat{Y},S'')$. Therefore, $R(Y,S') \subsetneq R(\widehat{Y},S'')$ and $\widehat{Y}$ is an (S',t)-cut as $S' \subseteq R(Y,S')$. Cutset Y is no longer important for (S',t), which is a contradiction. □

Let $Y_1, Y_2, \ldots, Y_{M_p}$ denote all the important (s,t)-cuts with size at most p, where s is a source: $s \in S$ and value M_p indicates their number. From Lemma 2.2, we know that

$M_p \leq k4^p$. These cuts are single-component. Indeed, let us suppose that one important (s,t)-cut Y_j is not: the separated vertices of Y_j form at least two connected components. One of these two components does not contain s, so some edges of Y_j do not separate s from t. In other words, cut Y_j is not minimal for terminals (s,t) which is a contradiction. For any Y_j, we denote by S_j the set of sources that are separated from t in $G\backslash Y_j$, and $R_j = R(Y_j, S_j)$. We prove that at least one solution of EDGE POTC is the union of certain of these important (s,t)-cuts.

Theorem 2.3. *If (G,S,t,r,p) is a feasible instance for* EDGE POTC*, there is a solution Y^* which is the union of some single-component important cuts among $Y_1,\ldots,Y_{M_p}$: $Y^* = \bigcup_{1\leq i\leq \ell} Y_{\sigma(i)}$, where*

- *value $\ell \leq p$ is the number of connected components in $G[R(Y^*,S')]$, where set S' stands for the set of sources separated from t in $G\backslash Y^*$,*
- *injective function $\sigma : \{1,\ldots,\ell\} \to \{1,\ldots,M_p\}$ indexes cuts Y_j which build up Y^*,*
- *cuts $Y_{\sigma(i)}$ have no edge in common, $\bigcap_{1\leq i\leq \ell} Y_{\sigma(i)} = \emptyset$ and they separate different vertices, $\bigcap_{1\leq i\leq \ell} R_{\sigma(i)} = \emptyset$.*

Proof. Let X be a minimum solution for this instance: no solution X' fulfils $|X'| < |X|$. We use cutset X to build Y^*. We denote $S' \subseteq S$ the set of sources that are separated from t in $G\backslash X$. Obviously, $|S'| \geq r$. After writing $R = R(X,S')$, set R is naturally partitioned $R = \bigcup_{1\leq i\leq \ell} R^{(i)}$, where each $R^{(i)}$ corresponds to a connected component of subgraph $G[R]$. In other words, for $(v_i,v_q) \in R^{(i)} \times R^{(q)}$, $i \neq q$, vertices v_i and v_q are not only separated from t by X but pairwise separated as well. We partition set S' in a similar way: $S' = \bigcup_{1\leq i\leq \ell} S^{(i)}$, where $S^{(i)} = S' \cap R^{(i)}$.

We prove that $X = \bigcup_{1\leq i\leq \ell} X_i$, where $X_i = \delta(R^{(i)})$ and $\bigcap_{1\leq i\leq \ell} X_i = \emptyset$. If $e = (u,v) \in X$ with $u \in R$ and $v \in V(G)\backslash R$, then there is always $R^{(i)}$ such that $u \in R^{(i)}$, so $e \in \delta\left(R^{(i)}\right)$. As the statement holds for all $e \in X$, we deduce $X \subseteq \bigcup_{1\leq i\leq \ell} X_i$.

Conversely, let $e = (u,v) \in X_i$ with $u \in R^{(i)}$ and $v \in V(G)\backslash R^{(i)}$: if $v \in V(G)\backslash R$ then $e \in X$. Otherwise, if $v \in R\backslash R^{(i)}$, then cut $X\setminus\{e\}$ also disconnects sources S' from t which is a contradiction as X is supposed to be minimum. Therefore, we obtain $X = \bigcup_{1\leq i\leq \ell} X_i$. Moreover, if $e = (u,v) \in X_i \cap X_q$ with $u \in R_i$ and $v \in R_q$, the same argument yields a contradiction as $X\setminus\{e\}$ separates R from t. We thus have $\bigcap_{1\leq i\leq \ell} X_i = \emptyset$ and $|X| = \sum_{1\leq i\leq \ell} |X_i|$. As any X_i contains at least one edge and $|X| \leq p$, we have $\ell \leq p$.

For any $1 \leq i \leq \ell$, we compute the minimum important $(R^{(i)},t)$-cut $Y^{(i)}$ and we study cutset $Y^* = \bigcup_{1\leq i\leq \ell} Y^{(i)}$. It is illustrated on an example in Figure 2.3: reachable vertices $R^{(i)}$ (in blue areas) separated by edges in X_i (leaving $R^{(i)}$), important cuts $Y^{(i)}$ (black edges), and sources (red vertices). All $Y^{(i)}$ are single-component: if u and v are reachable from a vertex of $R^{(i)}$ in graph $G\backslash Y^{(i)}$, there is a (u,v)-path as $G[R^{(i)}]$ is connected. According to Lemma 2.5, $Y^{(i)}$ is an important (s,t)-cut for all $s \in S^{(i)}$. Consequently, $Y^{(i)}$ is one of the $Y_1,\ldots,Y_{M_p}$ cuts and its index is given by $\sigma(i)$. In summary, $Y^{(i)} = Y_{\sigma(i)}$ and $Y^* = \bigcup_{1\leq i\leq \ell} Y_{\sigma(i)}$.

Now we prove that Y^* is a solution. Cutset Y^* separates from t at least r sources which belong to R. As X is minimum, $|X| \leq |Y^*|$. Conversely, as $Y_{\sigma(i)}$ is a minimum $(R^{(i)},t)$-cut, we have $\left|Y_{\sigma(i)}\right| \leq |X_i|$ and, finally,

$$|Y^*| \leq \sum_{i=1}^{\ell} \left|Y_{\sigma(i)}\right| \leq \sum_{i=1}^{\ell} |X_i| = |X|\,.$$

This means that $|Y^*| \leq p$. Eventually, the argument used above to prove $\bigcap_{1\leq i\leq \ell} X_i = \emptyset$, is still valid as Y^* is also a minimum solution: $\bigcap_{1\leq i\leq \ell} Y_{\sigma(i)} = \emptyset$. For the same reason, cutsets

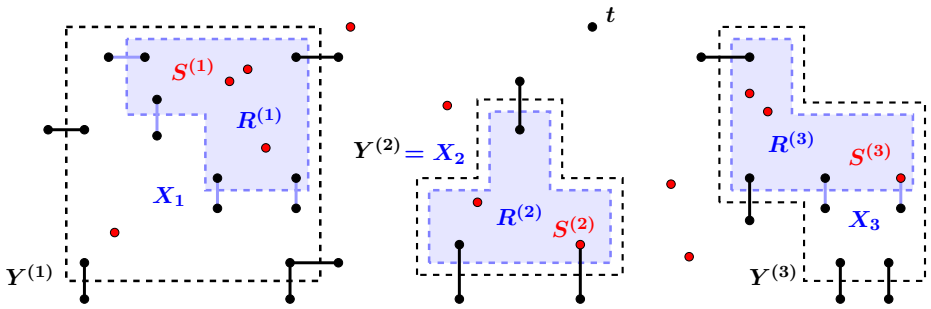


Figure 2.3: Cutsets X_i, $Y^{(i)} = Y_{\sigma(i)}$ and vertex sets $S^{(i)}$ and $R^{(i)}$.

$Y_{\sigma(i)}$ separate disjoint parts: $\bigcap_{1\leq i\leq \ell} R_{\sigma(i)} = \emptyset$. Indeed, if $R_{\sigma(i)} \cap R_{\sigma(i')} \neq \emptyset$ *ad absurdum* for $i \neq i'$, there is at least one edge leaving $R_{\sigma(i)} \cap R_{\sigma(i')}$: $\left|\delta(R_{\sigma(i)} \cap R_{\sigma(i')})\right| > 0$. The submodularity of cut-function δ (evoked in [48, 63]) gives:

$$\left|\delta(R_{\sigma(i)} \cup R_{\sigma(i')})\right| < \left|\delta(R_{\sigma(i)})\right| + \left|\delta(R_{\sigma(i')})\right| = \left|Y_{\sigma(i)}\right| + \left|Y_{\sigma(i')}\right|.$$

As cutset $Y_{\sigma(i)} \cup Y_{\sigma(i')}$ is not minimum for separating $S_{\sigma(i)} \cup S_{\sigma(i')}$ from t, Y^* is not minimum, which terminates the proof. □

Theorem 2.3 ensures the existence of a particular solution of EDGE POTC for any feasible instance. We will prove that it can be identified either in time $2^{O(r)}4^p$ (Theorem 2.4) or $2^{O(p^2)}$ (Theorem 2.7). For the former, we show that Y^* is produced by the solution of the partial set cover problem [22].

Definition 2.8 (Partial Set Cover)**.**
Input: *Universe U, sets $A_j \subseteq U$ of weights ω_j, positive integers p and $r \leq |U|$.*
Question: *Is there a collection $\mathcal{C}$ of sets A_j which covers at least r elements in U such that $\sum_{A_j\in\mathcal{C}} \omega_j \leq p$?*

A partial set cover is computed in time $O^*\left(2^{O(r)}\right)$ multiplied by the number of sets A_j of the instance [22].

Theorem 2.4. EDGE POTC *can be solved in time $O^*\left(2^{O(r)}4^p\right)$.*

Proof. We reduce EDGE POTC to PARTIAL SET COVER. Collection $\mathcal{C}^* = \left\{S_{\sigma(i)}\right\}$ is a solution for the partial set cover instance $U = S$, $A_j = S_j$, $\omega_j = |Y_j|$. It contains more than r elements of the "universe" of sources S and its total weight is smaller than p.
Any solution for this instance produces a solution for EDGE POTC satisfying the property given in Theorem 2.3. All cuts Y_j are enumerated in time $O^*\left(4^p\right)$. The number of covering sets $A_j = S_j$ is $M_p \leq k4^p$. The running time obtained is $O^*\left(2^{O(r)}4^p\right)$. □

As a consequence, EDGE POTC is FPT$\langle r, p\rangle$. We focus now on its complexity when it is parameterized by p only.

Definition of edge passes

We prove that there is a tractable algorithm for EDGE POTC when only p is small. To do so, we make use of combinatorial properties specific to important cuts Y_j.
There are three kinds of relationships which exist between two important cuts Y_j and Y_q:

- these cuts can be disjoint ($R_j \cap R_q = \emptyset$),
- one can be included into another (either $R_j \subseteq R_q$ or $R_q \subseteq R_j$),

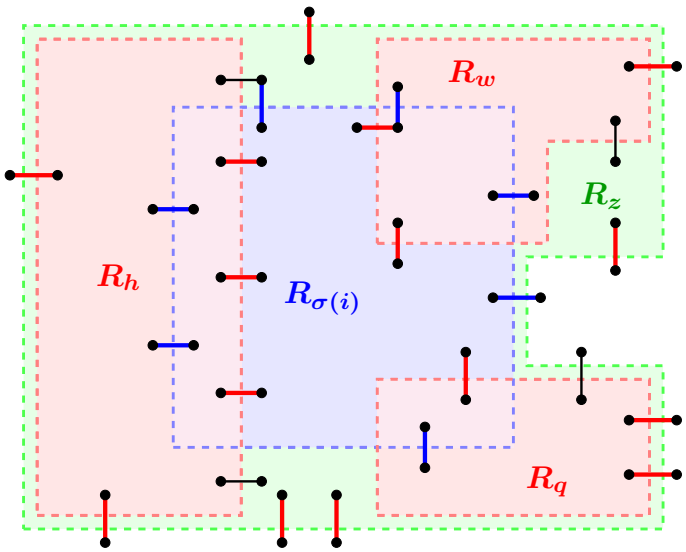


Figure 2.4: Painted cutset $Y_{\sigma(i)}$ with secant cuts Y_h, Y_q, Y_w and included in Y_z.

- they are *secant* if they do not fulfil any of the previous cases. In other words, Y_j and Y_q are secant if $R_j \cap R_q \neq \emptyset \wedge R_j \backslash R_q \neq \emptyset \wedge R_q \backslash R_j \neq \emptyset$. We use notation $(Y_j, Y_q) \in \Pi$ to express this relationship.

Cut Y_j is characterized by its *edge passes*:

Definition 2.9 (Edge passes)**.** *Edge $e = (u, v)$ is a pass of Y_j if:*

- *vertices u and v are both in R_j: $u, v \in R_j$,*
- *there is a secant cut Y_q, $(Y_j, Y_q) \in \Pi$, such that $e \in Y_q$.*

Edge passes of cut Y_j, assembled in set P_j, are contained in the subgraph $G[R_j]$ of G separated from t by Y_j. Each edge contained in P_j is in at least one of the secant cuts of Y_j.
We prove that under a certain hypothesis, EDGE POTC is FPT$\langle p \rangle$. This hypothesis is that edges of G are colored either in blue or in red in a particular way. Here, colorings must not be understood in the classical sense of edge coloring problems (they are not proper edge colorings). Thanks to derandomization tools, we get rid of this hypothesis later as for any j, the number of edge passes P_j is small.

Definition 2.10. *We say that an important cut Y_j is painted if:*

- *all edges in cut Y_j are blue,*
- *all edge passes P_j of Y_j are red,*
- *for any $1 \leq q \leq M_p$, if $R_j \subsetneq R_q$, then edges in $Y_q \backslash Y_j$ are red.*

We remind that Y^* denotes the solution satisfying properties listed in Theorem 2.3.
We say that graph G is *cut-painted* for solution Y^* if all cuts $Y_{\sigma(i)}$ are painted. Figure 2.4 illustrates a cut $Y_{\sigma(i)}$ (blue edges and dashed line) with three secant cuts Y_h, Y_q, Y_w (red dashed lines), and it is included in Y_z (green dashed line). Edge passes of $Y_{\sigma(i)}$ and edges in $Y_z \backslash Y_{\sigma(i)}$ are in red. Any color put on black edges makes cutset $Y_{\sigma(i)}$ be painted. The following theorem says that if we only know to cut-paint graph G for Y^* in FPT$\langle p \rangle$ time, EDGE POTC is FPT$\langle p \rangle$.

Theorem 2.5. *If graph G is cut-painted for Y^*, a solution of* EDGE POTC *can be found in time $O^*\left(4^{2p}\right)$.*

Proof. Among cuts $Y_1, \ldots, Y_{M_p}$, we pick up cuts Y_j that are painted and we obtain a collection $\mathcal{D}$. For two different painted cuts Y_j, Y_q, no source can be separated from t by both of them: $S_j \cap S_q = \emptyset$. Otherwise, cuts Y_j, Y_q are either one included into another or secant. The first case ($R_j \subsetneq R_q$, for example) leads to a contradiction because edges in $Y_q \backslash Y_j$ must be red as $Y_j \in \mathcal{D}$ and blue as $Y_q \in \mathcal{D}$. The second case, if Y_j and Y_q are secant, edges in Y_q must be blue ($Y_q \in \mathcal{D}$) but, if some of them are edge passes of Y_j, they must also be in red ($Y_j \in \mathcal{D}$). We prove that there is at least one edge which must be colored with two colors in the same time. As Y_j is single-component, some edges connect the two parts of R_j, $R_j \cap R_q$ and $R_j \backslash R_q$. Consequently, there is necessarily an edge leaving R_q with its two endpoints in R_j: an edge pass of Y_j. This argument brings the contradiction for the second case. In summary, cuts in $\mathcal{D}$ are disjoint ($R_j \cap R_q = \emptyset$ and $S_j \cap S_q = \emptyset$).
As G is cut-painted for Y^*, which is composed only of single-component important cuts, all cutsets $Y_{\sigma(i)}$ belong to collection $\mathcal{D}$. Solution Y^* is the union of $\ell \leq p$ cutsets from $\mathcal{D}$. The cardinality of $\mathcal{D}$ is at most k, as each cutset Y_j in $\mathcal{D}$ separates its own set of sources S_j.
As cutsets in $\mathcal{D}$ are disjoint, finding a subset of $\mathcal{D}$ of cardinality at most p that separates the maximum number of sources is equivalent to solving the 0-1 KNAPSACK problem. Each $Y_j \in \mathcal{D}$ can be seen as a knapsack item of utility value $|S_j|$ (number of sources separated) and weight $|Y_j|$ (the cut size). With the knapsack capacity equal to p, an optimal solution is made up of these cutsets from $\mathcal{D}$ which separate from t the maximum number of sources. Thanks to cutset Y^*, at least one solution is found and reaches a value greater or equal to r. Consequently, the optimal knapsack is a solution of EDGE POTC as it separates more than r sources with less than p edges.
Evaluating the execution time, we observe that the most expensive operation consists in computing collection $\mathcal{D}$. For each important cut Y_j, we determine its relationship with other cutsets and check whether it is painted or not. This consists in a double loop enumerating all pairs of cuts (Y_j, Y_q) and checking for any of them whether colors are correct (execution time $O^*(M_p^2) = O^*\left(4^{2p}\right)$). A standard dynamic programming produces in our case the knapsack packing in $O(kp)$ with the number of items k and capacity p, which is negligible compared to the computation of collection $\mathcal{D}$. □

We show a manner to cut-paint graph G for solution Y^* in FPT$\langle p \rangle$ time. In this way, an FPT$\langle p \rangle$ algorithm becomes apparent naturally thanks to Theorem 2.5.

Derandomization

We present a way to generate a list of colorings for G such that at least one of them paints Y^*. For any feasible instance, computing the knapsack packing algorithm for each of these colorings necessarily finds a solution as graph G is cut-painted for Y^* with one of them. Our reasoning begins with the proof that the cardinality of edge passes is bounded by p only, which allows us to produce this list of colorings in FPT$\langle p \rangle$ time.

Theorem 2.6 (Cardinality of secant cuts and edge passes)**.** *For any cutset Y_j, there are at most $p4^p$ secant cuts Y_q, $(Y_j, Y_q) \in \Pi$, which implies that $|P_j| \leq p^2 4^p$.*

Proof. For a vertex $v \in V(G)$, we focus on cutsets Y_q separating it from t, *i.e.* $v \in R_q$. Such a cutset Y_q is an important (v,t)-cut as Y_q is single-component and $\{v\} \subseteq R_q$ (Lemma 2.5). Therefore, there are at most 4^p sets R_q covering vertex v (Lemma 2.2).
Let B_j be the *border* of set R_j, *i.e.* vertices in R_j which are endpoints of cut edges in Y_j: $B_j = \{v \in R_j : (u,v) \in Y_j\}$. Border B_j contains at most p vertices. For any secant cut Y_q, $(Y_j, Y_q) \in \Pi$, there is an edge of Y_j with both its endpoints in R_q. Indeed, sets $R_j \cap R_q$ and

$R_q \backslash R_j$ have to be connected as Y_q is single-component. So, an edge $(u, v) \in Y_j$ verifies $u \in R_j \cap R_q$ and $v \in R_q \backslash R_j$. Thus, at least one vertex in the border B_j belongs to R_q.
In summary, secant cuts Y_q, $(Y_j, Y_q) \in \Pi$, verify:

- set R_q covers at least one border vertex $v_{j,q}$ of Y_j: $R_q \cap B_j \neq \emptyset$,
- there are at most 4^p sets covering $v_{j,q}$ as R_q.

Consequently, the cardinality of secant cuts is bounded by $4^p |B_j| \leq p4^p$. It follows that there are at most $p^2 4^p$ edge passes because $|Y_q| \leq p$. □

The number of edges with the same color (either blue or red) obligatorily assigned is bounded from above for G which is cut-painted for Y^*. These bounds depend on p only.
Only edges of cutset Y^* have to be blue (Definition 2.10). As $|Y^*| \leq p$, at most p edges have to be blue. They are denoted by $E_{Y^*,\mathsf{blue}}$. Theorem 2.6 allows us to obtain an upper bound for red edges in function of p only. As the number of border vertices in Y^* is less than p, there are less than $p^2 4^p$ edge passes for Y^*. Furthermore, for any $1 \leq i \leq \ell$, there are at most $4^p - 1$ cuts Y_q with $R_{\sigma(i)} \subsetneq R_q$ because any vertex of $R_{\sigma(i)}$ has at most 4^p important cuts separating it from t. As a consequence, the number of edges belonging to a set $Y_q \backslash Y_{\sigma(i)}$ with $R_{\sigma(i)} \subsetneq R_q$ is less than $\ell p(4^p - 1)$, which can be upper-bounded by $p^2 4^p$.
In a nutshell, at most $2p^2 4^p$ edges must be colored in red. They are denoted by $E_{Y^*,\mathsf{red}}$.
We define $\varphi_{\mathsf{blue}}(p) = p$, $\varphi_{\mathsf{red}}(p) = 2p^2 4^p$ as upper bounds of $\left|E_{Y^*,\mathsf{blue}}\right|$ and $\left|E_{Y^*,\mathsf{red}}\right|$, respectively. Value $\varphi(p) = \varphi_{\mathsf{blue}}(p) + \varphi_{\mathsf{red}}(p) = O\left(p^2 4^p\right)$ bounds from above the number of edges obligatorily colored with a certain color: $|E_{Y^*}| \leq \varphi(p)$.
When colors are assigned to edges uniformly, the probability to get G cut-painted for Y^* is greater than $\left(\frac{1}{2}\right)^{\varphi(p)}$. In other words, there is an FPT$\langle p \rangle$ algorithm solving EDGE POTC with probability $\left(\frac{1}{2}\right)^{\varphi(p)}$. We derandomize this algorithm by using a result [67] on (n, a, a^2)-*splitters*.

Definition 2.11 (from [67]). *A (n, a, a^2)-splitter is a collection H of functions $h : \{1, \ldots, n\} \to \{1, \ldots, a^2\}$ such that, for any subset $A \subsetneq \{1, \ldots, n\}$, $|A| = a$, there is a function $h \in H$ which is injective over A.*

Naor *et al.* [67] showed that there is a (n, a, a^2)-splitter with $a^6 \log n \log a$ functions, listed in time linear to its cardinality.
We compute the $(m, \varphi(p), \varphi(p)^2)$-splitter $H_{G,p}$, where $m = |E(G)|$. We order edges of G with indices from $\{1, \ldots, m\}$ arbitrarily. Any pair (h, I_{blue}) with $h \in H_{G,p}$ and $I_{\mathsf{blue}} \subsetneq \{1, \ldots, \varphi(p)^2\}$, $|I_{\mathsf{blue}}| = p$ is interpreted as an edge-coloring of G: edges of index in $h^{-1}(I_{\mathsf{blue}})$ are colored in blue, otherwise in red. The following lemma establishes a relationship between splitters and cut paintings of graph G.

Lemma 2.6. *There is a pair (h, I_{blue}) with $h \in H_{G,p}$ and $I_{\mathsf{blue}} \subsetneq \{1, \ldots, \varphi(p)^2\}$, $|I_{\mathsf{blue}}| = p$ which paints graph G for cut Y^*.*

Proof. Set E_{Y^*} contains edges that have to be colored either in blue or in red: $E_{Y^*} = E_{Y^*,\mathsf{blue}} \cup E_{Y^*,\mathsf{red}}$. By Definition 2.11, there is $h \in H_{G,p}$ such that h is injective over E_{Y^*}. As $E_{Y^*,\mathsf{blue}} \cap E_{Y^*,\mathsf{red}} = \emptyset$, the image of these sets is also disjoint for h: $h(E_{Y^*,\mathsf{blue}}) \cap h(E_{Y^*,\mathsf{red}}) = \emptyset$. Let $I_{\mathsf{blue}} = h(E_{Y^*,\mathsf{blue}})$. As h is injective over $E_{Y^*,\mathsf{blue}} \subsetneq E_{Y^*}$, we have $|I_{\mathsf{blue}}| = p$. It comes that edges of $h^{-1}(I_{\mathsf{blue}}) = E_{Y^*,\mathsf{blue}}$ are colored in blue. As $h(E_{Y^*,\mathsf{red}})$ is disjoint from I_{blue}, edges of $E_{Y^*,\mathsf{red}}$ are colored in red. □

Lemma 2.6 ensures us that graph G is cut-painted for Y^* for at least one pair (h, I_{blue}). There are $O^*\left(\varphi(p)^6\right) = O^*\left(4^{6p}\right)$ functions $h \in H_{G,p}$. Then, there are exactly $\binom{\varphi(p)^2}{p} = O^*\left(4^{2p^2}\right)$ sets I_{blue} of size p in $\{1, \ldots, \varphi(p)^2\}$. In summary, the total number of such pairs is upper-bounded by $O^*\left(2^{O(p^2)}\right)$. Therefore, the algorithm presented below is FPT$\langle p \rangle$:

Theorem 2.7. EDGE POTC *can be solved in time* $O^*\left(2^{O(p^2)}\right)$.

Proof. We summarize the steps to obtain a solution for EDGE POTC:

- Step 1: Compute all the important (s,t)-cuts of size at most p for any source $s \in S$,
- Step 2: Generate an $(m, \varphi(p), \varphi(p)^2)$-splitter $H_{G,p}$,
- Step 3: For any pair (h, I_{blue}) with $h \in H_{G,p}$ and $I_{\text{blue}} \subsetneq \{1, \ldots, \varphi(p)^2\}$, $|I_{\text{blue}}| = p$, color graph G in line with it and find a solution among painted cuts (Theorem 2.5).

For a feasible EDGE POTC instance, solution Y^* exists. Therefore, at least one pair (h, I_{blue}) paints Y^* and a solution is necessarily found in Step 3. If the instance is not feasible, Step 3 cannot produce a solution, despite colors assigned to edges of G.
The execution time is $O^*\left(4^p\right)$ for Step 1, $O^*\left(2^{O(p)}\right)$ for Step 2, and $O^*\left(2^{O(p^2)}4^{2p}\right)$ for Step 3. The overall complexity is $O^*\left(2^{O(p^2)}\right)$. □

In summary, we proposed two algorithms to solve EDGE POTC. The first one is FPT$\langle r, p\rangle$ and its execution time is $O^*\left(2^{O(r)}4^p\right)$. The second one implies a stronger result as it is FPT$\langle p\rangle$. Its running time is $O^*\left(2^{O(p^2)}\right)$. Our objective is now to show whether EDGE POTC is FPT or W[1]-hard for parameter r only. Moreover, we focus on the parameterized complexity of VERTEX POTC.

2.2.2 Hardness results

We provide two W[1]-hardness proofs: one for EDGE POTC parameterized by r only and another one for the vertex version of POTC, VERTEX POTC, parameterized by $p + r$.

Hardness of Edge Partial One-Target Cut for parameter r only

The problem of finding the minimum edge $(r, n-r)$-cut in a graph G which separates a set of vertices V^* from $V(G)\backslash V^*$, with $|V^*| = r$ is called CUTTING r VERTICES in [44].

Definition 2.12 (Cutting r vertices)**.**
Input: *Undirected graph G, parameters r and p.*
Question: *Is there an edge cutset X and a set of vertices V^* such that X disconnects V^* from $V(G)\backslash V^*$ with $|X| \leq p$ and $|V^*| = r$?*

Downey *et al.* [44] proved that CUTTING r VERTICES$\langle r\rangle$ is W[1]-hard with a reduction from CLIQUE.[1] We show that EDGE POTC is W[1]-hard for parameter r.

Theorem 2.8. EDGE POTC *is W[1]-hard parameterized by* r.

Proof. We reduce CUTTING r VERTICES (abbreviated to r-CV) to EDGE POTC. We construct an instance of EDGE POTC $\left(\widehat{G}, \widehat{S}, \widehat{t}, \widehat{r}, \widehat{p}\right)$ from an instance of r-CV, which is composed of graph G, parameter r, and cutset size p.
We index the vertices of G, $V(G) = \{v_i\}_{1 \leq i \leq n}$. Set $V(\widehat{G})$ contains n sources $\widehat{S} = \{\widehat{s}_i\}$ (which are copies of vertices v_i) and target $\widehat{t}$. We also duplicate edges of G on $\widehat{G}$: for any $(v_i, v_j) \in E(G)$, there is $(\widehat{s}_i, \widehat{s}_j) \in E(\widehat{G})$. We connect all vertices $\widehat{s}_i$ to target $\widehat{t}$ with $p+1$ edge-disjoint $(\widehat{s}_i, \widehat{t})$-paths which belong to a set $\widehat{Q}_i$. Figure 2.5 illustrates sets $\widehat{Q}_i$ and $\widehat{Q}_h$ for vertices $\widehat{s}_i, \widehat{s}_h \in \widehat{S}$. All paths in $\widehat{Q}_i$ are composed of two edges:

$$\widehat{Q}_i = \{\widehat{s}_i \cdot \widehat{v}_{i,j} \cdot t : 1 \leq j \leq p+1\}.$$

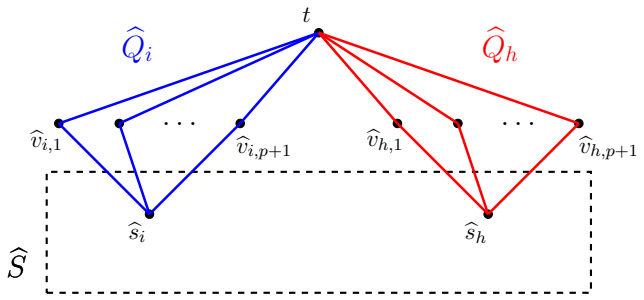


Figure 2.5: Structure of the graph $\widehat{G}$ with edge-disjoint paths.

We put $\widehat{p} = r(p+1)+p$ and $\widehat{r} = r$. Let us suppose that there is a cut X in G with size at most p that separates some vertices V^*, $|V^*| = r$, from the others. We define $\widehat{X} = \widehat{X}_{\widehat{S}\to t} \cup \widehat{X}_{\widehat{S}\to\widehat{S}}$, where:

$$\widehat{X}_{\widehat{S}\to t} = \{(\widehat{s}_i, \widehat{v}_{i,j}) : v_i \in V^*, 1 \leq j \leq p+1\} \text{ and } \widehat{X}_{\widehat{S}\to\widehat{S}} = \{(\widehat{s}_i, \widehat{s}_j) : (v_i, v_j) \in X\}.$$

By definition, $\left|\widehat{X}\right| \leq \widehat{p}$. We show that edges in $\widehat{X}$ separate set $\widehat{S}' = \{\widehat{s}_i : v_i \in V^*\}$, $\left|\widehat{S}'\right| = r$, from t. Ignoring edges $(\widehat{s}_i, \widehat{t})$ in graph $\widehat{G}$ makes set $\widehat{S}'$ be isolated because of $\widehat{X}_{\widehat{S}\to\widehat{S}}$. Cut $\widehat{X}_{\widehat{S}\to t}$ disconnects $\widehat{S}'$ from $\widehat{t}$, so no path from any source in $\widehat{S}'$ to $\widehat{t}$ exists because of $\widehat{X}$. Conversely, we suppose that there is a cut $\widehat{X} \subsetneq E(\widehat{G})$ and a set $\widehat{S}' \subsetneq \widehat{S}$, $\left|\widehat{S}'\right| \geq r$, such that $\widehat{S}'$ is separated from $\widehat{t}$, and the size of $\widehat{X}$ is less than $\widehat{p}$. Each path in $\widehat{Q}_i$ with $\widehat{s}_i \in \widehat{S}'$ necessarily contains an edge of $\widehat{X}$, otherwise $\widehat{s}_i$ and $\widehat{t}$ are connected. On the contrary, vertices $\widehat{s}_h \notin \widehat{S}'$ and $\widehat{t}$ are connected via at least one path of $\widehat{Q}_h$, otherwise $\left|\widehat{X}\right| \geq (r+1)(p+1) > \widehat{p}$. For the same reason, $\left|\widehat{S}'\right| = r$. The edges of $\widehat{X}$ on the edge-disjoint paths are denoted by $\widehat{X}_{\widehat{S}'\to\widehat{t}}$. We can split $\widehat{X}$ into two sets $\widehat{X}_{\widehat{S}'\to\widehat{t}}$ and $\widehat{X}_{\widehat{S}'\to\widehat{S}'}$, where $\widehat{X}_{\widehat{S}'\to\widehat{S}'}$ contain edges from $\widehat{S}\times\widehat{S}$ and its size is at most p. We write:

$$X = \left\{(v_i, v_j) \in E(G) : (\widehat{s}_i, \widehat{s}_j) \in \widehat{X}_{\widehat{S}'\to\widehat{S}'}\right\}.$$

Let $v_i, v_j \in V(G)$ be such that $\widehat{s}_i \in \widehat{S}'$ and $\widehat{s}_j \notin \widehat{S}'$. We prove that no (v_i, v_j)-path in $G\backslash X$ exists. We suppose *ad absurdum* that v_i and v_j are connected. As a consequence, $\widehat{s}_i$ and $\widehat{s}_j$ are also connected in $\widehat{G}\backslash\widehat{X}$. As at least one edge is not removed from paths $\widehat{Q}_j$ in $\widehat{G}\backslash\widehat{X}$, one can build a $(\widehat{s}_i, \widehat{t})$-path traversing $\widehat{s}_j$ which is a contradiction as $\widehat{s}_i \in \widehat{S}'$. So, vertices $V^* = \left\{v_i : \widehat{s}_i \in \widehat{S}'\right\}$ are separated from other vertices of G: cutset X is a solution for r-CV. □

We observe that Theorem 2.8 also shows the NP-completeness of EDGE POTC. Indeed, its proof is a polynomial reduction from a NP-hard problem to it.
Let us introduce a W[1]-hardness proof for VERTEX POTC now.

Hardness of Vertex Partial One-Target Cut

The definition of the vertex version of POTC is given below. Naturally, terminal t is excluded from cuts, otherwise the trivial solution $X = \{t\}$ would separate all sources from the target.

[1]Even if CUTTING r VERTICES was originally defined for undirected weighted graphs, the reduction from CLIQUE in [44] only uses edges with unitary weights.

Definition 2.13 (Vertex Partial One-Target Cut)**.**
Input: *Graph G, sources $S = \{s_1, \ldots, s_k\}$, target t, positive integers p and $r \leq k$.*
Question: *Is there a cutset $X \subseteq V \setminus \{t\}$, $|X| \leq p$, which disconnects at least r sources from target t?*

We prove that CLIQUE parameterized by the size of the solution reduces to VERTEX POTC$\langle r, p \rangle$, as formulated in the following theorem.

Theorem 2.9. VERTEX POTC$\langle r, p \rangle$ *is W[1]-hard.*

Proof. When solving the CLIQUE problem, we look for a subgraph of G which is a clique of size $c \geq 0$. From graph G, we build an instance of POTC composed of graph G' with $k = |E(G)|$ sources, one target t, and parameters $r = \frac{c(c-1)}{2}$ and $p = c + \frac{c(c-1)}{2}$. We arbitrarily attribute identifiers from $\{1, 2, \ldots, n\}$ to vertices of G. For any $x, y \in V(G)$, we say that $x < y$ if the identifier of x is less than the identifier of y. Set $V(G')$ consists of four parts: $V(G') = S \cup \{t\} \cup V_1 \cup V_2$ (see Figure 2.6), where:

- set S contains vertices indexed by edges (x, y) of G which play the role of sources: $S = \{s_{x,y} : (x, y) \in E(G), x < y\}$,
- vertex t is the common target,
- set V_1 contains vertices indexed by vertices x of G: $V_1 = \{v_x : x \in V(G)\}$,
- set V_2 is the duplicate of S: $V_2 = \{v_{x,y} : (x, y) \in E(G), x < y\}$.

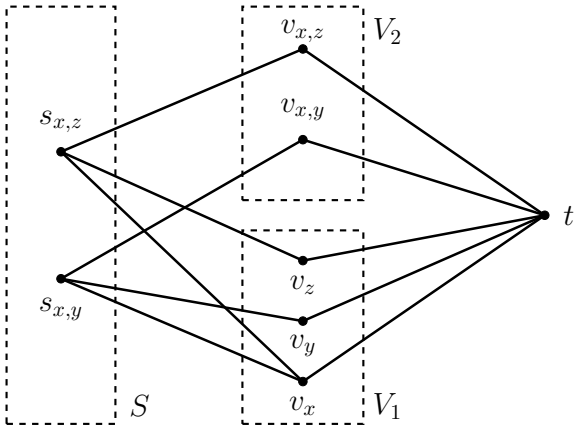


Figure 2.6: Construction of the VERTEX POTC instance: vertex sets S, V_1, V_2, and target t.

We connect a source $s_{x,y}$ to $v_x, v_y \in V_1$ and $v_{x,y} \in V_2$. Then, we connect all vertices of V_1 and V_2 to t. We suppose that there is a clique K of size c in graph G. The following cut X, with $|X| = p = c + \frac{c(c-1)}{2}$, disconnects $r = \frac{c(c-1)}{2}$ sources:

$$X = \{v_x : x \in K\} \cup \left\{v_{x,y} : (x, y) \in K^2, x < y\right\}. \tag{2.1}$$

No source $s_{x,y}$ such that $(x, y) \in K^2$ and $x < y$ is connected to t when vertices from X are deleted. No path exists between $s_{x,y}$ and t iff vertices v_x, v_y, and $v_{x,y}$ are removed.
Conversely, let us suppose that there is a vertex cut X' in G', $|X'| = p$, such that sources $S_{X'}$, $|S_{X'}| = r$, are not connected to t. We write $K' = \{x \in V(G) : v_x \in X'\}$ and $|K'| = a$. Our objective is to prove that K' is indeed a clique of size c.

- **If $a < c$**: any source $s_{x,y}$, such that at least one of two vertices v_x, v_y is not in X', is still connected to t. In other words, if $s_{x,y} \in S_{X'}$ then both $v_x, v_y \in X'$. We provide an

upper bound of $|S_{X'}|$ as function of a. At best, for any pair $(v_x, v_y) \in X'^2$, source $s_{x,y}$ is separated from t. By definition, the number $|X' \cap V_1|$ of vertices removed from V_1 is equal to $|K'| = a < c$, so there are at most $\binom{a}{2} = \frac{a(a-1)}{2} < r$ sources in S_X, which is a contradiction.

- **If $a > c$**: we write $B' = \{(x, y) \in E(G) : v_{x,y} \in X', x < y\}$ and $b = |B'| = |X' \cap V_2|$. We have $a+b = |X'|$, so $b < \frac{c(c-1)}{2}$. If source $s_{x,y}$ is in $S_{X'}$, then necessarily $v_{x,y} \in X'$. Consequently, as r sources have to be separated from t, set B' shall contain at least $r = \frac{c(c-1)}{2}$ elements which yields a contradiction.

Therefore, $a = c$ and $b = \frac{c(c-1)}{2}$. Now we prove that K' is a clique, *i.e.* if $(x, y) \in K'^2$, then $(x, y) \in E(G)$. As $|X' \cap V_1| = c$, then there are at most $\frac{c(c-1)}{2}$ sources disconnected from t (one source $s_{x,y}$ for any pair $(v_x, v_y) \in X'^2$). In summary, if $(x, y) \in K'^2$ then $(v_x, v_y) \in X'^2$ and $s_{x,y} \in S_{X'}$ which is only possible if $(x, y) \in E(G)$. □

Minor changes in the above proof lead to the conclusion that DIRECTED POTC$\langle r, p\rangle$ (POTC$\langle r, p\rangle$ on directed graphs) is W[1]-hard for both edge and vertex versions. For VERTEX DIRECTED POTC, it suffices to orient all edges towards the target and the proof does not change. For EDGE DIRECTED POTC, an extra modification consists in replacing all vertices in $V_1 \cup V_2$ by arcs and replacing all edges by p arcs in parallel in order to obtain W[1]-hardness.

2.2.3 Summary

We have not discussed the complexity of POTC for parameter k yet. In fact, a trivial FPT$\langle k\rangle$ algorithm exists. For small k and $r < k$, it is sufficient to enumerate all sets of sources $S' \subseteq S$ with cardinality $|S'| \geq r$ and compute the minimum (S', t)-cut for all of them. Therefore, POTC is solvable in time $O^*(2^k)$ for its vertex and edge versions. Table 2.1 summarizes the complexity results we established for POTC.

Parameters	EDGE POTC	VERTEX POTC
k	**FPT**: 2^k	**FPT**: 2^k
r	**W[1]-hard**, Theorem 2.8	**W[1]-hard**, Theorem 2.9
p	**FPT**: $2^{O(p^2)}$, Theorem 2.7	**W[1]-hard**, Theorem 2.9
r, p	**FPT**[a]: $2^{O(p^2)}$, Theorem 2.7	**W[1]-hard**, Theorem 2.9

[a] Another FPT algorithm designed specifically for both parameters r and p performs $2^{O(r)}4^p$, see Theorem 2.4

Table 2.1: Overview on the complexity of POTC regarding the parameters k, r, p and $p + r$.

This table summarizing the results on POTC for parametrizations involving k, p, and r is complete. We observe a difference between the complexity of EDGE POTC and VERTEX POTC when p is a parameter. Such a complexity gap on cut problems also exists for the problem called MINIMUM BISECTION [39]. Given a graph and parameter p, it asks for two sets A, B partitioning V such that $|A| = \lfloor \frac{n}{2} \rfloor$, $|B| = \lceil \frac{n}{2} \rceil$, and the number of edges between A and B is at most p. Our results confirm that the complexity of edge and vertex cut problems may differ. Perhaps this is also the case for PARTIAL MULTICUT, which is more general than POTC. A consequence of our hardness result for VERTEX POTC is that VERTEX PARTIAL MULTICUT is W[1]-hard for parameter p only. However, the complexity of EDGE PARTIAL MULTICUT parameterized by p remains open.

2.3 Counting minimum (S,T)-cuts

After the study of the cut decision problem POTC, we focus on one of the most natural counting cut problems, COUNTING MIN-(S,T)-CUTS, to find the exact number of minimum (S,T)-cuts in an undirected graph G. As it is #P-complete [7] and thus unlikely solvable in polynomial time, we study its fixed-parameter tractability. The size of the minimum (S,T)-cuts p is chosen as a parameter.
As with cut decision problems, the vertex version of this counting problem may be harder than the edge one (Theorem 2.2, page 20). Said differently, an FPT$\langle p\rangle$ algorithm for the counting of minimum vertex (S,T)-cuts can be used to solve the counting of minimum edge (S,T)-cuts in time FPT$\langle p\rangle$ as well.
COUNTING MIN-(S,T)-CUTS is FPT$\langle p\rangle$ because an algorithm can be deduced from two results reported in the literature [20, 62]. It performs in running time $O^*\left(2^{2^p}\right)$. Unfortunately, this double exponential makes it intractable even for small values of p. Our objective is to devise an algorithm which counts the minimum (S,T)-cuts in time $O^*(2^{\mathsf{poly}(p)})$, where poly is a polynomial function.
First, in Section 2.3.1, we propose an algorithm counting only the minimum edge (S,T)-cuts in $O^*\left(2^{O(p^2)}\right)$ [16]. It uses the concepts of drainages and dry instances that we defined. Second, in Section 2.3.2, we refined this algorithm in order to count minimum vertex (S,T)-cuts in a reasonable time. The concepts used to deal with edge cuts have been adapted to vertex (S,T)-cuts. Moreover, new concepts, as the local drainage, have been introduced for the second algorithm which counts the minimum vertex (S,T)-cuts in time $O^*\left(2^{O(p\log p)}\right)$ [12]. We lower the time complexity dedicated to the edge cut counting, as the edge-to-vertex reduction allows us to count the minimum edge (S,T)-cuts with this new contribution.

2.3.1 Counting minimum edge (S,T)-cuts with exponential factor $2^{O(p^2)}$

The first part of this section details the tools needed to design our algorithm counting minimum edge (S,T)-cuts in time $O^*\left(2^{O(p^2)}\right)$. The two fundamental concepts we introduce are the drainage and the dry instances. Then, we present our recursive algorithm and provide elements proving its FPT$\langle p\rangle$ complexity.

Construction of the drainage

We build the *drainage*, a collection of minimum cuts Z_i, $i \in \{1,\ldots,k\}$, where $k < n$, such that at least one edge of any minimum (S,T)-cut X belongs to $\bigcup_{i=1}^{k} Z_i$. First, we list the properties of the drainage. Second, we present the method to find this structure in polynomial time. In particular, we will construct cuts Z_i as the successive closest cuts of instance $\mathcal{I} = (G,S,T)$.
The drainage $\mathcal{Z}(\mathcal{I}) = (Z_1,\ldots,Z_k)$ of an instance $\mathcal{I} = (G,S,T)$ is a collection of disjoint minimum (S,T)-cuts Z_i, $|Z_i| = p$, satisfying the following properties:

- there are less than n cuts Z_i, *i.e.* $1 \leq k < n$,
- the source sides of cuts Z_i fulfil $R(Z_i,S) \subsetneq R(Z_{i+1},S)$ for $i \in \{1,\ldots,k-1\}$,
- for any minimum (S,T)-cut X, there is at least one cut Z_i which has edges with X in common: $X \cap Z_i \neq \emptyset$.

Observe that a similar tool was devised by Marx *et al.* in [62]. It is also a succession of cuts X_i satisfying $R(X_i,S) \subsetneq R(X_{i+1},S)$. Given an integer $\ell \geq p$, they compute in polynomial time a collection of (S,T)-cuts $X_1,\ldots,X_q$, $q \leq n$, such that any edge of a minimal (S,T)-cut of size at most ℓ is in $\cup_{i=1}^{q} X_i$. It characterizes all minimal (S,T)-cuts of size at most ℓ,

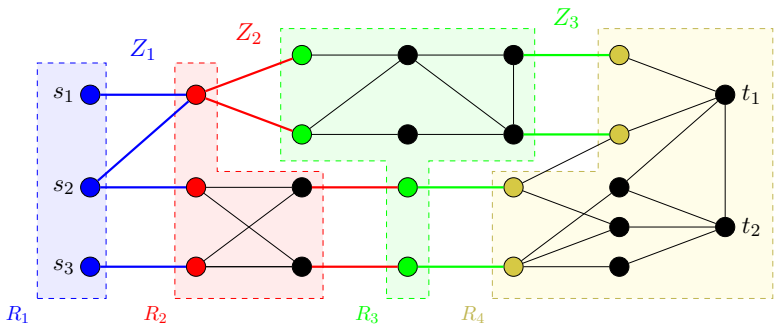


Figure 2.7: The drainage (cuts Z_i, sets S_i and R_i) for an instance containing graph G, sources $S = \{s_1, s_2, s_3\}$ and targets $T = \{t_1, t_2\}$. Here, $S_1 = R_1$ (in general, $S_1 \subseteq R_1$).

not only the minimum ones, although it may contain cuts X_i, X_{i+1} which are not disjoint, $X_i \cap X_{i+1} \neq \emptyset$. Our drainage, however, is such that any edge is in at most one cut Z_i. For this reason, our drainage not only characterizes minimum (S,T)-cuts but contains less cuts than the structure defined in [62] as well. The drainage property $Z_i \cap Z_{i+1} = \emptyset$, for any $i \in \{1, \ldots, k\}$, is a necessary condition as our algorithm benefits from it.
We construct the drainage iteratively. Let $S_1 = S$ and Z_1 be the minimum closest (S_1, T)-cut. We fix $R_1 = R(Z_1, S)$. Let S_2 be the set of vertices incident to edges of Z_1 inside $R(Z_1, T)$: $S_2 = V^T(Z_1) = \{v \notin R_1, (u, v) \in Z_1\}$.
Next, we construct Z_2 which is the minimum closest (S_2, T)-cut in $G \backslash R(Z_1, S)$. If $|Z_2| > p$, the drainage construction stops. Otherwise, if $|Z_2| = p$, set R_2 follows the same scheme as R_1, $R_2 = R(Z_2, S_2)$ in graph $G \backslash R(Z_1, S)$. We repeat the process until no more minimum (S_i, T)-cut Z_i of size p can be found. We denote by k the number of cuts Z_i produced and fix $R_{k+1} = R(Z_k, T)$. Cuts Z_i form the *minimum drainage cuts* of $\mathcal{I}$.
Figure 2.7 provides us with an example of graph G with $S = \{s_1, s_2, s_3\}$ and $T = \{t_1, t_2\}$ and indicates its drainage. The size of minimum (S,T)-cuts is $p = 4$. Blue, red, and green edges represent minimum drainage cuts Z_1, Z_2, and Z_3, respectively. Similarly, blue, red, green, and yellow vertices represent sets $S_1 = S$, S_2, S_3, and S_4. Source sides R_1, R_2, R_3, and R_4 are also appropriately colored. As the size of the minimum cut between S_4 (yellow vertices) and T in graph $G \backslash R(Z_3, S)$ is greater than p, we have $k = 3$.
We emphasize that set R_i, which is $R(Z_i, S_i)$ taken in $G \backslash R(Z_{i-1}, S)$, and set $R(Z_i, S)$ are different for $i \neq 1$. On the one hand, set $R(Z_i, S) = \bigcup_{\ell=1}^{i} R_\ell$ contains the vertices reachable from S in graph G deprived of Z_i. On the other hand, set R_i can be written $R_i = R(Z_i, S) \backslash R(Z_{i-1}, S)$. Sets R_i and R_{i+1} are disjoint and nonempty, as $S_i \subseteq R_i$ and $S_{i+1} \subseteq R_{i+1}$. Another crucial property is that the minimum drainage cuts are disjoint: $Z_i \cap Z_j = \emptyset$. The number k of minimum drainage cuts is less than n. The running time needed to construct the drainage is in $O(mnp)$.

Theorem 2.10. *The drainage $\mathcal{Z}(\mathcal{I})$ is obtained in time $O(mnp)$.*

Proof. According to Lemma 2.4, the minimum closest cut of any instance is computed in $O(mp)$. As there are less than n drainage cuts in $\mathcal{Z}(\mathcal{I})$, all cuts Z_i are retrieved in $O(mnp)$. □

The source sides of cuts Z_i are included one into another: $R(Z_i, S) \subsetneq R(Z_{i+1}, S)$. The following theorem shows that, for any minimum (S,T)-cut X, there is a cut Z_i containing edges of X. Among cuts Z_i sharing edges with X, we are interested in the one with the smallest index.

Definition 2.14 (Front of X)**.** *The front of a minimum (S,T)-cut X, $i(X)$, $1 \leq i(X) \leq k$ is the smallest index i such that $Z_i \cap X \neq \emptyset$.*

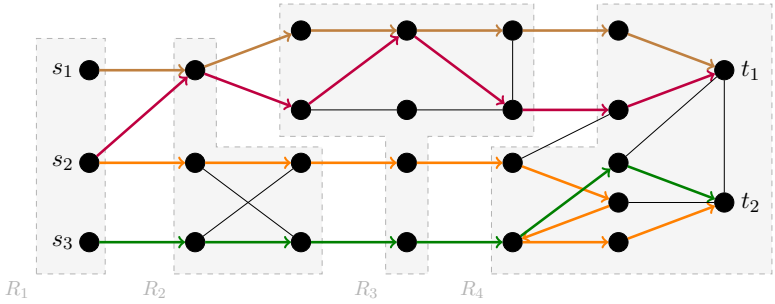


Figure 2.8: Menger's paths in graph G with sources $S = \{s_1, s_2, s_3\}$, targets $T = \{t_1, t_2\}$.

The next theorem states the properties of $i(X)$ for any minimum (S,T)-cut.

Theorem 2.11. *Any minimum (S,T)-cut X admits a front $i(X)$ and verifies $X \cap E\left[R(Z_{i(X)}, S)\right] = \emptyset$.*

Proof. First, cut X cannot be entirely included in $E\left[R_{k+1}\right]$. If it was, it would be a minimum (S_{k+1}, T)-cut of size p, which contradicts the drainage definition. So, some edges of X are incident to $R(Z_k, S)$.
Second, no edge of X belongs to $E\left[R(Z_1, S)\right]$ as cut Z_1 is the minimum closest (S,T)-cut. Therefore, there is an index $i \geq 1$ such that no edge of X belongs to $E\left[R(Z_i, S)\right]$ but at least one has an endpoint in R_{i+1}.
Obviously, if an edge of X belongs to Z_i, the theorem holds: $i = i(X)$. We study the case where no edge of X belongs to Z_i and there is an edge e of X, $e \in E\left[R_{i+1}\right]$. According to the definition of index i, no edge of X belongs to $E\left[R(Z_i, S)\right]$, and therefore all edges of X have to be on the target side $E\left[R(Z_i, T)\right]$. Therefore, X is a minimum (S_{i+1}, T)-cut in graph $G \backslash R(Z_i, S)$. Either cut X is a minimum closest (S_{i+1}, T)-cut (and then we fix $Z = X$) or the minimum closest (S_{i+1}, T)-cut Z is different than X and it fulfils $R(Z, S_{i+1}) \subsetneq R(X, S_{i+1})$. Since there is an edge $e \in X \cap E\left[R_{i+1}\right]$, then one of its endpoints $v \in R_{i+1}$ belongs to $R(X,T)$. Consequently, $v \notin R(Z, S_{i+1})$. This brings a contradiction: cutset Z_{i+1} is the unique minimum closest (S_{i+1}, T)-cut and $R(Z_{i+1}, S_{i+1}) = R_{i+1}$. As vertex v can be reached from S_{i+1} after the removal of Z_{i+1} but not after the removal of Z, cuts Z and Z_{i+1} differ. Thus, cut Z cannot be the minimum closest (S_{i+1}, T)-cut.
In summary, there is an index i such that no edge of X belongs to $E\left[R(Z_i, S)\right]$ and, moreover, $X \cap Z_i \neq \emptyset$. This means that there is no index $\ell < i$ such that $X \cap Z_\ell \neq \emptyset$. Consequently, index i is the front of X: $i = i(X)$. □

The reader can verify that any minimum (S,T)-cut of G contains some edges of at least one cut Z_1, Z_2, Z_3 in Figure 2.7.

Dry instances and closest dams

For any minimum edge (S,T)-cut X, we know that some of its edges of X also belong to one of the minimum drainage cuts Z_i. We introduce a concept called the *dry instance* which characterizes the edges of X which are not in Z_i but in its target side. The definition of the dry instance requires a certain preliminary effort.
We said in Section 2.1.3 that an arbitrary collection of p edge-disjoint (S,T)-paths $\mathcal{Q} = \{Q_1, \ldots, Q_p\}$ can be computed in time $O(mp)$. Paths $Q_1, \ldots, Q_p$, called Menger's paths, are used to devise our algorithm. In Figure 2.8, Menger's paths are indicated in the instance (G, S, T) depicted in Figure 2.7.
We begin by the definition of *dams* which are subsets of cuts Z_i of the drainage of G.

Definition 2.15 (Dam)**.** *A dam B_i is a nonempty subset of a minimum drainage cut Z_i,* i.e. *$B_i \subseteq Z_i$, $B_i \neq \emptyset$.*

Thanks to this definition, Theorem 2.11 together with the concept of the front makes us observe that any minimum (S,T)-cut X contains a *front dam*:

Definition 2.16 (Front dam)**.** *The front dam of a minimum (S,T)-cut X is $B_{i(X)} = X \cap Z_{i(X)}$.*

We know that all edges in $X \backslash B_{i(X)}$ belong to the target side of $Z_{i(X)}$, $E\left[R(Z_{i(X)}, T)\right]$, and the source side of $Z_{i(X)}$ is empty, $X \cap E\left[R(Z_{i(X)}, S)\right] = \emptyset$. If $X \backslash B_{i(X)} = \emptyset$, then $X = Z_{i(X)}$. A dam B_i is characterized by:

- its *level*, *i.e.* the index i of the cut Z_i it belongs to,
- its *signature* $\sigma(B_i) = \{Q_j : B_i \cap Q_j \neq \emptyset\}$, *i.e.* the set of Menger's paths passing through it.

Choking graph G with dam $B_{i(X)}$ puts in evidence a subgraph which still connects S and T through $X \backslash B_{i(X)}$. Our idea is to dam a graph gradually in order to dry it completely.
The description of the method we devised to reach this goal requires a transformation of G into G_D which is actually G with certain edges directed (G_D is a mixed graph). If edge e does not belong to a Menger's path, it stays undirected. For path $Q_j : v_1^{(j)} \cdot v_2^{(j)} \cdot v_3^{(j)} \cdots$, edges $(v_i^{(j)}, v_{i+1}^{(j)})$ become arcs $(v_i^{(j)}, v_{i+1}^{(j)})$, respecting the natural flow from sources to targets.
Figure 2.8 illustrates graph G_D. Arrows indicate the arcs while bare segments represent its edges. According to our preliminary work on Menger's paths in Lemma 2.1, any minimum (S,T)-cut of G is made up of arcs in G_D. Minimum drainage cuts Z_i are thus composed of arcs, directed from R_i to R_{i+1}. We insist on the fact that graph G_D is only used to define the notion of dry area, we do not count minimum cuts in it.

Definition 2.17 (Dry area)**.** *The dry area of B_i is the set $A^*(B_i)$ which contains the vertices of G which are not reachable from S in graph G_D deprived of B_i,* i.e. *$G_D \backslash B_i$.*

In a less formal way, set $A^*(B_i)$ keeps vertices which are dried as B_i is the only means to irrigate them. The definition of the dry instance follows.

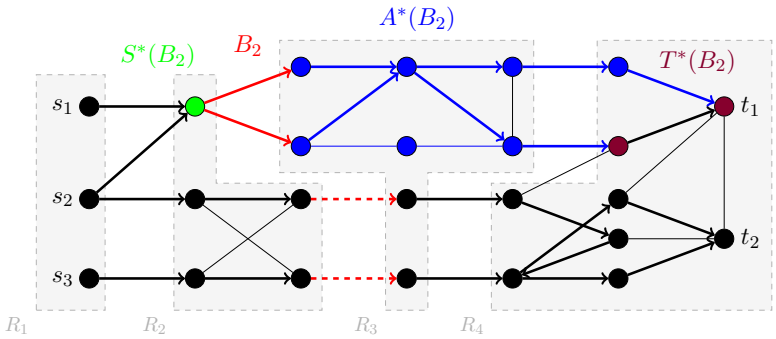


Figure 2.9: An example of dam B_2 and its dry instance $\mathcal{D}(\mathcal{I}, B_2) = (G^*(B_2), S^*(B_2), T^*(B_2))$

Definition 2.18 (Dry instance)**.** *The dry instance induced by a dam B_i is an instance $\mathcal{D}(\mathcal{I}, B_i) = (G^*(B_i), S^*(B_i), T^*(B_i))$ with graph $G^*(B_i) = (V^*(B_i), E^*(B_i))$. In particular,*

- *set $S^*(B_i)$ keeps vertices reachable from S "just before" dam B_i. Formally, it contains the tails of arcs in B_i: $S^*(B_i) = \{u : (u,v) \in B_i\}$,*
- *set $T^*(B_i)$ keeps vertices placed "after" dam B_i which become irrigated in $G_D \backslash B_i$. Formally, it contains the heads of arcs which have their tail either inside $S^*(B_i)$ or inside $A^*(B_i)$ and their head outside: $T^*(B_i) = \{v \notin A^*(B_i) : (u,v) \in E, u \in S^*(B_i) \cup A^*(B_i)\}$,*

- *set $V^*(B_i)$ is the union: $V^*(B_i) = S^*(B_i) \cup A^*(B_i) \cup T^*(B_i)$,*
- *set $E^*(B_i)$ stores edges of G which lie inside the dry area of B_i or on its border (one endpoint is outside) in G_D. Formally, it is composed of edges with two endpoints in $V^*(B_i)$ and at least one of them in $A^*(B_i)$: $E^*(B_i) = \{(u,v) \in E : u \in A^*(B_i), v \in V^*(B_i)\}$.*

Figure 2.9 gives an example of dam $B_2 \subseteq Z_2$ and the dry instance it induces in G. Its arcs are drawn in red, arcs of $Z_2 \backslash B_2$ are red and dashed. Blue vertices represent the vertices unreachable from S in $G_D \backslash B_2$, *i.e.* set $A^*(B_2)$. Sets $S^*(B_2)$ and $T^*(B_2)$ are drawn in green and purple, respectively. Set $E^*(B_2)$ is composed of dam B_2 (red arcs) and blue edges/arcs.
An important property of dry areas is that there is no arc (u,v) of G_D "entering" in the dry area $A^*(B_i)$, except for arcs in B_i.

Lemma 2.7. *For any dam B_i, there is no arc (u,v) in G_D such that $u \notin A^*(B_i)$ and $v \in A^*(B_i)$, except for arcs in B_i. Moreover, there is no undirected edge with exactly one endpoint in $A^*(B_i)$.*

Proof. Suppose that such an arc $(u,v) \notin B_i$ exists. As $u \notin A^*(B_i)$, it is reachable from S in $G_D \backslash B_i$. Therefore, v can be reached too: this contradicts $v \in A^*(B_i)$. For the same reason, there is no undirected edge (u,v) with only endpoint in $A^*(B_i)$ as the existence of this edge makes both its endpoints be reachable from S in $G_D \backslash B_i$. □

In Theorem 2.12 (page 37), we provide a characterization of any minimum (S,T)-cut which is based on dry instances and on *closest dams*. We start by:

Definition 2.19. *A dam B_h is closer than dam B_i if:*

- $h < i$,
- $\sigma(B_h) = \sigma(B_i)$,
- *edges in B_i are the only edges of level i inside the dry instance of B_h: $E^*(B_h) \cap Z_i = B_i$.*

As a consequence, the dry area of B_i is included in the dry area of B_h when B_h is closer than B_i: $A^*(B_i) \subsetneq A^*(B_h)$. Indeed, if a vertex is unreachable from S in $G_D \backslash B_i$, then it is also unreachable from S in $G_D \backslash B_h$ as arcs of B_i cannot be attained from S in $G_D \backslash B_h$ according to Definition 2.19.

Definition 2.20 (Closest dam)**.** *Dam B_i is a closest dam if no dam B_h, $h < i$ is closer than B_i.*

For any dam B_i, either B_i is a closest dam or there is a closest dam $B_h \neq B_i$, closer than B_i. Each dam B_i admits a closest dam (itself or B_h) which is unique.

Lemma 2.8. *Any dam B_i has a unique closest dam.*

Proof. If B_i is already a closest dam, then it is its own unique closest dam.
Now, we suppose that there are two closest dams of B_i, denoted by B_{h_1} and B_{h_2}. Necessarily, $h_1 \neq h_2$, otherwise $B_{h_1} = B_{h_2}$ as, according to Definition 2.19, they have the same signature.
We prove that under the hypothesis $h_1 < h_2$, dam B_{h_1} is closer than B_{h_2}. Suppose, towards a contradiction, that there is an arc $e_2 = (u_2, v_2) \in B_{h_2}$ which does not belong to set $E^*(B_{h_1})$. As $e_2 \notin E^*(B_{h_1})$, vertex u_2 is not inside $A^*(B_{h_1})$, otherwise, according to Definition 2.19, e_2 would belong to $E^*(B_{h_1})$. So, there is a path $\widehat{Q}$ connecting sources from

S with u_2, which avoids arcs in B_{h_1}. Arc e_2 belongs to a Menger's path Q_j. A section of this path, denoted by $\widehat{Q}'_j$, connects u_2 with a tail u_3 of an arc (u_3, v_3) in B_i, because Q_j passes through B_i: $\sigma(B_i) = \sigma(B_{h_2})$. The concatenated path $\widehat{Q} \cdot \widehat{Q}'_j$ connects S with B_i while avoiding B_{h_1}: this is a contradiction as $B_i \subseteq E^*(B_{h_1})$. Consequently, $B_{h_2} \subseteq E^*(B_{h_1}) \cap Z_{h_2}$. The equality $B_{h_2} = E^*(B_{h_1}) \cap Z_{h_2}$ comes from the fact that B_{h_1} and B_{h_2} have the same signature $\sigma(B_i)$. Arcs in dam $E^*(B_{h_1}) \cap Z_{h_2}$ belong to different Menger's paths. If $B_{h_2} \neq E^*(B_{h_1}) \cap Z_{h_2}$, then we have: $|\sigma(B_{h_2})| < |\sigma\left(E^*(B_{h_1}) \cap Z_{h_2}\right)| \leq |\sigma(B_{h_1})|$. Therefore, $B_{h_2} = E^*(B_{h_1}) \cap Z_{h_2}$, so B_{h_1} is closer than B_{h_2} which is contradictory to our assumption that B_{h_2} is a closest dam. □

Moreover, if B_h is a closest dam then its complement $\overline{B}_h = Z_h \backslash B_h$ is also a closest dam. This property will be used to prove the fixed-parameter tractability of our algorithm.

Lemma 2.9. *If B_h is a closest dam, then $\overline{B}_h = Z_h \backslash B_h$ is also a closest dam.*

Proof. Suppose that B_h is closest and $\overline{B}_h$ is not: let $\overline{B}_\alpha$ denote the closest dam of $\overline{B}_h$, $\alpha < h$.
First, we focus on the dry instance of dam $\overline{B}_\alpha$ between levels α and h. We prove that no edge/arc (u, v), with the exception of arcs from dam $\overline{B}_\alpha$, has one endpoint inside the dry instance of $\overline{B}_\alpha$ before level h (*i.e.* in the source side of cut Z_h) and one outside. We distinguish two cases:

- Case 1: If arc (u, v) is such that $u \notin A^*\left(\overline{B}_\alpha\right)$ and $v \in A^*\left(\overline{B}_\alpha\right)$, then Lemma 2.7 brings the contradiction. This argument also holds when (u, v) is undirected.
- Case 2: If arc (u, v) is such that $u \in A^*\left(\overline{B}_\alpha\right)$ is before level h and $v \notin A^*\left(\overline{B}_\alpha\right)$, then a Menger's path Q_j leaves the dry instance of $\overline{B}_\alpha$ through this arc. However, dams $\overline{B}_\alpha$ and $\overline{B}_h$ have the same signature, so path Q_j also contains an arc of $\overline{B}_h$ placed after arc (u, v). Consequently, there exists an arc (u', v'), $u' \notin A^*\left(\overline{B}_\alpha\right)$ and $v' \in A^*\left(\overline{B}_\alpha\right)$, to make path Q_j go back inside $\mathcal{D}(\mathcal{I}, \overline{B}_\alpha)$. This contradicts Case 1. Figure 2.10 illustrates the explanations given in Case 2 on a graph G with dams B_h, $\overline{B}_h$, B_α, and $\overline{B}_\alpha$. In this example, vertices u' and v are identical.

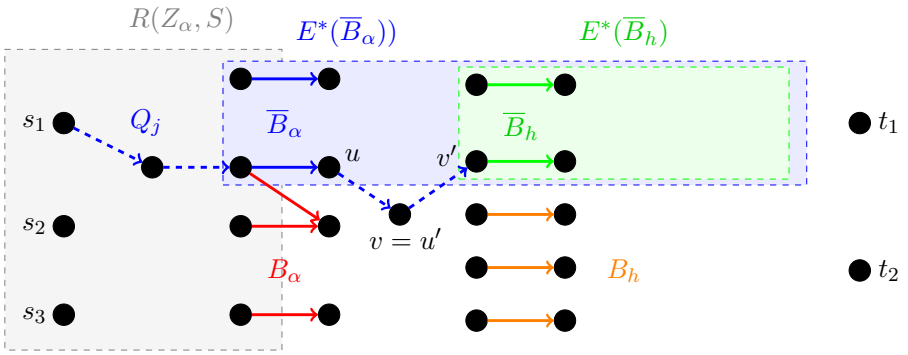


Figure 2.10: Illustration of the contradiction we arose for Case 2 in the proof of Lemma 2.9.

Second, we show that any vertex of $V^S(B_h)$ is unreachable from S in $G_D \backslash B_\alpha$, where $B_\alpha = Z_\alpha \backslash \overline{B}_\alpha$. We suppose that a path Q inside graph $G_D \backslash B_\alpha$ connects a source $s \in S$ with a vertex $w \in V^S(B_h)$. Path Q necessarily traverses level α, so it contains an arc (u, v) of $\overline{B}_\alpha$, as the complement dam B_α has been removed. As dam $\overline{B}_\alpha$ is closer than $\overline{B}_h$, vertices of $V^T(\overline{B}_\alpha)$ form a subset of $A^*\left(\overline{B}_\alpha\right)$, otherwise one vertex of $V^T(\overline{B}_\alpha)$ is reachable in $G_D \backslash B_\alpha$ and Menger's paths make a vertex in $V^S(\overline{B}_h)$ be reachable too, which is impossible. For this reason, path Q must contain a vertex $v \in A^*\left(\overline{B}_\alpha\right)$. Therefore, it connects a vertex v inside the dry instance of $\overline{B}_\alpha$ with vertex w which is outside. As a consequence, there

is an edge/arc leaving the dry instance of $\overline{B}_\alpha$ on path Q, which is a contradiction with our reasoning in Case 2.
Eventually, all vertices of $V^S(B_h)$ are unreachable from S in $G_D \backslash B_\alpha$, so arcs of B_h belong to the dry instance of B_α. Conversely, arcs of $\overline{B}_h$ do not belong to $E^*(B_\alpha)$, as the Menger's paths containing arcs of $\overline{B}_\alpha$ connect S with $\overline{B}_h$ despite the removal of B_α. Therefore, $E^*(B_\alpha) \cap Z_h = B_h$. Moreover, $\sigma\left(\overline{B}_\alpha\right) = \sigma\left(\overline{B}_h\right)$ as $\overline{B}_\alpha$ is closer than $\overline{B}_h$, so their complement dams also have the same signature: $\sigma(B_\alpha) = \sigma(B_h)$. Dam B_α is thus closer than B_h, which is a contradiction because B_h is supposed to be a closest dam. □

Observe that the dry areas of a dam B_i and of its complement $\overline{B}_i$, $A^*(B_i)$ and $A^*(\overline{B}_i)$ respectively, are disjoint because any vertex is reachable from S either in $G \backslash B_i$ or in $G \backslash \overline{B}_i$ or in both of them.
Theorem 2.12 provides us with the keystone to build our FPT$\langle p \rangle$ algorithm. It combines the concepts of dry instance and closest dam: given a minimum (S,T)-cut X and its front dam $B_{i(X)}$, either $X \backslash B_{i(X)} = \emptyset$ and $X = Z_{i(X)}$ or edges in $X \backslash B_{i(X)} \neq \emptyset$ belong to the dry instance of the dam $\overline{B}_{h(X)} = Z_{h(X)} \backslash B_{h(X)}$, where $B_{h(X)}$ is the closest dam of $B_{i(X)}$.

Theorem 2.12. *If $X \neq Z_{i(X)}$ is a minimum cut for $\mathcal{I}$, $B_{i(X)}$ its front dam, and $B_{h(X)}$ the closest dam of $B_{i(X)}$, then set $X \backslash B_{i(X)}$ is a minimum cut for the dry instance of $\overline{B}_{h(X)} = Z_{h(X)} \backslash B_{h(X)}$,* i.e. $\mathcal{D}\left(\mathcal{I}, \overline{B}_{h(X)}\right)$.

Proof. We prove that all edges in $X \backslash B_{i(X)}$ belong to the dry instance of $\overline{B}_{h(X)}$. Let us suppose *ad absurdum* that an edge $e = (u,v) \in X \backslash B_{i(X)}$ is reachable from S in graph G_D deprived of the dam $\overline{B}_{h(X)}$. Edge e is an arc (u,v) in G_D. We denote by P_e a path in G_D starting from a source $s \in S$ and terminating with (u,v), deprived of arcs of $\overline{B}_{h(X)}$, $P_e : s \cdots u \cdot v$.
Due to the iterative construction of cuts Z_i, path P_e necessarily contains one edge $e_h = (u_h, v_h)$ of level $h(X)$ when it arrives at a level greater than $i(X) \geq h(X)$. We know that this edge e_h does not belong to $\overline{B}_{h(X)}$, so $e_h \in B_{h(X)}$ and $P_e : s \cdots u_h \cdot v_h \cdots u \cdot v$. From now on, we focus on the segment of path P_e, denoted by $P_e^{(h)}$, which contains all edges between e_h and e, *i.e.* $P_e^{(h)} : u_h \cdot v_h \cdots u \cdot v$. The proof goes in two steps:
Step 1: Path $P_e^{(h)}$ contains an arc of $B_{i(X)}$.
Step 2: The existence of a path in G_D containing both an arc of $B_{i(X)}$ (proven in Step 1) and arc e contradicts the definition of cut X.
For Step 1, let us suppose that the path $P_e^{(h)}$ does not contain arcs of $B_{i(X)}$. This means that path $P_e^{(h)}$ passes by the dry instance of $B_{h(X)}$ between levels $h(X)$ and $i(X)$, otherwise it would necessarily contain an arc of $B_{i(X)}$. So, there is an edge/arc of this path, $\tilde{e} = (\tilde{u}, \tilde{v})$, where $\tilde{u}$ is in the dry area of $B_{h(X)}$ but not in the dry area of $B_{i(X)}$, and $\tilde{v}$ lies outside the dry area of $B_{h(X)}$. In brief, $\tilde{u} \in A^*(B_{h(X)}) \backslash A^*(B_{i(X)})$ and $\tilde{v} \notin A^*(B_{h(X)})$. Edge $\tilde{e}$ must be an arc $(\tilde{u}, \tilde{v})$ in G_D according to Lemma 2.7. Arc $\tilde{e}$ belongs to a Menger's path $Q_{j(\tilde{e})}$ passing through $B_{h(X)}$, *i.e.* $Q_{j(\tilde{e})} \in \sigma(B_{h(X)})$. Path $Q_{j(\tilde{e})}$ must traverse an arc of $B_{i(X)}$ as $\sigma(B_{i(X)}) = \sigma(B_{h(X)})$. As a consequence, path $Q_{j(\tilde{e})}$ connects a vertex $\tilde{v}$ outside the dry area of $B_{h(X)}$ with the tail u_i' of an arc $e_i' = (u_i', v_i')$ of $B_{i(X)}$. This is a contradiction, as vertex u_i' is supposed not to be reachable from S in G_D deprived of cut $B_{h(X)}$. Path $P_e^{(h)}$ thus contains an arc of $B_{i(X)}$ denoted by $e_i = (u_i, v_i)$.
For Step 2, let $\widehat{e} = (\widehat{u}, \widehat{v}) \neq e_i$ be the first arc of cut X in path $P_e^{(h)}$ which arrives after e_i on this path (Figure 2.11). In this way, we ensure that no edge of X lies on path $P_e^{(h)}$ between vertices v_i and $\widehat{u}$. Arc $\widehat{e}$ exists as e is a potential candidate to be one.
As vertex v_i is the head of e_i in graph G_D, $v_i \in R(X,T)$ according to Lemma 2.1 in page 18. For the same reason, vertex $\widehat{u} \in R(X,S)$ as it is the tail of arc $\widehat{e} \in X$. We know that path $P_e^{(h)}$ in G_D connects these two vertices and there is no arc of X on the segment of $P_e^{(h)}$

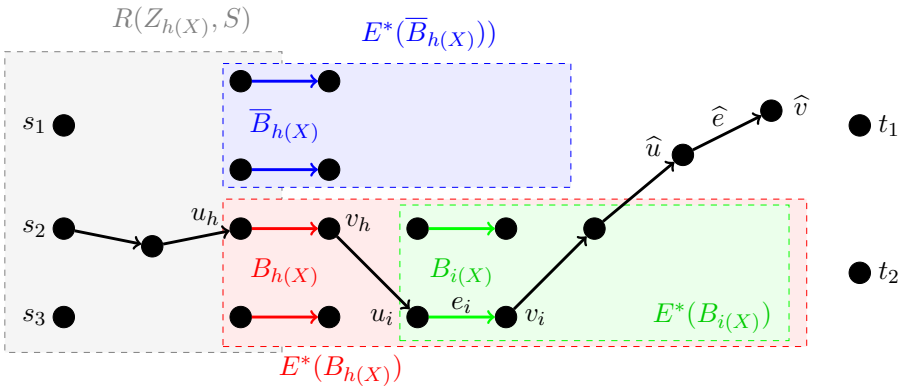


Figure 2.11: Illustration of the segment of path P_e between S and $\widehat{e}$, traversing dams B_h and B_i.

connecting them. Let us go back now to the initial undirected graph G. In graph $G \backslash X$, vertices $v_i \in R(X,T)$ and $\widehat{u} \in R(X,S)$ are connected whereas they must be separated by X. The presence of an edge (u,v) of X outside $E^*(\overline{B}_{h(X)})$ yields a contradiction.
In summary, all edges in $X \backslash B_{i(X)}$ belong to the dry instance of $\overline{B}_{h(X)}$. They necessarily form a cut in instance $\mathcal{D}\left(\mathcal{I}, \overline{B}_{h(X)}\right)$, otherwise X would not separate S and T in $\mathcal{I}$. The $p - \left|B_{h(X)}\right|$ Menger's paths in signature $\sigma(\overline{B}_{h(X)})$ are edge-disjoint inside $\mathcal{D}\left(\mathcal{I}, \overline{B}_{h(X)}\right)$, so the minimum cut size of this instance is greater than $p - \left|B_{h(X)}\right|$. As $X \backslash B_{i(X)}$ contains $p - \left|B_{i(X)}\right| = p - \left|B_{h(X)}\right|$ edges, we conclude that it is a minimum cut of $\mathcal{D}\left(\mathcal{I}, \overline{B}_{h(X)}\right)$. □

Therefore, any minimum (S,T)-cut X, which is not a minimum drainage cut $Z_{i(X)}$ itself, can be partitioned into two sets, $B_{i(X)}$ and $X \backslash B_{i(X)}$, such that:

- $B_{i(X)}$ is a minimum cut of instance $\mathcal{D}\left(\mathcal{I}, B_{h(X)}\right)$ and a dam of $\mathcal{I}$,
- $X \backslash B_{i(X)}$ is a minimum cut of instance $\mathcal{D}\left(\mathcal{I}, \overline{B}_{h(X)}\right)$ and all its edges belong to the target side of $Z_{i(X)}$, $E\left[R(Z_{i(X)}, T)\right]$.

Conversely, given a closest dam B_h and its complement $\overline{B}_h = Z_h \backslash B_h$, the union $B_i \cup X_{\overline{B}_h}$, where the closest dam of B_i is B_h and $X_{\overline{B}_h}$ is a minimum cut of $\mathcal{D}\left(\mathcal{I}, \overline{B}_h\right)$, separates S from T.

Theorem 2.13. *Let B_h be a closest dam of $\mathcal{Z}(\mathcal{I})$ and $\overline{B}_h = Z_h \backslash B_h$. Let B_i be a dam such that B_h is closer than B_i and $X_{\overline{B}_h}$ a minimum cut of $\mathcal{D}\left(\mathcal{I}, \overline{B}_h\right)$. Then, $B_i \cup X_{\overline{B}_h}$ is a minimum (S,T)-cut for instance $\mathcal{I}$.*

Proof. As dam B_h is closer than B_i, the edges of B_i form a minimum cut of $\mathcal{D}(\mathcal{I}, B_h)$. Indeed, they are the edges of level i inside $\mathcal{D}(\mathcal{I}, B_h)$, so they separate $S^*(B_h)$ from $T^*(B_h)$. Moreover, we know there is a set of $|\sigma(B_h)|$ edge-disjoint paths from $S^*(B_h)$ to $T^*(B_h)$ in $\mathcal{D}(\mathcal{I}, B_h)$. As $|\sigma(B_h)| = |\sigma(B_i)| = |B_i|$, set B_i is a minimum cut of instance $\mathcal{D}(\mathcal{I}, B_h)$.
We suppose that there is an open (S,T)-path Q in undirected graph G deprived of edges $B_i \cup X_{\overline{B}_h}$. Path Q cannot avoid level h of the drainage and passes through one edge of Z_h. As $B_h \cup \overline{B}_h = Z_h$, some edges of path Q belong either to the dry instance of B_h or to the dry instance of $\overline{B}_h$, or to both of them.
First, from Lemma 2.7 we know that no edge of graph G has one endpoint in the dry area $A^*(B_h)$ of B_h and the another one in the dry area $A^*(\overline{B}_h)$ of $\overline{B}_h$.
Second, we show that the existence of path Q yields a contradiction with the definition of the dry instance. As sets $A^*(B_h)$ and $A^*(\overline{B}_h)$ cannot be connected by an edge of G, path Q "traverses" completely at least one of the dry instances $\mathcal{D}(\mathcal{I}, B_h)$ or $\mathcal{D}\left(\mathcal{I}, \overline{B}_h\right)$, with no

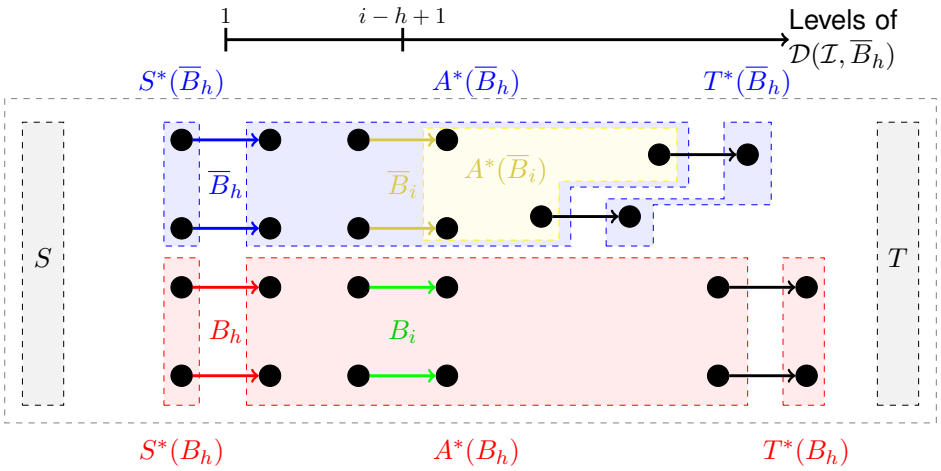


Figure 2.12: Illustration of Theorem 2.14: for any minimum cut with the front dam B_i, the tails of arcs in $X \backslash B_i$ belong to the yellow zone.

loss of generality we say $\mathcal{D}(\mathcal{I}, B_h)$. In other words, a segment of Q connects $S^*(B_h)$ and $T^*(B_h)$. This is not possible because dam B_i separates these two sets of vertices. With the dry instance $\mathcal{D}\left(\mathcal{I}, \overline{B}_h\right)$, we obtain the same contradiction as $X_{\overline{B}_h}$ separates $S^*(\overline{B}_h)$ and $T^*(\overline{B}_h)$. Therefore, $B_i \cup X_{\overline{B}_h}$ separates S from T with p edges, as $\left|X_{\overline{B}_h}\right| = |\sigma(\overline{B}_h)| = p - |B_i|$. □

We now prove a stronger result for set $X \backslash B_{i(X)}$. In fact, edges of set $X \backslash B_{i(X)}$ lie in the target side of level $i(X) - h(X) + 1$ in the drainage of instance $\mathcal{D}\left(\mathcal{I}, \overline{B}_{h(X)}\right)$. This statement is formulated in the theorem below.

Theorem 2.14. *Let X be a minimum (S,T)-cut of G and let $(Z'_1, \ldots, Z'_{k'})$ be the drainage of instance $\mathcal{D}\left(\mathcal{I}, \overline{B}_{h(X)}\right)$. Then, set $Z'_{i(X)-h(X)+1}$ is equal to $\overline{B}_{i(X)} = Z_{i(X)} \backslash B_{i(X)}$ and edges $X \backslash B_{i(X)}$ belong to the target side of $Z'_{i(X)-h(X)+1}$ inside instance $\mathcal{D}\left(\mathcal{I}, \overline{B}_{h(X)}\right)$.*

Proof. According to Theorem 2.12, we know that edges of $X \backslash B_{i(X)}$ belong to the dry instance of $\overline{B}_{h(X)}$, *i.e.* $E^*(\overline{B}_{h(X)})$. Moreover, they are also in the target side of $Z_{i(X)}$, as $B_{i(X)} \subsetneq Z_{i(X)}$ is the front dam of X. We denote by $\overline{B}_{i(X)}$ the complement of $B_{i(X)}$ in $Z_{i(X)}$, $\overline{B}_{i(X)} = Z_{i(X)} \backslash B_{i(X)}$. We want to prove that dam $\overline{B}_{i(X)}$ is the minimum drainage cut of level $i(X) - h(X) + 1$ in instance $\mathcal{D}\left(\mathcal{I}, \overline{B}_{h(X)}\right)$. For this purpose, we first prove that $\overline{B}_{h(X)}$ is closer than $\overline{B}_{i(X)}$.

Dam $\overline{B}_{i(X)}$ has the same signature as $\overline{B}_{h(X)}$ because their respective complement fulfil $\sigma(B_{i(X)}) = \sigma(B_{h(X)})$. Moreover, we prove that arcs of $\overline{B}_{i(X)}$ are in the dry instance of $\overline{B}_{h(X)}$. Suppose an arc $e_i = (u_i, v_i)$ of $\overline{B}_{i(X)}$ does not belong to $E^*\left(\overline{B}_{h(X)}\right)$. Let Q_j be the Menger's path containing arc e_i. Then, no arc of Q_j after e_i is inside instance $\mathcal{D}\left(\mathcal{I}, \overline{B}_{h(X)}\right)$. This is a contradiction as path Q_j must contain an edge of cut X. Indeed, path Q_j contains neither an arc of $B_{i(X)}$ as $\sigma(B_{i(X)}) \cap \sigma(\overline{B}_{i(X)}) = \emptyset$ nor an arc of $X \backslash B_{i(X)}$ as all its arcs after level $i(X)$ do not belong to $\mathcal{D}\left(\mathcal{I}, \overline{B}_{h(X)}\right)$ (contradiction with Theorem 2.12). In summary, dam $\overline{B}_{h(X)}$ is closer than $\overline{B}_{i(X)}$.

All dams $\overline{B}_\ell$ such that $\sigma(\overline{B}_{h(X)}) = \sigma(\overline{B}_\ell)$ and $h(X) < \ell < i(X)$ have a common closest dam: $\overline{B}_{h(X)}$. Indeed, if it is not the case for a dam $\overline{B}_\ell$, there is an arc $(u_\ell, v_\ell) \in \overline{B}_\ell$ where $u_\ell \notin A^*\left(\overline{B}_{h(X)}\right)$. As a consequence, dam $\overline{B}_{i(X)}$ is not contained in $E^*(\overline{B}_{h(X)})$, as arc (u_ℓ, v_ℓ) belongs to a Menger's path containing an arc of $\overline{B}_{i(X)}$.

We focus now on the drainage of instance $\mathcal{D}\left(\mathcal{I}, \overline{B}_{h(X)}\right)$. The minimum drainage cut of level one in this instance is $\overline{B}_{h(X)}$ itself, as it is the minimum closest $\left(S^*(\overline{B}_{h(X)}), T^*(\overline{B}_{h(X)})\right)$-cut.

Let $\overline{B}_{h(X)+1}$ be the dam of level $h(X)+1$ in $\mathcal{I}$ fulfilling $\sigma(\overline{B}_{h(X)}) = \sigma(\overline{B}_{h(X)+1})$ and $B_{h(X)+1}$ its complement. We prove that $\overline{B}_{h(X)+1}$ is the minimum drainage cut of level two in $\mathcal{D}\left(\mathcal{I}, \overline{B}_{h(X)}\right)$, *i.e.* it is the minimum closest cut between $V^T(\overline{B}_{h(X)})$ and $T^*(\overline{B}_{h(X)})$ in graph $G^*(\overline{B}_{h(X)})$ deprived of $R\left(\overline{B}_{h(X)}, S^*(\overline{B}_{h(X)})\right)$. Suppose that there is another minimum closest cut $Z_{\overline{B}_{h(X)}} \neq \overline{B}_{h(X)+1}$. Set $X_{\overline{B}_{h(X)}} = Z_{\overline{B}_{h(X)}} \cup B_{h(X)+1}$ is a minimum (S,T)-cut of instance $\mathcal{I}$ according to Theorem 2.13. As its edges belong to the target side of $Z_{h(X)}$, it is a minimum $(S_{h(X)+1}, T)$-cut. Based on the definition of $Z_{\overline{B}_{h(X)}}$, the source side $X_{\overline{B}_{h(X)}}$ is necessarily included into the source side of $Z_{h(X)+1} = B_{h(X)+1} \cup \overline{B}_{h(X)+1}$ in graph $G \backslash R(Z_{h(X)}, S)$. This is a contradiction to the construction of the drainage, as $Z_{h(X)+1}$ is the unique minimum closest $(S_{h(X)+1}, T)$-cut in graph $G \backslash R(Z_{h(X)}, S)$. Consequently, $\overline{B}_{h(X)+1}$ is the minimum drainage cut of level two inside $\mathcal{D}\left(\mathcal{I}, \overline{B}_{h(X)}\right)$.

We can iterate these arguments on dams $\overline{B}_{h(X)+2}$, $\overline{B}_{h(X)+3}$, etc. For example, dam $\overline{B}_{h(X)+2}$, $\sigma(\overline{B}_{h(X)+2}) = \sigma(\overline{B}_{h(X)})$, is the minimum closest $\left(V^T(\overline{B}_{h(X)+1}), T\right)$-cut when graph $G^*(\overline{B}_{h(X)})$ is deprived of $R\left(\overline{B}_{h(X)+1}, S^*(\overline{B}_{h(X)})\right)$. Otherwise, it would imply that $Z_{h(X)+2}$ is not the minimum closest $(S_{h(X)+2}, T)$-cut in graph $G \backslash R(Z_{h(X)+1}, S)$, which contradicts the construction of the drainage. Eventually, dam $\overline{B}_{h(X)+2}$ is the minimum drainage cut of level three inside $\mathcal{D}\left(\mathcal{I}, \overline{B}_{h(X)}\right)$, dam $\overline{B}_{h(X)+3}$ of level four, etc. Then, dam $\overline{B}_{i(X)}$ is the minimum drainage cut of level $i(X) - h(X) + 1$ in instance $\mathcal{D}\left(\mathcal{I}, \overline{B}_{h(X)}\right)$.

Coming back to Theorem 2.12, the edges of $X \backslash B_{i(X)}$ belong to both sets $E^*(\overline{B}_{h(X)})$ and $E\left[R(Z_{i(X)}, T)\right]$. They are in the target side of dam $\overline{B}_{i(X)}$ inside instance $\mathcal{D}\left(\mathcal{I}, \overline{B}_{h(X)}\right)$. □

Description of the algorithm

Our algorithm starts by computing the drainage $\mathcal{Z}(\mathcal{I})$ and the Menger's paths of input instance $\mathcal{I}$. For all dams B_i, it counts the minimum cuts of size p in $\mathcal{I}$ which admit the front dam B_i. If $B_i \neq Z_i$, it does this recursively by counting the minimum cuts in instance $\mathcal{D}\left(\mathcal{I}, \overline{B}_h\right)$ which only contains edges from the target side of the internal level $i - h + 1$ of $\mathcal{D}\left(\mathcal{I}, \overline{B}_h\right)$, where B_h is the closest dam of B_i. The minimum cut size in $\mathcal{D}(\mathcal{I}, \overline{B}_h)$ is at most $p - 1$.

We denote by $C_0(\mathcal{I}) = C(\mathcal{I})$ the total number of minimum (S,T)-cuts of instance $\mathcal{I}$. We define $C_\ell(\mathcal{I})$ as the number of minimum cuts of instance $\mathcal{I}$ which are composed of edges from $E\left[R(Z_\ell, T)\right]$ only. For example, $C_2(\mathcal{I})$ gives the number of minimum (S,T)-cuts in instance $\mathcal{I}$ without edges of $Z_1 \cup Z_2$. Value $C_\ell(\mathcal{I})$, $0 \leq \ell \leq k-1$, can be written:

$$C_\ell(\mathcal{I}) = k - \ell + \sum_{\substack{\text{Closest} \\ \text{dam } B_h \subsetneq Z_h}} \sum_{\substack{i \,:\, i > \ell, \\ \exists B_i : B_h \\ \text{closer than } B_i}} C_{i-h+1}\left(\mathcal{D}\left(\mathcal{I}, \overline{B}_h\right)\right). \tag{2.2}$$

The first $k - \ell$ cuts are the minimum drainage cuts of $\mathcal{I}$ with level greater than ℓ, *i.e.* cuts $Z_{\ell+1}, \ldots, Z_k$. The second term counts cuts taking edges only from $E\left[R(Z_\ell, T)\right]$ and admitting a front dam $B_{i(X)} \neq Z_{i(X)}$. Theorems 2.12 and 2.14 guarantee that any of these minimum (S,T)-cuts is counted at least once. Indeed, for any front dam B_i and its closest dam B_h, we compute the number of cuts in instance $\mathcal{D}\left(\mathcal{I}, \overline{B}_h\right)$ such that all their edges belong to the target side of $\overline{B}_i$, which is the internal level $i - h + 1$ in $\mathcal{D}\left(\mathcal{I}, \overline{B}_h\right)$. In the event that the drainage of $\mathcal{D}\left(\mathcal{I}, \overline{B}_h\right)$ has less than $i - h + 1$ levels, then $C_{i-h+1}\left(\mathcal{D}\left(\mathcal{I}, \overline{B}_h\right)\right) = 0$, as it means no minimum cut of $\mathcal{I}$ has the front dam B_i.

Conversely, the unicity of a closest dam ensures us that each minimum cut is counted exactly once. A minimum (S,T)-cut $X \neq Z_{i(X)}$ has a unique front dam $B_{i(X)}$ and the closest dam $B_{h(X)}$ of $B_{i(X)}$ is unique (Lemma 2.8). Finally, Theorem 2.13 guarantees that all cuts counted with Eq. (2.2) are minimum (S,T)-cuts.

Value $C_0(\mathcal{I})$ is computed thanks to recursive calls on multiple instances $\mathcal{D}\left(\mathcal{I}, \overline{B}_h\right)$. From now on, we distinguish the input instance $\mathcal{I}$ (for which we want to compute $C_0(\mathcal{I})$) with other instances (denoted by $\mathcal{J}$ later on) of the recursive tree. The base cases of the recursion, *i.e.* the leaves of the recursive tree, are the computation of values $C_\ell(\mathcal{J})$ either in instances $\mathcal{J}$ where the minimum cut size is one or in instances where no minimum cut admits a front dam $B_i \neq Z_i$, $i > \ell$. In both cases, the only minimum cuts of $\mathcal{J}$ are its minimum drainage cuts. Each recursive call of the algorithm makes the minimum cut size decrease: for example, if the minimum cut size of $\mathcal{J}$ is q, then it is $\left|\overline{B}_h\right| < q$ for an instance $\mathcal{D}\left(\mathcal{J}, \overline{B}_h\right)$. Therefore, the recursive tree is not deeper than $p-1$.

Figure 2.13 illustrates the recursive scheme of our algorithm with a tree describing the relationship between the instances. Three instances $\mathcal{I}$, $\mathcal{J}$, and $\mathcal{J}'$ of the recursive tree are represented. For example, instance $\mathcal{J} = \mathcal{D}\left(\mathcal{I}, \overline{B}_h\right)$ is the son of instance $\mathcal{I}$ in the tree as it is one of its dry instances.

In parallel, another graph (black dashed arcs in Figure 2.13) contains arcs with endpoints $C_\ell\left(\mathcal{J}\right)$. An arc connects two "compartments" $C_\ell\left(\mathcal{J}\right)$ and $C_{\ell'}\left(\mathcal{J}'\right)$ when the computation of $C_\ell\left(\mathcal{J}\right)$ depends on $C_{\ell'}\left(\mathcal{J}'\right)$. The minimum (S,T)-cut size of $\mathcal{J}'$ is smaller than the one of $\mathcal{J}$. This is why the graph made of arcs between compartments is a DAG, *i.e.* a directed graph without oriented cycles.

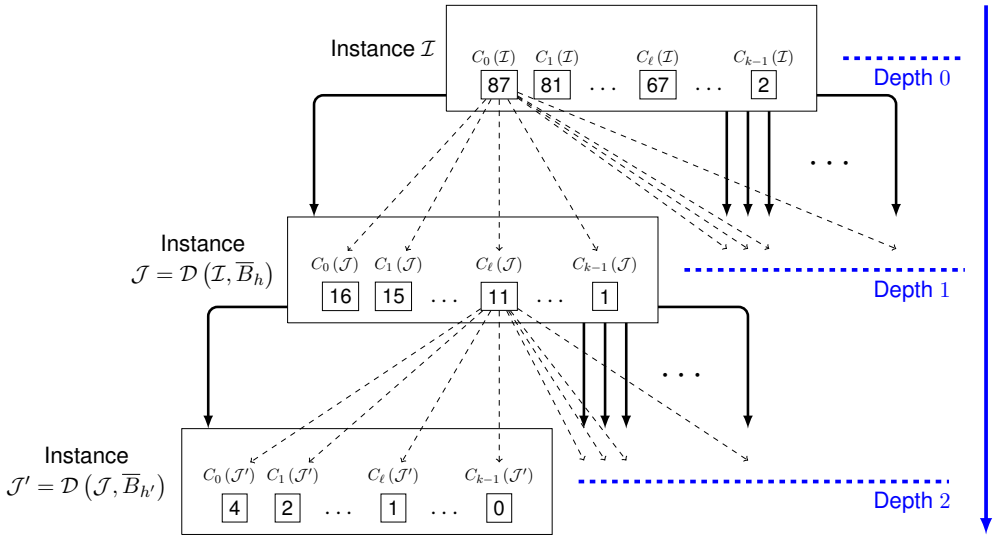


Figure 2.13: Recursive calls used to compute values $C_\ell\left(\mathcal{I}\right)$.

Then, we present the proof of Theorem 2.15 which allows us to declare the fixed-parameter tractability of COUNTING MIN-(S,T)-CUTS.

Theorem 2.15. *There are at most $2^{p^2}m$ instances in the recursive tree.*

Proof. The depth of instance $\mathcal{I}$ in the recursive tree is zero, we say $\Delta(\mathcal{I}) = 0$. For any closest dam B_h of $\mathcal{Z}(\mathcal{I})$, the depth of the dry instance of B_h is one: $\Delta\left(\mathcal{D}\left(\mathcal{I}, \overline{B}_h\right)\right) = 1$. More generally, if $\mathcal{J}$ is an instance of depth $d \geq 0$ and B_h a closest dam of $\mathcal{Z}(\mathcal{J})$, then instance $\mathcal{D}\left(\mathcal{J}, B_h\right)$ is at depth $d+1$.

We prove that, for any edge e in graph G, there are at most 2^{pd} instances $\mathcal{J}$ of depth d such that edge e belongs to the graph of $\mathcal{J}$. This fact makes the total number of instances in the recursive tree be upper-bounded by $2^{p^2}m$.

We proceed by induction. There is one instance defined for depth $d = 0$: it is $\mathcal{I}$ and it obviously contains edge e, so the number of instances with depth $d = 0$ containing e is thus $2^{pd} = 1$. Let $d \geq 1$ and $\mathcal{J}'$ be an instance of depth d containing edge e: there is an instance $\mathcal{J}$ of depth $d-1$ and one of its closest dam B_h such that $\mathcal{J}' = \mathcal{D}\left(\mathcal{J}, \overline{B}_h\right)$. As the graph of instance $\mathcal{J}'$ is a subgraph of those of $\mathcal{J}$, the latter contains edge e.

Using the induction hypothesis, there are at most $2^{p(d-1)}$ instances $\mathcal{J}$ of depth $d-1$ containing edge e. Now, given an instance $\mathcal{J}$ with $\Delta(\mathcal{J}) = d-1$, we bound the number of dams $\overline{B}_h$ (they are closest dams according to Lemma 2.9) of $\mathcal{J}$ such that $\mathcal{D}\left(\mathcal{J}, \overline{B}_h\right)$ contains edge e. We distinguish two cases:

- Case 1: edge e belongs to a minimum drainage cut Z_i of instance $\mathcal{J}$. We focus on the dams B_i of level i containing edge e. Their cardinality is bounded by 2^p. The edges of level i belonging to the dry instance of $\overline{B}_h$, $\mathcal{D}\left(\mathcal{J}, \overline{B}_h\right)$, form one of these dams B_i. As each dam B_i admits a unique closest dam (Lemma 2.8), there cannot be more than 2^p closest dams $\overline{B}_h$ such that $\mathcal{D}\left(\mathcal{J}, \overline{B}_h\right)$ contains edge e.
- Case 2: edge e is located between two minimum drainage cuts Z_i and Z_{i+1}, $e \in R_{i+1}$. Consequently, the level of any closest dam $\overline{B}_h$ such that $\mathcal{D}\left(\mathcal{J}, \overline{B}_h\right)$ contains e is less than i: $i \geq h$. Therefore, the edges of level i belonging to the dry instance of $\overline{B}_h$ form a dam B_i. Thus, the argument used in Case 1 arises the same conclusion: there cannot be more than 2^p closest dams such that $\mathcal{D}\left(\mathcal{J}, \overline{B}_h\right)$ contains edge e.

Finally, the number of instances written as $\mathcal{J}' = \mathcal{D}\left(\mathcal{J}, \overline{B}_h\right)$ where $\Delta(\mathcal{J}) = d-1$ and $\mathcal{J}'$ contains e, is upper-bounded by $2^{p(d-1)}2^p = 2^{pd}$. We conclude that there are less than 2^{pd} instances of depth d containing edge e. The total number of instances is thus smaller than $\sum_{d=0}^{p-1} 2^{pd} m \leq 2^{p^2} m$. □

For any instance $\mathcal{J}$ of the recursive tree, the algorithm computes its drainage $\mathcal{Z}(\mathcal{J})$, its Menger's paths and all instances $\mathcal{D}(\mathcal{J}, B_h)$ where B_h is a closest dam of $\mathcal{Z}(\mathcal{J})$. This third operation is done by enumerating all dams B_i of $\mathcal{Z}(\mathcal{J})$, verifying whether there is another dam B_h which is closer than B_i, and (if B_i is a closest dam) identifying the vertices/edges of $\mathcal{D}(\mathcal{J}, B_i)$ thanks to a depth-first search in $G_D \backslash B_i$. As there are at most $2^p n$ dams in $\mathcal{Z}(\mathcal{J})$, its execution time is $O(2^{2p} n^3)$. As the drainage of any instance is obtained in $O(mnp)$ (Theorem 2.10), the overall complexity is $O\left(2^{p^2} m(mnp + 2^{2p} n^3)\right) = O\left(2^{p(p+2)} pmn^3\right)$.

Theorem 2.16. *The counting of minimum edge (S,T)-cuts can be solved in time $O(2^{p(p+2)} pmn^3)$ on undirected graphs.*

This counting algorithm is actually more powerful than needed. It can be used to sample the minimum (S,T)-cuts, *i.e.* to pick up a minimum (S,T)-cut at random.

Sampling minimum edge (S,T)-cuts

The resilience analysis of a network issued from a practical application would be exhaustive when all minimum (S,T)-cuts were enumerated. However, this might be infeasible as the number $C(\mathcal{I})$ of minimum (S,T)-cuts might be exponential. An alternative consists in selecting only one or several of these cuts without bias. The selection of several "typical" minimum (S,T)-cuts can be made through a sampling procedure. In other words, an algorithm sampling the minimum (S,T)-cuts returns one of these cuts following the uniform distribution.

We sketch the algorithm which produces one of the minimum (S,T)-cuts according to the uniform distribution over all minimum (S,T)-cuts. We run our counting algorithm and execute a post-processing, described below.

We distinguish the input instance $\mathcal{I}$ from the other instances $\mathcal{J}$ of the recursive tree. Our method to sample minimum cuts consists in searching in the recursive tree, already filled out with values $C_\ell(\mathcal{J})$ during the counting. A minimum cut of $\mathcal{I}$ is extracted thanks to a randomly driven descent in the recursive tree.

We start at root $C_0(\mathcal{I})$. With probability $\frac{k}{C_0(\mathcal{I})}$, the sampling algorithm returns one of the minimum drainage cuts of $\mathcal{Z}(\mathcal{I})$ taken uniformly over them. Said differently, each cut

Z_i has probability $\frac{1}{C_0(\mathcal{I})}$ to be produced. With probability $1 - \frac{k}{C_0(\mathcal{I})}$, we will go one step down the tree. Concretely, for any dam B_i of $\mathcal{Z}(\mathcal{I})$ and its closest dam B_h, we visit node $C_{i-h+1}\left(\mathcal{D}(\mathcal{I}, \overline{B}_h)\right)$ of depth 1 with probability $\frac{C_{i-h+1}\left(\mathcal{D}(\mathcal{I}, \overline{B}_h)\right)}{C_0(\mathcal{I})}$. The sampling algorithm returns the union of B_i with the cut obtained by a recursive call on $C_{i-h+1}(\mathcal{D}(\mathcal{I}, \overline{B}_h))$. The algorithm applied on $C_{i-h+1}(\mathcal{D}(\mathcal{I}, \overline{B}_h))$ either selects a minimum drainage cut of level greater than $i - h + 1$ in $\mathcal{D}(\mathcal{I}, \overline{B}_h)$ (uniform selection among these cuts) or visits a node at depth 2, etc.
In this way, we ensure that the minimum (S, T)-cuts are sampled uniformly. Indeed, a cut with front dam B_i is chosen with probability $\frac{C_{i-h+1}\left(\mathcal{D}(\mathcal{I}, \overline{B}_h)\right)}{C_0(\mathcal{I})}$ which is the ratio of the number $C_{i-h+1}\left(\mathcal{D}(\mathcal{I}, \overline{B}_h)\right)$ of minimum cuts with front dam B_i by the total number $C_0(\mathcal{I})$ of minimum cuts in instance $\mathcal{I}$.

2.3.2 Counting minimum vertex (S, T)-cuts with exponential factor $2^{O(p \log p)}$

After an algorithm which counts the minimum edge (S, T)-cuts, we propose another one which counts the minimum vertex (S, T)-cuts in time $O^*(2^{O(p \log p)})$. It outperforms the algorithm issued from the treewidth reduction whose running time is $O^*(2^{2^p})$ [20, 62]. Moreover, it improves our algorithm devised specifically for the edge cut counting and executed in time $O^*(2^{O(p^2)})$, as the new one can also be used to count the minimum edge (S, T)-cuts thanks to the edge-to-vertex reduction shown in Section 2.1.3.
This second algorithm also benefits from the structures exploited by the first one: the drainage and the dry instances. The specificity of the vertex cuts is that some edges, called *leaks*, leave the dry instance, while they do not exist for edge cuts. They disrupt the recursivity (Figure 2.13, page 41) put in place to count minimum edge (S, T)-cuts. We thus designed new concepts for vertex cuts in order to catch the undesired phenomena caused by the leaks: the local drainage and the enclosed instances.
We introduce properties distinctive for the minimum vertex (S, T)-cuts. Then, we propose a vertex version of the drainage defined for edges in Section 2.3.1. We extend it with the *local drainage*, a drainage which only contains cuts inside the source side of a given *frontier*. The definition of the dry instance is also adapted to vertex (S, T)-cuts and we present its generalization, the *enclosed instance*. Eventually, we describe our algorithm based on all these notions and show that its running time is $O^*(2^{(p \log p)})$.

Properties of minimum vertex (S, T)-cuts

We state a property concerning two distinct minimum (S, T)-cuts X_1 and X_2. According to Lemma 2.1, each of their edges belongs to exactly one Menger's path. We denote by $X_1^{(c)} \subseteq X_1$ the set containing the ancestors of X_2 on paths $\sigma(X_1^{(c)})$. Then, let $X_1^{(d)}$ denote the edges of X_1 which are the descendants of X_2 on paths $\sigma(X_1^{(d)})$. Cut X_1 can be seen, in presence of X_2, as a sum of disjoint vertex sets:

$$X_1 = X_1^{(c)} \cup (X_1 \cap X_2) \cup X_1^{(d)}.$$

The similar construction is applied to X_2 to produce sets $X_2^{(c)}$ and $X_2^{(d)}$. If $X_1^{(c)}$ is empty, then X_2 is closer than X_1. In Figure 2.14, red vertices belong to $X_1^{(c)}$ and $X_1^{(d)}$ while green ones belong to $X_2^{(c)}$ and $X_2^{(d)}$. The black vertex is in $X_1 \cap X_2$. Black arrows illustrate the natural orientation of the Menger's paths, despite the fact that graph G is undirected.
We define the *closer-cut product* (CCP) Π_c of X_1 and X_2 as the union of "source-side" sets $X_1^{(c)}$, $X_2^{(c)}$, and $X_1 \cap X_2$.

Definition 2.21. *The CCP of cuts X_1 and X_2 is:* $\Pi_c(X_1, X_2) = X_1^{(c)} \cup X_2^{(c)} \cup (X_1 \cap X_2)$.

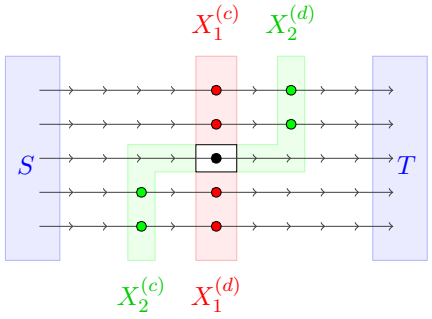


Figure 2.14: Sets $X_i^{(c)}$ and $X_i^{(d)}$ for two minimum X_i, $i \in \{1, 2\}$

The *dominating-cut product* (DCP) Π_{d} of X_1 and X_2 is the union of "target-side" sets $X_1^{(d)}$, $X_2^{(d)}$, and $(X_1 \cap X_2)$.

Definition 2.22. *The DCP of cuts X_1 and X_2 is:* $\Pi_d(X_1, X_2) = X_1^{(d)} \cup X_2^{(d)} \cup (X_1 \cap X_2)$.

Using these two terms we write the following property:

Lemma 2.10. *For two minimum (S,T)-cuts X_1 and X_2, sets $\Pi_c(X_1, X_2)$ and $\Pi_d(X_1, X_2)$ are minimum (S,T)-cuts.*

Proof. Suppose, towards a contradiction, that set $\Pi_{\mathsf{c}}(X_1, X_2)$ is not an (S,T)-cut. There is a simple (S,T)-path P which does not go either through $X_1^{(c)}$, or through $X_2^{(c)}$, or through $X_1 \cap X_2$. The initial vertex of P is thus in both $R(X_1, S)$ and $R(X_2, S)$ while its final vertex is in both $R(X_1, T)$ and $R(X_2, T)$. As X_1 and X_2 are (S,T)-cuts, path P must pass through at least one vertex of X_1 and at least one vertex of X_2. So, a vertex of $X_1^{(d)}$ and a vertex of $X_2^{(d)}$ are in P.
We go along path P from its start and "catch" the first vertex v of P which also is in $X_1^{(d)} \cup X_2^{(d)}$. We say, without loss of generality, that $v \in X_1^{(d)}$. According to Lemma 2.1, the vertices of $X_1^{(d)}$ are in the target side $R(X_2, T)$ of cut X_2. Therefore, the section of path P going from S to vertex v connects $R(X_2, S)$ and $R(X_2, T)$ without going through any vertex of X_2. This is a contradiction as X_2 is an (S,T)-cut.
Swapping sets S and T and inverting the direction of Menger's paths suffices to prove that $\Pi_{\mathsf{d}}(X_1, X_2)$ is an (S,T)-cut too as the graph is undirected. The size of cuts $\Pi_{\mathsf{c}}(X_1, X_2)$ and $\Pi_{\mathsf{d}}(X_1, X_2)$ is naturally at least p. As $|X_1| + |X_2| = |\Pi_{\mathsf{c}}(X_1, X_2)| + |\Pi_{\mathsf{d}}(X_1, X_2)| = 2p$, then these cuts are necessarily minimum. □

The second property captures the difference between minimum edge and vertex (S,T)-cuts. An edge (S,T)-cut X, not necessarily minimum, splits a connected graph into connected components which necessarily contain elements of $S \cup T$. The definition of X forbids, however, any of these connected components to contain vertices from both source and target sets.
For a vertex (S,T)-cut, connected components which do not contain vertices either from S or T can emerge in $G \backslash X$. We call *unreachable areas* all the connected components of $G \backslash X$ without any vertex from $S \cup T$. Their union is denoted by $R^*(X)$. With this definition, we have $R(X, S) \cup R(X, T) \cup R^*(X) \cup X = V$. Without loss of generality, we assume later on that $R^*(X)$ is actually a single connected component. Figure 2.15 illustrates such an inaccessible set $R^*(X)$. Our objective is to prove that, for any minimum vertex (S,T)-cut X, no Menger's path traverses an unreachable area of X. Consequently, according to Lemma 2.1, no other minimum (S,T)-cut $X' \neq X$ shares vertices with $R^*(X)$. This result allows us to neglect sets $R^*(X)$ in the study of minimum vertex cuts.

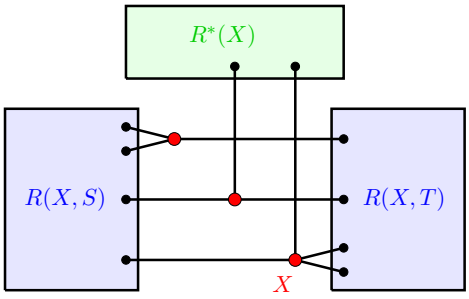


Figure 2.15: An example of single-component unreachable area $R^*(X)$

Lemma 2.11. *Let X be a minimum (S,T)-cut. No Menger's path of $\mathcal{Q}$ passes through the unreachable areas $R^*(X)$ of cut X.*

Proof. The neighborhood of set $R^*(X)$, *i.e.* the set $N(R^*(X))$ of vertices adjacent to $R^*(X)$, is made of vertices in X. A Menger's path Q_j passing through $R^*(X)$ must enter through a vertex of $N(R^*(X)) \subseteq X$ and leave through another vertex of $N(R^*(X))$ as Q_j is a simple path. For this reason, $|N(R^*(X))| \geq 2$. Therefore, there are two vertices $x_1, x_2 \in X$ such that $(x_1,u),(v,x_2) \in Q_j$ and $u,v \in R^*(X)$. Observe that Q_j, which traverses $R^*(X)$, cannot actually be a Menger's path as it contains two elements of X. □

In summary, for any minimum vertex (S,T)-cut X, the vertices of any other minimum vertex cut X' belong either to $R(X,S)$ or to $R(X,T)$.

Drainage for the minimum vertex (S,T)-cuts

The notion of drainage for minimum vertex (S,T)-cuts is inspired by the edge version given in Section 2.3.1. For now, we call it the *global drainage* and keep its prior notation $\mathcal{Z}(\mathcal{I})$ as in Section 2.3.1. The *local drainage* $\mathcal{Z}(\mathcal{I},U)$ is a new concept where the drainage construction is limited in function of a minimum (S,T)-cut U, called the *frontier*. All drainage cuts of the local drainage are in the augmented source side of U, *i.e.* $R^+(U,S) = R(U,S) \cup U$. The local drainage characterizes exclusively the minimum (S,T)-cuts with at least one vertex in $R^+(U,S)$. It is used by our algorithm in the remainder to point out the minimum cuts which are located in a certain area of the dry instance.

Global drainage. The global drainage $\mathcal{Z}(\mathcal{I}) = (Z_1,\ldots,Z_k)$ of an instance $\mathcal{I} = (G,S,T)$ is a collection of minimum vertex (S,T)-cuts Z_i, $|Z_i| = p$. It satisfies the properties listed for the edge case. These cuts are disjoint: $Z_i \cap Z_j = \emptyset$ when $i \neq j$. As a consequence, the collection $\mathcal{Z}(\mathcal{I})$ contains at most $\frac{n}{p}$ cuts Z_i, *i.e.* $k \leq \frac{n}{p}$. Moreover, the source sides of these cuts are included one into another: $R(Z_i,S) \subsetneq R(Z_{i+1},S)$.
We construct it iteratively: let $S_1 = S$, Z_1 be the minimum closest (S_1,T)-cut and $R_1 = R(Z_1,S)$. Then, we remove set R_1 from the graph and define a new set of sources $S_2 = Z_1$. The minimum closest (S_2,T)-cut is computed again and is denoted by Z_2. We remove R_2, the source side of Z_2 on the current graph, and fix $S_3 = Z_2$. At iteration $i+1$, we work on graph $G_{i+1} = G \backslash R(Z_i,S)$: $S_{i+1} = Z_i$ and Z_{i+1} is the minimum closest (S_{i+1},T)-cut in graph G_{i+1}. The construction terminates when the size of the minimum (S_{i+1},T)-cut is greater than p. The minimum (S,T)-cuts Z_i are called *minimum drainage cuts*. Eventually, we naturally fix $S_{k+1} = Z_k$ and $R_{k+1} = R(Z_k,T)$. Figure 2.16 illustrates the drainage of an instance $\mathcal{I} = (G,S,T)$ with $S = \{s_1,s_2,s_3\}$ and $T = \{t_1,t_2,t_3,t_4\}$. There are $k = 3$ minimum drainage cuts of size $p = 3$. Each color in the drawing refers to indices: blue for 1, etc.
Sets R_i are disjoint: when set R_{i+1} is deduced from the computation of Z_{i+1}, sets $R_1,\ldots,R_i$ have been already removed from the original graph G. For any minimum drainage cut Z_i, its

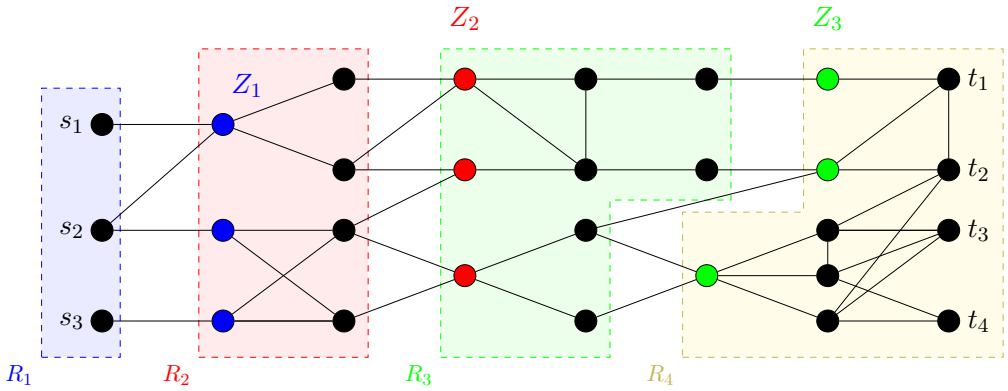


Figure 2.16: Drainage of an instance $\mathcal{I} = (G, S, T)$

source side $R(Z_i, S)$ in G is the union $\bigcup_{\ell=1}^{i} R_\ell$. The construction of the global drainage consists in k computations of minimum closest cuts, so its execution time is $O(mpk) = O(mn)$ according to Lemma 2.4. As for edges, the global drainage puts in evidence certain elements of any minimum (S, T)-cut. Now, the front dam is made up of vertices.

Theorem 2.17. *For any minimum (S, T)-cut X, there is a unique minimum drainage cut Z_i such that $X \cap Z_i \neq \emptyset$ and $X \cap R(Z_i, S) = \emptyset$.*

Proof. First, cut X cannot be entirely included in R_{k+1}. If it was, it would be a minimum (S_{k+1}, T)-cut of size p, which contradicts the drainage construction ending. As a consequence, some vertices of X belong to either Z_k or its source side $R(Z_k, S)$.
Second, no vertex of X belongs to $R(Z_1, S)$ as cut Z_1 is the minimum closest (S, T)-cut and is unique (Lemma 2.3). Therefore, there is an index $i \geq 1$ such that no vertex of X belongs to the source side of Z_i in G, *i.e.* $R(Z_i, S)$, but at least one is in R_{i+1}.
If a vertex of X belongs to Z_i, the theorem holds. We examine the case where no vertex of X belongs to Z_i but at least one is in $R_{i+1} \backslash Z_i$. According to the definition of index i, no vertex of X belongs to $R(Z_i, S)$. Therefore, X is a minimum (S_{i+1}, T)-cut in graph G_{i+1}. Either cut X is the minimum closest (S_{i+1}, T)-cut (and then we fix $Z = X$) or the minimum closest (S_{i+1}, T)-cut Z is different than X and it fulfils $R(Z, S_{i+1}) \subsetneq R(X, S_{i+1})$ in G_{i+1}. However, we know that Z_{i+1} is the minimum closest (S_{i+1}, T)-cut in G_{i+1}. In brief, we must have $Z = Z_{i+1}$. As at least one vertex $x \in X$ is in the source side of Z_{i+1} in G_{i+1}, *i.e.* R_{i+1}. this contradicts the fact that Z is closer than X in graph G_{i+1}.
In summary, there is an index i such that no vertex of X belongs to $R(Z_i, S)$ and, moreover, $X \cap Z_i \neq \emptyset$. This index is unique because Z_i and Z_j, $i < j$, are disjoint: $X \cap R(Z_j, S) \neq \emptyset$ as $Z_i \subseteq R(Z_j, S)$ and $X \cap Z_i \neq \emptyset$. □

We associate the notion of dam with the concept of global drainage. It follows:

Definition 2.23 (Dam of $\mathcal{Z}(\mathcal{I})$)**.** *A dam B_i is a nonempty subset of a minimum drainage cut Z_i,* i.e. *$B_i \subseteq Z_i$, $B_i \neq \emptyset$.*

For any minimum (S, T)-cut X, the set $X \cap Z_i$ mentioned in Theorem 2.17 is either Z_i (in this case, $X = Z_i$ is a minimum drainage cut) or is a dam $B_i \subsetneq Z_i$. For the latter, the *front index* of X, denoted by $i(X)$, is the index such that $X \cap Z_{i(X)} \neq \emptyset$ and $X \cap R(Z_{i(X)}, S) = \emptyset$. The definition of the front dam follows.

Definition 2.24 (Front dam in $\mathcal{Z}(\mathcal{I})$)**.** *The front dam of a minimum (S, T)-cut X is $B_{i(X)} = X \cap Z_{i(X)}$.*

From Theorem 2.17, we have that any minimum (S, T)-cut which is not a drainage cut admits a unique front dam.
Eventually, we list the properties of the global drainage presented so far.

Summary 1 (Properties of the global drainage)**.** *Collection $\mathcal{Z}(\mathcal{I})$, composed of the minimum drainage cuts Z_i, fulfils:*

- *cuts Z_i are disjoint: for any $i, j \in \{1, \ldots, k\}$, $i \neq j$, $Z_i \cap Z_j = \emptyset$,*
- *there are less than $\frac{n}{p}$ cuts Z_i: $k \leq \frac{n}{p}$,*
- *the reachable sets of cuts Z_i fulfil $R(Z_i, S) \subsetneq R(Z_{i+1}, S)$ for $i \in \{1, \ldots, k-1\}$,*
- *for any minimum (S,T)-cut X, there is a unique Z_i such that $X \cap Z_i \neq \emptyset$ and $X \cap R(Z_i, S) = \emptyset$. Set $B_i = X \cap Z_i$ is called the front dam of X when $B_i \subsetneq Z_i$.*

The existence of the unique front dam for any minimum (S,T)-cut $X \neq Z_i$ is the key for our algorithm. In brief, the vertex version of the global drainage has the same properties as the edge version. One difference is that value k is at most $\frac{n}{p}$ as the drainage cuts are now made up of vertices. However, the number of minimum edge drainage cuts was only upper-bounded by n or $\frac{m}{p}$.

Local drainage. The notion of drainage is generalized to a *local drainage* which depends not only on an instance $\mathcal{I}$ but also an arbitrary minimum (S,T)-cut U. We denote by $\mathcal{Z}(\mathcal{I}, U)$ the local drainage of $\mathcal{I}$ truncated on U, *i.e.* a collection of minimum (S,T)-cuts $Z_1^U, \ldots, Z_k^U$ that characterizes the minimum cuts of $\mathcal{I}$ which contain at least one vertex in $R^+(U,S) = R(U,S) \cup U$.
Cuts Z_i^U verify $R(Z_i^U, S) \subseteq R(U,S)$. Any minimum (S,T)-cut X with at least one vertex in $R^+(U,S)$ has a front dam: it contains vertices in a minimum drainage cut of $\mathcal{Z}(\mathcal{I}, U)$. The idea is to use the local drainage to count the minimum (S,T)-cuts present in different areas of a dry instance. On the one hand, the minimum (S,T)-cuts with a vertex in $R^+(U,S)$ are put in evidence by the local drainage. On the other hand, the minimum (S,T)-cuts included in $R(U,T)$ are counted in another instance, obtained after removing $R^+(U,S)$ and considering U as the new source set. This separation is the keystone of our algorithm. It allows us to obtain a slightly super-exponential time $O^*(2^{O(p \log p)})$.

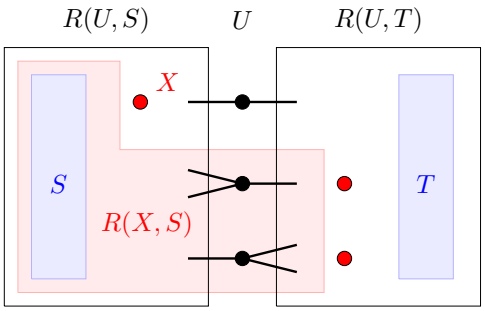


Figure 2.17: A frontier U and a minimum (S,T)-cut X, whose vertices are in red, where $X \cap R^+(U,S) \neq \emptyset$.

We call set U the *frontier* of the local drainage $\mathcal{Z}(\mathcal{I}, U)$. Figure 2.17 presents a structural view of an instance $\mathcal{I}$ with a frontier U and a cut X fulfilling $X \cap R^+(U,S) \neq \emptyset$. The local drainage $\mathcal{Z}(\mathcal{I}, U)$ is built to put in evidence the minimum (S,T)-cuts with at least one vertex in $R^+(U,S)$.
The tactic consists in computing the successive minimum closest (S,T)-cuts until one Z_i^U intersects U. When it happens, we force the vertices of U present in the drainage, *i.e.* $Z_i^U \cap U$, to be contained in all the next drainage cuts: Z_{i+1}^U, Z_{i+2}^U, etc. So, the construction starts as for the global drainage: we compute the successive minimum closest (S_i,T)-cut Z_i^U, but also sets S_i^U, R_i^U, and graph G_i^U in the same way than Z_i, S_i, R_i, and G_i, respectively. This process changes when a cut Z_i^U contains at least one vertex of U.

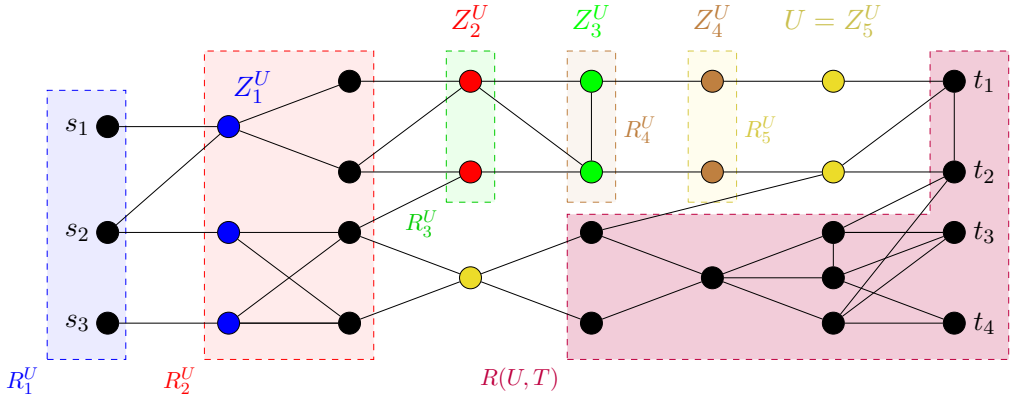


Figure 2.18: Local drainage $\mathcal{Z}(\mathcal{I}, U)$.

Suppose that $Z_i^U \cap U \neq \emptyset$. We fix $S_{i+1}^U = Z_i^U \backslash U$ and remove $Z_i^U \cap U$ from the graph. Graph G_{i+1}^U is obtained from G_i^U after the removal of vertices in sets R_i^U and $Z_i^U \cap U$. Cut Z_{i+1}^U is the union of the minimum closest (S_{i+1}^U, T)-cut with $Z_i^U \cap U$, if its size is p, otherwise the construction terminates. Naturally, the last drainage cut is the frontier, as we seek the cuts as close as possible from U: $Z_k^U = U$ (Lemma 2.12). In brief, the definition of sets S_i^U and R_i^U is: $S_{i+1}^U = Z_i^U \backslash U$ and $R_{i+1}^U = R(Z_{i+1}^U, S) \backslash R(Z_i^U, S)$.
The cuts in $\mathcal{Z}(\mathcal{I}, U)$ are not necessarily disjoint and, moreover, $Z_i^U \cap Z_j^U \subseteq U$ for $i \neq j$. Their source sides are included one into others: $R(Z_i^U, S) \subsetneq R(Z_{i+1}^U, S)$. For this reason, there are at most n minimum drainage cuts in $\mathcal{Z}(\mathcal{I}, U)$. No minimum drainage cut arrives after U, *i.e.* $R(Z_i^U, S) \cap U = \emptyset$ for every $i \in \{1, \ldots, k\}$. The following lemma states that the last drainage cut Z_k^U is exactly the frontier U: $Z_k^U = U$.

Lemma 2.12. *No vertex of the minimum drainage cuts in $\mathcal{Z}(\mathcal{I}, U)$ belongs to the target side $R(U,T)$ of U. Moreover, the last drainage cut of $\mathcal{Z}(\mathcal{I}, U)$ is the frontier U: $Z_k^U = U$.*

Proof. As Z_1^U is the minimum closest (S,T)-cut in G, all its vertices are in $R^+(U,S)$ by definition. Inductively, we suppose that Z_i^U is in $R^+(U,S)$ and we focus on cut Z_{i+1}^U. As $S_{i+1} = Z_i^U \backslash U$, set $U \backslash Z_i^U$ separates S_{i+1} from T, otherwise U would not be an (S,T)-cut.
The minimum closest (S_{i+1}, T)-cut in G_{i+1}, denoted by $\widehat{Z}_{i+1}^U$, is closer than $U \backslash Z_i^U$. Consequently, no vertex of $\widehat{Z}_{i+1}^U \subseteq Z_{i+1}^U$ is in $R(U,T)$. The other vertices of cut Z_{i+1}^U, *i.e.* in set $Z_i^U \cap U$ are also not in $R(U,T)$. Finally, no drainage cut in $\mathcal{Z}(\mathcal{I}, U)$ contains vertices of $R(U,T)$.
Suppose that $Z_k^U \neq U$. Vertices $Z_k^U \cap U$ are removed from the graph G_{k+1}^U. Its non-empty source set is $Z_k^U \backslash (Z_k^U \cap U)$. Set $U \backslash (Z_k^U \cap U)$ is a source-free minimum cut for graph G_{k+1}^U. As Z_k^U is supposed to be the last drainage cut, this is a contradiction. □

We give an example of local drainage in Figure 2.18. The frontier U is represented with yellow vertices. Cut Z_2^U contains not only the red vertices but also the yellow one of U just below. Cuts Z_3^U and Z_4^U also contain this vertex which was not drawn in their respective colors to keep the figure readable. Cut Z_2^U is the first drainage cut in $\mathcal{Z}(\mathcal{I}, U)$ to intersect the frontier U. In this illustration, $k = 5$ and $Z_5^U = U$.
Finally, we prove that any minimum (S,T)-cut X such that $X \cap R^+(U,S) \neq \emptyset$ admits a front dam in $\mathcal{Z}(\mathcal{I}, U)$.

Theorem 2.18. *For any minimum (S,T)-cut X such that $X \cap R^+(U,S) \neq \emptyset$, there is a minimum drainage cut Z_i^U such that $X \cap Z_i^U \neq \emptyset$ and $X \cap R(Z_i^U, S) = \emptyset$.*

Proof. Set U is a minimum (S,T)-cut and we know that $Z_k^U = U$ (Lemma 2.12). For any minimum (S,T)-cut X such that $U \not\subseteq R(X,S)$, no vertex of X can be in set $R_{k+1}^U = R(U,T)$. Said differently, at least one vertex of X belongs to $R^+(Z_k^U, S) = R^+(U,S)$.

Moreover, no vertex of X is inside $R(Z_1^U, S)$ because of the unicity of the minimum closest (S,T)-cut in G. There is an index $i \geq 1$ such that no vertex of X belongs to the source side of Z_i^U but at least one belongs to R_{i+1}^U. We examine the case where no vertex of X is in Z_i^U but one vertex $x \in X$ belongs to $R_{i+1}^U \backslash Z_i^U$. Vertices in $Z_i^U \cap U$ are not in graph G_{i+1}^U. The minimum closest (S_{i+1}^U, T)-cut in G_{i+1}^U, denoted by $\widehat{Z}_{i+1}^U$, is set Z_{i+1}^U deprived of vertices $Z_i^U \cap U$. Vertex x is located between $Z_i^U \backslash U$ and $\widehat{Z}_{i+1}^U$.
We show that $\widehat{Z}_{i+1}^U$ cannot be the minimum closest (S_{i+1}^U, T)-cut because of the identified vertex x. To meet this goal, we form another minimum closest cut which contradicts the unicity.
Let $\widehat{X}_U$ be the CCP $\Pi_c(U, X)$ in graph G. Cut $\widehat{X}_U$ is closer than U and contains vertex x. As $\widehat{X}_U$ is made up of vertices of U and X, there is no vertex of $\widehat{X}_U$ in set $R(Z_i^U, S)$. As a consequence, $Z_i^U \cap U \subsetneq \widehat{X}_U$ because the vertices of $\widehat{X}_U$ on the Menger's paths $\sigma(Z_i^U \cap U)$ can be neither descendants of U nor ancestors of Z_i^U. Set $\widehat{X}_U \cap (Z_i^U \cap U)$ thus forms a minimum (S_{i+1}^U, T)-cut in G_{i+1}^U. This is a contradiction as vertex x is the ancestor of a vertex of $\widehat{Z}_{i+1}^U$ on path $\sigma(x)$. Although $\widehat{Z}_{i+1}^U$ is supposed to be the closest (S_{i+1}^U, T)-cut, vertex x is in its source side. □

Theorem 2.18 highlights the existence of a drainage cut Z_i^U such that $X \cap Z_i^U \neq \emptyset$ and $X \cap R(Z_i^U, S) = \emptyset$. But this drainage cut is not necessarily unique as the cuts in $\mathcal{Z}(\mathcal{I}, U)$ are not disjoint. We would like to identify on $\mathcal{Z}(\mathcal{I}, U)$ a unique front dam $B_i = X \cap Z_i^U$ for any minimum (S,T)-cut X fulfilling $X \cap R^+(U,S) \neq \emptyset$ and $X \neq Z_i^U$. This property allows us to justify that each minimum (S,T)-cut is counted exactly once by our algorithm. For this reason, Definition 2.15 in page 33 has to be modified to capture the notion of dam in the local drainage.

Definition 2.25 (Dam of $\mathcal{Z}(\mathcal{I}, U)$)**.** *A dam B_i is a nonempty subset of a minimum drainage cut Z_i^U,* i.e. *$B_i \subseteq Z_i^U$, $B_i \neq \emptyset$. Furthermore, for any $h < i$, that there is no subset B_h of Z_h^U such that $B_h \subseteq B_i$.*

Let us consider the minimum drainage cut Z_i^U with the smallest index i such that $X \cap Z_i^U \neq \emptyset$ and $X \cap R(Z_i^U, S) = \emptyset$. Suppose $B_i = X \cap Z_i^U$ is not a dam: there is a dam $B_h \subseteq B_i$ with $h < i$. Consequently, $B_h \subsetneq X$ and the drainage cut Z_h^U verifies $X \cap R(Z_h^U, S) = \emptyset$, as $R(Z_h^U, S) \subsetneq R(Z_i^U, S)$. This is a contradiction because the index of cut Z_h^U is smaller than i. So, set B_i is a dam of $\mathcal{Z}(\mathcal{I}, U)$. When $X \neq Z_i^U$, we say the front index of X in this case is $i(X) = i$. The definition of the front dam follows.

Definition 2.26 (Front dam in $\mathcal{Z}(\mathcal{I}, U)$)**.** *The front dam of a minimum (S,T)-cut X such that $X \cap R^+(U,S) \neq \emptyset$ is $B_{i(X)} = X \cap Z_{i(X)}^U$.*

As $X \cap R(U,S) \neq \emptyset$ in Figure 2.17, cut X admits a front dam which is necessarily the only vertex of X in $R(U,S)$. We list the properties of the local drainage $\mathcal{Z}(\mathcal{I}, U)$.

Summary 2 (Properties of the local drainage)**.** *Collection $\mathcal{Z}(\mathcal{I}, U)$, composed of the minimum drainage cuts Z_i^U, fulfils:*

- *the intersection of any cuts Z_i^U and Z_j^U, $i \neq j$, lies in U,* i.e. *$Z_i^U \cap Z_j^U \subseteq U$,*
- *there are less than n cuts Z_i^U,*
- *the reachable sets of cuts Z_i^U fulfil $R(Z_i^U, S) \subsetneq R(Z_{i+1}^U, S)$ for $i \in \{1, \ldots, k-1\}$,*
- *for any minimum (S,T)-cut X, there is a unique drainage cut $Z_{i(X)}^U$ such that $X \cap Z_{i(X)}^U \neq \emptyset$, $X \cap R(Z_{i(X)}^U, S) = \emptyset$, and no drainage cut Z_h^U, $h < i(X)$ fulfils $X \cap Z_h^U = \emptyset$.*

The computation of both drainages $\mathcal{Z}(\mathcal{I})$ and $\mathcal{Z}(\mathcal{I},U)$ consists in a succession of at most n executions of Ford-Fulkerson's algorithm to obtain closest cuts. As with the edge-cut drainage, they are retrieved in time $O(mnp)$.

Theorem 2.19. *The global and local vertex drainages of $\mathcal{I}$ are computed in time $O(mnp)$.*

Proof. There are at most $\frac{n}{p}$ drainage cuts in global $\mathcal{Z}(\mathcal{I})$ because the cuts Z_i are pairwise disjoint. For the local drainage $\mathcal{Z}(\mathcal{I},U)$, sets $R^+(Z^U_{i+1},S)\backslash R^+(Z^U_i,S)$, $1\leq i\leq k$ are pairwise disjoint and nonempty as $R(Z^U_i,S)\subsetneq R(Z^U_{i+1},S)$. So, there are at most n drainage cuts in $\mathcal{Z}(\mathcal{I},U)$. Finally, both drainages are obtained by computing at most n closest cuts in different graphs. The overall running time of such an operation is $O(mnp)$. □

From now on, we considered an arbitrary cut U to define the local drainage $\mathcal{Z}(\mathcal{I},U)$. We explain in the remainder which frontiers are relevant to make the number of recursive calls of our algorithm be only FPT$\langle p\rangle$.

Dry and enclosed instances

As in Section 2.3.1, we associate with any dam of $\mathcal{Z}(\mathcal{I})$ or $\mathcal{Z}(\mathcal{I},U)$ a *dry instance* which covers all sets $X\backslash B_i$, where X is a minimum (S,T)-cut of $\mathcal{I}$ and B_i is its front dam.
We focus on a global drainage $\mathcal{Z}(\mathcal{I})$ to illustrate the concept of dry instance, without any loss of generality, as all arguments remain valid for $\mathcal{Z}(\mathcal{I},U)$.
Any dam B_i is characterized by its *level*, *i.e.* the index i of the cut Z_i (or Z^U_i) it belongs to, but also its *signature* $\sigma(B_i)=\{Q_j : B_i\cap Q_j\neq\emptyset\}$.

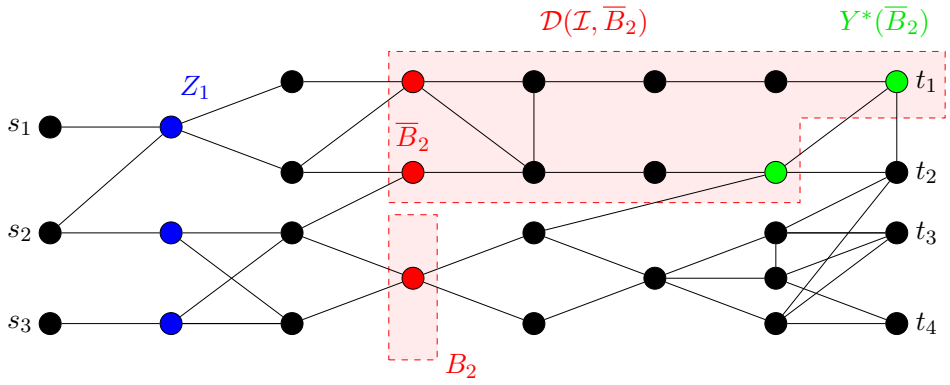


Figure 2.19: Dry instance $\mathcal{D}(\mathcal{I},\overline{B}_2)$

Choking graph G with dam B_i puts in evidence a sub-instance of $\mathcal{I}$, denoted by $\mathcal{D}(\mathcal{I},\overline{B}_i)$. Dam $\overline{B}_i$ is the complementary of B_i in $\mathcal{Z}(\mathcal{I})$, *i.e.* $\overline{B}_i=Z_i\backslash B_i$. It is the source of the dry instance $\mathcal{D}(\mathcal{I},\overline{B}_i)$. The definition of $\mathcal{D}(\mathcal{I},\overline{B}_i)$ follows:

Definition 2.27 (Dry instance $\mathcal{D}(\mathcal{I},\overline{B}_i)$)**.** *Let $Y^*(\overline{B}_i)$ be the minimum important (S,T)-cut obtained in graph $G\backslash B_i$. The dry instance $\mathcal{D}(\mathcal{I},\overline{B}_i)$ is a triplet $\big(G^*(\overline{B}_i),S^*(\overline{B}_i),T^*(\overline{B}_i)\big)$ with graph $G^*(\overline{B}_i)=\big(V^*(\overline{B}_i),E^*(\overline{B}_i)\big)$. In particular,*

- *graph $G^*(\overline{B}_i)$ is made up of the vertices of the augmented source side $R^+(Y^*(\overline{B}_i),S)$ in $G\backslash B_i$ which are located either on dam $\overline{B}_i$ or in its target side: $V^*(\overline{B}_i)=R^+(Y^*(\overline{B}_i),S)\cap R^+(\overline{B}_i,T)$. Indeed, dam $\overline{B}_i$ is a minimum (S,T)-cut in $G\backslash B_i$. Its edges are induced by set $V^*(\overline{B}_i)$: $E^*(\overline{B}_i)=E\left[V^*(\overline{B}_i)\right]$,*
- *source $S^*(\overline{B}_i)$ is the set $\overline{B}_i$: $S^*(\overline{B}_i)=\overline{B}_i$,*
- *target $T^*(\overline{B}_i)$ is made up of the important cut $Y^*(\overline{B}_i)$: $T^*(\overline{B}_i)=Y^*(\overline{B}_i)$.*

Figure 2.19 provides an example of dam $B_2 \subsetneq Z_2$ and a dry instance $\mathcal{D}(\mathcal{I}, \overline{B}_2)$ on the global drainage already given in Figure 2.16.
The target side of Z_i in G, $R(Z_i, T)$, can be divided into three parts relative to cut $Y^*(\overline{B}_i)$ and its source and target sides. Its vertices are either in $R(Y^*(\overline{B}_i), S)$, in cut $Y^*(\overline{B}_i)$, or in $R(Y^*(\overline{B}_i), T)$.
The Menger's paths in $\sigma(\overline{B}_i)$ traverse the dry instance $\mathcal{D}(\mathcal{I}, \overline{B}_i)$: they enter in $\overline{B}_i$ and leave it through $Y^*(\overline{B}_i)$ as no edge connects $R(Y^*(\overline{B}_i), S)$ and $R(Y^*(\overline{B}_i), T)$. As a consequence, the sections of all paths in $\sigma(\overline{B}_i)$ going from the source $\overline{B}_i$ to the target $Y^*(\overline{B}_i)$ form a collection of vertex-disjoint paths for the instance $\mathcal{D}(\mathcal{I}, \overline{B}_i)$.
The main property of the dry instance $\mathcal{D}(\mathcal{I}, \overline{B}_i)$ is the bijective relationship between the cuts of $\mathcal{I}$ with front dam B_i and the cuts of $\mathcal{D}(\mathcal{I}, \overline{B}_i)$. Indeed, for any minimum (S,T)-cut X with front dam B_i, set $X \backslash B_i$ is a minimum cut for the instance $\mathcal{D}(\mathcal{I}, \overline{B}_i)$. Conversely, all minimum cuts of $\mathcal{D}(\mathcal{I}, \overline{B}_i)$ assembled with dam B_i form a minimum cut of $\mathcal{I}$. We begin with the proof of the latter.

Theorem 2.20. *If $\widehat{X}$ is a minimum cut for instance $\mathcal{D}(\mathcal{I}, \overline{B}_i)$, then $X = B_i \cup \widehat{X}$ is a minimum cut for $\mathcal{I}$.*

Proof. We suppose that X is not an (S,T)-cut: there is a simple (S,T)-path P in $G \backslash X$. Path P necessarily traverses the drainage cut Z_i, more particularly dam $\overline{B}_i$ as $B_i \subsetneq X$. Therefore, a section $\widehat{P}$ of path P goes from $\overline{B}_i$ to T. Path $\widehat{P}$ must traverse the cut $Y^*(\overline{B}_i)$ of graph $G \backslash B_i$. As $\widehat{X}$ is a cut of instance $\mathcal{D}(\mathcal{I}, \overline{B}_i)$, it separates its source $\overline{B}_i$ from its target $Y^*(\overline{B}_i)$. As a consequence, section $\widehat{P}$ passes through cut $\widehat{X}$ which is a contradiction. □

The converse result is stated in the following theorem.

Theorem 2.21. *If X is a minimum (S,T)-cut with front dam B_i, then set $X \backslash B_i$ is a minimum cut for instance $\mathcal{D}(\mathcal{I}, \overline{B}_i)$.*

Proof. The minimum (S,T)-cut with front dam B_i which admits the largest source side is $Y = B_i \cup Y^*(\overline{B}_i)$ by definition and it is unique. Therefore, $R(X,S) \subseteq R(Y,S)$, so X is an (S,Y)-cut. As B_i is the front dam of X, we know that $B_i \subsetneq X$ and set $X \backslash B_i$ is included in $R(Z_i, T)$. Consequently, set $X \backslash B_i$ separates $\overline{B}_i$ and $Y^*(\overline{B}_i)$ in graph $G \backslash B_i$, otherwise X would not be an (S,Y)-cut. As each of its vertices belongs to exactly one path in $\sigma(\overline{B}_i)$ (Lemma 2.1), set $X \backslash B_i$ is inside instance $\mathcal{D}(\mathcal{I}, \overline{B}_i)$. In summary, $X \backslash B_i$ is a cut for instance $\mathcal{D}(\mathcal{I}, \overline{B}_i)$. Suppose that $X \backslash B_i$ is not minimum: there is a cut $\widehat{X}$ in $\mathcal{D}(\mathcal{I}, \overline{B}_i)$ verifying $\left|\widehat{X}\right| < |X \backslash B_i|$. According to Theorem 2.20, $\widehat{X} \cup B_i$ is a cut for $\mathcal{I}$ and $\left|\widehat{X} \cup B_i\right| < |X|$. This is incoherent as X is a minimum (S,T)-cut. □

In summary, a set $\widehat{X}$ is a minimum cut for $\mathcal{D}(\mathcal{I}, \overline{B}_i)$ iff $B_i \cup \widehat{X}$ is a minimum cut for $\mathcal{I}$. We observed previously that the sections of paths $\sigma(\overline{B}_i)$ from $\overline{B}_i$ to $Y^*(\overline{B}_i)$ are vertex-disjoint paths for the instance $\mathcal{D}(\mathcal{I}, \overline{B}_i)$. In fact, this collection forms a set of Menger's paths for $\mathcal{D}(\mathcal{I}, \overline{B}_i)$. According to Theorem 2.21, dam $\overline{B}_i$ is a minimum cut of $\mathcal{D}(\mathcal{I}, \overline{B}_i)$ as $B_i \cup \overline{B}_i = Z_i$ is minimum for $\mathcal{I}$. So, $\left|\sigma(\overline{B}_i)\right| = \left|\overline{B}_i\right|$ is equal to the minimum cut size of this instance. In brief, there is no need to launch new computations in order to obtain the Menger's paths of $\mathcal{D}(\mathcal{I}, \overline{B}_i)$: they can be deduced directly from the subset $\sigma(\overline{B}_i)$ of $\mathcal{Q}$.
There is a relationship between the number of minimum cuts of an instance $\mathcal{I}$ and the number of minimum cuts in its dry instances.

Corollary 2.1. *The number of minimum (S,T)-cuts in $\mathcal{I}$ with front dam B_i is equal to the number of minimum $\left(S^*(\overline{B}_i), T^*(\overline{B}_i)\right)$-cuts in $\mathcal{D}(\mathcal{I}, \overline{B}_i)$. The total number $C\left[\mathcal{I}\right]$ of minimum (S,T)-cuts in $\mathcal{I}$ can be thus written:*

$$C\left[\mathcal{I}\right] = |\mathcal{Z}(\mathcal{I})| + \sum_{\substack{\text{dam } B_i \text{ of } \mathcal{Z}(\mathcal{I}) \\ 1 \leq i \leq |\mathcal{Z}(\mathcal{I})|}} C\left[\mathcal{D}(\mathcal{I}, \overline{B}_i)\right]. \tag{2.3}$$

Proof. According to Theorems 2.20 and 2.21, the number of minimum cuts in $\mathcal{I}$ with front dam B_i is equal to the number of minimum cuts in $\mathcal{D}(\mathcal{I}, \overline{B}_i)$. There are $|\mathcal{Z}(\mathcal{I})|$ minimum drainage cuts and the other ones admit a front dam. In this way, we naturally obtain Eq. (2.3). □

A recursive algorithm counting the minimum (S,T)-cuts is a direct consequence of Eq. (2.3). It consists in computing the global drainage of the input instance $\mathcal{I}$. Then, it counts its minimum drainage cuts, enumerates all dams B_i and is applied recursively on the dry instances $\mathcal{D}(\mathcal{I}, \overline{B}_i)$ for each B_i. Unfortunately, its execution time is not FPT$\langle p \rangle$ as the number of dry instances computed throughout the recursion is $\Omega(n^p)$ in certain cases. We should introduce additional concepts which allow us to decrease considerably this running time.
As the unicity of the front dam is also ensured on a local drainage $\mathcal{Z}(\mathcal{I}, U)$, a set $X = B_i \cup \widehat{X}$ is a minimum cut of $\mathcal{I}$ fulfilling $X \cap R^+(U,S) \neq \emptyset$, with front dam B_i, iff $\widehat{X}$ is a minimum cut of $\mathcal{D}(\mathcal{I}, \overline{B}_i)$. Corollary 2.1 should be tailored to fit local drainages. The proof uses the same arguments as above. Let $C\left[\mathcal{I}, U\right]$ be the number of minimum (S,T)-cuts X such that $X \cap R^+(U,S) \neq \emptyset$. We have:

Corollary 2.2. *The number of minimum (S,T)-cuts in $\mathcal{I}$ with front dam B_i is equal to the number of minimum $\left(S^*(\overline{B}_i), T^*(\overline{B}_i)\right)$-cuts in $\mathcal{D}(\mathcal{I}, \overline{B}_i)$. Value $C\left[\mathcal{I}, U\right]$ can be written:*

$$C\left[\mathcal{I}, U\right] = |\mathcal{Z}(\mathcal{I}, U)| + \sum_{\substack{\text{dam } B_i \text{ of } \mathcal{Z}(\mathcal{I},U) \\ 1 \leq i \leq |\mathcal{Z}(\mathcal{I},U)|}} C\left[\mathcal{D}(\mathcal{I}, \overline{B}_i)\right]. \tag{2.4}$$

We present a structural property of the dry instances. It states that the set of edges with one endpoint inside $\mathcal{D}(\mathcal{I}, \overline{B}_i)$ and one outside is limited to the ones incident to either B_i, $\overline{B}_i$, or $Y^*(\overline{B}_i)$. This allows us to explain how a path can pass through the dry instance as only a few edges are the entry and exit points for $\mathcal{D}(\mathcal{I}, \overline{B}_i)$.

Theorem 2.22. *Let B_i be a dam of $\mathcal{Z}(\mathcal{I})$, or $\mathcal{Z}(\mathcal{I}, U)$, and $\mathcal{D}(\mathcal{I}, \overline{B}_i)$ be the dry instance of its complementary $\overline{B}_i$. We focus on the edges with one endpoint in $V^*(\overline{B}_i)$ and another one in $V \backslash V^*(\overline{B}_i)$. They can be divided into three families:*

- *the* entry edges *with one endpoint in $R(Z_i, S)$ and another one in $\overline{B}_i$,*
- *the* exit edges *with one endpoint in $Y^*(\overline{B}_i)$ and another one in $R(Z_i, T)$,*
- *the* leaks *with one endpoint in $V^*(\overline{B}_i)$ and another one in B_i.*

Proof. First, we focus on edges $e = (u,v)$ leaving $\mathcal{D}(\mathcal{I}, \overline{B}_i)$ with an endpoint $u \in V^*(\overline{B}_i)$ which is neither in $\overline{B}_i$ nor in $Y^*(\overline{B}_i)$. If $v \in R(Z_i, S)$ in graph G, then Z_i is not an (S,T)-cut anymore as $u \in R(Z_i, T)$: a contradiction. Now suppose that $v \in R(Z_i, T)$. In graph $G \backslash B_i$, vertex v belongs necessarily to $R\left(Y^*(\overline{B}_i), T\right)$ because $v \notin V^*(\overline{B}_i)$. This is a contradiction as edge e overpasses cut $Y^*(\overline{B}_i)$. The only possibility is to have $v \in Z_i$: therefore, $v \in B_i$ as $\overline{B}_i \subseteq V^*(\overline{B}_i)$. Such edges are leaks of the dry instance $\mathcal{D}(\mathcal{I}, \overline{B}_i)$.
Second, we treat edges $e = (u,v)$ with $u \in \overline{B}_i$ and $v \notin V^*(\overline{B}_i)$. Vertex v does not belong to set $R\left(Y^*(\overline{B}_i), T\right)$ because $Y^*(\overline{B}_i)$ is an (S,T)-cut in $G \backslash B_i$. So, either v is in $R(Z_i, S)$ (edge e is an entry edge) or v is in B_i (edge e is a leak).
Eventually, let us suppose that $u \in Y^*(\overline{B}_i)$ and $v \notin V^*(\overline{B}_i)$. As previously, vertex v is not in $R(Z_i, S)$ otherwise edge e overpasses cut Z_i. Consequently, either $v \in R\left(Y^*(\overline{B}_i), T\right)$ in $G \backslash B_i$ (edge e is an exit edge) or $v \in B_i$ (edge e is a leak). □

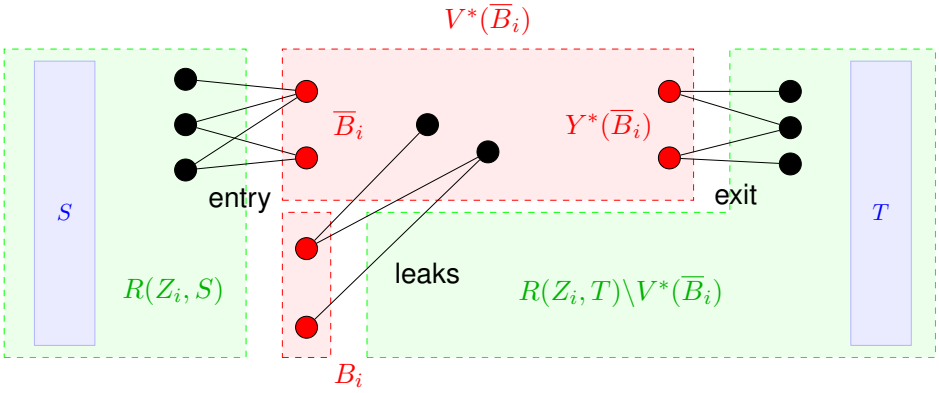


Figure 2.20: Entry, exit edges and leaks of the dry instance $\mathcal{D}(\mathcal{I}, \overline{B}_i)$

The proof of Theorem 2.22 only uses the property that Z_i is an (S,T)-cut in G and $Y^*(\overline{B}_i)$ an (S,T)-cut in $G\backslash B_i$. It thus can be generalized to a dry instance of the local drainage $\mathcal{Z}(\mathcal{I}, U)$. Figure 2.20 illustrates the edges enumerated in Theorem 2.22. The set $R(Z_i,T)\backslash V^*(\overline{B}_i)$ corresponds to the target side $R\left(Y^*(\overline{B}_i), T\right)$ in $G\backslash B_i$, which is mentioned in the proof above.
The notion of dry instances is now generalized. Enclosed instances $\mathcal{D}(\mathcal{I}, \overline{B}_i, H_i)$ are derived from the dry instances $\mathcal{D}(\mathcal{I}, \overline{B}_i)$. They are defined not only for a certain dam $\overline{B}_i$ but also for a subset $H_i \subseteq B_i$ of its complementary B_i. They are obtained from $\mathcal{D}(\mathcal{I}, \overline{B}_i)$ by forbidding a subset of its leaks.

Definition 2.28 (Enclosed instances $\mathcal{D}(\mathcal{I}, \overline{B}_i, H_i)$)**.** *We consider graph G_{i+1},* i.e. *graph G after the removal of $R(Z_i, S)$. Let $Y^*(\overline{B}_i, H_i)$ be the minimum important $(\overline{B}_i, T)$-cut obtained in graph $G_{i+1}\backslash H_i$. The enclosed instance $\mathcal{D}(\mathcal{I}, \overline{B}_i, H_i)$ is a triplet containing:*

- *graph $G^*(\overline{B}_i, H_i) = \left(V^*(\overline{B}_i, H_i), E^*(\overline{B}_i, H_i)\right)$ is made up of the vertices of the source side $R^+(Y^*(\overline{B}_i, H_i), \overline{B}_i)$: $V^*(\overline{B}_i, H_i) = R^+(Y^*(\overline{B}_i), \overline{B}_i)$. Its edges are obtained from the induced set: $E^*(\overline{B}_i, H_i) = E\left[V^*(\overline{B}_i, H_i)\right]$,*
- *source $S^*(\overline{B}_i, H_i) = \overline{B}_i$,*
- *target $T^*(\overline{B}_i, H_i) = Y^*(\overline{B}_i, H_i)$.*

All the enclosed instances of $\overline{B}_i$ have the same source $S^*(\overline{B}_i, H_i) = \overline{B}_i$. The particular enclosed instance $\mathcal{D}(\mathcal{I}, \overline{B}_i, B_i)$ is identical to the dry instance $\mathcal{D}(\mathcal{I}, \overline{B}_i)$ defined earlier: $\mathcal{D}(\mathcal{I}, \overline{B}_i, B_i) = \mathcal{D}(\mathcal{I}, \overline{B}_i)$. Indeed, an equivalent formulation of Definition 2.18 is that cut $Y^*(\overline{B}_i)$ is the minimum important $(\overline{B}_i, T)$-cut in the graph $G_{i+1}\backslash B_i$.
Theorem 2.22 can be generalized for the enclosed instances. Enclosed instances also admit entry and exit edges. In contrast, the leaks of the enclosed instance $\mathcal{D}(\mathcal{I}, \overline{B}_i, H_i)$ only connect $B_i\backslash H_i$ and $V^*(\overline{B}_i, H_i)$. Said differently, there is no edge with one endpoint in $B_i\backslash H_i$ and the other in $V^*(\overline{B}_i, H_i)$, as stated below. The smaller set H_i is, the more difficult it is to escape from the instance $\mathcal{D}(\mathcal{I}, \overline{B}_i, H_i)$.

Theorem 2.23. *There is no edge in graph G between sets $B_i\backslash H_i$ and $V^*(\overline{B}_i, H_i)$*

Proof. Sections of paths $\sigma(\overline{B}_i)$ from $\overline{B}_i$ to T form a set of vertex-disjoint paths in $G_{i+1}\backslash H_i$. As $\left|\sigma(\overline{B}_i)\right| = \left|Y^*(\overline{B}_i, H_i)\right|$, they are Menger's paths and each vertex of $Y^*(\overline{B}_i, H_i)$ belongs to a path in $\sigma(\overline{B}_i)$.
Suppose that an edge e connects $B_i\backslash H_i$ and $V^*(\overline{B}_i, H_i)$. Thanks ot this edge, a path can be formed between source $\overline{B}_i$ and $B_i\backslash H_i$. This path can be extended to T with a path of $\sigma(B_i\backslash H_i)$ which is vertex-disjoint with any path in $\sigma(\overline{B}_i)$, and consequently with cut

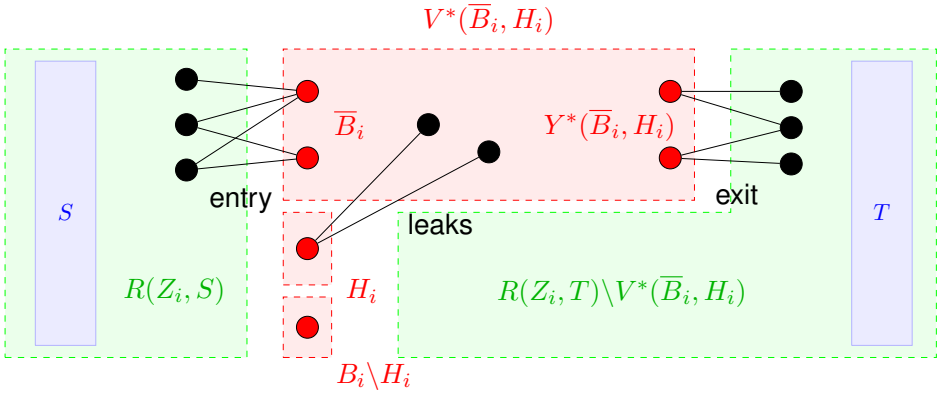


Figure 2.21: Entry, exit edges and leaks of the enclosed instance $\mathcal{D}(\mathcal{I}, \overline{B}_i, H_i)$

$Y^*(\overline{B}_i, H_i)$, too. The existence of a path between $\overline{B}_i$ and T which avoids $Y^*(\overline{B}_i, H_i)$ is a contradiction. □

In summary, as shown in Figure 2.21, the edges leaving the enclosed instance $\mathcal{D}(\mathcal{I}, \overline{B}_i, H_i)$, *i.e.* belonging to $V^*(\overline{B}_i, H_i) \times V\backslash V^*(\overline{B}_i, H_i)$, can be divided into three families: the entry, exit edges, and the leaks connecting H_i with $V^*(\overline{B}_i, H_i)$ according to Theorem 2.23. This observation is the key to prove a major result, Theorem 2.24, which highlights a relationship between the enclosed instances of dams $\overline{B}_i$ and $\overline{B}_{i+1}$, where $\sigma(\overline{B}_i) = \sigma(\overline{B}_{i+1})$.
We suppose that $\overline{B}_{i+1}$ is inside the enclosed instance $\mathcal{D}(\mathcal{I}, \overline{B}_i, H_i)$, *i.e.* $\overline{B}_{i+1} \subseteq V^*(\overline{B}_i, H_i)$. Independently from the nature of the drainage, set $\overline{B}_{i+1}$ is in this case a minimum cut for $\mathcal{D}(\mathcal{I}, \overline{B}_i, H_i)$, according to the following lemma.

Lemma 2.13. *If $\overline{B}_{i+1} \subseteq V^*(\overline{B}_i, H_i)$, then $\overline{B}_{i+1}$ is a minimum cut for instance $\mathcal{D}(\mathcal{I}, \overline{B}_i, H_i)$.*

Proof. If set $\overline{B}_{i+1}$ is not a cut in $\mathcal{D}(\mathcal{I}, \overline{B}_i, H_i)$, then a path Q connects $\overline{B}_i$ with $Y^*(\overline{B}_i, H_i)$ and bypasses $\overline{B}_{i+1}$. Let $u \in \overline{B}_i$ and $v \in Y^*(\overline{B}_i, H_i)$ be the two extremities of path Q. Then, path Q can be extended to an (S,T)-path which bypasses Z_{i+1} (or Z^U_{i+1}) by glueing to it both an (S,u)-path and an (v,T)-path taken from the Menger's paths. Indeed, a Menger's path in $\sigma(\overline{B}_i)$ connects a vertex of S with u and another Menger's path connects v with T, as $\sigma(\overline{B}_i)$ is the signature of the vertices in $Y^*(\overline{B}_i, H_i)$. As a consequence, an (S,T)-path bypasses the drainage cut of level i, which is a contradiction. □

We focus on the part of instance $\mathcal{D}(\mathcal{I}, \overline{B}_i, H_i)$ which is in the target side of its cut $\overline{B}_{i+1}$. In fact, the set in question coincides with an enclosed instance of $\overline{B}_{i+1}$. Formally, the set $V^*(\overline{B}_i, H_i) \cap R(Z^U_{i+1}, T)$ is equal to a certain $V^*(\overline{B}_{i+1}, H_{i+1})$. It follows:

Theorem 2.24. *If $\overline{B}_{i+1} \subseteq V^*(\overline{B}_i, H_i)$, then set $V^*(\overline{B}_i, H_i) \cap R(Z^U_{i+1}, T)$ in G contains exactly the vertices of the enclosed instance $\mathcal{D}(\mathcal{I}, \overline{B}_{i+1}, H_{i+1})$, where $H_{i+1} = H_i \cap B_{i+1}$ and B_{i+1} is the complementary of $\overline{B}_{i+1}$:*

$$V^*(\overline{B}_i, H_i) \cap R(Z^U_{i+1}, T) = V^*(\overline{B}_{i+1}, H_{i+1}).$$

Proof. We begin with the proof that dam $\overline{B}_{i+1}$ is a cut for graph $G_{i+1}\backslash H_i$, based upon both the nature of the edges leaving the enclosed instances (Theorem 2.23) and the fact that $\overline{B}_{i+1}$ is a cut for $\mathcal{D}(\mathcal{I}, \overline{B}_i, H_i)$ (Lemma 2.13). The leaks of $\mathcal{D}(\mathcal{I}, \overline{B}_i, H_i)$ are withdrawn from graph $G_{i+1}\backslash H_i$ as they admit an endpoint in H_i. Therefore, a path connecting $\overline{B}_i$ and T in $G_{i+1}\backslash H_i$ necessarily leaves the enclosed instance $\mathcal{D}(\mathcal{I}, \overline{B}_i, H_i)$ via the exit edges. No path can both attain these edges and bypass dam $\overline{B}_{i+1}$ which is a cut of $\mathcal{D}(\mathcal{I}, \overline{B}_i, H_i)$. The absence of such path implies that $\overline{B}_{i+1}$ is a cut for graph $G_{i+1}\backslash H_i$.

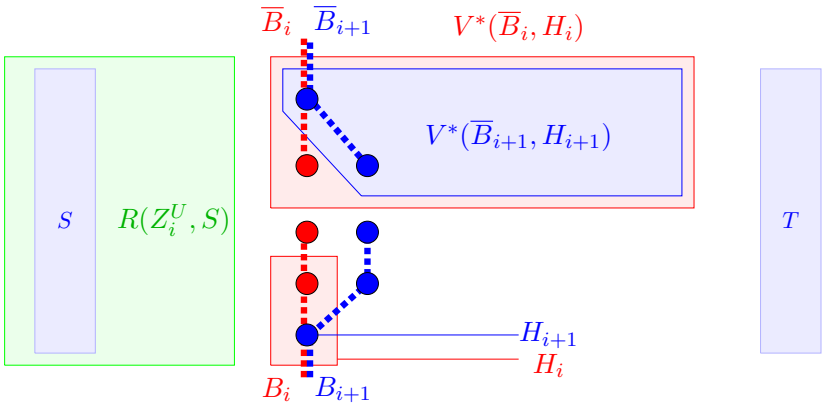


Figure 2.22: Instance $\mathcal{D}(\mathcal{I}, \overline{B}_{i+1}, H_{i+1})$ coincides with the target side of $\overline{B}_{i+1}$ in $\mathcal{D}(\mathcal{I}, \overline{B}_i, H_i)$.

Cut $Y^*(\overline{B}_i, H_i)$ dominates cut $\overline{B}_{i+1}$ in graph $G_{i+1} \setminus H_i$. So, set $Y^*(\overline{B}_i, H_i)$ is the minimum important $(\overline{B}_{i+1}, T)$-cut in graph $G_{i+1} \setminus H_i$.
If we limit our attention on the target side of Z_{i+1}^U in this graph, then $Y^*(\overline{B}_i, H_i)$ is also the minimum important $(\overline{B}_{i+1}, T)$-cut in graph G_{i+2} deprived of the vertices H_i. The vertices in $H_i \cap R(Z_{i+1}^U, S)$ have been already removed from graph G_{i+2} but the vertices in $H_i \cap B_{i+1}$ remain in G_{i+2}. Therefore, the target sets of instances $\mathcal{D}(\mathcal{I}, \overline{B}_i, H_i)$ and $\mathcal{D}(\mathcal{I}, \overline{B}_{i+1}, H_{i+1})$ are equal and they coincide with each other. □

This statement has been presented for dams B_i and B_{i+1} belonging to a local drainage $\mathcal{Z}(\mathcal{I}, U)$. It also holds for a global drainage. In this case, the set $H_{i+1} = H_i \cap B_{i+1}$ is necessarily empty as $H_i \subseteq B_i$ and B_{i+1} are disjoint. In brief, set $V^*(\overline{B}_i, H_i) \cap R(Z_{i+1}^U, T)$ in G contains exclusively all vertices of the enclosed instance $\mathcal{D}(\mathcal{I}, \overline{B}_{i+1}, \emptyset)$.
Figure 2.22 illustrates the relationship between instances $\mathcal{D}(\mathcal{I}, \overline{B}_i, H_i)$ and $\mathcal{D}(\mathcal{I}, \overline{B}_{i+1}, H_{i+1})$ when dam $\overline{B}_{i+1}$ is included in set $V^*(\overline{B}_i, H_i)$. The thick dotted lines represent dams B_i, $\overline{B}_i$, B_{i+1}, and $\overline{B}_{i+1}$. We suppose these dams belong to a local drainage $\mathcal{Z}(\mathcal{I}, U)$, so the set $Z_i^U \cap Z_{i+1}^U$ is not necessarily empty. Set $H_{i+1} = B_{i+1} \cap H_i$ is a single vertex in this example: this is the blue vertex in B_{i+1}.
Thanks to Theorem 2.24, we see that any minimum cut in $\mathcal{D}(\mathcal{I}, \overline{B}_{i+1}, H_{i+1})$ with $H_{i+1} = H_i \cap B_{i+1}$ is also a minimum cut in $\mathcal{D}(\mathcal{I}, \overline{B}_i, H_i)$. The converse is not necessarily true. This property is exploited by our algorithm as it allows us to limit the number of recursive calls with an FPT$\langle p \rangle$ function.

Dynamic programming to count minimum vertex (S, T)-cuts

We are now ready to design the algorithm that counts minimum vertex (S, T)-cuts. It uses the dynamic programming principle: the computation of the number $C(\mathcal{I})$ of minimum cuts in $\mathcal{I}$ depends on the number of minimum cuts of certain of its dry instances, and so on. We prove that its execution time is $O^*(2^{O(p \log p)})$.

Description of the algorithm. From now on, letter $\mathcal{I}$ denotes the input instance only: our objective is to compute $C(\mathcal{I})$. Other instances mentioned in the description are denoted by letters $\mathcal{J}$ or $\mathcal{H}$. The number returned by our algorithm is given by function count. It goes without saying that we prove later: count$(\mathcal{I}) = C(\mathcal{I})$.
We define a *Menger's sequence* as a sequence of sets which forms a partition of Menger's paths $\mathcal{Q}$.

Definition 2.29 (Menger's sequence)**.** *Let $\mathcal{K} = K_1 \cdot K_2 \cdots K_r$ be a sequence of sets K_i containing Menger's paths. We say that $\mathcal{K}$ is a Menger's sequence if the collection $K_1, \ldots, K_r$*

1: **Input:** Instance $\mathcal{I} = (G, S, T)$
2: res $\leftarrow 0$;
3: $p \leftarrow$ minimum (S,T)-cut size of G;
4: $\mathcal{Z}(\mathcal{I}) = (Z_1, \ldots, Z_k) \leftarrow$ global drainage of $\mathcal{I}$;
5: $\mathcal{K}^{(2)} \leftarrow K_2 \cdots K_r$;
6: **for** *all Menger's sequences* $\mathcal{K} = K_1 \cdots K_r$ **do**
7: | **if** $K_1 = K_r = \bigcup_{j=1}^{p} Q_i$ **then**
8: | | res $\leftarrow$ res $+ k$;
| **else**
9: | | **for** *all dams* B_i *such that* $\sigma(B_i) = K_1$ **do**
10: | | | $\overline{B}_i \leftarrow Z_i \backslash B_i$;
11: | | | res $\leftarrow$ res $+$ count_encl$(\mathcal{I}, \textbf{null}, \overline{B}_i, B_i, \mathcal{K}^{(2)})$;
12: | | **endfor**
| **endif**
13: | /* The value added to variable res during each loop is count$(\mathcal{I}, \mathcal{K})$. */
14: **endfor**
15: **return** res;

Algorithm 1: The count algorithm

is a partition of $\mathcal{Q}$*:* $\bigcup_{j=1}^{r} K_j = \bigcup_{j=1}^{p} Q_j$ *and* $K_h \cap K_\ell = \emptyset$ *for any* $h, \ell \in \{1, \ldots, r\}$*,* $h \neq \ell$*.*

Given a partition $(K_1, \ldots, K_r)$ of $\mathcal{Q}$, there are at most $r!$ Menger's sequences composed of these sets. Furthermore, $r \leq p$ as $|\mathcal{Q}| = p$. Therefore, as the number of partitions of $\mathcal{Q}$ is equal to the Bell number $\mathcal{B}_p = O(p^p)$, the total number of Menger's sequences is $O(2^{p \log p})$.
We begin with an overview of the algorithm. Later, we specify the manner of associating to any minimum (S,T)-cut X a unique Menger's sequence $\mathcal{K}_X$. Without going into details at the moment, $K_1 \cdot K_2 \cdots$ is the Menger's sequence of cut X if K_1 is the signature of the front dam B_i of X, then K_2 is the signature of the front dam of $X \backslash B_i$ in the dry instance $\mathcal{D}(\mathcal{I}, \overline{B}_i)$, and so on. Our algorithm executes $O(2^{p \log p})$ independent tasks. Put differently, the computation of count$(\mathcal{I})$ consists in launching one execution for each Menger's sequence. Each "thread" returns a value count$(\mathcal{I}, \mathcal{K})$ which is the number of minimum (S,T)-cuts X such that $\mathcal{K}_X = \mathcal{K}$. The "global" algorithm returns the sum of values count$(\mathcal{I}, \mathcal{K})$. Written formally,

$$\text{count}(\mathcal{I}) = \sum_{\mathcal{K}} \text{count}(\mathcal{I}, \mathcal{K}).$$

From now on, we give the description of the counting algorithm with two inputs: an instance $\mathcal{I}$ and a Menger's sequence $\mathcal{K} = K_1 \cdot K_2 \cdots K_r$. We show how value count$(\mathcal{I}, \mathcal{K})$ is obtained (Algorithm 1).
We define a recursive routine count_encl. It determines the number of minimum cuts of enclosed instances, and consequently dry instances, as $\mathcal{D}(\mathcal{J}, \overline{B}_i) = \mathcal{D}(\mathcal{J}, \overline{B}_i, B_i)$ for any instance $\mathcal{J}$ and dam B_i. The algorithm count calls count_encl on the dry instances of $\mathcal{I}$. Algorithm count_encl has five arguments: an instance $\mathcal{H}$ with a frontier U, a dam $\overline{B}_i$ of the local drainage $\mathcal{Z}(\mathcal{H}, U)$, a subset H_i of its complementary B_i, and a Menger's sequence $\mathcal{K}$. Value count_encl$(\mathcal{H}, U, \overline{B}_i, H_i, \mathcal{K})$ gives the number of minimum cuts of the enclosed instance $\mathcal{D}(\mathcal{H}, \overline{B}_i, H_i)$ associated with Menger's sequence $\mathcal{K}$. Argument U is optional: if $U =$ **null**, then $\overline{B}_i$ is a dam of the global drainage $\mathcal{Z}(\mathcal{H})$. Algorithm count_encl uses the dynamic programming (DP) principle: it makes recursive calls to instances with a smaller minimum (S,T)-cut size and stores the result obtained for any input 5-uplet considered to avoid multiple calls for the same data.
A special case occurs when computing value count$(\mathcal{I}, \mathcal{K})$. A sequence $\mathcal{K}$ made up of a single set $K_1 = \bigcup_{j=1}^{p} Q_j$ (line 7 of Algorithm 1). In this situation, value count$(\mathcal{I}, \mathcal{K})$ is the

number of minimum drainage cuts $|\mathcal{Z}(\mathcal{I})|$ (line 8). Otherwise, we enumerate all dams B_i of the drainage $\mathcal{Z}(\mathcal{I})$ satisfying $\sigma(B_i) = K_1$. For each of them, we compute the dry instance of their complementary $\overline{B}_i$. The idea is to determine the number of minimum cuts with front dam B_i, *i.e.* the number of minimum cuts for the instances $\mathcal{D}(\mathcal{I}, \overline{B}_i)$. To do so, we call the algorithm count_encl. More precisely speaking, we compute count_encl$(\mathcal{I}, \textbf{null}, \overline{B}_i, B_i, \mathcal{K}^{(2)})$, where $\mathcal{K}^{(2)} = K_2 \cdots K_r$ for any dam B_i of $\mathcal{Z}(\mathcal{I})$ with a signature K_1 (line 11).
The following equation reveals how value count$(\mathcal{I}, \mathcal{K})$ is computed. This can be seen as a revised version of Eq. (2.3) in Corollary 2.1 which expressed $C(\mathcal{I})$ as a function of values $C(\mathcal{D}(\mathcal{I}, \overline{B}_i))$.

$$\text{count}(\mathcal{I}, \mathcal{K}) = \begin{cases} |\mathcal{Z}(\mathcal{I})| = k, \text{ if } r = 1 \\ \sum\limits_{\substack{\text{dam } B_i \\ \sigma(B_i)=K_1}} \text{count\_encl}(\mathcal{I}, \textbf{null}, \overline{B}_i, B_i, \mathcal{K}^{(2)}), \text{ otherwise} \end{cases}$$

We now begin the description of the algorithm count_encl. Its pseudocode is given in Algorithm 2. Before giving its general formulation, we explain how the number count_encl$(\mathcal{I}, \textbf{null}, \overline{B}_i, B_i, \mathcal{K}^{(2)})$ of minimum cuts of a dry instance $\mathcal{D}(\mathcal{I}, \overline{B}_i)$ of $\mathcal{I}$ is calculated in order to provide the intuition behind this algorithm.
For any dry instance $\mathcal{D}(\mathcal{I}, \overline{B}_i)$, no recursive call is made and the result is returned directly if the Menger's sequence $\mathcal{K}^{(2)}$ contains only one element, *i.e.* $K_2 = K_r$.

$$\text{count\_encl}(\mathcal{I}, \textbf{null}, \overline{B}_i, B_i, \mathcal{K}^{(2)}) = |\mathcal{Z}(\mathcal{J})|, \text{ where } \mathcal{J} = \mathcal{D}(\mathcal{I}, \overline{B}_i).$$

Otherwise, the algorithm verifies whether the vertices of $\overline{B}_{i+1}$ are in the dry instance $\mathcal{J} = \mathcal{D}(\mathcal{I}, \overline{B}_i)$, where $\overline{B}_{i+1}$ has the same signature as $\overline{B}_i$ but is on the next level, $\overline{B}_{i+1} \subsetneq Z_{i+1}$. We distinguish three cases depending on the position of the vertices in $\overline{B}_{i+1}$.

- **Case 1:** No vertex of dam $\overline{B}_{i+1}$ is inside the dry instance $\mathcal{D}(\mathcal{I}, \overline{B}_i)$, *i.e.* $\overline{B}_{i+1} \cap V^*(\overline{B}_i) = \emptyset$. Figure 2.23 provides a schematic view of a dam $\overline{B}_i$ satisfying this case. In this example, the signature of both dams $\overline{B}_i$ and $\overline{B}_{i+1}$ is $\{Q_1, Q_2\}$.

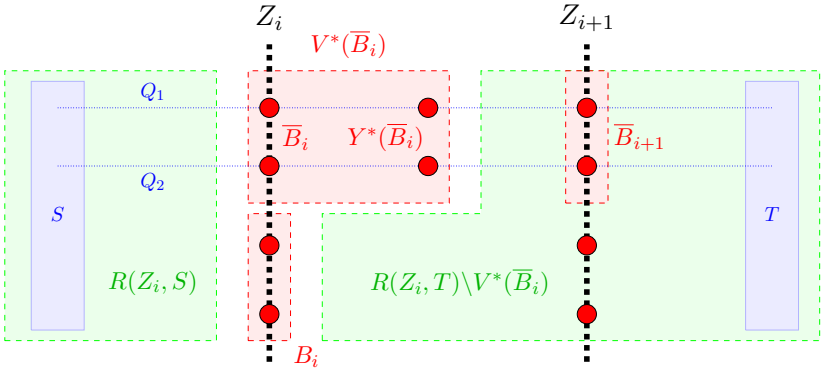


Figure 2.23: Illustration of Case 1: dam $\overline{B}_{i+1}$ arrives after $Y^*(\overline{B}_i)$.

When Case 1 occurs, we compute the global drainage $\mathcal{Z}(\mathcal{J})$ of instance $\mathcal{J} = \mathcal{D}(\mathcal{I}, \overline{B}_i)$. Set $\sigma(\overline{B}_i) = \bigcup_{j=2}^{r} K_j$ contains the Menger's paths of $\mathcal{D}(\mathcal{I}, \overline{B}_i)$. For any dam B_ℓ, where $\sigma(B_\ell) = K_2$, the algorithm is applied recursively for the dry instance of its complementary $\overline{B}_\ell$ in $\mathcal{Z}(\mathcal{J})$. The sum of these recursive calls is returned. Formally,

$$\text{count\_encl}\left(\mathcal{I}, \textbf{null}, \overline{B}_i, B_i, \mathcal{K}^{(2)}\right) = \sum_{\substack{\text{dam } B_\ell \in \mathcal{Z}(\mathcal{J}) \\ \sigma(B_\ell)=K_2}} \text{count\_encl}(\mathcal{J}, \textbf{null}, \overline{B}_\ell, B_\ell, \mathcal{K}^{(3)}), \tag{2.5}$$

where $\mathcal{K}^{(3)} = K_3 \cdots K_r$. In the general case, a frontier U and a subset $H_i \subsetneq B_i$ may be given as arguments. We consider an instance $\mathcal{H}$ and two dams B_i, $\overline{B}_i$ of the local drainage $\mathcal{Z}(\mathcal{H}, U)$. Let $\mathcal{K}^{(j)} = K_j \cdot K_{j+1} \cdots K_r$ be the input Menger's sequence. Eq. (2.5) becomes:

$$\text{count\_encl}\left(\mathcal{H}, U, \overline{B}_i, H_i, \mathcal{K}^{(j)}\right) = \sum_{\substack{\text{dam } B_\ell \in \mathcal{Z}(\mathcal{H},U) \\ \sigma(B_\ell)=K_j}} \text{count\_encl}\left(\mathcal{H}', \textbf{null}, \overline{B}_\ell, B_\ell, \mathcal{K}^{(j+1)}\right), \tag{2.6}$$

where $\mathcal{H}' = \mathcal{D}(\mathcal{H}, \overline{B}_i, H_i)$ and $\mathcal{K}^{(j+1)} = K_{j+1} \cdots K_r$. Line 13 in Algorithm 2 computes value $\text{count\_encl}\left(\mathcal{H}, U, \overline{B}_i, H_i, \mathcal{K}^{(j)}\right)$ following this formula.

- **Case 2:** Entire dam $\overline{B}_{i+1}$ is included in the dry instance $\mathcal{D}(\mathcal{I}, \overline{B}_i)$, *i.e.* $\overline{B}_{i+1} \subseteq V^*(\overline{B}_i)$. An illustration of this case is given in Figure 2.24.

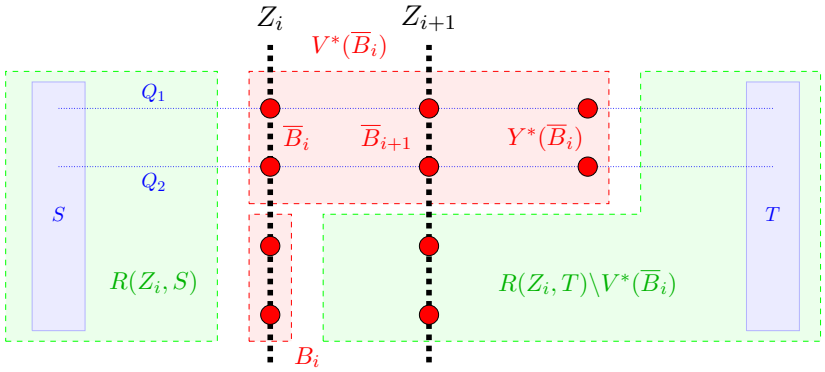


Figure 2.24: Illustration of Case 2: dam $\overline{B}_{i+1}$ is in $V^*(\overline{B}_i)$.

In this case, the dam $\overline{B}_{i+1}$ is a minimum cut for instance $\mathcal{D}(\mathcal{I}, \overline{B}_i)$ according to Lemma 2.13. We compute the local drainage of instance $\mathcal{J} = \mathcal{D}(\mathcal{I}, \overline{B}_i)$ with frontier $\overline{B}_{i+1}$. There are two types of minimum cuts in instance $\mathcal{J}$. Some of them contain at least one vertex in the source side of $\overline{B}_{i+1}$ and admit a front dam in the local drainage $\mathcal{Z}(\mathcal{J}, \overline{B}_{i+1})$ (Theorem 2.18). The other dams entirely belong to the target side of $\overline{B}_{i+1}$. We know from Theorem 2.24 that these cuts are the cuts of an enclosed instance of $\overline{B}_{i+1}$. We exploit these two theorems to determine the functioning of count_encl for Case 2. A recursive call is executed on the enclosed instance of $\overline{B}_{i+1}$ highlighted in Theorem 2.24. Moreover, we enumerate all dams B_ℓ with the signature K_2 of the local drainage $\mathcal{Z}(\mathcal{J}, \overline{B}_{i+1})$ (line 18). For each of them, the dry instance of their complementary is computed, *i.e.* instance $\mathcal{D}(\mathcal{J}, \overline{B_\ell})$. We make a recursive call on these instances and add up the values returned:

$$\text{count\_encl}\left(\mathcal{I}, \textbf{null}, \overline{B}_i, B_i, \mathcal{K}^{(2)}\right) = \text{count\_encl}\left(\mathcal{I}, \textbf{null}, \overline{B}_{i+1}, H_{i+1}, \mathcal{K}^{(2)}\right) + \sum_{\substack{\text{dam } B_\ell \in \mathcal{Z}(\mathcal{J},\overline{B}_{i+1}) \\ \sigma(B_\ell)=K_2}} \text{count\_encl}(\mathcal{J}, \overline{B}_{i+1}, \overline{B}_\ell, B_\ell, \mathcal{K}^{(3)}), \tag{2.7}$$

where $H_{i+1} = B_i \cap B_{i+1}$, in accordance with Theorem 2.24. For the general case

(line 19), Eq. (2.7) becomes:

$$\text{count\_encl}\left(\mathcal{H}, U, \overline{B}_i, H_i, \mathcal{K}^{(j)}\right) = \text{count\_encl}\left(\mathcal{H}, U, \overline{B}_{i+1}, H_{i+1}, \mathcal{K}^{(j)}\right) + \sum_{\substack{\text{dam } B_\ell \in \mathcal{Z}(\mathcal{H}, \overline{B}_{i+1}) \\ \sigma(B_\ell) = K_j}} \text{count\_encl}(\mathcal{H}', \overline{B}_{i+1}, \overline{B}_\ell, B_\ell, \mathcal{K}^{(j+1)}). \tag{2.8}$$

Here, set $H_{i+1} \subseteq B_{i+1}$ is equal to $H_i \cap B_{i+1}$ in order to be relevant with Theorem 2.24.

- **Case 3**: Some vertices of dam $\overline{B}_{i+1}$ are in the dry instance $\mathcal{D}(\mathcal{I}, \overline{B}_i)$, *i.e.* $\overline{B}_{i+1} \cap V^*(\overline{B}_i) \neq \emptyset$ and $\overline{B}_{i+1} \not\subseteq V^*(\overline{B}_i)$. Figure 2.25 provides an example of dry instance of this case. A vertex of $\overline{B}_{i+1}$ is in $V^*(\overline{B}_i)$, while another one is outside.

  We associate a frontier F with the dry instance $\mathcal{D}(\mathcal{I}, \overline{B}_i)$. Contrary to Case 2, dam $\overline{B}_i$ cannot be chosen as it does not lie entirely in $V^*(\overline{B}_i)$. We define F as:

$$F = \left[\overline{B}_{i+1} \cap V^*(\overline{B}_i)\right] \cup \left[Y^*(\overline{B}_i) \cap R(Z_{i+1}, S)\right] \tag{2.9}$$

  In Figure 2.25, the vertices in F are drawn in orange. We prove that set F is a cut for instance $\mathcal{D}(\mathcal{I}, \overline{B}_i)$. No edge jumps over the vertices of $\overline{B}_{i+1} \cap V^*(\overline{B}_i)$, otherwise Z_{i+1} would not be an (S,T)-cut. Therefore, no path bypassing F goes from $\overline{B}_i$ to $Y^*(\overline{B}_i)$ in the instance $\mathcal{D}(\mathcal{I}, \overline{B}_i)$.

  As cut F contains some vertices of the target set $Y^*(\overline{B}_i)$, the Menger's paths $\sigma(Y^*(\overline{B}_i) \cap R(Z_{i+1}, S)) \neq \emptyset$ arrive at target $Y^*(\overline{B}_i)$ before meeting the frontier F. The minimum cuts of $\mathcal{D}(\mathcal{I}, \overline{B}_i)$ contain one vertex of these paths. So, any minimum cut has a vertex inside F or its source side. According to Theorem 2.18, all minimum cuts of $\mathcal{D}(\mathcal{I}, \overline{B}_i)$ in Case 3 admit a front dam in the local drainage $\mathcal{Z}(\mathcal{J}, F)$, where $\mathcal{J} = \mathcal{D}(\mathcal{I}, \overline{B}_i)$.

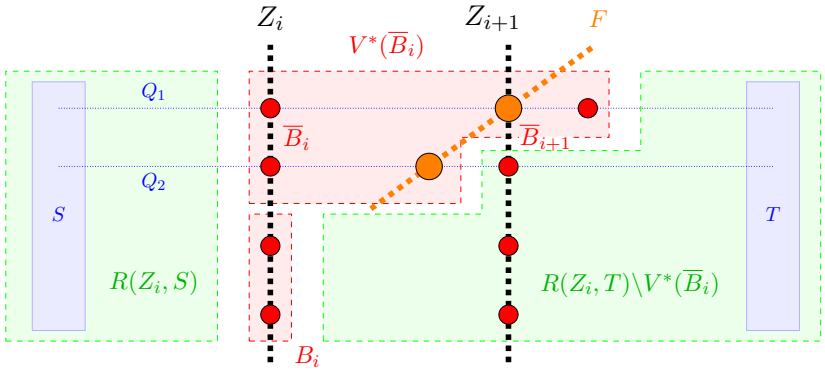


Figure 2.25: Illustration of Case 3: some vertices of dam $\overline{B}_{i+1}$ are in $V^*(\overline{B}_i)$.

Algorithm count_encl consists thus in computing the local drainage $\mathcal{Z}(\mathcal{J}, F)$ (line 24) and enumerating its dams $B_\ell \subsetneq Z_\ell^F$ fulfilling $\sigma(B_\ell) = K_2$ (line 27). For each of them, we determine the dry instance of their complementary $\mathcal{D}(\mathcal{J}, \overline{B_\ell})$. We terminate by making a recursive call for any of these instances and adding up the values returned:

$$\text{count\_encl}\left(\mathcal{I}, \textbf{null}, \overline{B}_i, B_i, \mathcal{K}^{(2)}\right) = \sum_{\substack{\text{dam } B_\ell \in \mathcal{Z}(\mathcal{J}, F) \\ \sigma(B_\ell) = K_2}} \text{count\_encl}(\mathcal{J}, F, \overline{B}_\ell, B_\ell, \mathcal{K}^{(3)}). \tag{2.10}$$

1: **Input:** Instance $\mathcal{H} = (G, S, T)$, its frontier U, dam $\overline{B}_i$, dam $H_i \subseteq B_i$, and Menger's sequence $\mathcal{K}^{(j)} = K_j \cdots K_r$ where $j \leq r$. /* U can be **null** */
2: res $\leftarrow 0$;
3: $p \leftarrow$ minimum (S,T)-cut size of G;
4: $\mathcal{Z}(\mathcal{H}, U) \leftarrow$ local drainage of $\mathcal{H}$ with frontier U;
5: /* If $U =$ **null**, then $\mathcal{Z}(\mathcal{H}, U) = \mathcal{Z}(\mathcal{H})$ is a global drainage. */
6: $\mathcal{H}' \leftarrow \mathcal{D}(\mathcal{H}, \overline{B}_i, H_i)$;
7: $\overline{B}_{i+1} \leftarrow$ dam of $\mathcal{Z}(\mathcal{H}, U)$ with level $i+1$ and signature $\sigma(\overline{B}_{i+1}) = \sigma(\overline{B}_i)$;
8: $\mathcal{K}^{(j+1)} \leftarrow K_{j+1} \cdots K_r$;
case *1* **do**
9: $\quad \mathcal{Z}(\mathcal{H}') \leftarrow$ global drainage of instance $\mathcal{H}'$;
10: $\quad$ **if** $j = r$ **then**
11: $\quad\quad$ **return** $|\mathcal{Z}(\mathcal{H}')|$;
$\quad$ **else**
12: $\quad\quad$ **for** *all dams* B_ℓ *of* $\mathcal{Z}(\mathcal{H}')$ *such that* $\sigma(B_\ell) = K_j$ **do**
13: $\quad\quad\quad$ res $\leftarrow$ res + count_encl$(\mathcal{H}', \textbf{null}, \overline{B}_\ell, B_\ell, \mathcal{K}^{(j+1)})$;
14: $\quad\quad$ **endfor**
$\quad$ **endif**
end
case *2* **do**
15: $\quad \mathcal{Z}(\mathcal{H}', \overline{B}_{i+1}) \leftarrow$ local drainage of instance $\mathcal{H}'$ with frontier $\overline{B}_{i+1}$;
16: $\quad$ **if** $j = r$ **then**
17: $\quad\quad$ **return** $\left|\mathcal{Z}(\mathcal{H}', \overline{B}_{i+1})\right|$ + count_encl $\left(\mathcal{H}, U, \overline{B}_{i+1}, H_{i+1}, \mathcal{K}^{(j)}\right)$;
$\quad$ **else**
18: $\quad\quad$ **for** *all dams* B_ℓ *of* $\mathcal{Z}(\mathcal{H}', \overline{B}_{i+1})$ *such that* $\sigma(B_\ell) = K_j$ **do**
19: $\quad\quad\quad$ res $\leftarrow$ res + count_encl$(\mathcal{H}', \overline{B}_{i+1}, \overline{B}_\ell, B_\ell, \mathcal{K}^{(j+1)})$;
20: $\quad\quad$ **endfor**
21: $\quad\quad H_{i+1} \leftarrow H_i \cap B_{i+1}$;
22: $\quad\quad$ res $\leftarrow$ res + count_encl $\left(\mathcal{H}, U, \overline{B}_{i+1}, H_{i+1}, \mathcal{K}^{(j)}\right)$;
$\quad$ **endif**
end
case *3* **do**
23: $\quad F \leftarrow \left[\overline{B}_{i+1} \cap V^*(\overline{B}_i)\right] \cup \left[Y^*(\overline{B}_i) \cap R(Z_{i+1}, S)\right]$;
24: $\quad \mathcal{Z}(\mathcal{H}', F) \leftarrow$ local drainage of instance $\mathcal{H}'$ with frontier F;
25: $\quad$ **if** $j = r$ **then**
26: $\quad\quad$ **return** $|\mathcal{Z}(\mathcal{H}', F)|$;
$\quad$ **else**
27: $\quad\quad$ **for** *all dams* B_ℓ *of* $\mathcal{Z}(\mathcal{H}', F)$ *such that* $\sigma(B_\ell) = K_j$ **do**
28: $\quad\quad\quad$ res $\leftarrow$ res + count_encl$(\mathcal{H}', F, \overline{B}_\ell, B_\ell, \mathcal{K}^{(j+1)})$;
29: $\quad\quad$ **endfor**
$\quad$ **endif**
end
30: **return** res;

Algorithm 2: The count_encl algorithm

When the input is made up of an enclosed instance $\mathcal{D}(\mathcal{H}, \overline{B}_i, H_i)$ where dam $\overline{B}_i$ belongs to the local drainage $\mathcal{Z}(\mathcal{H}, U)$ (line 28), Eq. (2.10) becomes:

$$\text{count\_encl}\left(\mathcal{H}, U, \overline{B}_i, H_i, \mathcal{K}^{(j)}\right) = \sum_{\substack{\text{dam } B_\ell \in \mathcal{Z}(\mathcal{H}', F) \\ \sigma(B_\ell) = K_j}} \text{count\_encl}(\mathcal{H}', F, \overline{B}_\ell, B_\ell, \mathcal{K}^{(j+1)}). \tag{2.11}$$

Frontier F is still defined as in Eq. (2.9). Now, set $\overline{B}_{i+1}$ is dam of the local drainage $\mathcal{Z}(\mathcal{H}, U)$ with the same signature as $\overline{B}_i$ and located on the next level.

The description of our DP-based algorithm is now complete. We will now prove not only that it returns the number of minimum (S,T)-cuts of the input graph G but also that it does it in FPT$\langle p \rangle$ time.

Analysis of the algorithm. We justify that the value count$(\mathcal{I})$ returned by the algorithm is equal to the number $C(\mathcal{I})$ of minimum cuts of instance $\mathcal{I}$.
The minimum drainage cuts of instance $\mathcal{I}$ are counted with the Menger's sequence composed of a single set, *i.e.* $\mathcal{K} = K_1$, $K_1 = \bigcup_{j=1}^{p} Q_j$ (line 8 in Algorithm 1). A consequence of Theorem 2.17 is that any other cut $X \neq Z_i$ admits a unique front dam $B_i \subsetneq X$ (Definition 2.24). Moreover, the remaining vertices $X \backslash B_i$ of cut X form a minimum cut of the dry instance $\mathcal{D}(\mathcal{I}, \overline{B}_i)$ (Theorems 2.21 and 2.20). Supposing that count_encl allows us to obtain the number of minimum cuts for any dry instance $\mathcal{D}(\mathcal{I}, \overline{B}_i)$, value count$(\mathcal{I})$ is equal to $C(\mathcal{I})$, as Algorithm 1 computes $C(\mathcal{I})$ by following Eq. (2.3).
Then, we focus on the algorithm count_encl. Value count_encl $\left(\mathcal{H}, U, \overline{B}_i, H_i, \mathcal{K}^{(j)}\right)$ is supposed to be the number of minimum cuts of the enclosed instance $\mathcal{D}(\mathcal{H}, \overline{B}_i, H_i)$ with its successive front dams having the signatures given by sequence $\mathcal{K}^{(j)} = K_j \cdots K_r$. If $j = r$, no recursive call is executed for Cases 1 and 3: the algorithm count_encl returns the number of minimum drainage cuts of the instance $\mathcal{D}(\mathcal{H}, \overline{B}_i, H_i)$. This is not as direct for Case 2, because $\mathcal{H}' = \mathcal{D}(\mathcal{H}, \overline{B}_i, H_i)$ coincides with an enclosed instance $\mathcal{D}(\mathcal{H}, \overline{B}_{i+1}, H_{i+1})$: we add up the number of cuts in $\mathcal{Z}(\mathcal{H}', \overline{B}_{i+1})$ with the value count_encl $\left(\mathcal{H}, U, \overline{B}_{i+1}, H_{i+1}, \mathcal{K}^{(j)}\right)$ returned for the instance $\mathcal{D}(\mathcal{H}, \overline{B}_{i+1}, H_{i+1})$ on the next level. In this way, we count for all Cases the minimum cuts of the instance $\mathcal{D}(\mathcal{H}, \overline{B}_i, H_i)$ which do not admit a front dam.
If $j < r$, the algorithm count_encl uses recursive calls on instances with a smaller minimum cut size.

- For Case 1, the algorithm count_encl computes the global drainage of $\mathcal{H}'$, enumerates all dams B_ℓ, $\sigma(B_\ell) = K_j$ and computes values count_encl $\left(\mathcal{H}', \textbf{null}, \overline{B}_\ell, B_\ell, \mathcal{K}^{(j+1)}\right)$ which are supposed to be the number of minimum cuts of the dry instances $\mathcal{D}(\mathcal{H}', \overline{B}_\ell)$. Adding up the results obtained recursively, we obtain the number of minimum cuts which admit a front dam with signature K_j, according to Theorems 2.20 and 2.21.

- For Case 2, cut $\overline{B}_{i+1}$ separates the vertex set of instance $\mathcal{H}'$ into two parts. According to Theorem 2.24, the target side of $\overline{B}_{i+1}$ coincides with the enclosed instance $\mathcal{D}(\mathcal{H}, \overline{B}_{i+1}, H_{i+1})$ where $H_{i+1} = H_i \cap B_{i+1}$. Our objective is to count the minimum cuts of $\mathcal{D}(\mathcal{H}, \overline{B}_i, H_i)$ such that the signature of its successive front dams follows sequence $\mathcal{K}^{(j)}$. Certain minimum cuts of $\mathcal{D}(\mathcal{H}, \overline{B}_i, H_i)$ have at least one vertex in $\overline{B}_{i+1}$ or its source side: in this case, they admit a front dam of the local drainage $\mathcal{Z}(\mathcal{H}', \overline{B}_{i+1})$. Otherwise, they are fully included in the target side of $\overline{B}_{i+1}$ and are also minimum cuts of the enclosed instance $\mathcal{D}(\mathcal{H}, \overline{B}_{i+1}, H_{i+1})$. This explains that the algorithm count_encl computes value count_encl$(\mathcal{H}, U, \overline{B}_{i+1}, H_{i+1}, \mathcal{K}^{(j)})$ in addition to the recursive calls count_encl$(\mathcal{H}', \overline{B}_{i+1},$
$\overline{B}_\ell, B_\ell, \mathcal{K}^{(j+1)})$ to the dry instances $\mathcal{D}(\mathcal{H}', \overline{B}_\ell, B_\ell)$, see Eq. (2.8).

- For Case 3, the dams of the local drainage $\mathcal{Z}(\mathcal{H}', F)$ are enumerated, where F is a frontier of $\mathcal{H}'$ containing vertices of both $\overline{B}_{i+1}$ and $Y^*(\overline{B}_i)$. As at least one vertex of any minimum cut of $\mathcal{H}'$ belongs to frontier F or its source side, all minimum cuts admit a front dam in $\mathcal{Z}(\mathcal{H}', F)$, according to Theorem 2.18. So, algorithm count_encl consists in making recursive calls to the dry instances of $\overline{B}_\ell$, where $\sigma(B_\ell) = K_j$. In this way, no minimum cut of $\mathcal{H}'$ admitting a front dam with signature K_j is forgotten.

Another important property ensuring that the algorithms count and count_encl return a correct result is the unicity of the front dam, obtained from Theorems 2.17 and 2.18 and

mentioned explicitly in Definitions 2.24 and 2.26. Thanks to it, any minimum cut cannot be counted twice. In particular, a minimum cut X of the enclosed instance $\mathcal{H}' = \mathcal{D}(\mathcal{H}, \overline{B}_i, H_i)$ is associated with only one Menger's sequence. Concretely, cut X is counted in count_encl$(\mathcal{H}, U, \overline{B}_i, H_i, \mathcal{K}^{(j)})$, where $\mathcal{K}^{(j)} = \mathcal{K}_X$, but not in count_encl$(\mathcal{H}, U, \overline{B}_i, H_i, \widehat{\mathcal{K}}^{(j)})$, if $\widehat{\mathcal{K}}^{(j)} \neq \mathcal{K}_X$. First, cut X admits a unique front dam B_ℓ in $\mathcal{H}'$ and the signature of B_ℓ is K_j, the first set of sequence $\mathcal{K}^{(j)}$. Second, the recursive call for dam B_ℓ formulated in Eqs. (2.6), (2.8), and (2.11) aims at counting the cuts $X \backslash B_\ell$ in the dry/enclosed instances of $\mathcal{H}'$. Such cuts also admit a unique front dam, with a signature K_{j+1}, the second set of sequence $\mathcal{K}^{(j)}$, and so on. In summary, a minimum cut X of $\mathcal{H}'$ cannot be associated with two Menger's sequences because of the unicity of the front dam for both global and local drainages.

Eventually, we say that any minimum cut is associated with a single Menger's sequence. Moreover, we showed that the value count_encl $\left(\mathcal{H}, U, \overline{B}_i, H_i, \mathcal{K}^{(j)}\right)$ returns the number of minimum cuts X in instance $\mathcal{D}(\mathcal{H}, \overline{B}_i, H_i)$ with a Menger's sequence $\mathcal{K}_X = \mathcal{K}^{(j)}$. Consequently, the execution of algorithm count allows us to count the number $C(\mathcal{I}) = \text{count}(\mathcal{I})$ of minimum (S,T)-cuts in the input instance $\mathcal{I}$.

We determine the running time of our algorithm. We prove that the execution time to obtain count$(\mathcal{I}, \mathcal{K})$ for any Menger's sequence $\mathcal{K}$ is at most $O^*(2^p)$. As there are $O(2^{p \log p})$ Menger's sequences, computing value count$(\mathcal{I})$ takes $O^*(2^{O(p \log p)})$.

The recursive calls executed by the algorithm count_encl can be represented with a directed graph without oriented cycles (DAG). Each vertex of the DAG is a 5-uplet formed by arguments of the recursive call. For example, if Case 1 is fulfilled for the 5-uplet $\left(\mathcal{H}, U, \overline{B}_i, H_i, \mathcal{K}^{(j)}\right)$, some arcs go from $\left(\mathcal{H}, U, \overline{B}_i, H_i, \mathcal{K}^{(j)}\right)$ to $\left(\mathcal{H}', \textbf{null}, \overline{B}_\ell, B_\ell, \mathcal{K}^{(j+1)}\right)$.

Figure 2.26 shows a fragment of the DAG: a 5-uplet $\left(\mathcal{I}, \textbf{null}, \overline{B}_i, B_i, \mathcal{K}^{(2)}\right)$, where $\mathcal{K}^{(2)} = K_2 \cdots K_r$ and some of its descendants. The 5-uplet $\left(\mathcal{I}, \textbf{null}, \overline{B}_i, B_i, \mathcal{K}^{(2)}\right)$ has no predecessor as it is called by the head algorithm count (line 11 of Algorithm 1). Dam B_i belongs to the drainage $\mathcal{Z}(\mathcal{I})$ and verifies $\sigma(B_i) = K_1$. In this example, $\left(\mathcal{I}, \textbf{null}, \overline{B}_i, B_i, \mathcal{K}^{(2)}\right)$ fulfils the conditions of Case 3. Its descendants thus contain the instance $\mathcal{J} = \mathcal{D}(\mathcal{I}, \overline{B}_i)$ with the frontier F, Eq. (2.10). One of them, $\left(\mathcal{J}, F, \overline{B}_\ell, B_\ell, \mathcal{K}^{(3)}\right)$, is represented in the figure. Dam B_ℓ belongs to the drainage $\mathcal{Z}(\mathcal{J}, F)$ and verifies $\sigma(B_\ell) = K_2$. Case 2 is satisfied: one of its successors is the enclosed instance $\left(\mathcal{J}, F, \overline{B}_{\ell+1}, H_{\ell+1}, \mathcal{K}^{(3)}\right)$ containing the same Menger's sequence $\mathcal{K}^{(3)}$, where $H_{\ell+1} = B_\ell \cap B_{\ell+1}$, Eq. (2.8). Its other successors look like $\left(\mathcal{J}', \overline{B}_{\ell+1}, \overline{B}_q, B_q, \mathcal{K}^{(4)}\right)$, where $\mathcal{J}' = \mathcal{D}(\mathcal{J}, \overline{B}_\ell)$. They contain a dam $\overline{B}_q$ of $\mathcal{Z}(\mathcal{J}', \overline{B}_{\ell+1})$ and sequence $\mathcal{K}^{(4)}$. The boxed numbers represent the values returned by function count_encl. In this figure, they are chosen arbitrarily.

For each 5-uplet of the DAG, our DP algorithm computes the dry instance $\mathcal{D}(\mathcal{H}, \overline{B}_i, H_i)$ and its drainage which depends on the Case satisfied. The dry instance $\mathcal{D}(\mathcal{H}, \overline{B}_i, H_i)$ is obtained by identifying the important cut $Y^*(\overline{B}_i, H_i)$ (in running time $O(mp)$ according to Lemma 2.4) and then determining its source side with a depth-first search (running time $O(n)$). Consequently, the time to retrieve both $\mathcal{D}(\mathcal{H}, \overline{B}_i, H_i)$ and its drainage is $O(mnp)$. It finally determines value count_encl$(\mathcal{H}, U, \overline{B}_i, H_i, \mathcal{K}^{(j)})$ which is a function of the results stored by its successors in the DAG. All these operations are done in polynomial time. Our objective is to show that the number of vertices in the DAG is FPT$\langle p \rangle$. More precisely, we prove that the number of 5-uplets considered by our algorithm is at most $2^p n$.

This upper bound is obtained thanks to the notion of *core*. For the 5-uplet $\left(\mathcal{H}, U, \overline{B}_i, H_i, \mathcal{K}^{(j)}\right)$, we say that its core is the set of vertices of instance $\mathcal{D}(\mathcal{H}, \overline{B}_i, H_i)$ which are also in the "augmented" source side of cut Z^U_{i+1} in $\mathcal{H}$, *i.e.* $R^+(Z^U_{i+1}, S)$. An equivalent formulation follows.

Definition 2.30 (Core of $\left(\mathcal{H}, U, \overline{B}_i, H_i, \mathcal{K}^{(j)}\right)$)**.** *Let $S_\mathcal{H}$ denote the sources of $\mathcal{H}$. The core of $\left(\mathcal{H}, U, \overline{B}_i, H_i, \mathcal{K}^{(j)}\right)$ contains the vertices of $V^*(\overline{B}_i, H_i)$ which are also in the source side of Z^U_{i+1} in instance $\mathcal{H}$: $V^*(\overline{B}_i, H_i) \cap R^+(Z^U_{i+1}, S)$.*

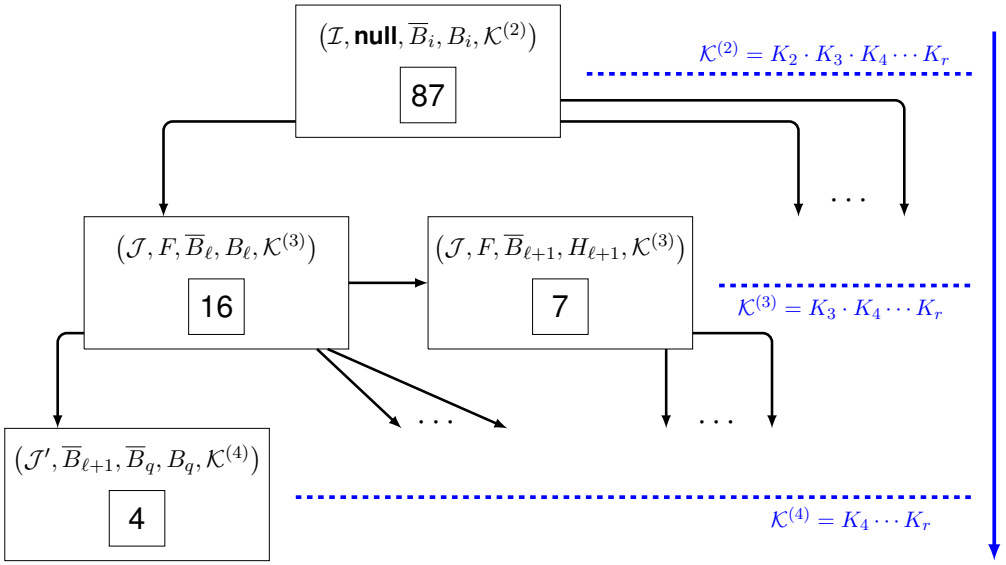


Figure 2.26: Illustration of the DAG describing the recursive calls to compute count$(\mathcal{I}, \mathcal{K})$

Figure 2.27 represents an instance $\mathcal{H}$ and the core of $(\mathcal{H}, U, \overline{B}_i, H_i, \mathcal{K}^{(j)})$ is highlighted by a crosshatched blue area inside set $V^*(\overline{B}_i, H_i)$.
We remind that sequence $\mathcal{K}^{(j)}$ is equal to $\mathcal{K}$ without its first $j-1$ sets. Let value p_{j-1} be the cardinality of the union $\bigcup_{h=1}^{j-1} K_h$ of sets withdrawn to obtain $\mathcal{K}^{(j)}$:

$$p_{j-1} = \left| \bigcup_{h=1}^{j-1} K_h \right| = \sum_{h=1}^{j-1} |K_h| \,. \tag{2.12}$$

We have $p_r = p$. For any vertex v of the input instance $\mathcal{I}$, we prove that v belongs to at most $2^{p_{j-1}}$ cores of 5-uplets in the DAG which contain the sequence $\mathcal{K}^{(j)}$.

Theorem 2.25. *Let v be a vertex of the instance $\mathcal{I}$. There are at most $2^{p_{j-1}}$ 5-uplets of the DAG containing the sequence $\mathcal{K}^{(j)}$ such that vertex v belongs to their core.*

Proof. We proceed inductively. Referring to line 11 of Algorithm 1, the 5-uplets of the DAG containing the sequence $\mathcal{K}^{(2)} = K_2 \cdots K_r$ are necessarily like $(\mathcal{I}, \textbf{null}, \overline{B}_i, B_i, \mathcal{K}^{(2)})$. There is one 5-uplet for each dam $\overline{B}_i$ of $\mathcal{Z}(\mathcal{I})$, where $\sigma(B_i) = K_1$. The core of $(\mathcal{I}, \textbf{null}, \overline{B}_i, B_i, \mathcal{K}^{(2)})$ contains the vertices between $\overline{B}_i$ and $\overline{B}_{i+1}$. So, the cores of all these 5-uplets are pairwise disjoint. Vertex v is in at most one of them, which is less than $2^{p_1} \geq 2$.
We suppose that the property holds for sequence $\mathcal{K}^{(j)}$ and prove it for $\mathcal{K}^{(j+1)}$. We focus on a 5-uplet $(\mathcal{H}, U, \overline{B}_i, H_i, \mathcal{K}^{(j)})$. Vertex v belongs to its core. We try to identify the descendants of $(\mathcal{H}, U, \overline{B}_i, H_i, \mathcal{K}^{(j)})$ in the DAG such that their core also contains v and their Menger's sequence is $\mathcal{K}^{(j+1)}$.
According to Eqs. (2.6), (2.8), and (2.11), the descendants of $(\mathcal{H}, U, \overline{B}_i, H_i, \mathcal{K}^{(j)})$ composed of the Menger's sequence $\mathcal{K}^{(j+1)}$ contain the instance $\mathcal{H}' = \mathcal{D}(\mathcal{H}, \overline{B}_i, H_i)$. Their frontier depends on the Case satisfied by the 5-uplet $(\mathcal{H}, U, \overline{B}_i, H_i, \mathcal{K}^{(j)})$: we denote it by U' ($U' =$ **null** for Case 1 for example).
There is a single drainage cut Z_ℓ^U in $\mathcal{Z}(\mathcal{H}, U)$ such that v is in $R^+(Z_{\ell+1}^U, S_\mathcal{H}) \backslash R^+(Z_\ell^U, S_\mathcal{H})$ because the source side of the drainage cuts are included one into another. Therefore, dam $\overline{B}_\ell$ is in the descendants we are looking for. This is a necessary condition to have v in the core of these 5-uplets.
We know that these descendants are composed of instance $\mathcal{H}'$, a frontier U' depending only on their ancestor $(\mathcal{H}, U, \overline{B}_i, H_i, \mathcal{K}^{(j)})$, dam $\overline{B}_\ell$, and sequence $\mathcal{K}^{(j+1)}$. Only the fourth component $H_\ell \subseteq B_\ell$ of these 5-uplets is unknown. As set B_ℓ is made up of exactly $|K_j|$

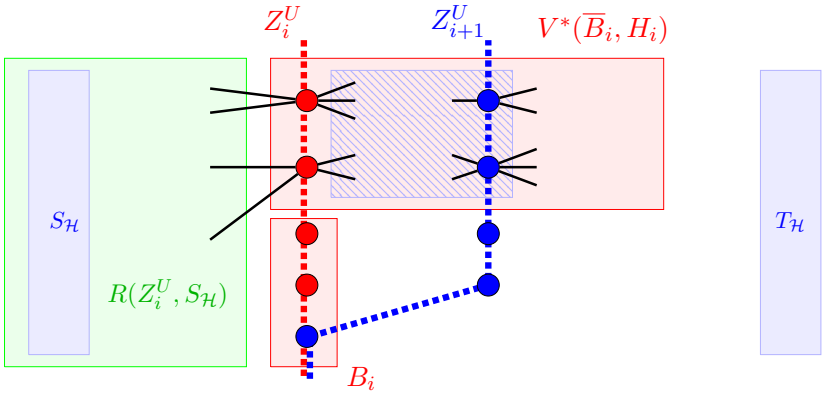


Figure 2.27: Instance $\mathcal{H}$ and core of a 5-uplet $(\mathcal{H}, U, \overline{B}_i, H_i, \mathcal{K}^{(j)})$

vertices, there are $2^{|K_j|}$ possible sets H_ℓ. In other words, at most $2^{|K_j|}$ descendants of the 5-uplet $(\mathcal{H}, U, \overline{B}_i, H_i, \mathcal{K}^{(j)})$ with sequence $\mathcal{K}^{(j+1)}$ have a core containing v.
Conversely, if the core of a 5-uplet with sequence $\mathcal{K}^{(j+1)}$ contains v, then we prove that at least one ancestor of this 5-uplet admits sequence $\mathcal{K}^{(j)}$ and has a core also containing v. Suppose that vertex v is in the core of $(\mathcal{H}', U', \overline{B}_\ell, H_\ell, \mathcal{K}^{(j+1)})$. According to Eqs. (2.6), (2.8), and (2.11), at least one ancestor of this 5-uplet is composed of an instance $\mathcal{H}$, a dam $\overline{B}_i$ of the drainage $\mathcal{Z}(\mathcal{H}, U)$ and a set H_i such that $\mathcal{H}' = \mathcal{D}(\mathcal{H}, \overline{B}_i, H_i)$. For all Cases, all the dams enumerated for the computation of count_encl$(\mathcal{H}, U, \overline{B}_i, H_i, \mathcal{K}^{(j)})$, as $\overline{B}_\ell$, are located between dams $\overline{B}_i$ and $\overline{B}_{i+1}$. Indeed, the instance $\mathcal{H}'$ lies entirely between Z_i^U and Z_{i+1}^U for Case 1 while the frontier chosen for Cases 2 and 3 does not go beyond $\overline{B}_{i+1}$. Consequently, vertex v is also between dams $\overline{B}_i$ and $\overline{B}_{i+1}$. Said differently, it belongs to the core of $(\mathcal{H}, U, \overline{B}_i, H_i, \mathcal{K}^{(j)})$.
In a nutshell, we know that each 5-uplet with sequence $\mathcal{K}^{(j+1)}$ such that v is in its core admits at least one ancestor with sequence $\mathcal{K}^{(j)}$ whose core also contains v. According to the induction hypothesis, at most $2^{p_{j-1}}$ of these ancestors exist. As each of them admits at most $2^{|K_j|}$ descendants with sequence $\mathcal{K}^{(j+1)}$ and a core containing v, the total number of 5-uplets with a sequence $\mathcal{K}^{(j+1)}$ satisfying the desired property is upper-bounded by $2^{p_{j-1}} 2^{|K_j|} = 2^{p_j}$. The induction process is thus completed. □

We have an upper bound on the number of 5-uplets with sequence $\mathcal{K}^{(j)}$ and a core containing a given vertex v, for any $j \in \{1, \ldots, r\}$. We deduce that the total number of instances in the DAG is FPT$\langle p \rangle$.

Theorem 2.26. *There are at most $2^p n$ 5-uplets in the DAG describing the recursive calls of the algorithm count_encl.*

Proof. Let $v \in V$. Sequence $(p_j)_{1 \leq j \leq r}$, defined in Eq. (2.12), takes values in $\{1, \ldots, p\}$ and is strictly increasing. The number of 5-uplets such that their core contains v is less than:

$$\sum_{j=1}^{r} 2^{p_{j-1}} \leq \sum_{j=1}^{p-1} 2^j \leq 2^p$$

Any 5-uplet admits a nonempty core. As there are n vertices in the graph, the total number of 5-uplets is at most $2^p n$. □

The time complexity of our algorithm depends on the operations made for the 5-uplets of the DAG which represents the recursive calls. We already explained that each 5-uplet requires a polynomial running time. The overall running time to compute count$(\mathcal{I}, \mathcal{K})$ is thus $O(2^p mn^2 p)$. Due to the number of Menger's sequences, the execution time to obtain count$(\mathcal{I})$ is $O(2^{O(p \log p)} mn^2)$.

Theorem 2.27. *The number of minimum vertex (S,T)-cuts is obtained in $O(2^{O(p\log p)}mn^2)$.*

The sampling of minimum vertex (S,T)-cuts is naturally deduced from the technique used to sample edge (S,T)-cuts in Section 2.3.1. Our algorithm counting the minimum vertex (S,T)-cuts is executed. First, each minimum drainage cut Z_i has a probability $\frac{1}{C(\mathcal{I})}$ to be selected. Second, for the minimum cuts admitting a front dam, each 5-uplet $(\mathcal{I}, \textbf{null}, \overline{B}_i, B_i, \mathcal{K}^{(2)})$ is associated with a probability $\frac{\text{count\_encl}(\mathcal{I},\textbf{null},\overline{B}_i,B_i,\mathcal{K}^{(2)})}{C(\mathcal{I})}$. One of these 5-uplets is selected in accordance with this probability distribution. As in Section 2.3.1, we go down the DAG, associating with each 5-uplet the probability distributed uniformly.

2.4 Conclusion

We identified two cut problems on undirected graphs which we solved in time $O^*(2^{\text{poly}(p)})$, where poly is a polynomial function. The first one, called POTC, is a decision problem asking for a small separation between r sources and a single target. The second one is the counting of minimum edge/vertex (S,T)-cuts.

The complexity of POTC parameterized by k, r, and p is now fully determined (Table 2.1). Our main result is the design of an FPT$\langle p\rangle$ algorithm computing a solution for EDGE POTC in time $O^*(2^{O(p^2)})$. It uses important cuts, color-coding techniques, and *edge passes*, a concept we devised. Parameterized by r only, EDGE POTC is W[1]-hard. Furthermore, we proved that VERTEX POTC is W[1]-hard for all parametrizations involving r or p.

These results have an impact on the more general problem PARTIAL MULTICUT, which asks for a cut X, $|X| = p$, separating r pairs of terminals (s_i, t_i) among k. The consequence is that VERTEX PARTIAL MULTICUT is W[1]-hard parameterized by r and p. This highlights an interesting complexity gap, as its sub-problem VERTEX MULTICUT is FPT$\langle p\rangle$ [63]. However, the parameterized complexity of EDGE PARTIAL MULTICUT remains open.

We built two algorithms to tackle the counting of minimum (S,T)-cuts in FPT$\langle p\rangle$ time, where p is the size of the minimum cuts. The first one is dedicated to edge (S,T)-cuts. It uses our concept, the *drainage*, which characterizes all minimum (S,T)-cuts of a graph G. This succession of at most n disjoint minimum cuts Z_i is such that any minimum (S,T)-cut X has a nonempty intersection with one of the drainage cuts. The recursion benefitting from this construction returns the number of minimum (S,T)-cuts in time $O^*(2^{O(p^2)})$.

The edge cut counting inspired us to count the minimum vertex (S,T)-cuts, which is a more general problem. For this occasion, we not only generalize the concepts used for the counting of edge cuts but also introduce new ones which shorten the execution time obtained before. We define the *local drainage*, a succession of minimum (S,T)-cuts Z_i^U such that all minimum (S,T)-cuts with at least one vertex in the source side of the frontier-cut U intersect one Z_i^U. This tool, in addition to structures devised to characterize minimum vertex (S,T)-cuts, make possible the design of a dynamic programming algorithm. Its running time, proportional to the number of recursive calls executed, is in $O^*(2^{O(p\log p)})$.

The sampling of minimum (S,T)-cuts, which consists in returning one of the minimum (S,T)-cuts with the uniform distribution, can be solved with these algorithms after minor changes. Therefore, we exhibit a method to sample minimum edge/vertex (S,T)-cuts in time $O^*(2^{O(p\log p)})$.

The counting of minimum (S,T)-cuts in directed graphs is more general than the counting of minimum vertex (S,T)-cuts in undirected graphs. As a consequence, we believe that the techniques introduced to count cuts in undirected graphs can help us to find an efficient FPT$\langle p\rangle$ algorithm for directed graphs. For the moment, the treewidth reduction from [62] provides an algorithm in $O^*(2^{2^p})$ for such graphs.

Chapter 3

Bypassing blockages in graphs

We investigated the parameterized complexity of certain cut problems to detect blockages which split a graph in disjoint components. The objective was to identify small blockage sets in a reasonable time.
This chapter deals with another facet of the handling of blockages in a graph. We focus on the problem where a traveller wants to go from s to t with the minimum cost, knowing that some edges of the graph may be blocked. These edges are hidden from the traveller. He discovers them when visiting one of their endpoints. This problem is known under the name of the Canadian Traveller Problem (CTP) [70]. A solution to the CTP is an online algorithm, also called a *strategy*, guiding the traveller during his trip. We provide the reader with our results on the competitive analysis of strategies for the CTP.
In Section 3.1, the state of the art for the CTP is given. We list the problems which, as the CTP, are generalizations of the SHORTEST PATH problem with uncertainty on graphs. Then, we remind results about the competitive ratio of deterministic and randomized strategies for the CTP known up to now. In the following sections, we describe our contributions on the competitive ratio of strategies for the CTP.
In Section 3.2, we justify our idea of splitting our results into two parts and discussing them separately. The contributions of the first group concern the competitiveness of strategies on the entire set of instances (global competitiveness). Those categorized in the second group deal with strategies dedicated to certain families of graphs (local competitiveness).
First, we introduce the results [13, 15, 14] on the global competitiveness of both deterministic and randomized strategies (Section 3.3). Second, we treat the local competitiveness side [18, 19], where the goal is to devise competitive strategies for certain types of graphs (Section 3.4). In Section 3.5 we draw conclusions on our contributions.

3.1 State of the art

We remind the definition of the CTP and one of its variants, k-CTP. The definition of the competitive ratio is also cited. We introduce the relationship between the CTP and other problems asking for shortest paths in graphs given with incomplete information. To complete this overview, certain variants of the CTP are evoked.
Then, we focus on the results from the literature related to our work. The state of the art for both global and local competitiveness is presented. We examine particularly the global competitiveness of deterministic strategies which has been fully treated. We also give the intuition of the most important strategies already proposed.

3.1.1 The CTP and the competitive ratio

We begin with the definition of the CTP before introducing the performance evaluation of strategies with the competitive ratio.

Definition of the CTP

The *Canadian Traveller Problem* (CTP) was introduced by Papadimitriou and Yannakakis and proven PSPACE-complete [70]. Given an undirected weighted graph $G = (V, E, \omega)$, $\omega : \mathbb{E} \to \mathbb{Q}^+$, and two vertices $s, t \in V$, the objective is to make a traveller walk from s to t on graph G in the most efficient way despite the fact that certain edges may be blocked. The traveller does not know which edges are blocked when he begins his walk and discovers them when he visits an endpoint of these edges. The set of blocked edges is denoted by $E_* \subsetneq E$. Let $G \backslash E_*$ denote graph G deprived of the blocked edges. The pair (G, E_*) is called a *road map*. Any road map considered in this study is feasible: there is an (s, t)-path in graph $G \backslash E_*$. In other words, the blocked edges do not separate source s from target t.

Definition 3.1 (The Canadian Traveller Problem)**.**
Input: *Undirected weighted graph* $G = (V, E, \omega)$, $\omega : \mathbb{E} \to \mathbb{Q}^+$, *vertices* s, t
Hidden input: *Blocked edges* $E_* \subsetneq E$.
Objective: *Traverse graph* G *from* s *to* t *with the minimum cost.*

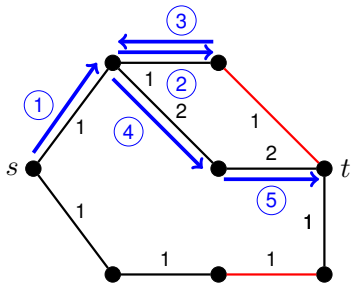


Figure 3.1: An example of trip bypassing two blocked edges (these in red).

Figure 3.1 illustrates a trip (blue arcs) of the traveller on a feasible road map. Blocked edges are drawn in red. The black numbers are the edge weights. The blue encircled numbers indicate the order in which the traveller may traverse edges.
The k-*Canadian Traveller Problem* (k-CTP) is the parameterized version of CTP, where k is an upper bound on the total number of blocked edges. The k-CTP is also PSPACE-complete [9, 70].

Definition 3.2 (The k-Canadian Traveller Problem)**.**
Input: *Undirected weighted graph* $G = (V, E, \omega)$, $\omega : \mathbb{E} \to \mathbb{Q}^+$, *vertices* s, t, *integer* $k \geq 1$
Hidden input: *Blocked edges* $E_* \subsetneq E$, $|E_*| \leq k$
Objective: *Traverse graph* G *from* s *to* t *with the minimum cost.*

Online algorithms, also called *strategies*, guide the traveller during his trip. The competitive ratio is a way to assess their performance. Strategies are either deterministic or randomized, if the decisions made depend on a random draw.

Competitive ratio

We remind the definition of the competitive ratio introduced in [26], in the context of the CTP. Letter A denotes a deterministic strategy. Let $\omega_A(G, E_*)$ be the distance traversed by the traveller from s to t guided by a strategy A on graph G with blocked edges E_*. The shortest (s, t)-path in graph $G \backslash E_*$ is called the *optimal offline path* of map (G, E_*) and its cost, noted $\omega_{\text{opt}} = \omega_{\text{min}}(G, E_*)$, is the optimal offline cost of map (G, E_*). The competitive ratio of a deterministic strategy A over a road map (G, E_*) is the value $c_A(G, E_*) = \frac{\omega_A(G, E_*)}{\omega_{\text{opt}}}$. The competitive ratio c_A of strategy A is thus the maximum value $c_A(G, E_*)$ taken over all road maps (G, E_*). We have:

$$c_A = \max_{(G,E_*)} c_A(G, E_*) = \max_{(G,E_*)} \frac{\omega_A(G, E_*)}{\omega_{\text{opt}}}.$$

For randomized strategies, the measure is based on the expected distance traversed by the traveller. It corresponds to the mean distance traversed by the traveller against an oblivious adversary setting down the blocked edges on the graph. This adversary is not able to guess the result of the random draws computed by the traveller. The competitive ratio of a randomized strategy A' over a road map (G, E_*) is the value $c_{A'}(G, E_*) = \frac{\mathbb{E}[\omega_{A'}(G,E_*)]}{\omega_{\text{opt}}}$. The competitive ratio $c_{A'}$ of strategy A' follows:

$$c_{A'} = \max_{(G,E_*)} c_{A'}(G, E_*) = \max_{(G,E_*)} \frac{\mathbb{E}[\omega_{A'}(G, E_*)]}{\omega_{\text{opt}}}.$$

For both deterministic and randomized strategies, the competitive ratio refers to these definitions in the literature. In this study, we call it the *global* competitive ratio because we aim at defining another way to measure the performance of strategies. Indeed, the definition of the competitive ratio can be adapted to a set $\mathcal{R}$ of road maps. This is what we call a *local* competitive ratio. Put formally, strategy A is $c_{A,\mathcal{R}}$-competitive if $c_{A,\mathcal{R}} = \max_{(G,E_*)\in\mathcal{R}} c_A(G, E_*)$.

Although our contributions concern the competitive ratio of strategies, we remark that another way to evaluate the performance of online algorithms is the worst-case criterion [10]. To the best of our knowledge, no past work deals with the worst-case performance of strategies for the CTP.

3.1.2 Results from the literature

The following paragraph summarizes the results reported in the scientific literature on the generalizations of the SHORTEST PATH problem with uncertainty on the input graph. The CTP and its variants, which are presented, belong to this class of problems. Then, we provide a detailed description of the results known for the CTP, related to our contributions.

Related problems

Both the CTP and the k-CTP belong to a class of problems generalizing the SHORTEST PATH problem where some information of the input graph is unknown. In our case, the uncertainty is that we do not know where the blocked edges are placed. We begin with the state-of-the art of problems for which the lack of information does not lie in the location of the blockages.

For a problem called DOUBLE-VALUED GRAPH, we do not know the weight function ω. DOUBLE-VALUED GRAPH is PSPACE-complete and was also introduced by Papadimitriou and Yannakakis [70]. It asks for the shortest trip from s to t on a graph where two possible costs are associated with some edges, called *traffic jams* in [60]. The traveller discovers the real weight of an edge when visiting one of its endpoints. Liao and Huang [60] studied the competitive ratio of deterministic and randomized strategies for DOUBLE-VALUED GRAPH. Their main result was to identify a deterministic strategy achieving the optimal competitive ratio $\min(r, k+1)$, where k is the number of traffic jams and r is the worst-case performance ratio. A generalization of this problem associates a cost distribution to any edge [69, 70].

Another problem related to the CTP is the OBSTACLE NEUTRALIZATION problem [3]. It models the navigation of a traveller from s to t on a plane with obstacles that can be neutralized at a cost added to the traversal length. A well-performing algorithm of weak time complexity, called *Penalty Search Algorithm*, is proposed in [3].

Let us recenter now on the variants derived directly from the CTP. On the one hand, the *Stochastic* CTP [70] associates to any edge e a probability $q(e)$ that e is blocked. The objective is to minimize the expected cost of the trip from s to t. The PSPACE-hardness of the

Stochastic CTP has been proven recently [51]. Bnaya *et al.* identified a polynomial-time optimal strategy for graphs made up only of disjoint paths [23]. Computational experiments were proposed on the U.S. Navy minefield dataset COBRA [1]. In this context, the Canadian traveller is a robot trying to reach t and avoiding land mines. Different algorithms using a Markov decision process formulation of the problem were compared: they compute the shortest expected walk in exponential time.
A variant of the *Stochastic* CTP with *remote-sensing* was also studied [23, 51]. Remote-sensing actions are used to reveal the status of a non-incident edge. Given the position v of the traveller and an edge e, a cost sc(v, e) is associated with the pair (v, e). With a cost sc(v, e), the traveller on vertex v knows whether edge e is blocked. It was proven that this variant is NP-hard even for graphs made up of vertex-disjoint (s, t)-paths [51].
On the other hand, the *Recoverable* CTP allows the traveller to clear the blockages incident to his position. This is possible with a certain cost, depending on both the position of the traveller and the blocked edge considered. This variant was studied in [9]. In the case where the recovery times are not long compared to the travel times, a polynomial-time strategy guarantees the shortest worst-case travel time [9].
Our work concerns the problems CTP, k-CTP, and k-CTP for multiple travellers. In the next paragraph, we go into details with the results already established for these problems.

Survey of the competitiveness

First, we present a survey for the CTP and the k-CTP with $L = 1$ traveller. The open questions on the competitive ratio of deterministic strategies for general graphs have been totally answered. We begin with an overview of the deterministic case. Then, we introduce the state of the art for randomized strategies.
Westphal [75] proved that no deterministic strategy for the k-CTP achieves a competitive ratio better than $2k+1$. This ratio is obtained by considering graphs made up of $k+1$ simple vertex-disjoint (s, t)-paths of similar cost, where k of them are blocked. In such an instance (Figure 3.2), the traveller has no choice but traversing successively the (s, t)-paths and hoping that they are open. When he is blocked, he comes to back to s and tries to traverse another path. He potentially traverses the open path at the latest, which produces the ratio $2k + 1$. The distance traversed on G^W is $2k + 1 + \varepsilon$ while the optimal offline cost is $1 + \varepsilon$. Making ε tend to zero terminates the proof. In fact, as these instances verify $|E_*| = k$, this also proves that no deterministic strategy drops below competitive ratio $2\,|E_*| + 1$ for the CTP.

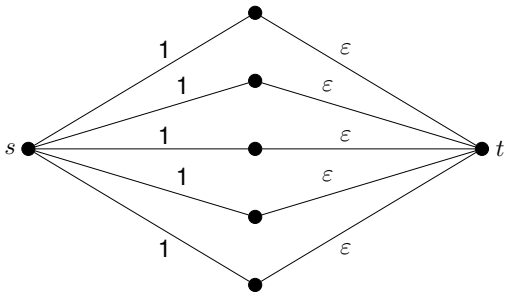


Figure 3.2: Graph G^W proposed in [75] to obtain the lower bound $2k + 1$.

The polynomial-time REPOSITION strategy [75] attains ratios $2\,|E_*|+1$ for the CTP and $2k+1$ for the k-CTP. It makes the traveller traverse the shortest (s, t)-path of the input graph. If there is a blocked edge (u, v) on this path, the traveller discovers it when he visits vertex u. Then, he comes back to s passing through the same path. The process restarts on $G\backslash E'_*$, graph G deprived of the blocked edges $E'_* \subseteq E_*$ identified until now. With this strategy, the traveller may be blocked k times. This is why he potentially achieves k trips, one for each

blockage, in both directions (towards t, then towards s) with an extra open trip to reach t in only one direction (towards t). Ratio $2|E_*|+1$ comes from this worst-case scenario.
The GREEDY strategy [77] is also a polynomial-time deterministic strategy. It is certainly the most intuitive one: it consists in choosing at each step of the walk the first edge of the shortest path between the current position of the traveller and the target t. However, its competitive ratio is $2^{k+1}-1$ for the k-CTP. Figure 3.3 provides the road map for $k=4$ on which GREEDY strategy attains this competitive ratio. The red edges are blocked. The GREEDY strategy makes the traveller traverse the edges with exponentially increasing costs.

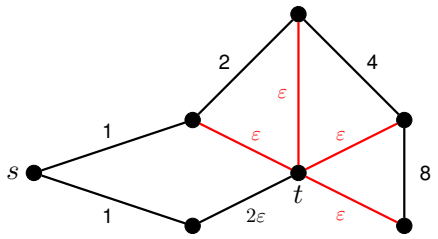


Figure 3.3: The road map for which the competitive ratio of GREEDY is $2^{k+1}-1$.

Another strategy called COMPARISON [77] is as competitive as REPOSITION. COMPARISON mixes the two previous strategies REPOSITION and GREEDY. When the traveller discovers a blockage (u,v) and stands on vertex u, he compares the shortest (u,t)-path $P_{\min}^{(u,t)}$ (cost $\omega_{\min}^{(u,t)}$) of $G\backslash E'_*$ with its shortest (s,t)-path $P_{\min}^{(s,t)}$ (cost $\omega_{\min}^{(s,t)}$). If $\omega_{\min}^{(s,t)} \leq \omega_{\min}^{(u,t)}$, the traveller moves as in REPOSITION: he comes back to s and traverses $P_{\min}^{(s,t)}$. If $\omega_{\min}^{(s,t)} > \omega_{\min}^{(u,t)}$, the traveller traverses the path $P_{\min}^{(u,t)}$ as he does with the GREEDY strategy. COMPARISON ensures that the "greedy option" is chosen only when it improves the performance of REPOSITION. As COMPARISON behaves as REPOSITION on graphs with vertex-disjoint (s,t)-paths, its competitive ratios are $2|E_*|+1$ for the CTP and $2k+1$ for the k-CTP.
In the literature, no distinction was made between the results on the CTP and the k-CTP. All the articles [11, 42, 72, 75, 79] provide results with the competitive ratio expressed as a function of k. However, in the original CTP, parameter k is absent. So, the competitive ratio must depend on $|E_*|$ as the strategies cannot make decisions depending on k here.
We observed that all the results given for the k-CTP can be extended to the classic CTP version after replacing k by the number of blocked edges $|E_*|$. In theory, differences of competitiveness between the CTP and the k-CTP might occur. In our work, all results presented for the k-CTP can also be extended to the CTP. However, we highlight this distinction as one of our results holds only for the CTP, not the k-CTP.
Westphal [75] proved, using graph G^W once again (Figure 3.2), that no randomized strategy attains a ratio smaller than $k+1$ for the k-CTP. Let us suppose that the traveller walks on G^W. All $k+1$ (s,t)-paths of G^W are indistinguishable. As a consequence, the most competitive randomized strategy for this graph consists necessarily in selecting successively one of the (s,t)-paths with uniform distribution. The probability to select the open (s,t)-path with the first draw is $\frac{1}{k+1}$. In this case, the total distance traversed is $1+\varepsilon$. The probability to be blocked on the path selected first and to traverse the open one during the second draw is $\frac{k}{k+1}\frac{1}{k}=\frac{1}{k+1}$. In this case, the total distance traversed is $3+\varepsilon$. Eventually, considering ε negligible compared to 1, the mean distance traversed is written:

$$\sum_{i=0}^{k} \frac{1}{k+1}(2i+1) = k+1.$$

This reasoning shows that no randomized strategy can defeat the competitive ratio $k+1$ on such a road map. Therefore, value $k+1$ is a lower bound for the competitive ratio of any randomized strategy. As the proof consists in studying the same instance as in the

deterministic case, it implies that any randomized strategy is at best $(|E_*|+1)$-competitive for the CTP.
However, the identification of an $(|E_*|+1)$-competitive randomized strategy for CTP (and a $k+1$-competitive randomized strategy for k-CTP) has not been achieved yet. Table 3.1 summarizes the state of the art of CTP and k-CTP and formulates two open questions.

	Deterministic strategies	
	CTP	**k-CTP**
Result	REPOSITION strategy is optimal and $(2(\|E_*\|)+1)$-competitive.	REPOSITION strategy is optimal and $(2k+1)$-competitive.
	Randomized strategies	
	CTP	**k-CTP**
Result	Any randomized strategy A is c_A-competitive with $c_A \geq \|E_*\|+1$.	Any randomized strategy A is c_A-competitive with $c_A \geq k+1$.
Open Question	Can we find a strategy which is $(\|E_*\|+1)$-competitive?	Can we find a strategy which is $(k+1)$-competitive?

Table 3.1: state of the art [11, 75] and open questions for the CTP and k-CTP

In fact, no strategy with competitive ratio $\beta k + O(1)$ with $\beta < 2$ has been proposed yet. We do not know whether a polynomial-time randomized strategy defeats REPOSITION in this way.
Two randomized strategies have been proposed for certain families of graphs. Demaine *et al.* [42] designed a strategy with a ratio $\left(1+\frac{\sqrt{2}}{2}\right)k+1$, executed in time of $O\left(k\lambda^2|E|^2\right)$, where parameter λ may be exponential. It is dedicated to graphs that can be transformed into apex trees using a polynomial-time process they devised. Apex trees are graphs composed of a tree rooted in t and edges connecting s with the vertices of the tree. Bender *et al.* studied in [11] the competitivenes of randomized strategies for the k-CTP on graphs composed of vertex-disjoint (s,t)-paths. They proposed a polynomial-time strategy with ratio $k+1$, which is optimal. Some details of the proof are added in [73].
For multiple travellers ($L > 1$), two criteria may orientate the definition of the competitive ratio: time and distance. Two articles [72, 79] presented the first results on the time competitive ratio of strategies for the k-CTP. Their authors assume that all the travellers' speed is the same and constant. On the one hand, the time competitive ratio of a strategy with $L \geq 1$ travellers is the ratio of the time taken by the travellers to reach t with this strategy and the optimal offline cost, which is the time which would have been taken by one traveller if blocked edges had been known in advance. On the other hand, the distance competitive ratio is the ratio of the distance traversed by all travellers and the distance traversed by one traveller knowing which edges are blocked.
For $L = 2$, the ALTERNATING strategy [79] reaches the optimal time ratio $k+1$. For $L \geq 3$, we only know that no deterministic strategy can drop below time ratio $2\lfloor\frac{k}{L}\rfloor+1$ but no strategy reaching this bound has been identified yet. To the best of our knowledge, no result on the distance competitive ratio was established either.

3.2 Global and local approaches for the competitiveness of strategies

Most of the results for the CTP and the k-CTP introduced in the previous section concern the competitive ratio of strategies for general graphs. We also mentioned two results on the competitive ratio of strategies for certain families of graphs [11, 42]. Any result on

a specific family of graphs has no consequence on the competitive ratio of strategies for general graphs. Moreover, certain graphs obtained from real-world applications belong to a well-known family of graphs. As a consequence, focusing locally on the competitive ratio can make emerge strategies to be used in practice, even if they are not optimal for the entire set of instances. For these two reasons, we believe that the global and the local perspectives should be separated.

We propose to divide the results on the competitive ratio in two classes. On the one hand, the *global competitiveness* refers to the results on the competitive ratio of both deterministic and randomized strategies for general graphs, which is the classic definition [26]. For example, the statement formulated in [75] that no randomized strategy defeats ratio $k+1$ for the k-CTP is global. In this context, the competitive ratio of a strategy is the maximum competitive ratio over all existing road maps (G, E_*), where $|E_*| \leq k$. On the other hand, the *local competitiveness* includes all results about the competitive ratio of strategies on a certain family of graphs. The statement formulated in [11] that a randomized strategy is $(k+1)$-competitive for graphs made up of vertex-disjoint (s,t)-paths is local. The competitive ratio of a strategy on a family $\mathcal{F}$ of graphs is the maximum competitive ratio over all road maps satisfying $G \in \mathcal{F}$ and $|E_*| \leq k$.

The global competitiveness of deterministic strategies for the k-CTP was fully treated as we know both a lower bound, $2k+1$, for the global competitive ratio and strategies, REPOSITION and COMPARISON, achieving this limit.

The main open question related to the global competitiveness of strategies for the CTP and the k-CTP is whether a randomized strategy defeats the deterministic ones, *i.e.* admits a competitive ratio less than $2k+1$. Two of our contributions try to answer this question. We show in [14] that the randomized memoryless strategies cannot drop below the competitive ratio $2k + O(1)$. Then, we prove that no randomized strategy reaches ratio $|E_*| + 1$ for the CTP only [15]. Moreover, a possible line of research is to pursue the study of the global competitiveness for multiple travellers [13]. We assess the distance competitive ratio of deterministic and randomized strategies with $L > 1$ travellers. Section 3.3 describes the over-mentioned studies.

Our last two contributions for the CTP deal with the local competitiveness for a single traveller (Section 3.4). We show how the traveller benefits from the maximum edge (s,t)-cut size. Concretely, we design a deterministic strategy achieving the competitive ratio $2\mu_{\mathsf{max}} + \sqrt{2}(k - \mu_{\mathsf{max}}) + 1$, where μ_{max} is the size of the largest minimal (s,t)-cut. Ratio $2k+1$ is thus defeated as soon as parameter k drops above μ_{max} [19]. Then, we study the competitiveness of deterministic strategies on chordal graphs [18]. In particular, we devise a strategy with ratio $\frac{2k+\omega_{\mathsf{opt}}}{\omega_{\mathsf{opt}}}$ for equal-weight chordal graphs.

3.3 Global competitive analysis

First, we focus on the competitive ratio of randomized strategies for general graphs. We prove that a randomized memoryless strategy cannot be more competitive than a deterministic strategy. Second, we show that the bound $|E_*| + 1$ for the CTP (Table 3.1) cannot be attained. Eventually, we evaluate the distance competitive ratio of both deterministic and randomized strategies for the k-CTP with multiple travellers.

3.3.1 Randomized memoryless strategies

We center on *memoryless strategies* (MS) [14]. More precisely speaking, we suppose that the traveller forgets the vertices he has already seen. In other words, a decision of an MS is independent of the vertices already visited. The term *memoryless* was used in the context of online algorithms (*e.g.* LIST UPDATE PROBLEM [2], PAGING PROBLEM [26]) which

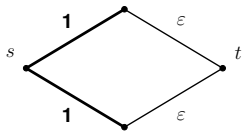
make decisions according to the current state, ignoring past events. An MS can be either deterministic or randomized. MSes are easy to be implemented as they do not need to memorize the edges already visited by the traveller. The only information they use is the graph $G\backslash E'_*$, which is the graph G deprived of the blocked edges discovered $E'_* \subseteq E_*$.

Definition 3.3 (Memoryless Strategies for the k-CTP)**.** *A deterministic strategy A is an MS if the next vertex w the traveller visits depends only on graph G deprived of blocked edges already discovered E'_* and the current traveller position v: $w = A(G\backslash E'_*, v)$. Similarly, a randomized strategy A is an MS if vertex w is the realization of a discrete random variable $X = A(G\backslash E'_*, v)$.*

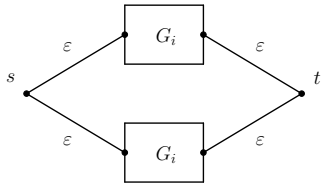
For example, the GREEDY strategy is a deterministic MS as it never makes decisions in function of the previous trips. In contrast, the REPOSITION strategy is not an MS as the past moves of the traveller allow him to know whether he shall go towards s or t. The polynomial-time strategies proposed in the literature do not use much memory information in the decision-making process. Either they are memoryless or they use a small amount of memory. As an illustration, REPOSITION can be implemented with one bit memory as the only information to retain is whether the traveller tries to reach t or returns to s.
The following process allows us to identify whether a deterministic strategy A is an MS. Let us suppose that a traveller T_1 follows strategy A: he has already visited certain vertices of the graph, he is currently at vertex v but he has not reached target t yet. Let us imagine a second traveller T_2 who is airdropped on vertex v of graph $G\backslash E'_*$ and is guided by strategy A. If the traveller T_2 always follows the same path as T_1 until reaching t, A is a deterministic MS. If T_1 and T_2 may follow different paths, then A is not an MS. Formally, proving that a strategy is a MS consists in finding the function which transforms the pair $(G\backslash E'_*, v)$ into vertex $w = A(G\backslash E'_*, v)$.
Let us define the competitive ratio c_{MS} of MSes as the minimum over competitive ratios of any MSes: $c_{\text{MS}} = \min_{A \text{ MS}} c_A$. Our goal is to prove that randomized memoryless strategies are not more competitive asymptotically than REPOSITION: $c_{\text{MS}} \geq 2k + O(1)$. To do this, we compute a lower bound $c_k = 2k + O(1)$ on the competitive ratio of any randomized memoryless strategy for a certain set $\mathcal{R}_k$ of road maps. For any value $k \geq 1$, set $\mathcal{R}_k$ is called a *road atlas*. Road maps in atlas $\mathcal{R}_k$ contain exactly k blocked edges. The idea is to build a road atlas $\mathcal{R}_k$ so that the randomized MSes cannot perform well on it. We know that if a randomized MS admits a ratio $2k+O(1)$ on set $\mathcal{R}_k$, then its competitive ratio for general graphs is necessarily larger, so $c_{\text{MS}} \geq c_k = 2k + O(1)$. Before specifying *road atlases* $\mathcal{R}_k$, we need to introduce the concepts used in their definition.

Road atlases $\mathcal{R}_k$

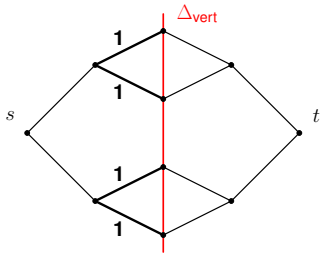
We define recursively a sequence of graphs G_i for $i \geq 1$ with weights from $\{1, \varepsilon\}$, $0 < \varepsilon \ll 1$. Graphs G_1 and G_{i+1} are represented in Figures 3.4a and 3.4b, graphs G_2 and G_3 are shown in Figures 3.4c and 3.4d. Edges with weight 1 are thicker than edges with weight ε (weights ε are omitted in Figures 3.4c and 3.4d). For any graph G_i, axis Δ_{vert} is its vertical axis of symmetry (Figures 3.4c and 3.4d).
We focus on road maps (G_i, E_*) composed of graph G_i but also at most i blocked edges which are on the right side of axis Δ_{vert}. Indeed, blocking edges on the left side of Δ_{vert} in G_i would affect negligibly the total distance traversed by a traveller. Let us suppose that a traveller traverses graph G_i and has already discovered some blocked edges $E'_* \subseteq E_*$. Then, he considers graph $G_i\backslash E'_*$ and tries to reach t, being unaware of the identity of the undiscovered blocked edges.
We denote by $\mathcal{G}$ the set of all the subgraphs of G_i, *i.e.* graphs $G_i\backslash E'_*$ with at most i edges in E'_* on the right side of Δ_{vert}, for any $i \geq 1$. We call them *diamond graphs* because of their appearance, diamonds joined together. Formally, we write $\mathcal{G} = \bigcup_{i=1}^{+\infty} \{G_i\backslash E'_* : |E'_*| \leq i\}$.



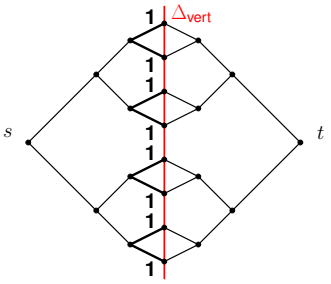
(a) Graph G_1



(b) Graph G_{i+1}



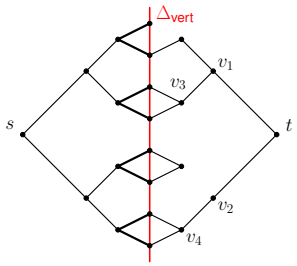
(c) Graph G_2 and axis Δ_{vert}



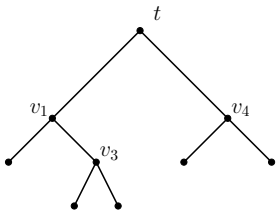
(d) Graph G_3 and axis Δ_{vert}

Figure 3.4: Recursive construction of graphs G_i

For any graph $G \in \mathcal{G}$, we partition its edges, denoted by E_G, into two sets $E_{G,\text{left}}$ (on the left side of axis Δ_{vert}) and $E_{G,\text{right}}$ (on the right side of axis Δ_{vert}).
To any diamond graph G of $\mathcal{G}$, we associate a *diamond binary tree* (DBT), denoted by T_G. Tree T_G, rooted in t, is obtained from the right half of graph G (on the right side of axis Δ_{vert}) by successive contractions of edges: any vertex with a single son is merged with its father (in Figure 3.5a: edge (t, v_2) is contracted, v_2 merges with t). We denote by $T_\varnothing$ the empty tree. Any nonempty tree is a triplet (v, T_a, T_b) with a root $v \in V$ and trees T_a and T_b. Figures 3.5a and 3.5b illustrate the construction of the DBT.



(a) Graph $G \in \mathcal{G}$, subgraph of G_3.



(b) The DBT T_G

Figure 3.5: An example of graph $G \in \mathcal{G}$ and its DBT T_G.

To put the definition of DBTs, let $L(v)$ denote the set of sons of vertex v which is only defined for the vertices on the right side of Δ_{vert}. For all $v \in \Delta_{\text{vert}}$, we have $L(v) = \emptyset$. For graph G of Figure 3.5a, $L(t) = \{v_1, v_2\}$, $L(v_2) = \{v_4\}$, for example.
Function BIN-TREE gives the construction of tree T_G, which is BIN-TREE (t):

$$\text{BIN-TREE}(v) = \begin{cases} T_\varnothing \text{ if } L(v) = \emptyset, \\ \text{BIN-TREE}(v_{\text{next}}) \text{ if } L(v) = \{v_{\text{next}}\}, \\ (v, \text{BIN-TREE}(v_{\text{up}}), \text{BIN-TREE}(v_{\text{down}})) \text{ if } L(v) = \{v_{\text{up}}, v_{\text{down}}\}. \end{cases}$$

We say that the depth of a vertex v in a DBT T, denoted by $d(v)$, is equal to the number of edges separating it from the root. We denote by $d_{\text{min}}(T)$ the minimum depth of all T leaves. For example, for DBT T_G in Figure 3.5b, $d_{\text{min}}(T_G) = 2$.

The depth of an edge (u, v), $D(u, v)$, is defined as $D(u, v) = \max\{d(u), d(v)\}$. We say that edge e' is the *mother* of edge e if these two edges share one endpoint and $D(e') = D(e)-1$, putting it shortly $e' = P(e)$. Conversely, we say e is the *daughter* of $P(e)$. Edge e^* is the *aunt* of edge e if e^* and $P(e)$ share one endpoint and $D(P(e)) = D(e^*)$. We indicate this fact as $e^* = U(e)$. We observe that the aunt of e and its mother share the same ancestor. For example, in Figure 3.5b, edge (t, v_4) is the aunt of (v_1, v_3).
Now we define the graphs contained in the road maps of atlas $\mathcal{R}_k$.

Definition 3.4 (Sets $\mathcal{D}_k$)**.** *Infinite set $\mathcal{D}_k$ contains graphs of $\mathcal{G}$ such that their DBT T_G fulfils $d_{min}(T_G) \geq k$: $\mathcal{D}_k = \{G \in \mathcal{G} : d_{min}(T_G) \geq k\}$.*

In other words, if graph G belongs to $\mathcal{D}_k$, then its DBT T_G induced on vertices of depth less than k forms a complete binary tree. For example, the DBT T_G in Figure 3.5b contains a complete binary tree of depth 2, so $G \in \mathcal{D}_2$. Finally, we define road atlases $\mathcal{R}_k$.

Definition 3.5 (Road atlas $\mathcal{R}_k$)**.** *Road atlas $\mathcal{R}_k$ is composed of road maps (G, E_*), where:*

- *Graph G belongs to $\mathcal{D}_k$: $G \in \mathcal{D}_k$,*
- *Set E_* becomes $\{\widehat{e}, U(\widehat{e}), U^2(\widehat{e}), \ldots, U^{k-1}(\widehat{e})\}$ in the DBT T_G, with $D(\widehat{e}) = k$.*

(a) Graph $G \in \mathcal{D}_2$ and its blocked edges. (b) The DBT T_G

Figure 3.6: An example of road map $(G, E_*) \in \mathcal{R}_2$, edges of E_* are dashed and blue.

In Figure 3.6a, we give an example of road map (G, E_*) in $\mathcal{R}_2$, where G is the graph initially drawn in Figure 3.5a and edges of E_* are dashed and in blue. In Figure 3.6b, we provide the corresponding DBT to see that the road map fulfils Definition 3.5 for $k = 2$.
For road map $(G, E_*) \in \mathcal{R}_k$, set $E_* \subsetneq E_{G,\text{right}}$ contains k edges and there is no two of them with the same depth in T_G. Moreover, there is a unique vertex $v_{k,j}$ among all vertices of depth k such that there is an open (s, t)-path containing $v_{k,j}$ in $G \backslash E_*$. In brief, any traveller on road map $(G, E_*) \in \mathcal{R}_k$ must traverse this vertex in order to reach t directly.

Competitiveness of randomized memoryless strategies

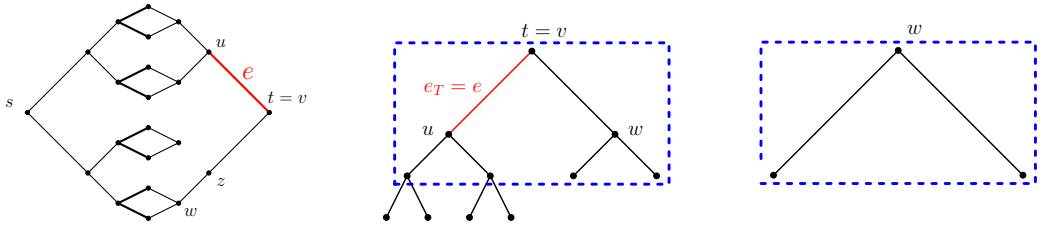
We study the competitiveness of randomized MSes for road atlases $\mathcal{R}_k$. The MS performance is determined by properties of the corresponding DBTs T_G. These properties result from relations which exist between DBT edges.
The following theorem states that cutting one edge from $G \in \mathcal{D}_k$ produces $G \backslash \{e\} \in \mathcal{D}_{k-1}$.

Theorem 3.1. *For any $G \in \mathcal{D}_k$ and edge $e \in E_{G,\text{right}}$, graph $G \backslash \{e\} \in \mathcal{D}_{k-1}$.*

Proof. Let $G \in \mathcal{D}_k$ and e be an edge in $E_{G,\text{right}}$. There is an edge e_T in T_G for which $T_{G\backslash\{e\}}$ is obtained by removing e_T and its descendants from T_G and next applying the edge contraction, if necessary. For example, if $e = (t, v_2)$ in Figure 3.5b, then $e_T = (t, v_4)$. Let v be the "shallower" endpoint of edge $e_T = \{u, v\}$, *i.e.* $d(v) < d(u)$. Edge e_T and its mother have this vertex in common, $v \in P(e_T)$. We distinguish two cases:

- **The depth of vertex u is greater or equal to $d_{\min}(T_G)$:** If u is the unique leaf of depth $d_{\min}(T_G)$, the depth of leaves of the DBT $T_{G\setminus\{e\}}$ is $d_{\min}(T_G) - 1 \geq k-1$. Otherwise, in DBT $T_{G\setminus\{e\}}$, the depth of leaves is still equal to $d_{\min}(T_G) \geq k$. In both cases, $G\setminus\{e\} \in \mathcal{D}_{k-1}$.
- **The depth of vertex u is strictly inferior to $d_{\min}(T_G)$:** Let T_v be the subtree of T_G with root v. We denote by w the brother of vertex u, *i.e.* the other son of vertex v (Figure 3.7b). When edge e is removed from G, edge e_T and its descendants are withdrawn in the DBT (in the DBT in Figure 3.7b, edge e has not been contracted, so $e = e_T$). Consequently, after the contraction, T_v becomes T_w, the subtree rooted in w. All the leaves of T_w have initially a depth greater than k, so by removing e from G, all the leaves of T_w have a depth greater than $k-1$. All leaves outside T_v, preserve their depth which is greater than k. Therefore, the depth of all leaves of $T_{G\setminus\{e\}}$ is greater than $k-1$.

After examining these two cases, we conclude that $G\setminus\{e\}$ belongs to $\mathcal{D}_{k-1}$. □



(a) Example of graph $G \in \mathcal{D}_2$. Edge e (red) is about to be removed from this graph.

(b) The DBT T_G. Blue frame covers vertices of depth 0 to 2. Two leaves are at depth 2.

(c) The DBT $T_{G\setminus\{e\}}$. Vertices of depth 0 to 1 are framed. Graph $G\setminus\{e\}$ belongs to $\mathcal{D}_1$.

Figure 3.7: Illustration of the proof of Theorem 3.1 on a subgraph of G_3.

Corollary 3.1. *For any road map $(G, E_*) \in \mathcal{R}_k$ and edge $e \in E_*$, we have:*

$$(G\setminus\{e\}, E_*\setminus\{e\}) \in \mathcal{R}_{k-1}.$$

Proof. Let $e = (u, v)$ and v be the shallowest endpoint of e. As $e \in E_*$, its depth is less than k. We know that $E_* = \{\widehat{e}, U(\widehat{e}), \ldots, U^{k-1}(\widehat{e})\}$ with $D(\widehat{e}) = k$. We denote edge e by $U^j(\widehat{e})$ with $0 \leq j \leq k-1$. As a consequence, the depth of edge e is $k-j$: $D(e) = k-j$.
As $G \in \mathcal{D}_k$, any edge of depth less than $k-1$ has two daughters. In graph $G\setminus\{e\}$, edge $P(e)$ has only one daughter as edge e disappeared. Consequently, vertices v and the sibling vertex of u are merged in the DBT of graph $G\setminus\{e\}$.
Now we prove that $U^{j+1}(\widehat{e})$ becomes the aunt of $U^{j-1}(\widehat{e})$ in the DBT $T_{G\setminus\{e\}}$, *i.e.* after the removal of $e = U^j(\widehat{e})$. Indeed, the daughters of the sibling edge of e in T_G are now the daughters of $P(e)$ in $T_{G\setminus\{e\}}$. So, edge $U^{j+1}(\widehat{e})$ which used to be the aunt of e is the aunt of $U^{j-1}(\widehat{e})$ in $T_{G\setminus\{e\}}$. Therefore, set $E_*\setminus\{e\}$ can be written $\{\widehat{e}, U(\widehat{e}), \ldots, U^{k-2}(\widehat{e})\}$ in $G\setminus\{e\}$. Thanks to Theorem 3.1, we have $G\setminus\{e\} \in \mathcal{D}_{k-1}$ which terminates the proof. □

We denote by c_k the competitive ratio of the best memoryless strategy for road atlases $\mathcal{R}_k$, $k \geq 1$. Formally:

$$c_k = \min_{A \in \text{ MS}} c_{A,\mathcal{R}_k}.$$

Our objective is to show that $c_k = 2k + O(1)$. As value c_k gives the competitiveness of MSes over a specific set of instances, it is a lower bound of c_{MS}. If our objective is achieved, then we are sure that randomized MSes are not asymptotically more competitive than deterministic strategies.

Theorem 3.2. *Any randomized MS competitive ratio is at least* $2k + O(1)$ *over atlas* $\mathcal{R}_k$.

Proof. We prove by induction that $c_k = 2k + 1 - \psi(k-1)$, where $\psi(k-1) = \sum_{j=0}^{k-1} \frac{c_j + 1}{2^{j+1}}$ is a convergent series bounded by a constant. Let A be the best MS over all road atlases $\mathcal{R}_k$, $k \geq 1$. We show that it achieves the same competitive ratio for a given k over any road map $(G, E_*) \in \mathcal{R}_k$: $c_A(G, E_*) = c_k$.

If $k = 0$, there is no blocked edge. The best MS for $k = 0$ consists in traversing an (s,t)-path of cost 1. So, $c_0 = 1$ and this competitive ratio is achieved for any road map in $\mathcal{R}_0$.

We assume that the induction hypothesis holds for index $k-1$. Let (G, E_*) be a road map of $\mathcal{R}_k$. As $G \in \mathcal{D}_k$, all leaves of T_G are at least at depth k. So, T_G is complete up to depth k and has 2^k vertices of depth k. We suppose that the traveller, guided by the most competitive MS A over atlas $\mathcal{R}_k$, is standing at source s and starts his walk on a road map $(G, E_*) \in \mathcal{R}_k$. We focus on value $c_A(G, E_*)$.

As strategy A is the most competitive, the traveller using it either reaches t directly with distance 1 or meets a blocked edge and thus traverses a total distance less than $2 + c_{k-1}$: distance 1 to reach the blockage, distance 1 to go back to a vertex on the left-hand side of Δ_{vert}, and at most distance c_{k-1} to reach t on the new road map which belongs to $\mathcal{R}_{k-1}$ (Corollary 3.1).

Indeed, we remember that when the traveller meets a blockage e^* for the first time, the only information taken into account by the MS A after this moment is the position of the traveller and the current graph $G \backslash \{e^*\}$. This fact justifies the use of the inductive term c_{k-1}, as strategy A is not influenced by the past and guides the traveller independently of its previous trips. The traveller, who necessarily returns to s after being blocked in an instance from $\mathcal{R}_k$, faces now an instance of $\mathcal{R}_{k-1}$.

For any $1 \leq j \leq 2^k$, let $p^A_{k,j}$ signify the probability that the traveller visits the j^{th} vertex at depth k, denoted by $v_{k,j}$ (index j passes from left to right in the DBT representation).

We denote by j^* the index of the only vertex v_{k,j^*} such that there is an open (s,t)-path containing it. Obviously, the traveller does not know the identity of vertex v_{k,j^*} as he is unaware of E_*. If he chooses luckily to walk on a simple (s,t)-path containing v_{k,j^*}, then he reaches t with distance 1. Otherwise, if he chooses an (s,t)-path traversing vertex $v_{k,j}$ with $j \neq j^*$, he meets a certain blocked edge $e_{k,j}$. We have:

$$c_A(G, E_*) = p^A_{k,j^*} + \sum_{j \neq j^*} p^A_{k,j} \left(2 + c_A(G \backslash \{e_{k,j}\}, E_* \backslash \{e_{k,j}\})\right). \tag{3.1}$$

According to Yao's principle [78], probabilities $p^A_{k,j}$ necessarily follow the uniform distribution and are all equal to $\frac{1}{2^k}$. From the traveller point of view, all vertices $v_{k,j}$ are indistinguishable. A strategy with a non-uniform distribution necessarily puts some vertices $v_{k,j}$ at a disadvantage, with $p^A_{k,j} < \frac{1}{2^k}$. Moreover, strategy A has to be competitive on any instance of $\mathcal{R}_k$. Applied on a road map of $\mathcal{R}_k$ where one of these penalized vertices is v_{k,j^*}, such a strategy makes the probability to reach t with distance 1 decrease and, therefore, the competitive ratio increases. Consequently, the best MS A fulfils $p^A_{k,j} = \frac{1}{2^k}$ for any vertex $v_{k,j}$.

According to the induction hypothesis, the best MS for road atlas $\mathcal{R}_{k-1}$ performs ratio c_{k-1} for any road map in $\mathcal{R}_{k-1}$. Thanks to this observation and the previous remark on the probability values $p_{k,j}$, we obtain from Eq. (3.1) that the competitive ratio of A is the same for all road maps of $\mathcal{R}_k$:

$$c_A(G, E_*) = \frac{1}{2^k} + \left(1 - \frac{1}{2^k}\right)(2 + c_{k-1}) = c_k.$$

We observe that $c_k - c_{k-1} = 2 - \frac{1}{2^k} - \frac{c_{k-1}}{2^k}$ and we obtain the following iterative formula thanks to the induction hypothesis:

$$c_k = 2 - \frac{1}{2^k} - \frac{c_{k-1}}{2^k} + 2(k-1) + 1 - \sum_{j=0}^{k-2} \frac{c_j + 1}{2^{j+1}} = 2k + 1 - \sum_{j=0}^{k-1} \frac{c_j + 1}{2^{j+1}}.$$

As $c_k \leq 2k + 1$, $\sum_{j=0}^{+\infty} \frac{c_j + 1}{2^{j+1}}$ converges and $c_k = 2k + O(1)$. For $k = 10^4$, the numerical computations give $\psi(10^4) = \sum_{j=0}^{10^4} \frac{c_j + 1}{2^{j+1}} = 3.213$, so value c_k is larger than $2k - 2.22$. □

In summary, $c_{\text{MS}} \geq c_k \geq 2k - 2.22$. As c_k represents the competitive ratio of the best competitive MS over road atlas $\mathcal{R}_k$, no MS can go below $2k + O(1)$ in terms of competitiveness. If we aim at designing a randomized strategy with competitive ratio $\beta k + O(1)$, $\beta < 2$, this result is important in our opinion. Indeed, it shows that such a randomized strategy is not memoryless. We shall focus on strategies which are not only randomized but use memory as well. Moreover, the more a strategy uses memory, the more difficult its competitive analysis is.

3.3.2 Absence of $(|E_*| + 1)$-competitive strategy

We expose our second contribution on the global competitiveness of randomized strategies [15]. We focus on the classic version CTP, without parameter k. An open question stated in Table 3.1 is whether a strategy reaches the competitive ratio $|E_*| + 1$ for the CTP. We answer it with the proof that no randomized strategy is $(|E_*| + 1)$-competitive on a certain graph G_α. As a consequence, the competitive ratio of the best randomized strategy is above value $|E_*| + 1$: the bound deduced from [75] is not tight.

Apex trees

This family of apex trees has already been mentioned in [42]. An apex tree is a graph composed of a tree rooted in t and vertex-disjoint paths that connect s to vertices of the tree. In fact, the over-mentioned graph G_α is an apex tree. As optimal strategies have been established for graphs with exclusively vertex-disjoint (s, t)-paths [11, 75], the question of the competitiveness of randomized strategies over apex trees, which represent a more general family of graphs, is of interest.

We work on a narrower family of graphs we define ourselves, called ε-apex trees (ε-ATs). Graph G_α, illustrated in Figure 3.8, is an ε-AT. An ε-AT is composed of a tree rooted in t whose all edges are of weight ε. Starting point s is connected to leaves of this tree with vertex-disjoint paths of arbitrary cost. We suppose that the traveller traverses an ε-AT G with blocked edges in E_* which all belong to the tree rooted in t (their weight is thus ε).

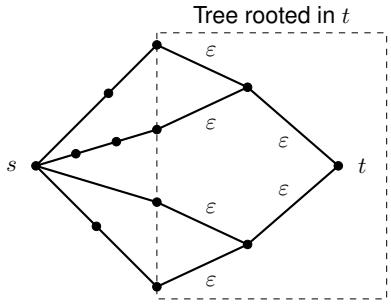


Figure 3.8: An example of ε-AT with four simple (s, t)-paths

Let $\mathcal{P}$ be the set of simple (s,t)-paths of G. There is a bijective relation between paths in $\mathcal{P}$ and the leaves of the tree: for any leaf of the tree, there is exactly one simple (s,t)-path passing through it.
We call the *memory* of the traveller the ordered set:

$$\mathcal{M} = \{(e_a^*, Q_a), (e_b^*, Q_b), \ldots, (e_z^*, Q_z)\},$$

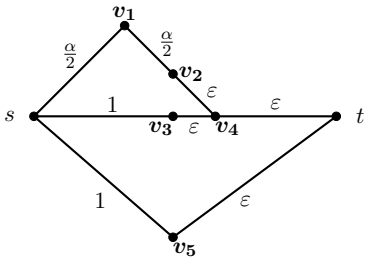
which indicates the blocked edges that the traveller discovered successively (e_a^* then e_b^*, etc.) and the simple (s,t)-path he was traversing at these moments (he was traversing Q_a when he discovered blockage e_a^*). The most competitive manner to traverse an ε-AT is to follow the randomized REPOSITION strategy guided by the distribution of the adequate discrete random variable X which, given the traveller memory $\mathcal{M}$, assigns a probability to remaining open paths in $\mathcal{P}$. In short:

1. Draw an open (s,t)-path $Q \in \mathcal{P}$ according to the distribution of X.
2. If the traveller discovers at vertex v of Q a blocked edge e^*, append pair (e^*, Q) to memory $\mathcal{M}$, go back to s on the shortest (v,s)-path and restart the process, otherwise terminate.

Indeed, the traveller has no alternative because of the structure of an ε-AT: each time the traveller meets a blockage, the only manner to reach t with minimum distance is to make a detour via vertex s. For this reason, the randomized REPOSITION strategy is the best for ε-ATs. Consequently, the optimal randomized strategy over ε-ATs is determined by the optimality of the distribution of random variable X. We prove next that no random variable X makes the randomized REPOSITION have competitive ratio $|E_*| + 1$ on G_α.

Farkas' lemma

We define the ε-AT G_α which depends on parameter α. It is an ε-AT composed of three simple (s,t)-paths, noted Q_a, Q_b, and Q_c (Figure 3.9).
We propose a road atlas $\mathcal{R}$ composed of maps with graph G_α and different sets of blocked edges. Our idea is to build an inequality system $B\mathbf{x} \leq \mathbf{d}$ such that it has a nonnegative solution iff there is a strategy which is $(|E_*| + 1)$-competitive over $\mathcal{R}$. We prove that this system has no nonnegative solution thanks to Farkas' lemma [46, 64].



(a) Graph G_α

Path	Sequence of vertices
Q_a	$s \to v_1 \to v_2 \to v_4 \to t$
Q_b	$s \to v_3 \to v_4 \to t$
Q_c	$s \to v_5 \to t$

(b) Paths Q_a, Q_b, and Q_c

Figure 3.9: Graph G_α and its three simple paths Q_a, Q_b, and Q_c

Proposition 3.1 (Farkas' lemma, Proposition 6.4.3 in [64])**.** *Let $B \in \mathbb{R}^{m \times l}$ be a matrix and $\mathbf{d} \in \mathbb{R}^m$ be a vector. The system $B\mathbf{x} \leq \mathbf{d}$ has a nonnegative solution iff every nonnegative vector $\mathbf{y} \in \mathbb{R}^m$ with $\mathbf{y}^T B \geq \mathbf{0^T}$ also satisfies $\mathbf{y}^T \mathbf{d} \geq 0$.*

We prove that ratio $|E_*| + 1$ cannot be attained. First, the road atlas $\mathcal{R}$ is defined. Second, we compute the system $B\mathbf{x} \leq \mathbf{d}$ such that it has no nonnegative solution iff bound $|E_*| + 1$ is not tight. After putting this system in its canonical form, we show it has no solution using Farkas' lemma.

Theorem 3.3. *There is no randomized strategy with competitive ratio $|E_*|+1$ on ε-ATs even with three simple (s,t)-paths.*

Proof. We focus on a road atlas $\mathcal{R}$ made for G_α composed of all feasible road maps, where only edges with weight $\varepsilon \ll 1$ can be blocked. First, we put two road maps into set $\mathcal{R}$, each one containing one blocked edge which is either (v_4,t) or (v_5,t).

$$\{(G_\alpha, E_*) : |E_*| = 1, E_* \subset \{((v_4,t),(v_5,t)\}\} \subset \mathcal{R},$$

Second, we put three road maps into $\mathcal{R}$, where two blocked edges are taken among (v_2,v_4), (v_3,v_4), and (v_5,t):

$$\{(G_\alpha, E_*) : |E_*| = 2, E_* \subset \{(v_2,v_4),(v_3,v_4),(v_5,t)\}\} \subset \mathcal{R}.$$

In the remainder of the proof, we neglect $\varepsilon \ll 1$ involved in calculations (weights in a CTP instance must be positive, this is why ε replaces zero). We make parameter α be in the interval $\left[\sqrt{2}, \frac{3}{2}\right[$.
Let A be a randomized REPOSITION strategy. We note p_a, p_b, and p_c the probabilities for the traveller to choose path Q_a, Q_b, and Q_c at the departure from s with strategy A, respectively. They obviously fulfil $p_a+p_b+p_c=1$. Let $p\left(Q_b|\left(v_2,v_4\right),Q_a\right)$ be the probability to select path Q_b after discovering blockage (v_2,v_4) on path Q_a. In other words, set $\{(v_2,v_4),Q_a\}$ is the memory of the traveller. We define similarly probabilities $p\left(Q_c|\left(v_2,v_4\right),Q_a\right)$, $p\left(Q_a|\left(v_3,v_4\right),Q_b\right)$, $p\left(Q_c|\left(v_3,v_4\right),Q_b\right)$, $p\left(Q_a|\left(v_5,t\right),Q_c\right)$, and $p(Q_b|(v_5$ $,t),Q_c)$. The sum of probabilities with the same condition is equal to 1, for example: $p\left(Q_b|\left(v_2,v_4\right),Q_a\right)+p\left(Q_c|\left(v_2,v_4\right),Q_a\right)=1$.
In Table 3.2, we define six variables $x_{.,.}$ resulting from the conditional probabilities presented above, arranged in a vector $\mathbf{x}_A=\left[x_{a,b}\ x_{a,c}\ x_{b,a}\ x_{b,c}\ x_{c,a}\ x_{c,b}\right]^T$.

Variable	Definition
$x_{a,b}$	$p\left(Q_b|\left(v_2,v_4\right),Q_a\right)p_a$
$x_{a,c}$	$p\left(Q_c|\left(v_2,v_4\right),Q_a\right)p_a$
$x_{b,a}$	$p\left(Q_a|\left(v_3,v_4\right),Q_b\right)p_b$
$x_{b,c}$	$p\left(Q_c|\left(v_3,v_4\right),Q_b\right)p_b$
$x_{c,a}$	$p\left(Q_a|\left(v_5,t\right),Q_c\right)p_c$
$x_{c,b}$	$p\left(Q_b|\left(v_5,t\right),Q_c\right)p_c$

Table 3.2: Definition of coordinates of $\mathbf{x}_A$

After supposing that the competitive ratio of strategy A is $|E_*|+1$, we produce the consequence of this assumption for each road map from $\mathcal{R}$. For road map $(G_\alpha,\{(v_2,v_4),(v_3,v_4)\})$, the optimal offline path is Q_a with cost α. If the traveller chooses Q_a (he does this with the probability $x_{a,b}+x_{a,c}$), he reaches t without discovering any blockage so the competitive ratio is 1. If he first chooses either path Q_b or Q_c and then Q_a (probability $x_{b,a}+x_{c,a}$), the competitive ratio is $\frac{2+\alpha}{\alpha}$. If the traveller traverses path Q_a after trying both Q_b and Q_c (probability $x_{b,c}+x_{c,b}$), the competitive ratio is $\frac{4+\alpha}{\alpha}$. Vector $\mathbf{x}_A$ thus fulfils:

$$(x_{a,b}+x_{a,c})+(x_{b,a}+x_{c,a})\frac{2+\alpha}{\alpha}+(x_{b,c}+x_{c,b})\frac{4+\alpha}{\alpha}\leq 3.$$

Similar linear inequalities can be written for all other road maps in $\mathcal{R}$ and vector $\mathbf{x}_A$ is a solution of an inequality system $B'\mathbf{x}\leq\mathbf{d}'$, with $\mathbf{x}\geq\mathbf{0}$:

$$\begin{bmatrix} 1 & 1 & \frac{2+\alpha}{\alpha} & \frac{4+\alpha}{\alpha} & \frac{2+\alpha}{\alpha} & \frac{4+\alpha}{\alpha} \\ 2\alpha+1 & 2\alpha+3 & 1 & 1 & 2\alpha+3 & 3 \\ 2\alpha+3 & 2\alpha+1 & 2\alpha+3 & 3 & 1 & 1 \\ \alpha+2 & \alpha+2 & 3 & 3 & 1 & 1 \\ \alpha & \alpha & 1 & 1 & 2+\alpha & 3 \end{bmatrix} \begin{bmatrix} x_{a,b} \\ x_{a,c} \\ x_{b,a} \\ x_{b,c} \\ x_{c,a} \\ x_{c,b} \end{bmatrix} \leq \begin{bmatrix} 3 \\ 3 \\ 3 \\ 2 \\ 2 \end{bmatrix}. \qquad (3.2)$$

Then, we write this system of inequalities in the canonical form and eliminate one redundant variable: we take $x_{c,b} = 1 - \sum_{i,j \neq c,b} x_{i,j}$. However, we must preserve the condition $x_{c,b} \geq 0$ which is equivalent to $\sum_{i,j \neq c,b} x_{i,j} \leq 1$. Finally, as strategy A is $(|E_*|+1)$-competitive on road atlas $\mathcal{R}$, vector $\mathbf{x}_A^c = [x_{a,b}\ x_{a,c}\ x_{b,a}\ x_{b,c}\ x_{c,a}]^T$ is a solution of the canonical system $B\mathbf{x} \leq \mathbf{d}$, $\mathbf{x} \geq \mathbf{0}$ (with $B \in \mathbb{R}^{6\times 5}$, $\mathbf{d} \in \mathbb{R}^6$):

$$\begin{bmatrix} -\frac{4}{\alpha} & -\frac{4}{\alpha} & -\frac{2}{\alpha} & 0 & -\frac{2}{\alpha} \\ 2(\alpha-1) & 2\alpha & -2 & -2 & 2\alpha \\ 2(\alpha+1) & 2\alpha & 2(\alpha+1) & 2 & 0 \\ \alpha+1 & \alpha+1 & 2 & 2 & 0 \\ \alpha-3 & \alpha-3 & -2 & -2 & \alpha-1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_{a,b} \\ x_{a,c} \\ x_{b,a} \\ x_{b,c} \\ x_{c,a} \end{bmatrix} \leq \begin{bmatrix} 2-\frac{4}{\alpha} \\ 0 \\ 2 \\ 1 \\ -1 \\ 1 \end{bmatrix}.$$

We define vector $\mathbf{y}$ such that $\mathbf{y}^T = \begin{bmatrix} \alpha(\alpha-1), & 0, & 0, & \alpha+1, & 2, & F(\alpha,\delta) \end{bmatrix}$, where $F(\alpha,\delta) = -2\alpha^2 + 5\alpha - 3 - \delta(\alpha-1)$. Polynomial $-2\alpha^2+5\alpha-3$ is positive for any $1 < \alpha < \frac{3}{2}$. We set $\delta > 0$ small enough to guarantee $F(\alpha,\delta) > 0$. Therefore, vector $\mathbf{y}$ is nonnegative. We check that $\mathbf{y}^T B \geq \mathbf{0^T}$. For this purpose, we note as $\mathbf{b}_1, \ldots, \mathbf{b}_5$ the column vectors of matrix B. We have, indeed

$$\begin{cases} \mathbf{y}^T\mathbf{b}_1 = -\frac{4}{\alpha}\alpha(\alpha-1) + (1+\alpha)^2 + 2(\alpha-3) + F(\alpha,\delta) = \alpha^2 - 2 + F(\alpha,\delta) \geq 0 \\ \mathbf{y}^T\mathbf{b}_2 = -\frac{4}{\alpha}\alpha(\alpha-1) + (1+\alpha)^2 + 2(\alpha-3) + F(\alpha,\delta) = \alpha^2 - 2 + F(\alpha,\delta) \geq 0 \\ \mathbf{y}^T\mathbf{b}_3 = -\frac{2}{\alpha}\alpha(\alpha-1) + 2\alpha(1+\alpha) - 4 + F(\alpha,\delta) = F(\alpha,\delta) \geq 0 \\ \mathbf{y}^T\mathbf{b}_4 = 2(1+\alpha) - 4 + F(\alpha,\delta) = 2(\alpha-1) + F(\alpha,\delta) \geq 0 \\ \mathbf{y}^T\mathbf{b}_5 = -\frac{2}{\alpha}\alpha(\alpha-1) + 2(\alpha-1) + F(\alpha,\delta) = F(\alpha,\delta) \geq 0 \end{cases}$$

Eventually, we obtain that $\mathbf{y}^T\mathbf{d} < 0$, as:

$$\begin{aligned} \mathbf{y}^T\mathbf{d} &= \alpha(\alpha-1)(2-\tfrac{4}{\alpha}) + 1 + \alpha - 2 + F(\alpha,\delta) \\ &= 2\alpha^2 - 6\alpha + 4 + 1 + \alpha - 2 - 2\alpha^2 + 5\alpha - 3 - \delta(\alpha-1) = -\delta(\alpha-1). \end{aligned}$$

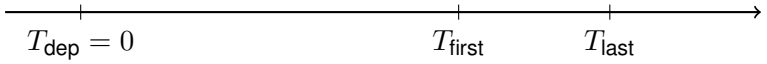
Farkas' lemma yields a contradiction: no vector $\mathbf{x}_A$ is a solution of our system $B\mathbf{x} \leq \mathbf{d}$. So, no randomized strategy is $(|E_*|+1)$-competitive. □

As no strategy reaches ratio $|E_*|+1$ for the road atlas $\mathcal{R}$ issued from graph G_α, we affirm that no randomized strategy is $(|E_*|+1)$-competitive for general graphs. The open question for the parameterized variant k-CTP in Table 3.1 remains however unanswered. In other words, we do not know whether there is (or not) a $(k+1)$-competitive strategy for general graphs. Indeed, our proof is not valid for the parameterized case as vector $\mathbf{d}'$ in Eq. (3.2) contains values $|E_*|+1$, not $k+1$ which would be all equal: $k+1=3$. The linear inequality with only a vector $\mathbf{d}'$ filled up with 3 has a solution.
All our results on the global competitiveness of randomized strategies for a single traveller have been introduced. In the next paragraph, we treat the case where there are several travellers ($L > 1$).

3.3.3 Distance competitive ratio for multiple travellers

We study the distance competitive ratio for the k-CTP with multiple travellers. For $L = 1$, the Canadian traveller can be seen as a vehicle traversing a map with blocked edges. The case $L \geq 2$ models a fleet of vehicles: our objective is to minimize the distance traversed by all vehicles. Therefore, we can consider ourselves as the manager of a transportation company who wants to limit the spendings on gasoline.
As no work has existed until now for the distance competitiveness, our objective is to give the bounds of competitiveness as tight as possible. Both deterministic and randomized strategies are treated. The travellers potentially communicate, so we compute these bounds for different communication levels.

Two scenarii can be distinguished when solving the k-CTP with multiple travellers, each one corresponding to a different practical application.
In the first one, a repair tool must absolutely be brought to a mechanical device on target t and therefore, several are sent at the same time.
In the second one, vehicles bring to t several pieces of a puzzle. In brief, the objective can be either to make one traveller reach t (at instant T_{first}) or all travellers reach t (at instant T_{last}). As expected, the ratio value depends upon the option chosen.
A group of travellers identified by indices from $\{1, \ldots, L\}$ starts its walk at source $s \in V$. The travellers' objective is to reach target $t \in V$ with a minimum cost (also called distance), which is the sum of the weights of edges traversed by each of them. On the timeline of an execution, three points are clearly marked: when the first traveller leaves s ($T_{\text{dep}} = 0$), when the first traveller reaches t (T_{first}), and when all travellers are standing on t (T_{last}).



The distance traversed by the travellers with strategy A on road map (G, E_*) at instant T_{first}, *i.e.* between T_{dep} and T_{first}, is denoted by $\omega_{A,\text{first}}(G, E_*)$. Naturally, value $\omega_{A,\text{last}}(G, E_*)$ denotes the distance traversed at instant T_{last}. Strategy A is $(c_{A,\text{first}}, c_{A,\text{last}})$-competitive if, for any road map (G, E_*) and some constant η:

$$\omega_{A,\text{first}}(G, E_*) \leq c_{A,\text{first}}\omega_{\text{min}}(G, E_*) + \eta \ \wedge \ \omega_{A,\text{last}}(G, E_*) \leq c_{A,\text{last}}\omega_{\text{min}}(G, E_*) + \eta.$$

On the one hand, the competitive ratio $c_{A,\text{first}}$ is used to compare the trip of the fleet, until one traveller arrives at destination, with the optimal trip of a single traveller. Therefore, the inequality $c_{A,\text{first}} < c_A$ means that multiple travellers perform better than a single one when the objective is to make only one traveller arrive at target t. On the other hand, the competitive ratio $c_{A,\text{last}}$ is used to compare the trip of the fleet until all travellers arrive at destination with the optimal trip of a single traveller.
We could also have defined value $c_{A,\text{last}}$ as the maximum ratio, over all road maps, of $\omega_{A,\text{first}}(G, E_*)$ with the cost $L\omega_{\text{min}}(G, E_*)$ of the optimal trip of the fleet, instead of the cost $\omega_{\text{min}}(G, E_*)$ of the optimal trip of a single traveller. It would allow us to compare the trip of a fleet following A with the trip of a fleet knowing set E_* in advance. However, we keep the version involving the denominator $\omega_{\text{min}}(G, E_*)$ for two reasons. First, the second version is directly obtained by dividing the version $c_{A,\text{last}}$ we selected by L. Second, the current version allows us to compare the competitive ratio of strategies, depending on the instant (T_{first} or T_{last}) which is considered as the end of the trip.

Communication levels

Our goal is to find the optimal competitive ratios for different communication levels and to measure their impact on the quality of strategies. We suppose that the communication time is negligible. We focus on four levels:

- Complete communication P_{complete} (initially noted P_1 in [79] but we rename it to improve readability of our text) authorizes each traveller to send and receive information to his teammates. They share all information about the blocked edges they discover, their location, etc.
- Partial communication P_{partial} (initially noted P_2 in [79]) uses an extra parameter $1 \leq L_1 \leq L$ because L_1 travellers can send and receive information, whereas $L - L_1$ can only receive. Case $L_1 = L$ corresponds to P_{complete}. We often obtain the same results with this communication level and with P_{complete}, except that variable L is replaced by L_1. To avoid repetitions, P_{partial} does not appear in the recapitulation charts (Table 3.3).

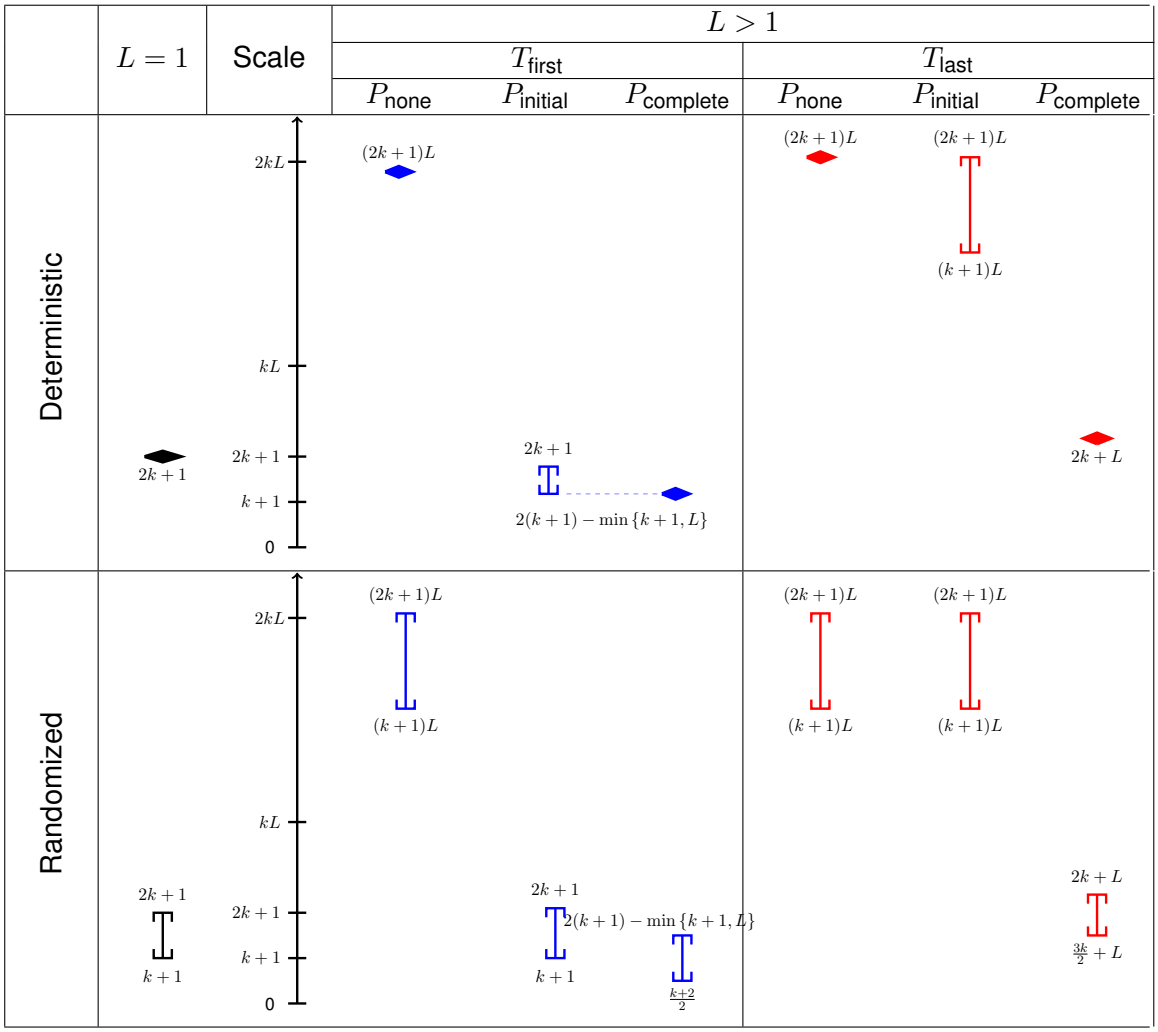


Table 3.3: Our results on the distance competitiveness for k-CTP with L travellers

- Initial communication P_{initial} is the first of the schemes we propose. The travellers are authorized to organize their journey before one of them leaves s but not to communicate after this moment. This level models drivers which communicate orally as long as they are parked in their base s. They cannot share information when they move because they do not have telecommunication equipment or the area is not covered by a cellular network.

- No communication P_{none} is the second one we add. The travellers do not communicate at all. We introduced this level to have a reference in our study of the impact of communication on the competitive ratio of the fleet.

Table 3.3 summarizes our results obtained at instants T_{first} and T_{last} for different communication levels. Case T_{first} is marked in blue, case T_{last} in red. Diamonds indicate that we obtained the optimal deterministic strategy and give its ratio. Intervals provide a lower bound such that a better ratio cannot be obtained and an upper bound which is the competitive ratio of the best strategy we have for now. For the deterministic case T_{first}, the dashed line was drawn to put in evidence that the lower bound for P_{initial} is equal to the exact value for P_{complete}.

Input: graph G, source s, target t, positive integers k, L
$i \leftarrow 1$ \\ counter of travellers
$E'_* \leftarrow \emptyset$ \\ set of blocked edges discovered
while *all travellers have not reached target* t **do**
  if *traveller* i *was informed of an open* (s,t)*-path* Q_{open} **then**
    he returns to s following the shortest path from his location and traverses Q_{open}
    $i \leftarrow (i \mod L) + 1$
  else
    if he is not on s, he goes to s following the shortest path from his location
    he computes the shortest (s,t)-path in $G \backslash E'_*$ and tries to traverse it
    if *a blocked edge* e *is on the path* **then**
      $E'_* \leftarrow E'_* \cup \{e\}$
    else
      he sends to other travellers the open (s,t)-path just traversed
    endif
    $i \leftarrow (i \mod L) + 1$
  endif
end

Algorithm 3: The MULTI-ALTERNATING strategy

Bounds of competitiveness: deterministic strategies

For all communication levels presented above, we provide either the optimal deterministic strategy together with its ratio or an interval containing the competitive ratio of the optimal strategy.

We start with complete communication P_{complete} and propose a deterministic strategy called MULTI-ALTERNATING (Algorithm 3). Only one traveller moves at a time. Traveller 1 computes the shortest (s,t)-path and traverses it. If he meets a blocked edge, he stops and hands over to traveller 2. Traveller 2 updates the graph by deleting the blocked edge discovered by traveller 1, computes the shortest (s,t)-path and so on. When traveller L finds a blocked edge, the algorithm continues by making the first traveller go back to s. Eventually, a traveller traverses an open (s,t)-path noted Q_{open} and alerts his teammates about his discovery.

The following theorem determines the competitive ratio of MULTI-ALTERNATING.

Theorem 3.4. *The competitive ratio of* MULTI-ALTERNATING *strategy* $c_{M\text{-}A}$ *is:*

$$2(k+1) - \min(k+1, L) \text{ for } T_{first},\ 2k + L \text{ for } T_{last}.$$

Proof. Traveller 1 traverses the shortest (s,t)-path of G, its cost is denoted ω_1. If he discovers a blocked edge e, he stays where he is and traveller 2 starts traversing the shortest (s,t)-path of $G \backslash \{e\}$, its cost is denoted ω_2, etc. At worst, the cost of the first open (s,t)-path traversed by a traveller is ω_{k+1}. Moreover, we obviously have $\omega_1 \leq \omega_2 \leq \ldots \leq \omega_{k+1}$.

- Case T_{first} and $k+1 \leq L$: before traveller $k+1$ reaches t, travellers with index in $\{1, \ldots, k\}$ tried to pass through a graph but have been stopped by a blocked edge they discovered. Travellers with index greater than $k+1$ are still at source s. Therefore, the total distance traversed is less than $\sum_{1 \leq i \leq k+1} \omega_i$ and the competitive ratio fulfils:

$$c_{\text{M-A,first}}(G, E_*) \leq \frac{1}{\omega_{k+1}} \left(\sum_{1 \leq i \leq k+1} \omega_i \right) \leq k+1.$$

- Case T_{first} and $k+1 > L$: MULTI-ALTERNATING makes all (s,t)-paths of cost ω_i, $1 \leq i \leq k+1$, be traversed once from s to t. At the time one traveller reaches t first, $L-1$ travellers are still standing on blocked paths (there are k blocked paths, as one path among $k+1$ is open), which have not been traversed in direction $t \to s$ yet:

$$c_{\text{M-A,first}}(G, E_*) \leq \frac{1}{\omega_{k+1}} \left(\sum_{1 \leq i \leq k+1} \omega_i + \sum_{1 \leq i \leq k-(L-1)} \omega_i \right) \leq 2(k+1) - L.$$

- Case T_{last} and $k+1 \leq L$: We know that when a traveller reaches t first, the total distance traversed is less than $\sum_{1 \leq i \leq k+1} \omega_i$. At this moment, the traveller arrived at t sends the open path he found to his k teammates stopped on a blocked path and those $L-(k+1)$ standing on s. So, the blocked travellers come back to s and all traverse Q_{open}. It is not surprising to obtain finally $2k+L$, as paths of cost $\omega_1, \ldots, \omega_k$ are traversed one in each direction and the open path Q_{open} of cost ω_{k+1} is traversed L times. We have:

$$c_{\text{M-A,last}}(G, E_*) \leq k+1 + \frac{1}{\omega_{k+1}} \left(\sum_{1 \leq i \leq k} \omega_i + k\omega_{k+1} \right) + L - (k+1) \leq 2k + L.$$

- Case T_{last} and $k+1 > L$: There are $L-1$ travellers stopped by blocked edges when the first traveller arrives at t. They all need to come back to s and traverse the open path discovered by their teammate. In summary, each (s,t)-path of cost ω_i with $i \leq k$ is traversed once in each direction and the (s,t)-path of cost ω_{k+1} is traversed by the L travellers in direction $s \to t$:

$$c_{\text{M-A,last}}(G, E_*) \leq \frac{1}{\omega_{k+1}} \left(\sum_{1 \leq i \leq k} 2\omega_i + L\omega_{k+1} \right) \leq 2k + L.$$

The competitiveness of MULTI-ALTERNATING under complete communication is summarized in Table 3.4. The competitive ratio for case T_{first} can be written $2(k+1) - \min(k+1, L)$ in order to omit the discontinuity between $k+1 \leq L$ and $k+1 > L$. □

	at instant T_{first}	at instant T_{last}
$k+1 \leq L$	$k+1$	$2k+L$
$k+1 > L$	$2(k+1)-L$	$2k+L$

Table 3.4: Competitive ratio of MULTI-ALTERNATING strategy for P_{complete}

We prove that on a certain road map, no strategy defeats MULTI-ALTERNATING. This road map instance is composed of graph G^W (Figure 3.2, page 70) and a set E_*^W of blocked edges containing k arbitrary edges of weight ε, so only one (s,t)-path of G^W is open. As a consequence, no deterministic strategy has a better competitive ratio than MULTI-ALTERNATING for general graphs.

Theorem 3.5. MULTI-ALTERNATING *is the best deterministic strategy for road map* $\left(G^W, E_*^W\right)$.

Proof. The simple (s,t)-paths of graph G^W are denoted $Q_1, \ldots, Q_{k+1}$. We need to consider three cases:

- Case T_{first} and $k+1 \leq L$: The travellers must try to traverse paths Q_i successively to reach t. For any deterministic strategy, the worst scenario occurs when all blocked paths have been traversed before finding out that Q_{k+1} is open. The distance traversed is at least $k+1$ and the optimal offline cost is $1+\varepsilon$. As $\varepsilon \to 0$, no deterministic strategy can drop below ratio $k+1$.

- Case T_{first} and $k+1 > L$: As above, the (s,t)-paths are traversed successively and the open path may come last. In this scenario, all (s,t)-paths are traversed once in direction $s \to t$, which results in distance $k+1$. As there are fewer travellers than paths, they sometimes must come back to s in order to visit the next (s,t)-path. When one traveller reaches t, at most $L-1$ blocked (s,t)-paths have not been traversed in direction $t \to s$ yet. Otherwise, this would mean that L travellers are stopped in front of a blocked edge, which is a contradiction because one traveller reached t. Furthermore, the open path is only traversed in one direction. Therefore, no deterministic strategy drops below the ratio $k+1+(k+1-L) = 2(k+1)-L$.

- Case T_{last}: The worst scenario occurs when the open path is traversed last. For any blocked path Q_i, at least one traveller traverses it, then he is forced to come back to s as he must arrive at t. Moreover, all travellers traverse necessarily path Q_{k+1}. Therefore, the distance traversed with any deterministic strategy is at least $2k+L$, which terminates the proof.

These lower bounds correspond to the competitive ratio of MULTI-ALTERNATING. □

The MULTI-ALTERNATING strategy is designed for P_{complete} communication. However, it can be adapted to P_{partial} communication. Indeed, the L_1 travellers who can receive and send messages follow the MULTI-ALTERNATING strategy as described in Algorithm 3. The $L-L_1$ travellers, who only receive messages, are waiting on s for information about the open path. Table 3.5 provides the competitive ratio of MULTI-ALTERNATING adapted to communication level P_{partial}.

	at instant T_{first}	at instant T_{last}
$k+1 \leq L_1$	$k+1$	$2k+L$
$k+1 > L_1$	$2(k+1)-L_1$	$2k+L$

Table 3.5: Competitive ratio of MULTI-ALTERNATING strategy for P_{partial}

When the competitive ratio is calculated for T_{last}, MULTI-ALTERNATING is not only the optimal deterministic strategy for level P_{complete} but also for P_{partial} with ratio $2k+L$. We also know that the competitive ratio of the optimal strategy for T_{first} is larger than $2(k+1)-\min(k+1,L)$ but cannot exceed $2(k+1)-\min(k+1,L_1)$.
We focus on the communication level P_{initial} where travellers are authorized to communicate before leaving s but not after their departure. Before starting their trip, the travellers decide together how to explore the map. They cannot inform their colleagues about the blocked edges discovered when being on the road towards t. We prove that under P_{initial} no deterministic strategy defeats the competitive ratio presented in Table 3.6.

Theorem 3.6. *No deterministic strategy for $P_{initial}$ has a competitive ratio smaller than:*

$$2(k+1)-\min(k+1,L) \text{ for } T_{first},\ L(k+1) \text{ for } T_{last}.$$

Proof. We show that this is the case on road map $\left(G^W, E_*^W\right)$. We treat three cases:

- Case T_{first}: As no strategy has a ratio less than $k+1$ for P_{complete} when $k+1 \leq L$, this is also true for P_{initial} which is more restrictive. No strategy has a ratio less than $2(k+1)-L$ for P_{complete} when $k+1 > L$, so this is also true for P_{initial}. The lack of communication in this case does not allow us to obtain a lower bound which is greater than the one for P_{complete}. In brief, the lower bound we establish for T_{first} follows from Theorem 3.5 and is $2(k+1) - \min(k+1, L)$.

- Case T_{last} and $k+1 \leq L$: The only degree of freedom offered by initial communication is on the spread of travellers over the (s,t)-paths at the beginning. This distribution is optimal when it is uniform, *i.e.* the same number of travellers is assigned to each path. Otherwise, we take the risk to lead more travellers to blocked (s,t)-paths. The best option on (G^W, E_*^W) is to make $\lfloor \frac{L}{k+1} \rfloor$ travellers traverse each path. The remaining travellers are arbitrarily sent on different paths of G^W. Then, each traveller cannot do better than obeying the optimal deterministic strategy REPOSITION [75], independently of his teammates: he traverses (s,t)-paths in the ascending order of path indices (and traversing Q_1 after being blocked on Q_{k+1}). Indeed, REPOSITION is the optimal strategy that each traveller can apply as no communication is allowed after the departure. If $\lfloor \frac{L}{k+1} \rfloor = \frac{L}{k+1}$, the mean distance traversed by the travellers is $\frac{1}{k+1}\left(\sum_{i=0}^{k} 2i\right) + 1 = k+1$, so the competitive ratio of the optimal strategy for $\left(G^W, E_*^W\right)$ is at least $L(k+1)$.

- Case T_{last} and $k+1 > L$: As previously, after starting their walk, travellers cannot communicate, so they follow the REPOSITION strategy. As there are more (s,t)-paths than travellers, only some of these paths have been assigned to travellers. The distribution of travellers on (s,t)-paths at departure must be as uniform as possible, otherwise the travellers visit the same (s,t)-paths and the distance traversed increases. For example, the paths assigned to travellers at the beginning could be $Q_1, Q_{\lfloor \frac{k+1}{L} \rfloor + 1}, Q_{2\lfloor \frac{k+1}{L} \rfloor + 1}$, etc. In this way, the number of travellers traversing a given path Q_i is minimum. Figure 3.10 illustrates how the distribution looks like when $\frac{k+1}{L}$ is an integer. At best, the mean distance traversed by the travellers is $k+1$. Thus, no strategy drops below ratio $L(k+1)$.

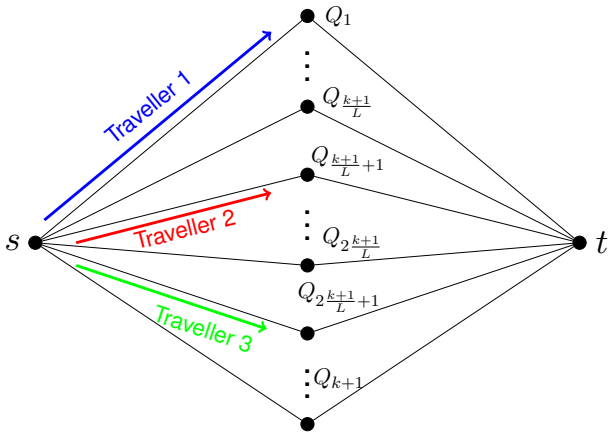


Figure 3.10: Dispatching of travellers at departure over the graph G^W when $L = 3$ and $\frac{k+1}{L} \in \mathbb{N}$

As no deterministic strategy can defeat these ratios on $\left(G^W, E_*^W\right)$, we obtain lower bounds of the competitiveness of deterministic strategies for communication level P_{initial}. □

The lower bounds established in Theorem 3.6 for P_{initial} are summarized in Table 3.6.

	at instant T_{first}	at instant T_{last}
$k+1 \leq L$	$k+1$	$L(k+1)$
$k+1 > L$	$2(k+1)-L$	$L(k+1)$

Table 3.6: Lower bound of the deterministic competitive ratio for P_{initial}.

Under communication level P_{none}, travellers cannot communicate with their teammates and they make decisions independently. The optimal competitive ratio for $L = 1$ is $2k+1$ with the REPOSITION strategy [75]. Consequently, for $L > 1$, the best strategy consists in making each traveller apply REPOSITION, so the optimal competitive ratio is $(2k+1)L$.
The impact of communication on the quality of strategies is significant. For example, when $L \geq k+1$, the competitive ratio of the best strategy under P_{complete} is $k+1$ at instant T_{first}, whereas it is $(2k+1)L$ without communication. Furthermore, at instant T_{last}, it becomes $2k+L$ under P_{complete}. This represents an improvement of the competitiveness by factor $2L$ and L, respectively. Then, it seems that initial communication improves the competitive ratio for some instances, especially on graphs with vertex-disjoint (s,t)-paths, as G^W. The question remains open when we consider all instances: does a strategy achieve a ratio less than $2k+1$ at instant T_{first} for P_{initial}?
Zhang *et al.* [79] put in evidence the interest of the map exploration with two travellers instead of one from the time point of view by proposing the $(k+1)$-competitive ALTERNATING strategy. Our appraisal from the distance point of view shows that the competitive ratio of the optimal strategy MULTI-ALTERNATING for complete communication tends to $k+1$ when L increases.

Bounds of competitiveness: randomized strategies

We focus our attention upon randomized strategies and give lower and upper bounds on their competitiveness in function of communication levels.
We first study competitive ratios for complete communication. We prove that no randomized strategy has a ratio less than $\frac{k+2}{2}$ at instant T_{first}, when $k+1 \leq L$.

Theorem 3.7. *No randomized strategy under $P_{complete}$ has a competitive ratio smaller than:*

	at instant T_{first}	at instant T_{last}
$k+1 \leq L$	$\frac{k+2}{2}$	$k+L$
$k+1 > L$	$k+2-L$	$k+L$

Table 3.7: Lower bounds of the competitive ratio of randomized strategies for P_{complete}

Proof. We provide these lower bounds for road map (G^W, E^W_*). Four cases are distinguished:

- Case T_{first} and $k+1 \leq L$: Travellers are obliged to traverse certain vertex-disjoint (s,t)-paths of G^W to reach t. As all these paths are indistinguishable, it is impossible to perform better than choosing a path uniformly. In other words, the optimal strategy consists in making traveller select a path with a uniform random draw (probability $\frac{1}{k+1}$ for all paths). Thus, the probability that target t is reached during the first try is $\frac{1}{k+1}$. If he is blocked, he stops, traveller 2 selects a path among the k remaining. The probability that traveller 2 reaches t during the second try is also $\frac{k}{k+1}\frac{1}{k} = \frac{1}{k+1}$, etc. The competitive ratio of this strategy is:

$$\lim_{\varepsilon \to 0} \sum_{i=1}^{k+1} \frac{1}{k+1}(i+\varepsilon) = \frac{k+2}{2}.$$

- Case T_{first} and $k+1 > L$: As previously, the best random draw to find the open path with minimum distance is uniform. Given that there are more (s,t)-paths than travellers, some of these paths are also traversed in direction $t \to s$ to allow travellers to come back to s and explore other paths. All (s,t)-paths traversed in direction $s \to t$ are also traversed in the opposite direction with exception to $L-1$ of them potentially on which travellers are standing when a traveller reaches t. In brief, the mean distance traversed is at best the same as the one for $L=1$ minus $L-1$ return trips. Thus, a lower bound of the competitive ratio is in this case:

$$(k+1) - (L-1) = k + 2 - L.$$

- Case T_{last} and $k+1 \leq L$: We know that the optimal mean distance traversed before a traveller reaches t is $\frac{k+2}{2}$. If the open path is found on the i^{th} attempt, this implies that $i-1$ blocked (s,t)-paths have been traversed in direction $s \to t$. These paths are necessarily traversed in direction $t \to s$, otherwise at least one traveller would be stopped on it and would never reach t. Moreover, all travellers traverse the open path. Finally, the optimal competitive ratio for this case cannot drop below the value:

$$\lim_{\varepsilon \to 0} \frac{k+2}{2} + \left(\sum_{i=1}^{k+1} \frac{1}{k+1}(i-1) \right) + (L-1)(1+\varepsilon) = k + L.$$

- Case T_{last} and $k+1 > L$: We know that the optimal mean distance traversed before a traveller reaches t is $k+2-L$. If the open path is found on the i^{th} attempt, then the $i-1$ blocked (s,t)-paths have been traversed in both directions $s \to t$ and $t \to s$, except at most $L-1$ of them. Moreover, the $L-1$ travellers who did not arrive at t have to traverse the open path. Finally, the optimal competitive ratio for this case is necessarily larger than:

$$\lim_{\varepsilon \to 0} k + 2 - L + L - 1 + (L-1)(1+\varepsilon) = k + L.$$

All those results are summarized in Table 3.7. □

The MULTI-ALTERNATING strategy produces an upper bound on the optimal competitive ratio of randomized strategies for P_{complete}. Indeed, any deterministic strategy, as MULTI-ALTERNATING, can be seen as a randomized one, designed only with probabilities 1 or 0. Therefore, one can say MULTI-ALTERNATING is a randomized strategy for level P_{complete} with competitive ratio $2(k+1) - \min(k+1, L)$ for T_{first} and $2k+L$ for T_{last}. This provides upper bounds for the level P_{complete}, as the best randomized strategy may be more competitive than MULTI-ALTERNATING.

We discuss the competitiveness of randomized strategies with multiple travellers for P_{none} and P_{initial}. Let us recapitulate the state of the art for a single traveller. We know that no randomized strategy does better than ratio $k+1$. However, no randomized strategy more competitive than REPOSITION has been proposed so far. In a nutshell, the competitive ratio of the best randomized strategy (which is still unknown) for one traveller is between $k+1$ and $2k+1$. Perhaps, no randomized strategy goes below ratio $2k+1$. In this case, REPOSITION would be the best randomized strategy as well because any deterministic strategy can be seen as a randomized one.

As for P_{none} travellers cannot communicate, they cannot decide which of them either start moving or stay waiting on s. Because all of them try to reach t, the competitive ratio of the best strategy lies between $(k+1)L$ and $(2k+1)L$. This interval is deduced from the competitive bounds for $L=1$ given above.

For P_{initial}, we give an upper bound on the competitive ratio at instant T_{first} with the following trivial strategy: we choose one traveller who moves while the others remain on s. The

moving traveller follows the REPOSITION strategy whose competitive ratio is $2k+1$. Furthermore, this ratio cannot be better than $\frac{k+2}{2}$. As with deterministic strategies, the initial communication makes algorithms perform better than when no communication is allowed. At instant T_{last}, the strategy which consists in making all travellers apply REPOSITION independently is $((2k+1)L)$-competitive. An interesting question is whether this strategy can be outperformed.

Summary

To summarize our work for multiple travellers, we remind our two main results which concern the communication level P_{complete}. With no restriction on the communication allowed, we devised an optimal deterministic strategy called MULTI-ALTERNATING. As REPOSITION for a single traveller, there is no hope of finding a deterministic strategy better than MULTI-ALTERNATING on general graphs. Then, we proposed a lower bound for randomized strategies. Under P_{complete}, no randomized strategy can drop below the competitive ratio $\frac{k+2}{2}$, which is smaller than the bound established for $L=1$. As for $L=1$ traveller, the main remaining open question is the identification optimal competitive ratio for randomized strategies, as we only pointed out an interval bounding this optimal ratio.

3.4 Local competitive analysis

We present our contributions on the local competitiveness of the k-CTP. Our objective was to identify families of graphs for which the local competitive ratio drops below ratio $2k+1$. Locally, we may have a chance to design deterministic strategies which offer better guarantees than REPOSITION. We put in evidence two of them.
First, we study the relationship between the competitive ratio of deterministic strategies and the size μ_{max} of the largest minimal (s,t)-cut [19] (Section 3.4.1). A deterministic strategy, DETOUR, with a competitive ratio $2\mu_{\text{max}}+\sqrt{2}(k-\mu_{\text{max}})+1$, is devised for graphs where $\mu_{\text{max}}<k$. When $\mu_{\text{max}}\geq k$, its competitive ratio is $2k+1$, as REPOSITION.
Second, we focus on chordal graphs [18] (Section 3.4.2). Without putting restrictions on weights, we prove that no deterministic strategy can defeat ratio $2k+1$. However, we propose a deterministic strategy, CHORD-WALK, with a competitive ratio $\frac{2k+\omega_{\text{opt}}}{\omega_{\text{opt}}}$ for equal-weight chordal graphs.

3.4.1 The k-CTP on graphs with small max-(s,t)-cut size

We establish a relationship between the size μ_{max} of the largest minimal (s,t)-cut of a graph G and the competitive ratio that can be obtained on G, for any configuration of blocked edges. Concretely, the competitive ratio of deterministic strategies on graphs, where $\mu_{\text{max}}<k$ is studied.
According to the proof of Lemma 2.1 in [75], for any value $\mu\in\mathbb{N}^*$, there is at least one graph, made up of vertex-disjoint (s,t)-paths only, such that $\mu_{\text{max}}=\mu$ and no deterministic strategy has a competitive ratio less than $2k+1$ on it if $\mu_{\text{max}}\geq k$. We focus on graphs fulfilling $\mu_{\text{max}}<k$: we assess the competitive ratio of the strategies REPOSITION and COMPARISON under this condition. We devise a more competitive strategy called DETOUR. In brief:

- for any value $\mu_{\text{max}}\geq 4$, we prove that there is at least one graph with $\mu_{\text{max}}<k$ for which both REPOSITION/COMPARISON strategies are $(2k+1)$-competitive,
- we propose a polynomial-time strategy DETOUR with competitive ratio $2\mu_{\text{max}}+\sqrt{2}(k-\mu_{\text{max}})+1$, when $\mu_{\text{max}}<k$. It outperforms the competitive ratio of the existing deter-

ministic strategies. Put differently, ratio $2k+1$ is defeated by a deterministic strategy on graphs G satisfying $\mu_{\text{max}} < k$.

The strategy DETOUR is also $(2k+1)$-competitive, when $\mu_{\text{max}} \geq k$.

Parameter μ_{max}

As said in Chapter 2, an (s,t)-cut X is minimal if none of its proper subsets $X' \subsetneq X$ is an (s,t)-cut. Let μ_{max} be the maximum cardinality of a minimal (s,t)-cut:

$$\mu_{\text{max}} = \max_{\substack{X \text{ minimal} \\ (s,t)-\text{cut}}} |X| \,. \tag{3.3}$$

Any (s,t)-cut X, where $|X| > \mu_{\text{max}}$, is not minimal. Parameter μ_{max} is smaller than the size of the commonly called *maximum* (s,t)*-cut* [52, 53], which is the largest set of edges separating two sets S and $V \backslash S$ in graph G, where $s \in S$ and $t \in V \backslash S$. Indeed, the maximum (s,t)-cut is not necessarily minimal.
We remind that, if X is a minimal (s,t)-cut, graph $G \backslash X$ contains exactly two connected components: one, denoted $R(X,s)$, contains all vertices reachable from s and another one, denoted $R(X,t)$, all vertices reachable from t. Largest minimal (s,t)-cuts $X_{\max}$, $|X_{\max}| = \mu_{\text{max}}$, are called max-$(s,t)$-cuts.

Competitive ratio of REPOSITION/COMPARISON when $\mu_{\text{max}} < k$

We study the family of graphs satisfying $\mu_{\text{max}} < k$. On such instances, we assess the competitiveness of the two best deterministic strategies known for now. Indeed, REPOSITION and COMPARISON [75, 77] are $(2k+1)$-competitive for general graphs. We prove that they do not benefit from the inequality $\mu_{\text{max}} < k$. We begin with the REPOSITION strategy.

Theorem 3.8. *For any $k > 4$, there is a road map $(G_k, E_{*,k})$, $\mu_{\text{max}} = 4$, such that the competitive ratio of* REPOSITION *on $(G_k, E_{*,k})$ is $2k+1$: $c_{\text{rep}}(G_k, E_{*,k}) = 2k+1$.*

Proof. The road map $(G_k, E_{*,k})$ is drawn in Figure 3.11. Graph G_k has a horizontal axis of symmetry Δ. On each side, there are $\lceil \frac{k}{2} \rceil$ *squares*, *i.e.* cycles of length 4, put in series. They are surrounded by two edges, one of weight 1 incident to s and one of weight $\varepsilon \ll 1$ incident to t. For any square above Δ, three of its edges are weighted with ε and the bottom left one is weighted with 3ε. All the top right edges are blocked (red edges in Figure 3.11). All squares below Δ are identical, except for the one closest to s (weights 2ε, ε, 4ε, and ε, see Figure 3.11). If k is even, as in Figure 3.11, the top edges on the right-hand side of all squares are blocked. If k is odd, there is no blockage on the square below Δ which is the closest to t. In this way, there are always k blocked edges in $E_{*,k}$ and the max-(s,t)-cut size of G_k is $\mu_{\text{max}} = 4$. Let $g(k) = 2\lceil \frac{k}{2} \rceil \in \{k, k+1\}$ be the total number of squares. The cost of the shortest (s,t)-path in G_k is $1 + (g(k)+1)\varepsilon$.

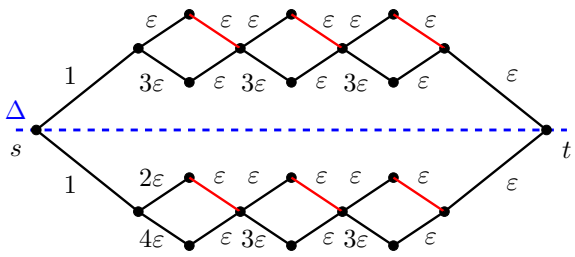


Figure 3.11: Graph G_6 and blocked edges $E_{*,6}$ in red

Guided by REPOSITION, the traveller traverses the shortest (s,t)-path which is above Δ and is blocked in the first square (distance $1+\varepsilon$). Set E'_* denotes the blocked edges discovered

during the execution: for now, $|E'_*| = 1$. The traveller comes back to s (distance $1+\varepsilon$). The shortest (s,t)-path in graph $G\backslash E'_*$ is now below axis Δ and its cost is $1+(g(k)+2)\varepsilon$ as it contains an edge of weight 2ε. The traveller traverses this path and is blocked in the first square below Δ (distance $1+2\varepsilon$). Then, the current shortest (s,t)-path in $G\backslash E_*$ is above Δ and its cost is $1+(g(k)+3)\varepsilon$, etc. In summary, the traveller is blocked k times traversing paths with cost larger than $1+\varepsilon$ in two directions. The total distance traversed d_{rep} satisfies $d_{\text{rep}} \geq 2k\,(1+\varepsilon) + \omega_{\text{opt}} \geq (2k+1)(1+\varepsilon)$. As $\omega_{\text{opt}} = 1+(2g(k)+1)\varepsilon$, the competitive ratio of REPOSITION c_{rep} is thus:

$$c_{\text{rep}} \geq (2k+1)\frac{1+\varepsilon}{1+(2g(k)+1)\varepsilon} \xrightarrow{\varepsilon\to 0} 2k+1.$$

With $\varepsilon \to 0$, there is always a road map on which REPOSITION achieves a ratio $2k+1-\delta$ for any arbitrarily small value $\delta > 0$. □

This result remains true for any value $\mu_{\text{max}} > 4$ as we can artificially add (s,t)-paths disjoint from G_k making μ_{max} increase. It suffices to assign a sufficiently large cost to these paths to make REPOSITION never guide the traveller to traverse them. Now, we examine the COMPARISON strategy.

Theorem 3.9. *For any $k > 3$, there is a road map $(G'_k, E'_{*,k})$, $\mu_{max} = 3$, such that the competitive ratio of* COMPARISON *on $(G'_k, E'_{*,k})$ is $2k+1$: $c_{\text{comp}}(G'_k, E'_{*,k}) = 2k+1$.*

Proof. Road map $(G'_k, E'_{*,k})$ is drawn in Figure 3.12. Axis Δ' is represented to facilitate the description of G'_k. Above Δ', $k-1$ squares are put in series and are surrounded as in G_k (see Theorem 3.8). On each square, the edge weights are ε, except for the bottom edges on the left-hand side weighted with value 1. The top left edges are blocked. Moreover, the edge incident to t above Δ' is also blocked, so $\left|E'_{*,k}\right| = k$. Below Δ', there is an open (s,t)-path with cost $1+2k\varepsilon$. The shortest (s,t)-path in G'_k, above Δ', costs $1+(2k-1)\varepsilon$. Graph G'_k is such that $\mu_{\text{max}} = 3$.

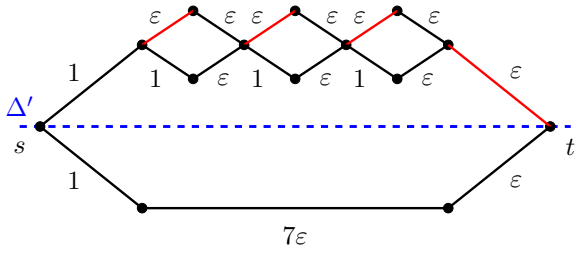


Figure 3.12: Graph G'_4 and blocked edges $E'_{*,4}$ in red

Guided by COMPARISON, the traveller traverses the shortest (s,t)-path and is blocked when he arrives on the first square (distance 1). Then, the cost of the shortest (s,t)-path in $G\backslash E'_*$, *i.e.* $1+2k\varepsilon$, is compared with the shortest distance between the current position of the traveller and t, *i.e.* $1+(2k-2)\varepsilon$. Since $1+(2k-2)\varepsilon < 1+2k\varepsilon$, the traveller chooses the shortest path between its current position and t, which is above Δ'. He meets a second blockage when arriving on the second square (distance $1+\varepsilon$). Then, he makes the same decision and traverses the squares above Δ'. Eventually, when he meets the last blockage incident to t, he travels back to s and finally passes through the optimal offline path, below Δ'. The total distance traversed is $d_{\text{comp}} = 2+2(k-1)(1+\varepsilon)+1+2k\varepsilon$. The competitive ratio c_{comp} of the COMPARISON strategy on the road map $(G'_k, E'_{*,k})$ follows:

$$c_{\text{comp}} = \frac{2+2(k-1)(1+\varepsilon)+1+2k\varepsilon}{1+2k\varepsilon} \xrightarrow{\varepsilon\to 0} 2k+1.$$

Making ε tend to zero terminates the proof. □

The existence of a deterministic strategy achieving a ratio less than $2k+1$ on graphs fulfilling $\mu_{\mathsf{max}} < k$ is still an open question after the results established in Theorems 3.8 and 3.9. Indeed, we showed that the existing strategies cannot defeat their global competitive ratio on this particular family of graphs. In the remainder, we devise a strategy outperforming REPOSITION and COMPARISON when $\mu_{\mathsf{max}} < k$.

Description of the DETOUR strategy

We introduce a parameterized strategy called α-DETOUR. It takes as input graph G, source s, target t, and a parameter α, $0 \leq \alpha \leq 1$. We provide an upper bound of its competitive ratio when $\mu_{\mathsf{max}} < k$. This bound is minimized for $\alpha = \frac{\sqrt{2}}{2}$ and is $2\mu_{\mathsf{max}} + \sqrt{2}(k - \mu_{\mathsf{max}}) + 1$. The strategy DETOUR corresponds to $\frac{\sqrt{2}}{2}$-DETOUR.

The α-DETOUR strategy is inspired by the COMPARISON strategy. In brief, the traveller traverses successively the shortest (s,t)-paths of the updated graph $G \backslash E'_*$, where E'_* is the set of discovered blocked edges. But when he backtracks towards source s, he verifies whether he can traverse a "short" detour from his position to t.

We present the α-DETOUR strategy in Algorithm 4. Variable pos keeps track of the traveller's current position. The idea is to perform successively two phases: an *exploration* followed by a *detour-backtracking*. The exploration starts when the traveller is on source s (line 10). He traverses the shortest (s,t)-path $P_{\mathsf{min}}^{(s,t)}$ called the *exploration path*. Its cost $\omega_{\mathsf{min}}^{(s,t)} = \omega_{\mathsf{min}}(G, E'_*)$ is stored in ω_{exp} (line 9). At this point, there are two possibilities:

1. The traveller reaches t and the execution terminates (line 15).
2. The traveller arrives at pos $= u$ and discovers a blocked edge $(u,v) \in P_{\mathsf{min}}^{(s,t)}$. Then, the detour-backtracking phase begins.

Each exploration followed by a detour-backtracking phase can be seen as a depth-first search (DFS). When the traveller is blocked on $P_{\mathsf{min}}^{(s,t)}$, we ask whether an *α-detour*, *i.e.* a (pos, t)-path with cost at most $\alpha\omega_{\mathsf{exp}}$, exists. If an α-detour exists, the traveller traverses the shortest path $P_{\mathsf{min}}^{(\mathsf{pos},t)}$ from the current position pos to target t (line 11). Obviously, its cost satisfies $\omega_{\mathsf{min}}^{(\mathsf{pos},t)} \leq \alpha\omega_{\mathsf{exp}}$. Otherwise, the traveller backtracks to the vertex before pos $= u$ on the exploration path (lines 16-18).

As in a DFS, we use a stack to store the previous vertices for backtracking. We denote by V_{stack} the set of vertices in the stack. We do not allow an α-detour $P_{\mathsf{min}}^{(\mathsf{pos},t)}$ to pass through any vertex $v \in V_{\mathsf{stack}}$, since the section $P_{\mathsf{min}}^{(v,t)}$ will be considered later on when pos $= v$. The vertices of an exploration path traversed by the traveller are naturally put into stack. Moreover, when the traveller is blocked on an α-detour $P_{\mathsf{min}}^{(\mathsf{pos},t)}$, the vertices of $P_{\mathsf{min}}^{(\mathsf{pos},t)}$ from pos to the endpoint of the blocked edge are put in stack. Finally, if the traveller backtracks to s, the algorithm goes back to the exploration phase. At this moment, the stack is empty. Variable G' contains the graph G deprived of the discovered blockages E'_* at any moment of the execution. At each iteration of the while loop, the variables are updated as follows: if the path $P_{\mathsf{min}}^{(u_0,t)}$ currently traversed (lines 10–11) does not contain any blockage, then the traveller reaches t, i.e. pos $\leftarrow t$. In this case, the algorithm terminates since the destination is reached. Otherwise, let $P_{\mathsf{min}}^{(u_0,t)} = u_0 \cdots u_i \cdot u_{i+1} \cdots u_r \cdot t$, where (u_i, u_{i+1}) is its first blocked edge. The traveller's position is updated from u_0 to u_i (line 12). Then, we update E'_* with the newly discovered blockages including (u_i, u_{i+1}), and $G' \leftarrow G \backslash E'_*$ (line 14). In addition, we push the traversed vertices $u_0, \ldots, u_{i-1}$ on the stack (except u_i) and update it accordingly $V_{\mathsf{stack}} \leftarrow V_{\mathsf{stack}} \cup \{u_0, \ldots, u_{i-1}\}$. In case there is no α-detour $P_{\mathsf{min}}^{(u_i,t)}$ in $G' \backslash V_{\mathsf{stack}}$, the algorithm backtracks by popping u_{i-1} from the stack and setting pos $\leftarrow u_{i-1}$ (lines 16–18).

If $\alpha = 0$, the algorithm does not take any detour. As a consequence, 0-DETOUR is equivalent to REPOSITION, as both procedures perform an exploration phase followed by backtracking

Input: graph G, source s, target t, parameter $\alpha \in (0,1)$
$E'_* \leftarrow \emptyset$; $G' \leftarrow G \backslash E'_*$;
pos $\leftarrow s$; $u_0 \leftarrow s$;
$\omega_{\text{exp}} \leftarrow \omega_{\text{min}}(G, \emptyset)$;
stack $\leftarrow$ Empty Stack; $V_{\text{stack}} \leftarrow \emptyset$;
while true **do**
  $u_0 \leftarrow$ pos;
  if $u_0 = s$ **then**
    $\omega_{\text{exp}} \leftarrow \omega_{\text{min}}(G, E'_*)$;
    traverse the shortest (s,t)-path $P_{\text{min}}^{(u_0,t)}$ in G';
  else
    traverse the shortest (u_0,t)-path $P_{\text{min}}^{(u_0,t)}$ in $G' \backslash V_{\text{stack}}$;
  endif
  update pos;
  push the vertices visited in $P_{\text{min}}^{(u_0,t)}$ except pos on stack;
  update E'_*, G', and V_{stack};
  if *pos* $= t$ **then break**;
  while *pos* $\neq s$ **and** *there is no* $P_{min}^{(pos,t)}$ *in* $G' \backslash V_{stack}$ *such that* $\omega_{min}^{(pos,t)} \leq \alpha\omega_{exp}$ **do**
    pos $\leftarrow$ pop(stack);
    $V_{\text{stack}} \leftarrow V_{\text{stack}} \backslash \{\text{pos}\}$;
  end
end

Algorithm 4: The α-DETOUR strategy

without taking any detour. In the following, we provide an upper bound of α-DETOUR's competitive ratio.

Competitive analysis of the DETOUR strategy

We denote by $P_1, \ldots, P_\ell$ the exploration paths $P_{\text{min}}^{(s,t)}$ such that the distance from s to the blocked edge discovered on it is greater than α multiplied by their own cost, *i.e.* $\alpha\omega_i$. In other words, the distance d_i traversed by the traveller on the exploration paths P_i, $1 \leq i \leq \ell$, satisfies $d_i \geq \alpha\omega_i$. Paths P_i are sorted in order to fulfil $\omega_1 \leq \cdots \leq \omega_\ell$. The exploration paths P_1,...,$P_{\ell-1}$ are blocked, while path P_ℓ can be open. If P_ℓ does not contain any blockage, then the algorithm terminates after traversing P_ℓ.
Let us split $P_1, \ldots, P_\ell$ into two sequences $S_1 = P_1, \ldots, P_{h-1}$ and $S_2 = P_h, \ldots, P_\ell$ such that $2\alpha\omega_{h-1} < \omega_\ell \leq 2\alpha\omega_h$. In the particular case where $\omega_\ell \leq 2\alpha\omega_1$, then $h = 1$ and the two sets are defined as $S_1 = \emptyset$ and $S_2 = P_1, \ldots, P_\ell$. We denote by $G[P_h, \ldots, P_\ell]$ the subgraph of G induced by paths $P_h, \ldots, P_\ell$, *i.e.* containing only the vertices and edges of paths P_i, $h \leq i \leq \ell$.

Theorem 3.10. *The max-(s,t)-cut size induced on graph $G[P_h, \ldots, P_\ell]$ is at least $\ell - h + 1$.*

Proof. We denote by b_i the blocked edge discovered on P_i, for $i \in \{h, \ldots, \ell\}$. We construct by induction a set $\{e_h, \ldots, e_\ell\}$ of edges satisfying the following induction hypotheses, for all $i \in \{h, \ldots, \ell\}$:

$H_1(i)$: $\{e_h, \ldots, e_i\}$ is a minimal (s,t)-cut of $G[P_h, \ldots, P_i]$,

$H_2(i)$: Either $e_i = b_i$ or e_i is an ancestor of b_i in P_i,

$H_3(i)$: For $j \in \{i+1, \ldots, \ell\}$, P_j does not pass through e_i.

Basis: For $i = h$, $G[P_h, \ldots, P_i]$ contains one path P_h only. We choose $e_h = b_h$, which fulfils $H_2(h)$. Since any edge of P_h is a max-(s,t)-cut of $G[P_h]$, it satisfies $H_1(h)$. Statement $H_3(h)$ is also true, as e_h is blocked.

Inductive step: Assume that $H_1(i)$, $H_2(i)$, and $H_3(i)$ are true, for $i \in \{h, \ldots, \ell - 1\}$. We construct e_{i+1} and prove the induction hypotheses $H_1(i+1)$ to $H_3(i+1)$. For simplicity, we denote sets $R(\{e_h, \ldots, e_i\}, s)$ and $R(\{e_h, \ldots, e_i\}, t)$ in graph $G[P_h, \ldots, P_i]$ by $R_i(s)$ and $R_i(t)$, respectively.

Let $P_{i+1}^{(v_0,v_p)} = v_0 \cdot v_1 \cdots v_p$ be the longest section in P_{i+1}, starting from $v_0 = s$, such that $v_0, \ldots, v_p \in R_i(s)$ and $p \in \mathbb{N}$. Section $P_{i+1}^{(v_0,v_p)}$ contains at least vertex $v_0 = s$. For $j \in \{h, \ldots, i\}$, all ancestors of e_j in P_j belong to $R_i(s)$, and all descendants belong to $R_i(t)$. Therefore, according to $H_2(i)$, all exploration paths' sections of the form $P_j^{(s,u)}$ are equal to the shortest open path from s to u, for $u \in R_i(s) \cap P_j$ and $j \in \{h, \ldots, i\}$. In particular, since $P_{i+1}^{(v_0,v_p)}$ is the shortest (v_0, v_p)-path, we deduce that it is open.

According to $H_3(i)$, $P_{i+1}^{(v_p,t)}$ is a new path connecting $R_i(s)$ to $R_i(t)$, which does not traverse any edge of the cut $\{e_h, \ldots, e_i\}$. Furthermore, no vertex in $P_{i+1}^{(v_{p+1},t)}$ belongs to $R_i(s)$. Indeed, suppose for the sake of contradiction that $u \in P_{i+1}^{(v_{p+1},t)}$ and $u \in R_i(s)$. There would exist $j \in \{h, \ldots, i\}$, such that $P_j^{(s,u)}$ is the shortest (s,u)-path, and all its vertices belong to $R_i(s)$. This contradicts with the fact that $P_{i+1}^{(s,u)}$ is also the shortest (s,u)-path and $v_{p+1} \notin R_i(s)$, by definition. Let $v_{p'}$ be the first vertex of P_{i+1} belonging to $R_i(t)$, *i.e.* $v_{p'} \in R_i(t)$ and $p < p'$. Such a vertex exists as t is a candidate. We derive that $P_{i+1}^{(v_p,v_{p'})}$ is the only path connecting $R_i(s)$ to $R_i(t)$. Figure 3.13 represents cut $\{e_h, \ldots, e_i\}$, path P_{i+1} and its vertices v_p and $v_{p'}$.

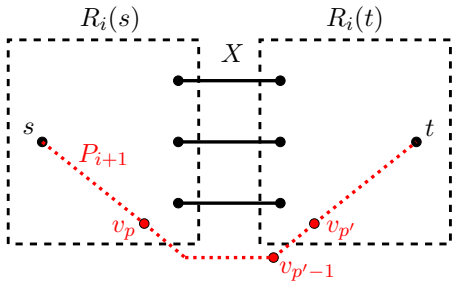


Figure 3.13: Cut $X = \{e_h, \ldots, e_i\}$, path P_{i+1}, and vertices v_p, $v_{p'-1}$, $v_{p'}$

We fix e_{i+1} differently, depending on the position of b_{i+1}. We already proved that $b_{i+1} \notin P_{i+1}^{(v_0,v_p)}$, the remaining cases are:

- If $b_{i+1} \in P_{i+1}^{(v_p,v_{p'})}$, then we set $e_{i+1} = b_{i+1}$. As $e_{i+1} \in E_*$, $H_3(i+1)$ is true.

- Otherwise, if $b_{i+1} \in P_{i+1}^{(v_{p'},t)}$, we choose $e_{i+1} = (v_{p'-1}, v_{p'})$. We prove that the cost of the current shortest $(s, v_{p'})$-path, $P_{i+1}^{(s,v_{p'})}$, is at least $\alpha\omega_h$. Indeed, as vertex $v_{p'}$ belongs to a certain path $P_{j'}$, $j' \in \{h, \ldots, i\}$, the cost of $P_{i+1}^{(s,v_{p'})}$ is at least the cost of $P_{j'}^{(s,v_{p'})}$. If we have $\omega_{i+1}^{(s,v_{p'})} \leq \alpha\omega_h$, the distance traversed by the traveller on $P_{j'}$ is less than $\alpha\omega_h \leq \alpha\omega_{j'}$, as $v_{p'} \in R_i(t)$. This contradicts the fact that $P_{j'} \in \{P_h, \ldots, P_\ell\}$. Moreover, after the $(i+1)$-th detour-backtracking phase, all remaining open $(v_{p'}, t)$-paths are longer than $\alpha\omega_{i+1} \geq \alpha\omega_h$. Therefore, the cost of any exploration (s,t)-path passing through $v_{p'}$ is greater than $2\alpha\omega_h$. This is impossible since the last exploration path P_ℓ satisfies $\omega_\ell \leq 2\alpha\omega_h$. As a consequence, statement $H_3(i+1)$ is true.

Both cases fulfil $H_2(i+1)$. It only remains to prove statement $H_1(i+1)$. We showed that $P_{i+1}^{(v_p,v_{p'})}$ is the only path connecting $R_i(s)$ to $R_i(t)$, and $e_{i+1} \in P_{i+1}^{(v_p,v_{p'})}$. Thus, $\{e_h, \ldots, e_{i+1}\}$ is an (s,t)-cut of $G[P_h, \ldots, P_{i+1}]$. If we reopen edge e_{i+1}, path $P_{i+1}^{(v_p,v_{p'})}$ connects $R_i(s)$ to $R_i(t)$. If we reopen e_j, $j < i+1$, there is a path in $G[P_h, \ldots, P_i]$ which connects $R_i(s)$ to $R_i(t)$ independently of $P_{i+1}^{(v_p,v_{p'})}$, according to the minimality of $\{e_h, \ldots, e_i\}$ in $H_1(i)$. As a consequence, no proper subset of $\{e_h, \ldots, e_{i+1}\}$ is an (s,t)-cut. Cut $\{e_h, \ldots, e_{i+1}\}$ is minimal.
In summary, we derive by induction that $\{e_h, \ldots, e_\ell\}$ is a minimal (s,t)-cut of $G[P_h, \ldots, P_\ell]$. The size of the max-(s,t)-cut is at least $\ell - h + 1$. □

The following lemma states that the max-(s,t)-cut size of graph $G[P_h, \ldots, P_\ell]$ cannot exceed the max-(s,t)-cut size of its super-graph G.

Lemma 3.1. *The max-(s,t)-cut size on graph $G[P_h, \ldots, P_\ell]$ is less than or equal to the max-(s,t)-cut size μ_{max} of the original graph G.*

Proof. Let X be one of the max-(s,t)-cuts in graph $G[P_h, \ldots, P_\ell]$. Cut X is minimal, so no subset $X' \subsetneq X$ is an (s,t)-cut. If X is an (s,t)-cut in G, then it is also minimal in G as none of its subsets can be an (s,t)-cut. Therefore, $|X| \leq \mu_{\text{max}}$.
Now, let us suppose that X is not an (s,t)-cut in G. We denote by Y the max-(s,t)-cut in graph G deprived of edges X, *i.e.* $G \backslash X$. Set $X \cup Y$ is thus a minimal (s,t)-cut in graph G as Y is minimal in $G \backslash X$. So, $|X| \leq |X \cup Y| \leq \mu_{\text{max}}$. In both cases, the max-(s,t)-cut size in $G[P_h, \ldots, P_\ell]$ is at most μ_{max}. □

According to Theorem 3.10 and Lemma 3.1, a relationship exists between values ℓ, h, and μ_{max}, which is $\ell - h + 1 \leq \mu_{\text{max}}$. After traversing an exploration path P_i, the traveller performs a detour-backtracking phase. The number of blockages discovered during this i-th detour-backtracking phase is denoted by q_i. We analyse the cost of traversing P_i and performing the i-th detour-backtracking phase in Lemma 3.2.

Lemma 3.2. *The total cost of both the i-th exploration phase and the i-th detour-backtracking phase is not greater than $(2 + 2\alpha q_i)\omega_i$.*

Proof. The filling of stack in Algorithm 4 ensures that each edge is traversed only twice: once when moving towards t on an exploration path or a detour, and once when backtracking. The exploration path costs ω_i and each detour costs no more than $\alpha\omega_i$. Besides, the number of detours is at most q_i. Hence, the total cost is at most $2\omega_i + q_i 2\alpha\omega_i$, which concludes the proof. □

We denote by k_1 the number of blocked edges discovered during the exploration and detour-backtracking phases associated with paths $P_1, \ldots, P_{h-1}$ (respectively $P_h, \ldots, P_\ell$). Similarly, the number of blocked edges discovered during the exploration and detour-backtracking phases associated with paths $P_h, \ldots, P_\ell$ is k_2. Let k_3 be the number of blockages discovered during the other phases, so that $k_1 + k_2 + k_3 = k$. We derive in Theorem 3.11 an upper-bound on the competitive ratio as a function of k_1, k_2, k_3, and α.

Theorem 3.11. *The competitive ratio of α-*DETOUR *is upper-bounded by*

$$\frac{k_1}{\alpha} + 2\mu_{max} + 2\alpha(k_2 + k_3 - \mu_{max}) + 1. \tag{3.4}$$

Proof. Since path P_ℓ is the shortest (s,t)-path in a certain graph $G \backslash E'_*$, where $E'_* \subseteq E_*$, the optimal offline cost satisfies

$$\omega_{\text{opt}} \geq \omega_\ell. \tag{3.5}$$

According to Lemma 3.2, the distance travelled during the exploration and detour-backtracking phases of $P_1, \ldots, P_{h-1}$ is not greater than

$$\sum_{j=1}^{h-1} (2 + 2\alpha q_j)\omega_j \leq 2\omega_{h-1} \sum_{j=1}^{h-1} (1 + q_j) = 2k_1\omega_{h-1}. \tag{3.6}$$

Inequality (3.6) comes from the fact that $\omega_1 \leq \cdots \leq \omega_{h-1}$ and $\sum_{j=1}^{h-1}(1+q_j) = k_1$.
We evaluate the cost of the phases associated with $P_h, \ldots, P_\ell$. Path P_ℓ is either open and traversed in one direction only (Case 1) or blocked and the traveller reaches t via a detour (Case 2).
Case 1: If P_ℓ does not contain any blockage, then the algorithm terminates after traversing it. This final exploration phase costs ω_ℓ. We have $q_\ell = 0$ and $k_2 = \sum_{j=h}^{\ell-1}(1+q_j)$. Given Lemma 3.2, the cost of phases from the h-th to the ℓ-th is less than:

$$\begin{aligned}
\sum_{j=h}^{\ell-1} (2 + 2\alpha q_j)\omega_j + \omega_\ell &= \sum_{j=h}^{\ell-1} (2\alpha + 2\alpha q_j)\omega_j + \sum_{j=h}^{\ell-1} (2 - 2\alpha)\omega_j + \omega_\ell \\
&\leq 2\alpha k_2\omega_\ell + (2 - 2\alpha)(\ell - h)\omega_\ell + \omega_\ell && (3.7)\\
&< 2\alpha k_2\omega_\ell + (2 - 2\alpha)\mu_{\max}\omega_\ell + \omega_\ell && (3.8)\\
&= 2\alpha(k_2 - \mu_{\max})\omega_\ell + 2\mu_{\max}\omega_\ell + \omega_\ell.
\end{aligned}$$

We deduce Inequality (3.7) from $\omega_h \leq \cdots \leq \omega_\ell$. By applying Theorem 3.10 and Lemma 3.1 on $S_2 = P_h, \ldots, P_\ell$, we conclude that $\ell - h \leq \mu_{\max} - 1 < \mu_{\max}$ in Inequality (3.8).
Case 2: Suppose that P_ℓ is blocked. The ℓ-th exploration and detour-backtracking phases cost at most $(2 + 2\alpha q_\ell)\omega_\ell + \alpha\omega_\ell$. Moreover, we have $k_2 = \sum_{j=h}^{\ell}(1+q_j)$. The distance traversed during the phases from the h-th to the ℓ-th is not greater than:

$$\begin{aligned}
\sum_{j=h}^{\ell-1}(2 + 2\alpha q_j)\omega_j + (2 + 2\alpha q_\ell + \alpha)\omega_\ell &= \sum_{j=h}^{\ell} (2 + 2\alpha q_j)\omega_j + \alpha\omega_\ell \\
&\leq 2\alpha k_2\omega_\ell + (2 - 2\alpha)(\ell - h + 1)\omega_\ell + \alpha\omega_\ell && (3.9)\\
&\leq 2\alpha k_2\omega_\ell + (2 - 2\alpha)\mu_{\max}\omega_\ell + \alpha\omega_\ell && (3.10)\\
&\leq 2\alpha(k_2 - \mu_{\max})\omega_\ell + 2\mu_{\max}\omega_\ell + \omega_\ell. && (3.11)
\end{aligned}$$

Inequality (3.9) follows from $\omega_h \leq \cdots \leq \omega_\ell$. We obtain Eq. (3.10) from $\ell - h + 1 \leq \mu_{\max}$. Finally, $\alpha \leq 1$ implies Eq. (3.11).
Contrary to $P_1, \ldots, P_\ell$, some exploration paths $\widehat{P}$ may be such that the distance traversed on them is at most α multiplied by their own cost $\widehat{\omega}$. The distance traversed during the phases which are not associated with $P_1, \ldots, P_\ell$ is the cost of these exploration paths $\widehat{P}$ and their α-detours. As $\widehat{\omega} \leq \omega_{\text{opt}}$, it is at most $2\alpha k_3\omega_{\text{opt}}$. Applying Eq. (3.5), the competitive ratio of α-DETOUR admits the following upper-bound:

$$\begin{aligned}
\frac{\omega_{\alpha-\text{DETOUR}}}{\omega_{\text{opt}}} &\leq \frac{2k_1\omega_{h-1} + 2\alpha(k_2 + k_3 - \mu_{\max})\omega_{\text{opt}} + 2\mu_{\max}\omega_{\text{opt}} + \omega_{\text{opt}}}{\omega_{\text{opt}}} \\
&\leq \frac{k_1\omega_\ell}{\alpha\omega_{\text{opt}}} + 2\mu_{\max} + 2\alpha(k_2 + k_3 - \mu_{\max}) + 1 && (3.12)\\
&\leq \frac{k_1}{\alpha} + 2\mu_{\max} + 2\alpha(k_2 + k_3 - \mu_{\max}) + 1. && (3.13)
\end{aligned}$$

Inequality (3.12) follows from the splitting $\{S_1, S_2\}$ which imposes $2\alpha\omega_{h-1} < \omega_\ell$. □

Let $c_{\text{det}}(k_1, k_2, k_3, \alpha)$ denote the value in Eq. (3.13). Parameters k_1, k_2, and k_3 depend on the road map (G, E_*), so only $\alpha \in (0, 1)$ can be tuned. Value $\alpha = \frac{\sqrt{2}}{2}$ minimizes $c_{\text{det}}(k_1, k_2, k_3, \alpha)$ under the condition $k_1 + k_2 + k_3 = k$ for any $k > \mu_{\text{max}}$. Formally,

$$\frac{\sqrt{2}}{2} = \operatorname*{argmin}_{0 \leq \alpha \leq 1} \max_{\substack{k_1, k_2, k_3 \in \mathbb{N} \\ k_1 + k_2 + k_3 = k}} c_{\text{det}}(k_1, k_2, k_3, \alpha).$$

Corollary 3.2. *The competitive ratio of* DETOUR *is at most* $2\mu_{max} + \sqrt{2}(k - \mu_{max}) + 1$.

Proof. We obtain this ratio by setting $\alpha = \frac{\sqrt{2}}{2}$ and $k_1 + k_2 + k_3 = k$ in Eq. (3.4). □

Discussion

Even if the global competitiveness of deterministic strategies for the k-CTP was fully treated by Westphal [75], families of graphs for which a competitive ratio better than $2k + 1$ can be achieved, can be identified. In this context, we designed the DETOUR strategy to improve significantly the competitive ratio on graphs satisfying $\mu_{\text{max}} < k$. Its competitive ratio is $2\mu_{\text{max}} + \sqrt{2}(k - \mu_{\text{max}}) + 1$.

The strategy DETOUR is as competitive as REPOSITION/COMPARISON for the range $\mu_{\text{max}} \geq k$ but better for the range $1 \leq \mu_{\text{max}} < k$. The slope of the competitive ratio of DETOUR when k varies is only $\sqrt{2}$ for $\mu_{\text{max}} < k$. Figure 3.14 gives the shape of the competitive ratios of REPOSITION (in blue) and DETOUR (in red) as a function of k.

The DETOUR strategy identifies the shortest (s, t)-paths and (pos, t)-paths at any moment of its execution. These paths are established by a single execution of Dijkstra's algorithm [43], performed between two discoveries of blockages with t as the starting point. Hence, the running time of DETOUR is $O(k(m + n \log n))$.

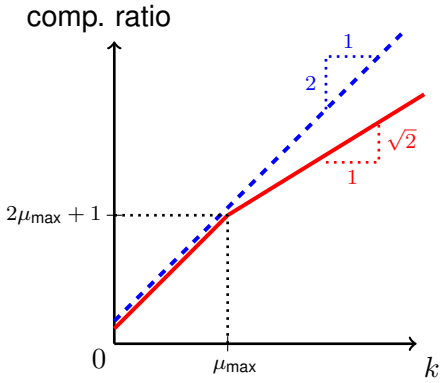


Figure 3.14: Competitive ratio of REPOSITION (blue) and DETOUR (red) in function of k

As for REPOSITION and COMPARISON, the DETOUR strategy can be launched when no upper bound k on the number of blockages is known and its competitive ratio becomes $2\mu_{\text{max}} + \sqrt{2}(|E_*| - \mu_{\text{max}}) + 1$.

The DETOUR strategy can be executed without knowing the value μ_{max}. Indeed, its competitive ratio depends on μ_{max} but no decision is based on μ_{max} in Algorithm 4.

One may wonder whether the estimation of a possible gain of competitiveness can be determined without launching our strategy. To answer this question, we need to verify whether $\mu_{\text{max}} < k$. From [56] we know that finding one of the largest minimal (s, t)-cuts is NP-hard even for planar graphs. Therefore, we cannot predict the quality of DETOUR for any input G in polynomial time.

A constant approximation ratio for the computation of μ_{max} would allow us, however, to obtain an upper bound of the competitive ratio of DETOUR in polynomial time. A linear time

algorithm computing μ_{max} exists for series-parallel graphs [32], so the competitive ratio of DETOUR can be estimated efficiently for this family of graphs.

3.4.2 The k-CTP on chordal graphs

The graphs which allow us to defeat ratio $2k+1$ may contain many detours. Intuitively, when the traveller is blocked on a vertex u by edge (u,v), there shall be a certain number of short (u,t)-paths to bypass the blockage. In this case, we are not forced to return to s every time and to accomplish a trip similar to REPOSITION's one. The family of chordal graphs seems to offer such a possibility. Chordal graphs do not contain any induced cycle of size greater or equal than four. Said differently, any cycle of size at least four in G has a *chord*, *i.e.* an edge between two vertices which are not adjacent in the cycle. Chords seem to offer lots of detours to a traveller placed in front of a blocked edge. This is why we believe we can benefit from their structure to design a competitive strategy. We study the local competitiveness of deterministic strategies on chordal graphs, reporting the results in [18].

First, we prove that no deterministic strategy achieves a competitive ratio smaller than $2k+1$ on the family of chordal graphs. We use the size of the largest minimal vertex (s,t)-cut, μ^V_{max}, as a factor involved in the competitive ratio of deterministic strategies. This statement can be extended to planar, bipartite graphs, and more generally the well-known families of graphs without any weight restrictions.

Second, we thus study chordal graphs when all edge weights are unitary: in *equal-weight* graphs [30]. The strategy CHORD-WALK we proposed, achieves the competitive ratio $\frac{2k+\omega_{\text{opt}}}{\omega_{\text{opt}}}$. As a trivial polynomial-time strategy handles the case $\omega_{\text{opt}} = 1$ with a competitive ratio 1, ratio $\frac{2k+\omega_{\text{opt}}}{\omega_{\text{opt}}}$ is less than $k+1$ when $\omega_{\text{opt}} \geq 2$. It becomes constant, $O(1)$, for values $\omega_{\text{opt}} = \Omega(k)$.

Vertex max-(s,t)-cut size as an indicator of competitiveness

We establish a relationship between the competitive ratio of strategies and value μ^V_{max}. Ratio $2k+1$ is a lower bound for all graphs verifying $k < \mu^V_{\text{max}}$.

Theorem 3.12. *Let G be an undirected graph with a maximum vertex (s,t)-cut X such that $k < |X| = \mu^V_{max}$. For any $\varepsilon > 0$, there is a weighting ω_ε such that any deterministic strategy is at least $(\frac{2k+1}{1+\varepsilon})$-competitive on graph (G, ω_ε).*

Proof. We define weights ω_ε. For all $e \in E[R(X,s)] \cup E[R(X,t)]$, we put $\omega_\varepsilon(e) = \frac{\varepsilon}{n}$. As cut X is minimal, any vertex $x \in X$ is adjacent to $R(X,s)$ and $R(X,t)$, otherwise x would be useless for the separation of s and t. For any $x \in X$, an arbitrary edge from $R(X,s) \times \{x\}$ has the unitary weight. In contrast, for an arbitrary edge e of $\{x\} \times R(X,t)$, we put $\omega_\varepsilon(e) = \frac{\varepsilon}{n}$. All other edges have the infinite weight: to be more rigorous, assigning weight $(2k+1)n+1$ is equivalent to put a weight $+\infty$ as traversing such edges becomes necessarily unefficient. Consequently, we can remove them and obtain a graph (Figure 3.15) corresponding to the one with vertex-disjoint paths proposed by Westphal in [75].

Suppose that set E_* only contains $(\frac{\varepsilon}{n})$-weighted edges from $X \times R(X,t)$. The traveller must pass through X at some point to arrive at t. At worst, each attempt to go through vertex $x \in X$ ends with a blockage (x,y) with $y \in R(X,t)$. As $k < \mu^V_{\text{max}}$, this can occur k times. The distance traversed between each blockage discovery is longer than 2. After discovering the k^{th} blocked edge and returning to set $R(X,s)$, the traveller goes to t with a distance longer than 1. So, the total distance traversed is at least $2k+1$ and no deterministic strategy has a competitive ratio lower than $\frac{2k+1}{\omega_{\text{opt}}}$. With the weight function ω_ε, we have $\omega_{\text{opt}} \leq 1+\varepsilon$. Therefore, the obtained lower bound is $\frac{2k+1}{1+\varepsilon}$. □

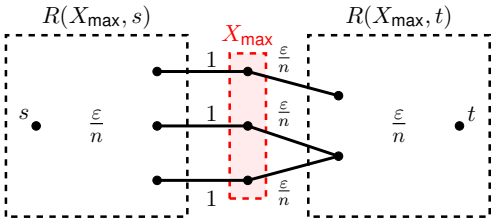


Figure 3.15: Structure of the graph obtained with weight function ω_ε

With $\varepsilon \to 0$, there is no deterministic strategy with ratio less than $2k+1$ for any family of graphs such that:

- no restriction is given for the weight function,
- it contains at least one road map with $k < \mu^V_{\max}$.

Corollary 3.3. *There is no deterministic strategy with competitive ratio lower than* $2k+1$ *for the family of chordal graphs:* $\{((G,\omega), E_*) : G \text{ chordal}\}$.

This statement remains valid for planar, bipartite graphs, and more generally any family containing graphs with $k < \mu^V_{\max}$. Taking advantage of Corollary 3.3, we draw our attention on a smaller family of graphs: chordal graphs where all edge weights are equal.

Description of the CHORD-WALK strategy

We study *equal-weight* graphs $\omega(e) = 1$ for any edge $e \in E$. This is equivalent to setting all weights $\omega(e)$ equal to any other value $a \neq 1$, from the competitive analysis point of view. We consider that $\omega_{\text{opt}} \geq 2$ because the case $\omega_{\text{opt}} = 1$ is trivial: there is an open edge (s,t), the traveller sees it, and traverses it. The competitive ratio obtained must be 1.
We propose a deterministic strategy, called CHORD-WALK, dedicated to equal-weight chordal graphs. Its competitive ratio, $\frac{2k+\omega_{\text{opt}}}{\omega_{\text{opt}}}$, is asymptotically optimal. As $\omega_{\text{opt}} \geq 2$, we observe that $\frac{2k+\omega_{\text{opt}}}{\omega_{\text{opt}}} \leq k+1$. CHORD-WALK is very efficient for a large value ω_{opt}: ratio $O(1)$ is obtained when $\omega_{\text{opt}} \geq k$.
We introduce the notation used to depict CHORD-WALK. At any moment of the traveller trip, G_{vis} denotes the subgraph of G composed of the vertices and edges that the traveller has already visited since his departure from source s. Set E'_* keeps a record of the blocked edges already discovered: $E'_* = \{e_1, e_2, \ldots\}$.
The particularity of the strategy CHORD-WALK is that any edge $(u,v) \in E$ in graph G is traversed at most twice. If an edge is traversed twice, it is done once in each direction: $u \to v$ and $v \to u$.
CHORD-WALK proceeds iteratively. Let P_i be the shortest (s,t)-path in graph G deprived of the blocked edges discovered during the previous iterations, *i.e.* graph $G \backslash E'_*$, when iteration i begins. Let the length of P_i be ω_i. We define the *memory* M_i as the subgraph of G_{vis} induced on edges which have been traversed once since the traveller has started his trip from s. The memory is a stack, updated step-by-step, for edges which can be traversed at most once. The top edge of the stack is incident to the traveller's position.
CHORD-WALK operates according to the following principle: at iteration $i+1$, the traveller looks for a vertex $h \in M_i$ which may be a starting point of a path arriving at t and built of unvisited vertices only. In other words, the traveller seeks a path $Q_h : h \cdots t$ such that all vertices but the starting point h have not been visited yet. When such a path exists, he traverses it and checks any of its intermediary vertex x, whether there is an open edge (x, r_{i+1}) between x and $r_{i+1} \in P_{i+1}$. When this happens, he abandons path Q_h, passes

through (x, r_{i+1}) and traverses section $P_{i+1}^{(r_{i+1},t)}$. Iteration $i+1$ terminates when either the traveller reaches target t or discovers a blockage on P_{i+1}.

The detailed description provided below is accompanied by a pseudocode (Algorithm 5). First of all, CHORD-WALK computes the shortest (s,t)-path P_1 in G. The traveller starts to traverse it and may discover a blocked edge $e_{\ell(1)} = e_1^* = (u_1^*, v_1^*)$ on P_1, where $\ell(1) \geq 1$. The traveller stores in set E'_* all blocked edges incident to the vertices visited on $P_1^{(s,u_1^*)}$. Being stopped by $e_{\ell(1)} = e_1^*$, he finally inserts it to E'_*: $E'_* = \{e_1, \ldots, e_{\ell(1)}\}$.

Iteration i terminates when the traveller either reaches t or is blocked (in u_i^*) by an edge $e_{\ell(i)} = e_i^* = (u_i^*, v_i^*)$ on P_i. We denote $E_*^{(i)}$ the set of blockages discovered on all iterations, from the very beginning: $E_*^{(i)} = \{e_1, \ldots, e_{\ell(i)}\}$.

Input: graph G, source s, target t, positive integer k
$E'_* \leftarrow \emptyset$; $i \leftarrow 1$; $G_{\text{vis}} \leftarrow (\{s\}, \emptyset)$
$P_i \leftarrow$ shortest (s,t)-path in $G \backslash E'_*$; traverse P_i;
if *the traveller is blocked by an edge* $e_i^* = (u_i^*, v_i^*)$ *on* P_i **then**
    $M_i \leftarrow P_i^{(s,u_i^*)}$;
    update E'_* and G_{vis};
endif
while true **do**
    compute (P_{i+1}, s_{i+1}), where P_{i+1} is a shortest (s,t)-path in $G \backslash E'_*$ and s_{i+1} is the closest vertex to u_i^* belonging to both $M_i^{(s,u_i^*)}$ and P_{i+1};
    $R_{i+1} \leftarrow M_i^{(u_i^*, s_{i+1})}$; $h \leftarrow u_i^*$;
    while $h \neq s_{i+1}$ **or** *the traveller is not blocked on* P_{i+1} **do**
        if *a path between* h *and* t *is made up of unvisited vertices* **then**
            $Q_h \leftarrow$ shortest (h,t)-path with unvisited vertices;
            $x \leftarrow$ successor of h on Q_h; go to x;
            if *there is an open edge* (x, r_{i+1}) *with* $r_{i+1} \in P_{i+1}$ **then**
                $x_{i+1} \leftarrow x$; traverse $x_{i+1} \cdot P_{i+1}^{(r_{i+1},t)}$;
            else
                $R_{i+1} \leftarrow x \cdot R_{i+1}$; $h \leftarrow x$;
            endif
        else
            $h \leftarrow$ successor of h on R_{i+1};
        endif
        update u_{i+1}^* and M_{i+1};
    end
    if $h = s_{i+1}$ **then**
        traverse path $P_{i+1}^{(s_{i+1},t)}$;
        update u_{i+1}^* and M_{i+1};
    endif
    if *the traveller reaches* t **then break**;
    else
        $i \leftarrow i+1$;
        update E'_* and G_{vis};
end

Algorithm 5: The CHORD-WALK strategy

We assume at this moment that M_i is a simple (s, u_i^*)-path for all $i \geq 1$. We will prove later that this assumption is true (see the last paragraph of this section). At iteration $i+1$, we compute the shortest (s,t)-path P_{i+1} in $G \backslash E_*$ and next determine the vertex s_{i+1} in both path P_{i+1} and memory M_i which is as close as possible to the current position u_i^* of

the traveller. Such a vertex exists as source s is always a candidate. If there are several shortest (s,t)-paths, we select the path P_{i+1} with the vertex s_{i+1} closest to u_i^*. Section $R_{i+1} = M_i^{(u_i^*, s_{i+1})}$, the reverse of $M_i^{(s_{i+1}, u_i^*)}$, is called the *trip back*. It represents the edges of M_i which have to be traversed to go back to vertex s_{i+1}. We go over R_{i+1}: each edge of the memory which is traversed back is removed from it. We check whether, for any vertex $h \in R_{i+1}$, it is possible to reach t by going through vertices which are not in G_{vis}. Such a section of unvisited vertices is called a *shortcut*. We thus distinguish two scenarii:

- **Scenario 1**: no shortcut is found on R_{i+1}. The traveller returns to vertex s_{i+1} and traverses $P_{i+1}^{(s_{i+1},t)}$ (line 20 of Algorithm 5). If he is blocked, he stops on u_{i+1}^* and the memory is updated $M_{i+1} \leftarrow M_i^{(s,s_{i+1})} \cdot P_{i+1}^{(s_{i+1},u_{i+1}^*)}$ (line 20).
- **Scenario 2**: there is at least one shortcut, *i.e.* an (h,t)-path made of vertices which have not been visited yet (except h). We denote by Q_h the shortest of them (line 12). We fix $M_{i+1} \leftarrow M_i^{(s,h)}$ for now: the memory will be updated as we go along. The traveller traverses Q_h little by little: at any vertex x of this shortcut, he verifies whether there is an open edge (x, r_{i+1}), where $r_{i+1} \in P_{i+1}^{(s_{i+1},t)}$ is as close as possible to t (line 14). If such an edge exists, the traveller goes to r_{i+1} and next traverses section $P_{i+1}^{(r_{i+1},t)}$ (line 15). We denote by x_{i+1} the vertex of the shortcut for which (x_{i+1}, r_{i+1}) is open. Figure 3.16 illustrates the situation described. Iteration $i+1$ terminates when the traveller either reaches t or is blocked by an edge e_{i+1}^* (line 10). In the latter, memory for iteration $i+1$ becomes $M_{i+1} \leftarrow M_{i+1} \cdot Q_h^{(h,x_{i+1})} \cdot P_{i+1}^{(r_{i+1},u_{i+1}^*)}$.

  If the traveller is blocked on x of Q_h before reaching a vertex $r_{i+1} \in P_{i+1}^{(s_{i+1},t)}$, the memory is updated, $M_{i+1} \leftarrow M_{i+1} \cdot Q_h^{(h,x)}$ (line 18). As he did not reach path $P_{i+1}^{(s_{i+1},t)}$ yet, iteration $i+1$ goes on. This iterative process, including the shortcut computation and the memory update, is restarted over the new trip back $R_{i+1} \leftarrow M_{i+1}^{(x,s_{i+1})}$ as long as the traveller does not arrive at a vertex of $P_{i+1}^{(s_{i+1},t)}$. In this situation, he may be blocked many times on shortcuts before reaching $P_{i+1}^{(s_{i+1},t)}$. The memory is frequently updated to ensure that it stores the edges visited only once.

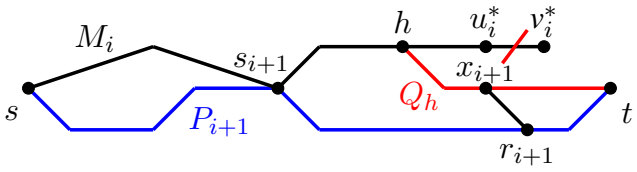


Figure 3.16: Situation evoked in Scenario 2 with a shortcut (red path)

To complete the description, we prove inductively that memory M_{i+1} is a simple (s, u_{i+1}^*)-path, as assumed at the opening. At the beginning, $M_1 = P_1^{(s,u_1^*)}$. Shortcuts contain vertices which have not been visited yet, so they are not memorized in M_i. For the same reason, vertices on $P_{i+1}^{(s_{i+1},t)}$ are not in M_i (except the starting point s_{i+1}) when iteration $i+1$ begins. In a nutshell, for both scenarii, the traveller traverses back edges of M_i and passes through "brand new" vertices which have not been memorized on the previous steps. Therefore, M_{i+1} does not form any cycle and is a simple (s, u_{i+1}^*)-path.

Competitive ratio of the CHORD-WALK strategy

We provide the reader with elements of our competitive analysis producing the competitive ratio of CHORD-WALK.

At any moment of the execution of CHORD-WALK, the vertices of G are split into three parts: those which have not been visited yet, those which are memorized, and the others that we call *dead ends*. A vertex v becomes a dead end when there is no (v,t)-path made up of unvisited vertices in $G\backslash E'_*$. Consequently, a dead end v with one of its incident edges are withdrawn from the current memory as they will not be used by the traveller anymore.
The main idea is to stamp certain blocked edges during the traveller's trip and to bound the distance traversed in function of the number of stamps. We denote by κ_i the number of edges stamped on iteration i and before: $\kappa_i \leq \kappa_{i+1}$. Each blocked edge can be stamped at most twice. We distinguish two types of stamps: A and B. We stamp a blocked edge when one of its endpoints is in $P_{i+1}^{(s_{i+1},t)}$ and the other one:

- is in a shortcut and is visited for the first time (type A),
- becomes a dead end (type B).

No edge can be stamped with A and B in the same time. Consequently, $\kappa_i = \kappa_i^A + \kappa_i^B \leq k$. A-stamped edges are incident to memory M_i. A blocked edge can at the beginning be A-stamped and become B-stamped. The restamping of an edge of type B is impossible.
For Scenario 1, all blocked edges which are incident to both $M_i^{(s_{i+1},u_i^*)}$ and $P_{i+1}^{(s_{i+1},t)}$ are B-stamped when the traveller goes through the vertices of the trip back, as they then become dead ends.
For Scenario 2, all blockages incident to both shortcut Q_h and section $P_{i+1}^{(s_{i+1},t)}$ are A-stamped. As in Scenario 1, we B-stamp the blockages incident to vertices of M_i becoming dead ends.
Let D_i be the total distance traversed by the traveller from his departure to vertex u_i^*. Theorem 3.13 announces that the stamps put in place make the distance traversed D_i on equal-weight chordal graphs be at most $\kappa_i^A + 2\kappa_i^B + \omega_i^{(s,u_i^*)}$, where $\omega_i^{(s,u_i^*)}$ is the length of section $P_i^{(s,u_i^*)}$.

Theorem 3.13. *The distance traversed verifies:* $D_i \leq \kappa_i^A + 2\kappa_i^B + \omega_i^{(s,u_i^*)}$.

The proof of Theorem 3.13 is deferred to the next paragraph because of its length. The competitive ratio of CHORD-WALK follows.

Theorem 3.14. *The competitive ratio of* CHORD-WALK *on equal-weight chordal graphs is* $\frac{2k+\omega_{opt}}{\omega_{opt}}$.

Proof. There is an iteration $i \leq k+1$ where the traveller reaches t ($u_i^* = t$). If $i = 1$, then the shortest (s,t)-path of G is open, so the traveller traverses the optimal offline path and the competitive ratio is 1.
If $i \geq 2$, then $\omega_{\text{opt}} \geq \omega_i$ because P_i is the shortest (s,t)-path in $G\backslash E'_*$ while the optimal offline path is the shortest (s,t)-path in $G\backslash E_*$ and $E'_* \subseteq E_*$.
As $\kappa_i^A + 2\kappa_i^B \leq 2\kappa_i \leq 2k$, we have $D_i \leq 2k + \omega_i^{(s,u_i^*)}$ from Theorem 3.13. The distance traversed from the departure to the moment the traveller arrives at t is at most $2k + \omega_i \leq 2k + \omega_{\text{opt}}$. The competitive ratio of CHORD-WALK is upper-bounded by $\frac{2k+\omega_{\text{opt}}}{\omega_{\text{opt}}}$. □

Upper bound on the number of stamped edges

Before starting, we introduce the notation. For vertices $a, b \in M_i$, let $\omega_{M_i}^{(a,b)}$ be the length of section $M_i^{(a,b)}$ and $\kappa_i^A[a,b]$ be the number of stamped edges incident to $M_i^{(a,b)}$. Theorem 3.15, formulated below, generalizes Theorem 3.13. We remind that memory M_i is a simple (s,u_i^*)-path.

Theorem 3.15. *Distance D_i verifies $D_i \leq 2\kappa_i^B + \omega_{M_i}^{(s,u_i^*)}$ and the length of memory M_i fulfils $\omega_{M_i}^{(s,u_i^*)} \leq \kappa_i^A + \omega_i^{(s,u_i^*)}$. For any $a \in M_i$ such that $a \in P_j$, $j \leq i$, we have $\omega_{M_i}^{(s,a)} \leq \omega_{min}^{(i)}[s,a] + \kappa_i^A[s,a]$, where $\omega_{min}^{(i)}[s,a]$ is the length of the shortest (s,a)-path in $G \backslash E_*^{(i)}$.*

Two cases are distinguished:

- **Case 1**: the concatenation $M_{i+1}^{(s_{i+1},u_i^*)} \cdot P_i^{(u_i^*,t)}$ is induced,
- **Case 2**: it is not.

Certain details of the proof of Theorem 3.15 change depending on these cases. For this reason, we split the proof into two chapters. The notion of *quasi-induced* path, characterizing our trip guided by CHORD-WALK, is explained before and transitional lemmas are proven.

Quasi-induced paths. The following lemma puts in evidence the adjacency of two (u,v)-paths P and Q, $u, v \in V$, where P is induced. In the remainder, "induced" means implicitly induced in the original graph G, not in $G \backslash E'_*$.

Lemma 3.3. *Let P and Q be two simple (u,v)-paths in G. Suppose that P is induced. As G is chordal, any vertex of P is either in Q or adjacent to Q.*

Proof. Let $y \in P$: if vertex y or its neighbors on P belong to path Q, then the lemma holds. We suppose that $y \notin Q$ and is not adjacent to Q. We denote by y_- and y_+ the predecessor and the successor of y on path P, respectively: $y_-, y_+ \notin Q$. Let x be the vertex both in P and Q which arrives before y on path P and is as close as possible to y. Similarly, vertex $z \in P \cap Q$ arrives after y as close as possible from it. The concatenation $P^{(x,z)} \cdot Q^{(z,x)}$ form a cycle C and $y_-, y, y_+ \in C$ (see Figure 3.17). As $y_-, y_+ \notin Q$, we have $x \neq y_-$ and $y_+ \neq z$. The length of C is at least five as it contains vertices x, y_-, y, y_+, and z. Cycle C admits chords because G is chordal. There is no chord $(y, \widehat{y})$ where $\widehat{y} \in P$ as path P is induced. Moreover, $\widehat{y} \notin Q$ because y is not adjacent to Q. In summary, no chord is incident to y.

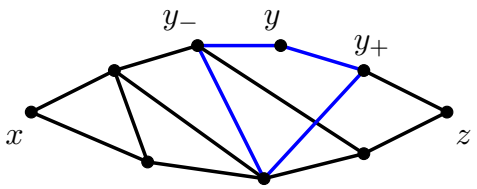


Figure 3.17: Illustration of cycles C and C^*

As y_- and y_+ are the neighbors of y, vertices y_-, y, and y_+ necessarily belong to an induced cycle C^* obtained from certain edges of C and chords. The induced cycle C^* cannot be composed only with these three vertices as $(y_-, y_+) \notin E$ (path P is induced). As a consequence, its size is at least four, which contradicts the chordal graph property. In Figure 3.17, edges in blue form the induced cycle C^*. □

We define the notion of *quasi-induced* paths. As we will see it later, many paths traversed by the traveller can be quasi-induced.

Definition 3.6 (Quasi-induced paths)**.** *Let P be a simple (u,v)-path of G (oriented from u to v). We denote by u_1 and u_2 the first two successors of vertex u, $P : u \cdot u_1 \cdot u_2 \cdots v$. We say that P is* quasi-induced *if there is an edge $(u, u_2) \in E$ and $P^{(u_1,v)}$ is induced.*

The following lemma is equivalent to Lemma 3.3 for the quasi-induced property.

Lemma 3.4. *Let P and Q be two simple (u,v)-paths of G. Suppose P is quasi-induced. As G is chordal, all vertices of P, except the successor of u, are either in Q or adjacent to Q.*

Proof. We denote by u_1 the successor of u and by u_2 the successor of u_1 in path P, *i.e.* $P : u \cdot u_1 \cdot u_2 \cdots v$. The path P' defined by $u \cdot u_2 \cdots v$ is induced. Applying Lemma 3.3, all vertices of P' (which means all vertices of P except u_1) are either in Q or adjacent to Q. □

Any shortest path containing vertices which have not been visited yet, except the departure, is either induced or quasi-induced. This is the case not only for shortcuts but also section $P_{i+1}^{(s_{i+1},t)}$.

Lemma 3.5. *Let $h \in R_{i+1}$ and let P be one of the shortest (h,t)-path of $G\backslash E'_*$ passing through unvisited vertices. Path P is either induced or quasi-induced.*

Proof. We fix $P : h \cdot h_1 \cdot h_2 \cdots t$. We prove that section $P^{(h_1,t)}$ is induced. It contains vertices which have not been visited yet (outside of G_{vis}). If an edge jumps over section $P^{(h_1,t)}$, the traveller sees it open, *i.e.* in $G\backslash E'_*$. There is an (h_1,t)-path shorter than $P^{(h_1,t)}$ as the edge weights are unitary. We obtain a contradiction: P is not a shortest (h,t)-path. As a consequence, path $P^{(h_1,t)}$ is necessarily induced.
Suppose that $j \geq 3$ is the minimum index such that $(h,h_j) \in E$. This edge is blocked, otherwise P is not a shortest (h,t)-path anymore. However, the cycle $h \cdot h_1 \cdots h_j \cdot h$ does not admit chords and its length is at least four. In summary, either P is induced or $(h,h_2) \in E$ (P is quasi-induced). □

Proof for Case 1. We are now ready for the proof of Theorem 3.15. We proceed inductively. The base case $i = 1$: the distance traversed when the traveller reaches vertex u_1^* on path P_1 is $D_1 = \omega_1^{(s,u_1^*)} = \omega_1^{(s,u_1^*)} + \kappa_1^A + 2\kappa_1^B$, as $\kappa_1^A = \kappa_1^B = 0$ (no shortcut, no dead end at the first iteration). Moreover, for $a \in M_1$, path $M_1^{(s,a)} = P_1^{(s,a)}$ is the shortest (s,a)-path in $G\backslash E_*^{(1)}$. So, $\omega_{M_1}^{(s,a)} = \omega_{\text{min}}^{(1)}[s,a]$.
The induction step: the result is assumed for iteration i and we prove it for $i+1$. The proof differs depending on shortcuts. We proceed by treating the two scenarii (see page 103) separately.
Scenario 1 (no shortcut). The induction hypothesis says that $D_i \leq 2\kappa_i^B + \omega_{M_i}^{(s,u_i^*)}$. We remind that memory M_{i+1} is the concatenation of section $M_i^{(s,s_{i+1})}$ with $P_{i+1}^{(s_{i+1},u_{i+1}^*)}$. At iteration $i+1$, sections $R_{i+1} = M_i^{(u_i^*,s_{i+1})}$ and $P_{i+1}^{(s_{i+1},u_{i+1}^*)}$ are traversed. The distance D_{i+1} can be thus written:

$$\begin{aligned} D_{i+1} &= D_i + \omega_{M_i}^{(s_{i+1},u_i^*)} + \omega_{i+1}^{(s_{i+1},u_{i+1}^*)} \\ &\leq 2\kappa_i^B + \omega_{M_i}^{(s,s_{i+1})} + 2\omega_{M_i}^{(s_{i+1},u_i^*)} + \omega_{i+1}^{(s_{i+1},u_{i+1}^*)} \\ &\leq 2\kappa_i^B + 2\omega_{M_i}^{(s_{i+1},u_i^*)} + \omega_{M_{i+1}}^{(s,u_{i+1}^*)}. \end{aligned}$$

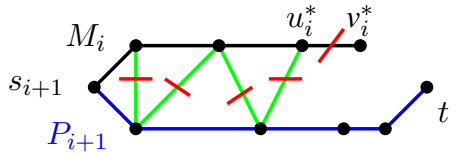


Figure 3.18: Illustration of B-stamped edges (in green) for Scenario 1

We apply Lemma 3.3 to (s_{i+1}, t)-paths $P = M_i^{(s_{i+1},u_i^*)} \cdot P_i^{(u_i^*,t)}$ and $Q = P_{i+1}^{(s_{i+1},t)}$, as P is induced (Case 1). Any vertex of the trip back, except s_{i+1}, is adjacent to $P_{i+1}^{(s_{i+1},t)}$. The number of B-stamped blockages (illustrated in Figure 3.18) at iteration $i+1$ is at least of the length of the trip back: $\kappa_{i+1}^B - \kappa_i^B \geq \omega_{M_i}^{(s_{i+1},u_i^*)}$:

$$D_{i+1} \leq 2\kappa_{i+1}^B + \omega_{M_{i+1}}^{(s,u_{i+1}^*)}.$$

The A-stamped edges are divided into two groups: they are either incident to $M_i^{(s,s_{i+1})}$ or to $M_i^{(s_{i+1},u_i^*)}$: $\kappa_i^A = \kappa_i^A\,[s, s_{i+1}] + \kappa_i^A\,]s_{i+1}, u_i^*]$. The A-stamped edges incident to $M_i^{(s_{i+1},u_i^*)}$ will be restamped with B-type as the traveller traverses the trip back. Therefore, $\kappa_{i+1}^A = \kappa_i^A\,[s, s_{i+1}]$ as no edge is A-stamped when the traveller traverses section $P_{i+1}^{(s_{i+1},u_{i+1}^*)}$. According to the induction hypothesis, $\omega_{M_{i+1}}^{(s,s_{i+1})} = \omega_{M_i}^{(s,s_{i+1})} \leq \kappa_{i+1}^A + \omega_{\mathsf{min}}^{(i)}\,[s, s_{i+1}]$. The shortest (s, s_{i+1})-path in $G \backslash E_*^{(i)}$ is necessarily $P_{i+1}^{(s,s_{i+1})}$, otherwise P_{i+1} would not be the current shortest (s,t)-path in $G \backslash E_*'$: $\omega_{\mathsf{min}}^{(i)}\,[s, s_{i+1}] = \omega_{i+1}^{(s,s_{i+1})}$. In summary,

$$\omega_{M_{i+1}}^{(s,u_{i+1}^*)} = \omega_{M_{i+1}}^{(s,s_{i+1})} + \omega_{i+1}^{(s_{i+1},u_{i+1}^*)} \leq \kappa_{i+1}^A + \omega_{i+1}^{(s,u_{i+1}^*)}.$$

This inequality will hold if we restrict the interval $\left[s, u_{i+1}^*\right]$ to $[s, a]$: for $a \in M_{i+1} \cap P_j$, $\omega_{M_{i+1}}^{(s,a)} \leq \kappa_{i+1}^A\,[s, a] + \omega_{\mathsf{min}}^{(i+1)}\,[s, a]$. If $a \in M_{i+1}^{(s,s_{i+1})}$, the induction hypothesis over $M_i^{(s,a)}$ produces the result. As no edge incident to $M_{i+1}^{(s_{i+1},u_{i+1}^*)}$ is A-stamped, $\omega_{M_{i+1}}^{(s,a)} = \omega_{\mathsf{min}}^{(i+1)}\,[s, a]$ when $a \in P_{i+1}^{(s_{i+1},u_{i+1}^*)} = M_{i+1} \cap P_{i+1}$.

Scenario 2 (at least one shortcut). As above, our first objective is to obtain the inequality $D_{i+1} \leq 2\kappa_{i+1}^B + \omega_{M_{i+1}}^{(s,u_{i+1}^*)}$. Lemma 3.3 still implies that each vertex of the trip back, except s_{i+1}, is adjacent to $P_{i+1}^{(s_{i+1},t)}$. Let r_h be the vertex of $P_{i+1}^{(s_{i+1},t)}$ which is adjacent to the starting point $h \neq s_{i+1}$ of shortcut Q_h and as close as possible to t. We prove that $r_h \neq s_{i+1}$. If h is the successor of s_{i+1} in section $M_i^{(s_{i+1},t)}$, then h must have a neighbor $r_h \neq s_{i+1}$, otherwise it belongs to an induced cycle longer than four containing s_{i+1}, itself and its successor. If h is located after the successor of s_{i+1}, an edge (s_{i+1}, h) would yield a contradiction with Case 1.

We compute the distance traversed between u_i^* and x_{i+1}. For any shortcut Q_h traversed, we apply Lemma 3.4 to paths $P' = Q_h$ and $Q' = h \cdot P_{i+1}^{(r_h,t)}$, as Lemma 3.5 says that Q_h is either quasi-induced or induced. Vertex h_2 and all its descendants on Q_h are adjacent to Q'. Path Q_h is either induced or quasi-induced which means that there is an edge (h, h_2). Edge (h, h_2) is blocked, otherwise path $h \cdot Q_h^{(h_2,t)}$ would be shorter than Q_h. If h_1 is not adjacent to $P_{i+1}^{(r_h,t)}$, we decide to A-stamp the blockage (h, h_2) (instead of a potential blockage (h_1, r_{h_1})). In this way, we ensure that value κ_{i+1}^B is incremented when an edge is traversed a second time.

We remind that no edge is A-stamped on the trip between r_{i+1} and u_{i+1}^*. Let $\widehat{\omega}_{i+1}$ be the number of edges traversed during iteration $i+1$ which remain in memory M_{i+1} when the iteration terminates. The distance traversed on iteration $i+1$ can be thus written: $D_{i+1} - D_i = \widehat{\omega}_{i+1} + \kappa_{i+1}^B - \kappa_i^B$.

The length of memory M_{i+1} decreases when the traveller passes through dead ends, so $\omega_{M_{i+1}}^{(s,u_{i+1}^*)} - \omega_{M_i}^{(s,u_i^*)} = \widehat{\omega}_{i+1} - \left(\kappa_{i+1}^B - \kappa_i^B\right)$. Combining these equations:

$$\begin{aligned} D_{i+1} &= D_i + \omega_{M_{i+1}}^{(s,u_{i+1}^*)} - \omega_{M_i}^{(s,u_i^*)} + 2\left(\kappa_{i+1}^B - \kappa_i^B\right) \\ &\leq 2\kappa_i^B + \omega_{M_{i+1}}^{(s,u_{i+1}^*)} + 2\left(\kappa_{i+1}^B - \kappa_i^B\right) \\ &\leq 2\kappa_{i+1}^B + \omega_{M_{i+1}}^{(s,u_{i+1}^*)}. \end{aligned}$$

A deeper analysis is needed to deal with A-stamps. We have already indicated that all vertices of shortcuts are adjacent to a blocked edge (except h_1 but this can be compensated with the A-stamped edge (h, h_2)). These edges are A-stamped when the traveller traverses a shortcut Q_h for the first time.

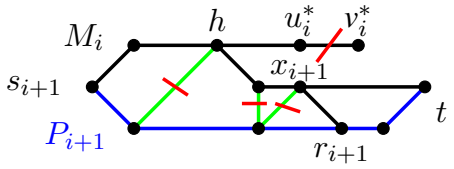


Figure 3.19: Illustration of A-stamped edges (in green) for Scenario 2

The collection of A-stamped edges when iteration $i+1$ terminates is composed of those incident to $M_i^{(s,h)}$ ($\kappa_i^A[s,h[$ edges) and those incident to $M_{i+1}^{(h,r_{i+1})}$ (κ_{i+1}^A edges, drawn in Figure 3.19), as no blockage is stamped on $M_{i+1}^{(r_{i+1},u_{i+1}^*)}$ in any manner:

$$\omega_{M_{i+1}}^{(h,r_{i+1})} = \kappa_{i+1}^A - \kappa_i^A[s,h[\,. \tag{3.14}$$

We denote by b the vertex of M_i as close as possible to h which was on a certain P_j, $j \leq i+1$. Indeed, vertex h may have been visited via a shortcut and not the shortest (s,t)-path. Eq. (3.14) can be generalized, as all edges of $M_i^{(b,h)} = M_{i+1}^{(b,h)}$ share an endpoint with A-stamped edges:

$$\omega_{M_{i+1}}^{(b,r_{i+1})} = \kappa_{i+1}^A - \kappa_i^A[s,b[\,.$$

As h, vertex b has a neighbor in section $P_{i+1}^{(s_{i+1},t)}$ according to Lemma 3.3: we denote it by r_b. We prove that vertex r_{i+1} arrives after r_b on path P_{i+1}, *i.e.* $r_{i+1} \in P_{i+1}^{(r_h,t)}$. Section $M_{i+1}^{(b,r_{i+1})}$ is made up of shortcuts. We consider the shortcut the traveller traverses when he passes through x_{i+1}, denoted by $Q_{h'}$. We apply Lemma 3.4 for paths $P = M_{i+1}^{(b,x_{i+1})} \cdot Q_{h'}^{(x_{i+1},t)}$ and $Q = b \cdot P_{i+1}^{(r_b,t)}$. Vertex x_{i+1} has at least one neighbor on $P_{i+1}^{(r_b,t)}$. We know that one of these neighbors, r_{i+1}, is as close as possible to t. So, r_{i+1} arrives after r_b. The length of memory M_{i+1} can be decomposed as:

$$\begin{aligned}
\omega_{M_{i+1}}^{(s,u_{i+1}^*)} &= \omega_{M_i}^{(s,b)} + \omega_{M_{i+1}}^{(b,r_{i+1})} + \omega_{i+1}^{(r_{i+1},u_{i+1}^*)} \\
&\leq \kappa_i^A[s,b] + \omega_{\mathsf{min}}^{(i)}[s,b] + \left(\kappa_{i+1}^A - \kappa_i^A[s,b]\right) + \omega_{i+1}^{(r_{i+1},u_{i+1}^*)} \\
&\leq \kappa_{i+1}^A + \omega_{\mathsf{min}}^{(i)}[s,b] + \omega_{i+1}^{(r_{i+1},u_{i+1}^*)}.
\end{aligned}$$

Edge (b,r_b) was considered open when path P_j was computed. The length of the (s,r_b)-paths $P_j^{(s,b)} \cdot r_b$ and $P_{i+1}^{(s,r_b)}$, both open at this moment, can be compared: $\omega_{i+1}^{(s,r_b)} \geq \omega_{\mathsf{min}}^{(j)}[s,b] - 1$. As no blockage is on the previous trip P_j, then $\omega_{\mathsf{min}}^{(j)}[s,b] = \omega_{\mathsf{min}}^{(i)}[s,b]$. We write $\omega_{i+1}^{(s,r_b)} \geq \omega_{\mathsf{min}}^{(i)}[s,b] - 1$. The equality $\omega_{i+1}^{(s,r_b)} = \omega_{\mathsf{min}}^{(i)}[s,b] - 1$ would yield a contradiction. Let $\widehat{b}$ be the successor of b on the (s,t)-path P_j, it follows:

- there is no edge $(r_b,\widehat{b}) \in E$, otherwise P_j would have been longer than $P_{i+1}^{(s,r_b)} \cdot P_j^{(\widehat{b},t)}$ at iteration j,
- there is no edge between b and the successor of r_b on P_{i+1} as r_b is taken as close as possible from t.

Vertices $\widehat{b}$, b, r_b, and its successor belong to a cycle formed by paths $P_j^{(b,t)}$ and $P_{i+1}^{(r_b,t)}$. They necessarily belong to a cycle of size at least four and without chords between these vertices. Therefore, $\omega_{i+1}^{(s,r_b)} \geq \omega_{\mathsf{min}}^{(i)}[s,b]$. Finally, as r_{i+1} arrives after r_b,

$$\omega_{M_{i+1}}^{(s,u_{i+1}^*)} \leq \kappa_{i+1}^A + \omega_{\mathsf{min}}^{(i+1)}[s,r_b] + \omega_{i+1}^{(r_{i+1},u_{i+1}^*)} \leq \kappa_{i+1}^A + \omega_{i+1}^{(s,u_{i+1}^*)}.$$

The same argument makes the "local" inequality $\omega_{M_{i+1}}^{(s,a)} \leq \kappa_{i+1}^A[s,a] + \omega_{\mathsf{min}}^{i+1}[s,a]$ valid for all $a \in M_{i+1} \cap P_j$, $j \leq i+1$. First, if $a \in M_{i+1}^{(s,h)} = M_i^{(s,h)}$, the induction hypothesis provides us with this formula. Second, if $a \in M_{i+1}^{(h,r_{i+1})}$, then we replace Eq. (3.14) by its local version: $\omega_{M_{i+1}}^{(h,a)} = \kappa_{i+1}^A[s,a] - \kappa_i^A[s,h[$. Decomposing the length of the memory into $\omega_{M_i}^{(s,b)} + \omega_{M_{i+1}}^{(b,a)}$ allows us to conclude. Eventually, if $a \in P_{i+1}^{(r_{i+1},u_{i+1}^*)}$, we simply replace u_{i+1}^* by vertex a in the whole reasoning for Scenario 2 and terminate the proof. □

Proof for Case 2. The proof for Case 1 is based on the property that any vertex of the trip back $R_{i+1} = M_i^{(u_i^*,s_{i+1})}$ is adjacent to section $P_{i+1}^{(s_{i+1},t)}$. Thanks to it, there is a relationship between the number of stamps and the distance $\omega_{M_i}^{(s_{i+1},u_i^*)}$. As far as path $M_i^{(s_{i+1},u_i^*)} \cdot P_i^{(v_i^*,t)}$ is induced, this property is true. In the general case, however, certain vertices of R_{i+1} can have no neighbor in $P_{i+1}^{(s_{i+1},t)}$ and are not incident to a potentially B-stamped edge. Our idea is to compensate for the absence of such edges. We also introduce slight modifications of the strategy CHORD-WALK to adapt it to Case 2.

We B-stamp certain blockages which were not taken into account before. If a vertex $h \in M_i^{(s_{i+1},u_i^*)}$ is not adjacent to $P_{i+1}^{(s_{i+1},t)}$, there is at least one jump going from an ancestor $y \neq h$ of h to one of its descendants $z \neq h$ on path $M_i^{(s_{i+1},u_i^*)} \cdot P_i^{(v_i^*,t)}$. According to Lemma 3.3, we know that in the absence of such jumps, then either h has a neighbor in $P_{i+1}^{(s_{i+1},t)}$ or G is not chordal.

We distinguish two kinds of jumps for $h \in M_i^{(s_{i+1},u_i^*)}$: these having an endpoint in $P_i^{(v_i^*,t)}$ (*long jumps*), others have two endpoints in $M_i^{(s_{i+1},u_i^*)}$ (*short jumps*). Long jumps (y,z) are necessarily blocked: when the traveller visits y for the first time, he aims at reaching t by the shortest unvisited (y,t)-path (via a shortcut or the current shortest (s,t)-path). As z is closer to t than y, the traveller would have gone to z directly in order to reach t in the shortest way if (y,z) had been open.

Short jumps can be either blocked or not. We transform our memory $M_i^{(s_{i+1},u_i^*)}$ into a shorter one $N_i^{(s_{i+1},u_i^*)}$ such that any short jump on section $N_i^{(s_{i+1},u_i^*)}$ is blocked. Memory $N_i^{(s_{i+1},u_i^*)}$ is obtained with a greedy procedure: while the current memory admits an open short jump, we take one arbitrarily, say (y,z), and replace section $M_i^{(y,z)}$ by the edge (y,z). Finally, memory $N_i^{(s_{i+1},u_i^*)}$ contains vertices of $M_i^{(s_{i+1},u_i^*)}$ and is shorter: the distance traversed by the traveller leaving u_i^* on the trip back is consequently smaller than the former one. Naturally, the trip back becomes $R_{i+1} \leftarrow N_i^{(u_i^*,s_{i+1})}$.

From now on, we denote $P = N_i^{(s_{i+1},u_i^*)} \cdot P_i^{(v_i^*,t)}$. In this situation, all jumps regardless of their type are blocked. Formally, we define a function φ which, given a vertex $h \in N_i^{(s_{i+1},u_i^*)}$ not adjacent to $P_{i+1}^{(s_{i+1},t)}$, computes a jump $\varphi(h)$ of path P. Then, we prove that function φ is injective: $h \neq h' \Rightarrow \varphi(h) \neq \varphi(h')$. The idea is to replace B-stamped blockages by edges $\varphi(h)$.

Among the jumps over h, we take one of them, $\varphi(h) = (y,z)$, such that no other jump (y',z') verifies $y' \in P^{(y,h)}$ and $z' \in P^{(h,z)}$. Intuitively, jump (y,z) does not "cover" another jump. Supposing that $h \neq h'$ and $\varphi(h) = \varphi(h')$, we obtain a contradiction. Section $P^{(y,z)}$ contains at least four vertices because y, h, h', and z are supposed to be all different. At the same time, it forms an induced cycle with edge (y,z): the existence of a chord would imply that another jump is covered by (y,z). As G is chordal, function φ is injective.

Certain B-stamped blockages are replaced by blocked edges $\varphi(h)$ when a vertex h of the memory is not adjacent to $P_{i+1}^{(s_{i+1},t)}$. As φ is injective, we preserve the property that each vertex of $N_i^{(s_{i+1},u_i^*)}$ visited a second time is associated with one B-stamped edge. For example, for Scenario 1, $\omega_{N_i}^{(s_{i+1},u_i^*)} = \kappa_{i+1}^B - \kappa_i^B$. □

3.5 Conclusion

Our contributions on the competitive analysis are separated into two categories. Some of them examine the global competitiveness of the CTP and the k-CTP. These results concern the competitive ratio of strategies for general graphs. The remaining contributions deal with the local competitiveness, *i.e.* results on the competitive ratio of strategies for certain families of graphs.

For a single traveller, we established that no randomized MS can defeat the optimal competitive ratio $2k+1$ of deterministic strategies [14]. As a consequence, either a randomized strategy has a competitive ratio less than $2k+1$ and uses memory, or no randomized strategy is more competitive than REPOSITION.

For the CTP without parameter k, we identified a graph such that the competitive ratio of any randomized strategy (memoryless or not) on it is greater than $|E_*|+1$ [15]. Therefore, the most competitive randomized strategy has a competitive ratio inside the interval $]|E_*|+1, 2|E_*|+1]$.

We studied the distance competitive ratio of both deterministic and randomized strategies for multiple travellers [13]. As for the case $L=1$, we identified an optimal deterministic strategy with competitive ratio $2(k+1)-\min(k+1,L)$ for general graphs under complete communication. This result puts in evidence the impact of communication on the performance of the fleet, as the optimal competitive ratio without communication is $(2k+1)L$. For randomized strategies, we proved that the competitive ratio $\frac{k+2}{2}$ cannot be defeated.

Then, we highlighted results for certain families of graphs. When k is at least the size μ_{max} of the largest minimal (s,t)-cut, strategy DETOUR achieves a competitive ratio $2\mu_{\text{max}} + \sqrt{2}(k-\mu_{\text{max}})+1 < 2k+1$ [19]. Contrary to the existing strategies REPOSITION and COMPARISON, our strategy benefits from large values of k compared to the edge max-(s,t)-cut size μ_{max}.

A second strategy, CHORD-WALK, is proposed for equal-weight chordal graphs [18]. Its competitive ratio is $\frac{2k+\omega_{\text{opt}}}{\omega_{\text{opt}}}$. As the case $\omega_{\text{opt}}=1$ can be easily identified and treated with a competitive ratio 1, we conclude that it improves ratio $k+1$ on this family of graphs.

Chapter 4

Conclusion

To conclude this study, we remind our contributions for both cut problems and the CTP and highlight their consequences. The possible lines of research which emerge from our work will be presented separately in Chapter 5.

4.1 Contributions on the parameterized complexity of cut problems

We investigated the fixed-parameter tractability of cut problems. We devised FPT$\langle p \rangle$ algorithms, where p is the cutset size. In particular, one solves EDGE POTC in $O^*(2^{O(p^2)})$ and another one counts the number of minimum (S,T)-cuts in $O^*(2^{O(p \log p)})$.
The parameterized complexity of multi-terminal cut problems has been intensively studied in the literature. A major result is the fixed-parameter tractability of MULTICUT [63], which asks for the smallest separation of k pairs of terminals (s_i, t_i). We oriented our research towards the parameterized complexity of the more general problem PARTIAL MULTICUT, asking for the smallest cut separating at least r pairs of terminals. We focused on a special case, where all targets are identical: $t_i = t$ for $1 \leq i \leq k$. We called this problem PARTIAL ONE-TARGET CUT (POTC).
We studied the complexity of POTC for all parametrizations involving k, r, and p. The edge and vertex versions, EDGE and VERTEX POTC respectively, have been treated separately as their parameterized complexity may differ. Any algorithm computing a solution for VERTEX POTC also solves EDGE POTC thanks to an edge-to-vertex cut reduction we reminded: EDGE POTC $\leq_{\text{fpt}}$ VERTEX POTC.
If k is a parameter, a trivial FPT$\langle k \rangle$ algorithm exists for both edge and vertex versions. Then, we brought to light a relationship between important cuts and the solutions of EDGE POTC: at least one solution Y^* is the union of certain important (s_i, t)-cuts. Using a reduction towards PARTIAL SET COVER, we identified an FPT algorithm parameterized by both r and p. Identifying particular edges of the graph associated with solution Y^*, called *edge passes*, we designed an FPT$\langle p \rangle$ algorithm based on a color-coding of both edges in Y^* and edge passes. Its running time is $O^*(2^{O(p^2)})$. This result is our main contribution for POTC.
Then, two hardness proofs are provided. EDGE POTC is W[1]-hard for parameter r only and VERTEX POTC is W[1]-hard for parameters r and p. Consequently, there is no chance that an FPT$\langle p \rangle$ algorithm for VERTEX POTC will be identified, contrary to EDGE POTC. In fact, VERTEX POTC is FPT for a single parametrization: k, p.
Next, we studied the fixed-parameter tractability of the counting of minimum (S,T)-cuts in undirected graphs. This problem consists in determining the number of minimum (S,T)-cuts in an input graph G. We designed two FPT$\langle p \rangle$ algorithms for it, dropping below the running time $O(2^{2^p})$ reported in the state of the art. The first one computes the number of minimum edge (S,T)-cuts in time $O^*(2^{O(p^2)})$. It is improved by the second one which not only counts minimum vertex (S,T)-cuts but also returns the result in $O^*(2^{O(p \log p)})$. Thanks

to the edge-to-vertex reduction, the second algorithm can be used to count minimum edge (S,T)-cuts in $O^*(2^{O(p \log p)})$ as well.

The first algorithm, counting minimum edge (S,T)-cuts in instance $\mathcal{I} = (G,S,T)$, benefited from the *drainage*, a structure we proposed. It is a succession of at most n disjoint minimum (S,T)-cuts Z_i such that their source sides are included one into another: $R(Z_i,S) \subsetneq R(Z_{i+1},S)$. If a minimum cut X is not a drainage cut, $X \neq Z_i$, then it admits a unique front dam $B_i \subsetneq X$, *i.e.* a subset of a cut Z_i such that no edge of X lies in the source side of Z_i. The algorithm first counts the minimum drainage cuts and then enumerates all potential front dams in order to count all cuts $X \backslash B_i$ of graph $G \backslash B_i$ recursively. To do so, we put in evidence for each dam B_i a sub-instance of $\mathcal{I}$ in polynomial time, called the *dry instance* and denoted $\mathcal{D}(\mathcal{I}, \overline{B}_i)$. Cut X admits a front dam B_i iff $X \backslash B_i$ is a minimum cut for instance $\mathcal{D}(\mathcal{I}, \overline{B}_i)$. The fast computation of the dry instances allows us to complete our recursion in FPT$\langle p \rangle$ time.

The second algorithm introduces new concepts to count minimum vertex (S,T)-cuts and decrease the execution time of the first one. The definition of the drainage is transposed to the minimum vertex (S,T)-cuts. Furthermore, we propose a *local drainage* which, given an instance $\mathcal{I}$ and frontier U, is a succession of minimum vertex (S,T)-cuts Z_i^U included in the source side of U. This allows us to characterize the minimum cuts which have at least one vertex in $R(U,S)$. Said differently, the minimum vertex (S,T)-cuts X included in the target side of U do not admit a front dam in this drainage. Compared to the first algorithm, the number of recursive calls goes down thanks to the local drainage and additional properties of the dry instance. Our computation based upon the DP principle produces the number of minimum vertex (S,T)-cuts in $O^*(2^{O(p \log p)})$.

These two algorithms can be easily adapted to sample minimum (S,T)-cuts, *i.e.* to return one of them at random, in FPT$\langle p \rangle$ time. However, they cannot be used to obtain an FPT$\langle p \rangle$ enumeration of the minimum (S,T)-cuts, as certain graphs contain $\Omega(n^p)$ minimum cuts.

4.2 Contributions on the Canadian Traveller Problem

We were interested in the competitiveness of both deterministic and randomized strategies for the Canadian Traveller Problem (CTP). Our contributions have been partitioned in two categories: global and local competitiveness.

The global competitiveness of strategies gathers all results related to the competitive analysis of strategies for general graphs. For the k-CTP, it was communicated in the literature that REPOSITION strategy is optimal and its ratio is $2k+1$. The optimal competitive ratio for randomized strategies was not identified.

We investigated this question and provided two results on this topic. First, we studied the impact of memory on the competitiveness of randomized strategies. We proved that memoryless strategies, which do not use the vertices visited previously by the traveller to make decisions, admit at best a global competitive ratio $2k + O(1)$. As a consequence, randomized memoryless strategies are asymptotically as competitive as REPOSITION in the best scenario. Suppose that a randomized strategy with competitive ratio $\beta k + 1$, where $\beta < 2$, exists: it necessarily uses memory. This is an important indication for the research of a randomized strategy outperforming the deterministic ones.

Second, we showed that no randomized strategy is $(|E_*|+1)$-competitive. It was proven in [75] that value $k+1$ is a lower bound on the competitive ratio of randomized strategies for the k-CTP, such as $|E_*|+1$ for the CTP. The proof that bound $|E_*|+1$ is not attained consists in transforming the problem of finding a $(|E_*|+1)$-competitive strategy into a linear inequality system. Farkas' lemma allowed us to affirm that this system has no solution. A consequence of this result is that, if a $(k+1)$-competitive strategy exists for the k-CTP, it must make decisions in function of k. Otherwise, it would mean that this strategy is

$(|E_*|+1)$-competitive for the CTP.

Our last contribution on the global competitiveness of strategies concerned the k-CTP with a group of L travellers, $L > 1$. We provided an overview of the distance competitive ratio of both deterministic and randomized strategies. In particular, we designed the optimal deterministic strategy, called MULTI-ALTERNATING, which makes one of the travellers reach t when the total distance traversed by the fleet is at most $2(k+1) - \min(k+1, L)$ times the optimal offline cost.

The global competitive ratio offers a guarantee of performance on general graphs. If our objective is to guide the traveller on a given family of graphs, this measure might be large compared to the performance we can get on this particular set of instances. This is why we focused on the local competitiveness of strategies. We designed deterministic strategies which defeat the ratio $2k+1$ on two families of graphs: the ones with small minimal (s,t)-cuts and equal-weight chordal graphs.

We proposed a deterministic strategy DETOUR which achieves a competitive ratio $2\mu_{\mathsf{max}} + \sqrt{2}(k-\mu_{\mathsf{max}})+1$ on graphs where value k is larger than the size μ_{max} of the largest minimal edge (s,t)-cut. This result puts in evidence a family of graphs for which a better guarantee than $2k+1$ exists. Moreover, for any graph, the evolution of the best competitive ratio in function of k changes when k overpasses value μ_{max}.

Eventually, we studied the competitive ratio of deterministic strategies on chordal graphs. We introduced a relationship between the size μ^V_{max} of the largest minimal vertex (s,t)-cut and the competitive ratio of deterministic strategies. Given a graph G satisfying $k < \mu^V_{\mathsf{max}}$, for any $\varepsilon > 0$, there is a weighting ω_ε for which the competitive ratio of any deterministic strategy is at least $2k+1-\varepsilon$. As some chordal graphs verify $k < \mu^V_{\mathsf{max}}$, no deterministic strategy is more competitive than REPOSITION on the family of chordal graphs. However, we devised a strategy CHORD-WALK dedicated to equal-weight chordal graphs. Its competitive ratio is $\frac{2k+\omega_{\mathsf{opt}}}{\omega_{\mathsf{opt}}}$. As a trivial strategy admits the competitive ratio 1 for the case $\omega_{\mathsf{opt}} = 1$, the competitive ratio of CHORD-WALK is less than for $k+1$ in general. Furthermore, it reaches a constant competitive ratio, $O(1)$ when values k and ω_{opt} are comparable, *i.e.* $\omega_{\mathsf{opt}} = \Omega(k)$.

Chapter 5

Further research

We could not, unfortunately, treat all questions which have been emerged and are of interest to extend or improve our results. We present several possible lines of research that could be followed.
In Section 5.1, open questions related to the parameterized complexity of cut problems, more particularly multi-terminal ones, are enumerated. We ask especially whether a polynomial kernel exists for EDGE POTC (Section 5.1.1) and whether an FPT$\langle p \rangle$ algorithm can be devised for EDGE PARTIAL MULTICUT (Section 5.1.2). We propose techniques which could be useful to answer these questions.
As we believe that determining the lowest competitive ratio is a challenging question for certain families of graphs, we identify two axes of research for the k-CTP in Section 5.2. Indeed, in its first part, we describe the relationship we detected [18, 19] between the k-CTP and the size of the minimal (s,t)-cuts. This aspect could be exploited to improve the performance of deterministic strategies for certain instances. In the second part of Section 5.2, we discuss the competitive ratio of randomized strategies on apex trees. In our opinion, they could offer us intuition for the design of a strategy achieving a global competitive ratio smaller than $2k+1$.

5.1 Deeper exploration of the parameterized complexity of cut problems

We examine some lines of further research derived from our parameterized complexity results on the POTC problem and the counting of minimum (S,T)-cuts.
First, we wonder how we could outperform the algorithms proposed in Chapter 2. We are particularly interested in the notion of *polynomial kernels* for POTC. Moreover, we investigate the manner to find lower bounds of the time complexity attainable for both POTC and counting of minimum (S,T)-cuts.
Second, we study the impact of our results for PARTIAL MULTICUT which is a natural generalization of both POTC and MULTICUT.

5.1.1 Polynomial kernels and lower bounds

A powerful technique to design not only efficient FPT algorithms but also approximation algorithms and heuristics as well is *kernelization*. Its definition is as follows.

Definition 5.1 (Kernelization)**.** *A parameterized problem admits a* kernel *if there is a polynomial-time algorithm which transforms an input instance (I,p) into another instance (I',p') of the same problem, where both the instance size $|I'|$ and the parameter value p' depend on value p only,* i.e. $|I'|, p' \leq f(p)$ *with an arbitrary function f.*

Any FPT parameterized problem has a kernel [37]. Conversely, any kernelizable problem is FPT as the execution time of the brute-force algorithm on the instance (I', p') is FPT.
Consequently, it is worth wondering which FPT parameterized problems admit a *polynomial kernel*. We say that a parameterized problem has a polynomial kernel if it has a kernel, where function f is polynomial, *i.e.* $f(p) = p^{O(1)}$. For example, VERTEX COVER parameterized by the solution size admits a polynomial kernel of size $O(p^2)$ [68].
We proved in Section 2.2 that POTC is FPT parameterized by the cutset size p. Looking for a polynomial kernel of POTC is a possible future work.

Open question 5.1. *Does* POTC *parameterized by* p *admit a polynomial kernel?*

Nevertheless, it was proven in [38] that, unless coNP $\subseteq$ NP/poly, several multi-terminal cut problems parameterized by the cutset size have no polynomial kernel: EDGE/VERTEX MULTICUT, DIRECTED MULTIWAY CUT, and k-WAY CUT. For this reason, our first approach to answer Question 5.1 would consist in producing a similar proof.
Boedlander *et al.* [24] provided an advanced technique to show that a parameterized problem does not admit a polynomial kernel. Let L be a parameterized problem and $\tilde{L}$ be its classical version, without parameters, which is NP-complete. It states that L may have no polynomial kernel if it admits a *composition*, which is defined below.

Definition 5.2 (Composition)**.** *A* composition *of a parameterized problem L is an algorithm which takes as an input a collection $((x_1, k), (x_2, k), \ldots, (x_r, k))$ of instances of L and outputs a single instance (y, k') of the same problem. It satisfies the following conditions:*

- *its running time is a polynomial of the size of all x_i's,* $\left(k + \sum_{1 \leq i \leq r} |x_i|\right)^{O(1)}$,
- *(y, k') is a solution for L iff some (x_i, k), $1 \leq i \leq r$, is a solution,*
- *value k' is bounded by a polynomial of k, $k' = k^{O(1)}$.*

Another key notion to understand the framework proposed in [24] is the *distillation*. It transforms a collection of instances of a classical problem into a single one which is feasible iff at least one of the instances inside the collection is feasible too.

Definition 5.3 (Distillation)**.** *A* distillation *of a classical problem $\tilde{L}$ is an algorithm which takes as an input a collection $(\tilde{x}_1, \ldots, \tilde{x}_r)$ of instances of $\tilde{L}$ and outputs a single instance $\tilde{y}$ of the same problem. It satisfies the following conditions:*

- *its running time is a polynomial of the size of all instances,* $\left(\sum_{1 \leq i \leq r} |x_i|\right)^{O(1)}$,
- *$\tilde{y}$ is a solution for L iff some $\tilde{x}_i$, $1 \leq i \leq r$, is a solution,*
- *the size of $\tilde{y}$ is a polynomial of the maximum size of the collection,* $|\tilde{y}| = \left(\max_{1 \leq i \leq r} |\tilde{x}_i|\right)^{O(1)}$.

If a distillation algorithm exists for any NP-complete problem, then coNP $\subseteq$ NP/poly and the polynomial hierarchy collapses [50]. So, no NP-complete problems may have a distillation algorithm. Boedlander *et al.* [24] showed that, considering a parameterized problem L and its classical version $\tilde{L}$, a composition followed by a polynomial kernelization on L produces a distillation algorithm for $\tilde{L}$, which is supposed to be impossible as $\tilde{L}$ is NP-complete. In summary, the existence of both a composition algorithm and a polynomial kernel may not occur.

Theorem 5.1 (from [24]). *Let L admit a composition and $\tilde{L}$ be NP-complete. Then, there is no polynomial kernel for L, unless coNP $\subseteq$ NP/poly.*

This theorem provides us with an idea for the proof that POTC has no polynomial kernel. The identification of a composition algorithm for POTC suffices. This is an axis to explore, as this technique was used to prove the absence of polynomial kernels for many well-known problems [24], such as k-PATH, asking for the existence of an induced path of length k in an input graph.
However, the proof that MULTICUT has no polynomial kernel [38] does not use this framework, but a slightly different one, proposed in [25]. This new technique, which consists in finding a *cross-composition* of the parameterized problem studied, is another serious candidate if we aim at answering Question 5.1.

Definition 5.4 (Cross-composition). *Let L be a parameterized problem and $\tilde{D}$ be a classical one, which can be different from $\tilde{L}$. We say $\tilde{D}$* cross-composes *L if we can find an equivalence relation $\mathcal{R}$ and an algorithm which, given a collection $(\tilde{x}_1, \ldots, \tilde{x}_r)$ of instances of $\tilde{D}$ in the same equivalence class for $\mathcal{R}$, returns a single instance (y, k) of L. It satisfies the following conditions:*

- *relation $\mathcal{R}$ is polynomial: it decides whether $\mathcal{R}(x, y)$ in time $(|x| + |y|)^{O(1)}$,*
- *the running time of the algorithm is a polynomial of the size of all x_i's, $\left(\sum_{1 \leq i \leq r} |x_i|\right)^{O(1)}$,*
- *(y, k) is a solution for L iff some $\tilde{x}_i$, $1 \leq i \leq r$, is a solution for $\tilde{D}$,*
- *value k is bounded by a polynomial in $\max_{1 \leq i \leq r} |\tilde{x}_i| + \log r$.*

In fact, designing a cross-composition is weaker than a composition, as only the collections containing instances of the same equivalence class for $\mathcal{R}$ have to be transformed in a single one. The same characterization as Theorem 5.1 was established in [25] for cross-compositions.

Theorem 5.2 (from [25]). *If a NP-complete problem $\tilde{D}$ cross-composes L, then there is no polynomial kernel for L, unless coNP $\subseteq$ NP/poly.*

The identification of a cross-composition for POTC would allow us to prove that it does not admit a polynomial kernel. Question 5.1 can thus be reformulated to suggest a research starting point.

Open question 5.2. *Is there a composition algorithm for* EDGE POTC*? A NP-complete problem cross-composing it?*

This represents, in our opinion, a challenging question for research on POTC.
Another prospect for research is the identification of lower bounds on the best running time obtainable not only for POTC but also for the counting of minimum (S, T)-cuts. We remind the usual methodology to find these lower bounds.
For decision parameterized problems, as POTC, the Exponential Time Hypothesis (ETH) is a tool to show that they do not admit an FPT running time below a certain bound [37]. We remind the definition of the ETH.

Definition 5.5 (ETH). *There is a constant $c > 0$ such that no algorithm solves 3-SAT in time $O^*(2^{cn})$, where n is the number of variables in the Boolean formula.*

A consequence of the ETH is that there is no algorithm solving 3-SAT in time $2^{o(n)}$. Some classical NP-hard problems, as VERTEX COVER, DOMINATING SET, and HAMILTONIAN CYCLE, cannot be solved in time $2^{o(n)}$ either, because a linear reduction transforms 3-SAT into these problems [37].
We suppose that there is a *linear parameterized reduction* between 3-SAT (or one of the problems listed above) and POTC, *i.e.* a polynomial-time reduction transforming an instance of size n of the problem satisfying the ETH into an instance of POTC, where $p = O(n)$. If an algorithm solves POTC in time $O^*(2^{o(p)})$, this reduction clarifies an algorithm for 3-SAT in time $2^{o(n)}$ for 3-SAT and the ETH collapses. Therefore, unless the ETH failed, there would be no algorithm solving POTC in time $O^*(2^{o(p)})$. The following question thus arises.

Open question 5.3. *Is there a linear parameterized reduction from a problem L to* POTC*, where L cannot be solved in time $2^{o(|L|)}$ assuming the ETH?*

For counting parameterized problems, a similar framework exists. It uses the #ETH [41], the counting version of ETH.

Definition 5.6 (#ETH)**.** *There is a constant $c > 0$ such that no algorithm counts the satisfying assignments of a 3-SAT formula in time $O^*(2^{cn})$.*

A counting reduction from either 3-SAT or another problem, admitting no $2^{o(n)}$-time algorithm for its counting version under the #ETH, to the counting of minimum (S, T)-cuts could allow us to prove that our running time $O^*(2^{O(p \log p)})$, obtained in [12], Section 2.3.2, is at worst very close to the optimal one.

Open question 5.4. *Is there a parameterized reduction from a counting problem L to the counting of minimum (S, T)-cuts preserving the number of solutions, where $p = O(|L|)$ and L cannot be solved in time $2^{o(|L|)}$ assuming the #ETH?*

Our techniques counting minimum (S, T)-cuts cannot produce a subexponential algorithm as they are based on the enumeration of dams, all $2^p - 1$ subsets of the minimum drainage cuts Z_i, $|Z_i| = p$. In other words, we cannot obtain a running time $O^*(2^{o(p)})$ by extending our two algorithms. So, either a subexponential algorithm exists and uses different techniques, or it does not exist and we should seek an appropriate parameterized reduction to benefit from the #ETH.

5.1.2 Generalization of MULTICUT

Our contributions have consequences for problems we did not treat explicitly. In particular, the hardness of PARTIAL MULTICUT for some parametrizations can be deduced from our results.
We present preliminary results on the complexity of PARTIAL MULTICUT (Definition 2.4, page 16) in which targets t_i can be distinct. A trivial algorithm allows us to affirm that PARTIAL MULTICUT is FPT$\langle k, p\rangle$ for both its edge and vertex versions. It consists in enumerating subsets $\mathcal{J}$ of indices from $\{1, \ldots, k\}$, with $|\mathcal{J}| = r$ and solving MULTICUT with the FPT algorithm presented in [63] for the pairs of terminals indexed with $\mathcal{J}$. Its overall execution time is $O^*\left(2^{k+O(p^3)}\right)$.
However, PARTIAL MULTICUT parameterized by k only is W[1]-hard for edge/vertex cuts because it generalizes MULTICUT which is NP-hard [40], even when $k = 3$. As $r \leq k$, the W[1]-hardness holds for parameter r only. Furthermore, as VERTEX POTC$\langle r, p\rangle$ is W[1]-hard (Theorem 2.9), its generalization to different targets is not FPT. The complexity of EDGE PARTIAL MULTICUT parameterized by p and $p + r$ remains, nevertheless, unknown. Table 5.1 summarizes the current state of the art on PARTIAL MULTICUT.

Open question 5.5. *Is* EDGE PARTIAL MULTICUT *FPT parameterized by p?*

Parameters	EDGE PARTIAL MULT.	VERTEX PARTIAL MULT.
k, p	**FPT**: $2^{k+O(p^3)}$	**FPT**: $2^{k+O(p^3)}$
k	**W[1]-hard**, [61]	**W[1]-hard**, [61]
r	**W[1]-hard**, [61]	**W[1]-hard**, [61]
p	**Open**	**W[1]-hard**, Theorem 2.9
r, p	**Open**	**W[1]-hard**, Theorem 2.9

Table 5.1: Parameterized classes and complexity of PARTIAL MULTICUT

In our opinion, the techniques proposed in [63] to show the fixed-parameter tractability of MULTICUT might be useful if we aim at designing an FPT$\langle p \rangle$ algorithm for EDGE PARTIAL MULTICUT. Some of them were also relevant to handle POTC in our work, in particular the random sampling of important separators. Marx and Razgon devised a proof [63] for MULTICUT based on the *iterative compression* technique. It consists in proving the fixed-parameter tractability of a problem for parameter value p, supposing that a solution W with size $|W| > p$ is given. As the extra input W provides some useful information for MULTICUT, this approach is a reasonable starting point to the design of an FPT$\langle p \rangle$ for PARTIAL MULTICUT.
As for EDGE POTC, an FPT$\langle p \rangle$ algorithm for EDGE PARTIAL MULTICUT would necessarily benefit from the fact that the cutset is made up of edges, as VERTEX PARTIAL MULTICUT is W[1]-hard. This might be the role of edge passes, the notion we proposed for EDGE POTC to derandomize efficiently the color-coding of edges.

5.1.3 Cut problems for directed graphs

We terminate with several words about open questions concerning the parameterized complexity of cut problems on directed graphs. Two of them draw our attention as they are strongly related to our contributions.
The first one asks whether an FPT algorithm can be designed for MULTICUT on directed graphs with parameter p and $k = 3$ terminal pairs. In other words, we wonder if an FPT$\langle p \rangle$ algorithm can decide whether a cut of size at most p separate s_1 from t_1, s_2 from t_2, and s_3 from t_3 in any directed graph.

Open question 5.6. *Is* DIRECTED MULTICUT*, parameterized by p, FPT or W[1]-hard with $k = 3$ terminal pairs?*

This question was formulated in several articles [57, 63, 71]. It is still open for $k = 3$ terminal pairs, but an FPT$\langle p \rangle$ algorithm was found for $k = 2$ only [34]. The problem becomes W[1]-hard [71] for any $k \geq 4$. Therefore, an answer for the case $k = 3$ would give us the number of terminal pairs being the complexity tipping point.
Two approaches can be followed to tackle Question 5.6. We could try a reduction from CLIQUE to our target problem, DIRECTED MULTICUT with $k = 3$, in order to show its W[1]-hardness. If it failed, our second idea would be to find a characterization of a cutset solution X, perhaps based on directed important cuts [61], specific to the case $k = 3$.
Another open question deals with the counting of minimum (S, T)-cuts in directed graphs. We know that an FPT algorithm exists for this problem, parameterized by the size p of the minimum (S, T)-cuts [20, 62] and its running time is $O^*(2^{2^p})$. As for undirected graphs, a natural question is whether we can count these minimum (S, T)-cuts without a tower of exponentials. Said differently:

Open question 5.7. *Is there an algorithm counting the minimum (S,T)-cuts in directed graphs in time $O^*(2^{poly(p)})$, where poly is a polynomial function?*

We believe that the concepts we proposed in [12, 16] (Section 2.3) could also be useful in this case. The drainage and the dry instances can be naturally generalized to directed graphs, following the definitions for vertex cuts in Section 2.3.2 (page 46 for the drainage, Definition 2.27 in page 51 for the dry instances).
The breaking point of our algorithm with directed graphs occurs because the arcs leaving a dry instance do not necessarily belong to entry, exit arcs or leaks, the three families listed in Theorem 2.22, page 52. As a consequence, a dry instance does not always coincide with an enclosed instance, as stated in Theorem 2.24, page 54, which is a key property to prove the validity of our algorithm. So, there is no doubt that new tools need to be introduced to handle the directed graphs.

5.2 Novel insights for the local competitiveness of the k-CTP strategies

We present certain families of graph we would like to study in order to pursue our research on the local competitiveness of strategies for the k-CTP.
First, we explain why we should try to extend our contribution in Section 3.4.1, the DETOUR strategy, in order to understand the real impact of the minimal (s,t)-cuts size on the competitiveness of deterministic strategies.
Second, we focus on the family of apex trees which have already been mentioned in Section 3.3.2. We believe that the competitive ratio on such graphs represents a challenging issue as it could be an intermediary step to answer the major question of this area, *i.e.* to obtain the best global competitive ratio for randomized strategies, as stated below.

Open question 5.8. *Is there a randomized strategy which achieves a global competitive ratio $\beta k + O(1)$, where $\beta < 2$?*

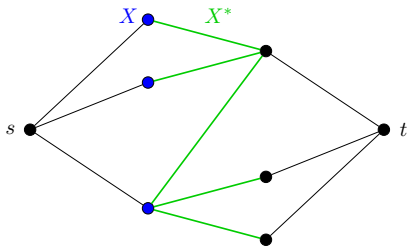
In particular, apex trees generalize graphs with vertex-disjoint (s,t)-paths for which a $(k+1)$-competitive strategy is known.

5.2.1 Relationship between the k-CTP and minimal (s,t)-cuts size

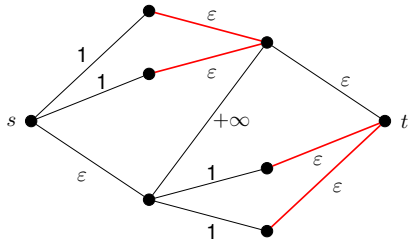
We provided in [19] (Section 3.4.1) a deterministic strategy DETOUR which achieves a competitive ratio $2\mu_{\text{max}} + \sqrt{2}(k - \mu_{\text{max}}) + 1$ for graphs where k is at least the size μ_{max} of the largest minimal edge (s,t)-cuts [19]. This significantly improves the ratio $2k+1$ of REPOSITION for such graphs. An open question is whether there is a smaller value $\mu < \mu_{\text{max}}$ for which we can find a deterministic strategy as DETOUR, dropping below ratio $2k+1$, when $k > \mu$.
A natural candidate for μ is the size μ^V_{max} of the largest minimal vertex (s,t)-cuts. This parameter verifies $\mu^V_{\text{max}} \leq \mu_{\text{max}}$: the largest minimal edge (s,t)-cuts are larger than the vertex ones. Indeed, for any minimal vertex (s,t)-cut X, the set made up of the edges between X and its source side $R(X,S)$ form a minimal edge (s,t)-cut X^*, where $|X^*| \geq |X|$ because no vertex of X is isolated from the source side $R(X,S)$. Figure 5.1a shows a graph with $\mu^V_{\text{max}} = 3$ and $\mu_{\text{max}} = 5$. One of the minimal vertex (s,t)-cuts X, with $|X| = \mu^V_{\text{max}}$ is drawn in blue. A minimal edge (s,t)-cut X^*, $|X^*| = \mu_{\text{max}}$, is in green.
We proved in [18] (Section 3.4.2) that, for any unweighted graph G fulfilling $k < \mu^V_{\text{max}}$, at least one weighting [18] of G prevents from designing a deterministic strategy with a ratio better than $2k+1$. Consequently, for any value $k \in \{1, \ldots, \mu^V_{\text{max}} - 1\}$, at least one weighted graph with k blocked edges is such that no deterministic strategy can defeat $2k+1$ on it.
In summary, the spectrum of values taken by k can be partitioned into three intervals:

- For any input graph where $k \in \{1, \ldots, \mu^V_{\text{max}} - 1\}$, the most competitive strategy known up to now is REPOSITION. Its competitive ratio is $2k + 1$. For certain of these graphs, a strategy more competitive than REPOSITION cannot be found.
- For any input graph where $k \in \{\mu^V_{\text{max}}, \ldots, \mu_{\text{max}} - 1\}$, the most competitive strategy known up to now is REPOSITION. We do not know whether a strategy more competitive than REPOSITION exists for all these graphs.
- For any input graph where $k \geq \mu_{\text{max}}$, the most competitive strategy known up to now is DETOUR and is better than REPOSITION.

An interesting open question issued from analysis is whether we can identify a deterministic strategy as DETOUR, more competitive than REPOSITION, for all graphs, where $\mu_{\text{max}} \neq \mu^V_{\text{max}}$ and $\mu^V_{\text{max}} \leq k < \mu_{\text{max}}$. The answer is no. Considering the graph drawn in Figure 5.1a, we determine a weighting in Figure 5.1b which ensures us that no deterministic strategy is more competitive than REPOSITION. Indeed, REPOSITION is optimal when the road map contains three blocked edges among the four red edges illustrated in Figure 5.1b.



(a) An unweighted graph which satisfies $\mu^V_{\text{max}} < \mu_{\text{max}}$.



(b) The optimal ratio for this weighting is $2k + 1$, with $k = 3$ blocked edges among the red ones.

Figure 5.1: The worst-case weighting of a graph satisfying $\mu^V_{\text{max}} \leq k < \mu_{\text{max}}$.

Can such a weighting be identified for all graphs verifying $\mu^V_{\text{max}} \leq k < \mu_{\text{max}}$? The answer to this last question would help us to understand the relationship between the competitiveness of deterministic strategies and the size of minimal (s, t)-cuts. This would put in evidence the importance of parameter μ_{max} in the evolution of the best competitive ratio depending on k.

Open question 5.9. *Is there a weighting ω_ε such that no deterministic strategy achieves a ratio smaller than $2k + 1 - O(\varepsilon)$ for any weighted graph (G, ω_ε), where G satisfies $\mu^V_{max} \leq k < \mu_{max}$,?*

5.2.2 Apex trees as a key to decrypt the global behavior of randomized strategies

The optimal randomized strategy, achieving the lowest global competitive ratio, is unknown. We do not even know whether a randomized strategy performs better than the optimal deterministic strategy, REPOSITION, with ratio $2k+1$. This is by far the most important open question related to both the CTP and the competitive analysis. We present an approach to answer this question based on a particular graph structure: apex trees.
An apex tree is a graph such that the removal of at least one vertex $v \in V$ produces a tree. The apex trees that have been studied for the k-CTP are such that the removal of the source s gives a tree. From now on, the notion of apex trees refers to this second definition, *i.e.* $G[V \setminus \{s\}]$ is a tree. An equivalent and more tangible definition consists in considering an apex tree as the union of a tree rooted in target t with vertex-disjoint paths between s and some vertices of the tree.
To the best of our knowledge, these graphs have been mentioned in two articles:

- In [42], the authors propose a randomized strategy, called TRAVERSE-TREE, dedicated to apex trees where the costs of all simple (s,t)-paths only differ from a factor $1+\alpha$, where $\alpha \geq 0$. Its competitive ratio is $(1+\alpha)k+1$. In particular, when $\alpha = 0$, TRAVERSE-TREE is $(k+1)$-competitive on apex trees, where all (s,t)-paths have the same cost. We know that TRAVERSE-TREE attains the optimal competitive ratio for such graphs with $\alpha = 0$, as Westphal [75] proved that we cannot drop below $k+1$ on graph G^W (Figure 3.2, page 70), which is an apex tree where all simple (s,t)-paths cost $1+\varepsilon$.

- In [15] (Section 3.3.2), we used an apex tree to prove that ratio $|E_*|+1$ cannot be reached by any randomized strategy for the CTP [15]. In this context, we focused on ε-ATs, which is a family composed of apex trees which are the union of the tree with vertex-disjoint paths connecting s with only some leaves of the tree. We showed that the optimal randomized strategy for ε-ATs is a randomized variant of REPOSITION. In other words, the best randomized strategy for these graphs consists in selecting a simple (s,t)-path according to a probability distribution (unknown for now), traversing the chosen path and coming back to s if a blockage is discovered. Then, the process restarts on the remaining simple (s,t)-paths.

Apex trees naturally generalize the graphs made up only of vertex-disjoint (s,t)-paths, for which a $(k+1)$-competitive strategy was identified [11]. Determining the optimal competitive ratio of randomized strategies on apex trees, or even the smaller family containing ε-ATs, is a promising intermediary step before studying the competitive ratio of randomized strategies for general graphs. According to our comments above, we already know the optimal strategy for apex trees when all simple (s,t)-paths have the same cost. Moreover, we know how the optimal strategy for ε-ATs looks like. We thus formulate the following question.

Open question 5.10. *Is there a randomized strategy which achieves a competitive ratio* $\beta k + O(1)$ *on apex trees, where* $\beta < 2$*?*

Demaine *et al.* provide us with ideas to capture the essential edges of any weighted graph into an equivalent apex tree [42]. Supposing Question 5.10 was solved, such a technique might allow us to extend a local result on apex trees into a global one and find the response to Question 5.8.

Bibliography

[1] V. Aksakalli, O. F. Sahin, and I. Ari. An AO* Based Exact Algorithm for the Canadian Traveler Problem. *INFORMS Journal on Computing*, 28(1):96–111, 2016.

[2] S. Albers. Online algorithms: a survey. *Math. Program.*, pages 3–26, 2003.

[3] A. F. Alkaya, V. Aksakalli, and C. E. Priebe. A penalty search algorithm for the obstacle neutralization problem. *Computers & OR*, 53:165–175, 2015.

[4] V. Arvind and V. Raman. Approximation algorithms for some parameterized counting problems. In *Proc. of ISAAC*, pages 453–464, 2002.

[5] M. O. Ball, C. J. Colbourn, and J. S. Provan. Network reliability. volume 7 of *Handbooks in Operations Research and Management Science*, pages 673 – 762. Elsevier, 1995.

[6] M. O. Ball and J. S. Provan. Calculating bounds on reachability and connectedness in stochastic networks. *Networks*, 13(2):253–278, 1983.

[7] M. O. Ball and J. S. Provan. The complexity of counting cuts and of computing the probability that a graph is connected. *SIAM J. Comput.*, 12(4):777–788, 1983.

[8] M. O. Ball and J. S. Provan. Computing network reliability in time polynomial in the number of cuts. *Operations Research*, 32(3):516–526, 1984.

[9] A. Bar-Noy and B. Schieber. The Canadian Traveller Problem. In *Proc. of ACM/SIAM SODA*, pages 261–270, 1991.

[10] S. Ben-David and A. Borodin. A new measure for the study of on-line algorithms. *Algorithmica*, 11(1):73–91, 1994.

[11] M. Bender and St. Westphal. An optimal randomized online algorithm for the k-Canadian Traveller Problem on node-disjoint paths. *J. Comb. Optim.*, 30(1):87–96, 2015.

[12] **P. Bergé**, W. Bouaziz, A. Rimmel, and J. Tomasik. An FPT counting of small minimum (S, T)-cuts. *in preparation for an international journal*, 2019.

[13] **P. Bergé**, J. Desmarchelier, W. Guo, A. Lefevbre, A. Rimmel, and J. Tomasik. Multiple Canadians on the road: minimizing the distance competitive ratio. *Journal of Comb. Optim., in print*, 2019.

[14] **P. Bergé**, J. Hemery, A. Rimmel, and J. Tomasik. On the Competitiveness of Memoryless Strategies for the k-Canadian Traveller Problem. In *Proc. of COCOA*, pages 566–576, 2018.

[15] **P. Bergé**, J. Hemery, A. Rimmel, and J. Tomasik. The Competitiveness of Randomized Strategies for Canadians via Systems of Linear Inequalities. In *Proc. of ISCIS*, pages 96–103, 2018.

[16] **P. Bergé**, B. Mouscadet, A. Rimmel, and J. Tomasik. Fixed-parameter tractability of counting small minimum (S, T)-cuts. In *Proc. of WG*, 2019.

[17] **P. Bergé**, A. Rimmel, and J. Tomasik. On the parameterized complexity of separating certain sources from the target. doi: 10.1016/j.tcs.2019.06.011. *Theoretical Computer Science*, 2019.

[18] **P. Bergé**, A. Rimmel, and J. Tomasik. The Canadian Traveller Problem on chordal graphs. *in preparation for an international journal*, 2020.

[19] **P. Bergé** and L. Salaün. Improved Deterministic Strategy for the Canadian Traveller Problem Exploiting Small Max-(s, t)-cuts. In *Proc. of WAOA*, 2019.

[20] I. Bezáková, E. W. Chambers, and K. Fox. Integrating and sampling cuts in bounded treewidth graphs. In *Advances in the Math. Sciences*, pages 401–415, 2016.

[21] I. Bezáková and A. J. Friedlander. Counting and sampling minimum (s, t)-cuts in weighted planar graphs in polynomial time. *Theor. Comput. Sci.*, 417:2–11, 2012.

[22] M. Bläser. Computing small partial coverings. *Inf. Process. Lett.*, 85(6):327–331, 2003.

[23] Z. Bnaya, A. Felner, and S. E. Shimony. Canadian traveler problem with remote sensing. In *Proc. of IJCAI*, pages 437–442, 2009.

[24] H. L. Bodlaender, R. G. Downey, M. R. Fellows, and D. Hermelin. On problems without polynomial kernels. In *Proc. of ICALP*, pages 563–574, 2008.

[25] H L. Bodlaender, B. M. P. Jansen, and S. Kratsch. Cross-composition: A new technique for kernelization lower bounds. In *Proc. of STACS*, pages 165–176, 2011.

[26] A. Borodin and R. El-Yaniv. *Online computation and competitive analysis*. Cambridge University Press, 1998.

[27] N. Bousquet, J. Daligault, and S. Thomassé. Multicut is FPT. In *Proc. of STOC*, pages 459–468, 2011.

[28] Y. Boykov and O. Veksler. Graph cuts in vision and graphics: Theories and applications. In *Handbook of Mathematical Models in Computer Vision*, pages 79–96. 2006.

[29] E. W. Chambers, K. Fox, and A. Nayyeri. Counting and sampling minimum cuts in genus g graphs. In *Proc. of SoCG*, pages 249–258, 2013.

[30] H. Chan, J. Chang, H. Wu, and T. Wu. The k-Canadian Traveller Problem on Equal-Weight Graphs. *Proc. of WCMCT*, pages 135–137, 2015.

[31] L. S. Chandran and L. S. Ram. On the number of minimum cuts in a graph. In *Proc. of COCOON*, pages 220–229, 2002.

[32] B. Chaourar. A Linear Time Algorithm for a Variant of the MAX CUT Problem in Series Parallel Graphs. *Adv. Operations Research*, 2017.

[33] J. Chen, Y. Liu, and S. Lu. An improved parameterized algorithm for the minimum node multiway cut problem. *Algorithmica*, 55(1):1–13, 2009.

[34] R. Chitnis, M. Hajiaghayi, and D. Marx. Fixed-parameter tractability of directed multiway cut parameterized by the size of the cutset. *SIAM J. Comput.*, 42(4):1674–1696, 2013.

[35] R. Curticapean. Counting matchings of size k is #W[1]-hard. In *Procs of ICALP*, pages 352–363, 2013.

[36] R. Curticapean. Counting problems in parameterized complexity. In *Proc. of IPEC*, pages 1–18, 2018.

[37] M. Cygan, F. V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, Ma. Pilipczuk, Mi. Pilipczuk, and S. Saurabh. *Parameterized Algorithms*. Springer, 2015.

[38] M. Cygan, S. Kratsch, Ma. Pilipczuk, Mi. Pilipczuk, and M. Wahlström. Clique cover and graph separation: New incompressibility results. In *Proc. of ICALP*, pages 254–265, 2012.

[39] M. Cygan, D. Lokshtanov, Ma. Pilipczuk, Mi. Pilipczuk, and S. Saurabh. Minimum bisection is fixed parameter tractable. In *Proc. of STOC*, pages 323–332, 2014.

[40] E. Dahlhaus, D. S. Johnson, C. H. Papadimitriou, P. D. Seymour, and M. Yannakakis. The complexity of multiterminal cuts. *SIAM J. Comput.*, 23(4):864–894, 1994.

[41] H. Dell, T. Husfeldt, D. Marx, N. Taslaman, and M. Wahlen. Exponential Time Complexity of the Permanent and the Tutte Polynomial. *ACM Trans. Algorithms*, 10(4):21:1–21:32, 2014.

[42] E. D. Demaine, Y. Huang, C.-S. Liao, and K. Sadakane. Canadians Should Travel Randomly. *Proc. of ICALP*, pages 380–391, 2014.

[43] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959.

[44] R. G. Downey, V. Estivill-Castro, M. R. Fellows, E. Prieto-Rodriguez, and F. A. Rosamond. Cutting Up is Hard to Do: the Parameterized Complexity of k-Cut and Related Problems. *Electr. Notes Theor. Comput. Sci.*, 78:209–222, 2003.

[45] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Monographs in Computer Science. Springer, 1999.

[46] J. Farkas. Uber die Theorie der Einfachen Ungleichungen. *J. fur die Reine und Angewandte Math.*, 124:1–27, 1902.

[47] J. Flum and M. Grohe. The parameterized complexity of counting problems. *SIAM J. Comput.*, 33(4):892–922, 2004.

[48] F. V. Fomin, P. A. Golovach, and J. H. Korhonen. On the parameterized complexity of cutting a few vertices from a graph. In *Proc. of MFCS 2013*, pages 421–432, 2013.

[49] L. R. Ford and D. R. Fulkerson. Maximal flow through a network. *Canadian Jour. of Math.*, (8):399–404, 1956.

[50] L. Fortnow and R. Santhanam. Infeasibility of instance compression and succinct PCPs for NP. In *Proc. of STOC*, pages 133–142, 2008.

[51] D. Fried, S. E. Shimony, A. Benbassat, and C. Wenner. Complexity of Canadian traveler problem variants. *Theor. Comput. Sci.*, 487:1–16, 2013.

[52] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[53] M. X. Goemans and D. P. Williamson. Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming. *J. ACM*, 42(6):1115–1145, 1995.

[54] D. Golovin, V. Nagarajan, and M. Singh. Approximating the k-multicut problem. In *Proc. of SODA*, pages 621–630, 2006.

[55] S. Guillemot and F. Sikora. Finding and counting vertex-colored subtrees. *Algorithmica*, 65(4):828–844, 2013.

[56] D. J. Haglin and S. M. Venkatesan. Approximation and Intractability Results for the Maximum Cut Problem and its Variants. *IEEE Trans. Computers*, 40(1):110–113, 1991.

[57] S. Kratsch, S. Li, D. Marx, Ma. Pilipczuk, and M. Wahlström. Multi-budgeted directed cuts. In *Proc. of IPEC*, pages 18:1–18:14, 2018.

[58] S. Kratsch, Ma. Pilipczuk, Mi. Pilipczuk, and M. Wahlström. Fixed-parameter tractability of Multicut in directed acyclic graphs. In *Proc. of ICALP*, pages 581–593, 2012.

[59] A. Levin and D. Segev. Partial multicuts in trees. *Theor. Comput. Sci.*, 369(1-3):384–395, 2006.

[60] C. Liao and Y. Huang. Generalized Canadian traveller problems. *J. Comb. Optim.*, 29(4):701–712, 2015.

[61] D. Marx. Parameterized graph separation problems. *Theor. Comput. Sci.*, 351(3):394–406, 2006.

[62] D. Marx, B. O'Sullivan, and I. Razgon. Finding small separators in linear time via treewidth reduction. *ACM Trans. Algorithms*, 9(4):30:1–30:35, 2013.

[63] D. Marx and I. Razgon. Fixed-parameter tractability of multicut parameterized by the size of the cutset. In *Proc. of STOC*, pages 469–478, 2011.

[64] J. Matousek and B. Gärtner. *Understanding and Using Linear Programming*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[65] K. Menger. Zur allgemeinen Kurventheorie. *Fundamenta Mathematicæ*, 10(1):96–115, 1927.

[66] H. Nagamochi, Z. Sun, and T. Ibaraki. Counting the number of minimum cuts in undirected multigraphs. *IEEE Trans. Reliab.*, 40:610–614, 1991.

[67] M. Naor, L. J. Schulman, and A. Srinivasan. Splitters and near-optimal derandomization. In *FOCS 1995*, pages 182–191, 1995.

[68] R. Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford Lecture Series in Math. and Its Applications. OUP Oxford, 2006.

[69] E. Nikolova and D. R. Karger. Route Planning under Uncertainty: The Canadian Traveller Problem. In *Proc. of AAAI*, pages 969–974, 2008.

[70] Ch. Papadimitriou and M. Yannakakis. Shortest paths without a map. *Theor. Comput. Sci.*, 84(1):127–150, 1991.

[71] Ma. Pilipczuk and M. Wahlström. Directed multicut is $W[1]$-hard, even for four terminal pairs. In *Proc. of SODA*, pages 1167–1178, 2016.

[72] D. Shiri and F. S. Salman. On the online multi-agent O-D k-Canadian Traveler Problem. *J. Comb. Optim.*, 34(2):453–461, 2017.

[73] D. Shiri and F. S. Salman. On the randomized online strategies for the k-canadian traveler problem. *J. Comb. Opt.*, pages 1–14, 2019.

[74] L. G. Valiant. The complexity of counting the permanent. *Theor. Comput. Sci.*, 8:189–201, 1979.

[75] St. Westphal. A note on the k-Canadian Traveller Problem. *Inform. Proces. Lett.*, 106(3):87–89, 2008.

[76] V. V. Williams and R. Williams. Finding, minimizing, and counting weighted subgraphs. *SIAM J. Comput.*, 42(3):831–854, 2013.

[77] Y. Xu, M. Hu, B. Su, B. Zhu, and Z. Zhu. The Canadian traveller problem and its competitive analysis. *J. Comb. Optim.*, 18(2):195–205, 2009.

[78] A. C. Yao. Probabilistic computations: Toward a unified measure of complexity. In *Proc. of FOCS*, pages 222–227, 1977.

[79] H. Zhang, Y. Xu, and L. Qin. The k-Canadian Travelers Problem with communication. *J. of Comb. Opt.*, pages 251–265, 2011.

Acknowledgments

First and foremost, I would like to express deep gratitude to my supervisors Prof. Joanna Tomasik and Dr. Arpad Rimmel. I thank them for accepting nothing less than excellence from me. We have been working together for five years, through student projects, an internship and then my Ph.D. Thanks to their help and support, I was able to overcome the obstacles I have been facing in my research. I am fully convinced that I will take advantage of their numerous advices throughout my academic career.

I would like to thank Bruno Escoffier, Ignasi Sau, and Stephan Westphal for their very insightful comments in the reviews of this thesis. I also express my gratitude to the other members of the jury for attending my Ph.D. defense: Cristina Bazgan, Olivier Bournez and Yannis Manoussakis.

I am grateful to all co-authors that are part of this work: Julien Hemery, Jean Desmarchelier, Wen Guo, Aurélie Lefebvre, Benjamin Mouscadet, Lou Salaün, and Wassim Bouaziz. I want to say thank you to the co-authors that are part of my previous works: Baptiste Cavarec, Kaourintin Le Guiban, Ghada Ben Hassine, Gregory Gutin, Jason Crampton, and Rémi Watrigant.

Eventually, I would like to thank my family, my friends and my colleagues. I will not forget the great support I received, especially from my mother Marielle, my father Éric, and my grandfather Henri.

université PARIS-SACLAY

ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre : Algorithmes pour voyager sur un graphe contenant des blocages

Mots clefs : problèmes de coupes, complexité paramétrée, problème du voyageur canadien, algorithmes on-line

Résumé : Nous étudions des problèmes NP-difficiles portant sur les graphes contenant des blocages.
Nous traitons les problèmes de coupes du point de vue de la complexité paramétrée. La taille p de la coupe est le paramètre. Étant donné un ensemble de sources $\{s_1, \ldots, s_k\}$ et une cible t, nous proposons un algorithme qui construit une coupe de taille au plus p séparant au moins r sources de t. Nous nommons ce problème NP-complet Partial One-Target Cut. Notre algorithme est FPT. Nous prouvons également que la variante de Partial One-Target Cut, où la coupe est composée de noeuds, est W[1]-difficile. Notre seconde contribution est la construction d'un algorithme qui compte les coupes minimums entre deux ensembles S et T en temps $2^{O(p \log p)} n^{O(1)}$.
Nous présentons ensuite plusieurs résultats sur le ratio de compétitivité des stratégies déterministes et randomisées pour le problème du voyageur canadien. Nous prouvons que les stratégies randomisées n'utilisant pas de mémoire ne peuvent pas améliorer le ratio $2k + 1$. Nous apportons également des éléments concernant les bornes inférieures de compétitivité de l'ensemble des stratégies randomisées. Puis, nous étudions la compétitivité en distance d'un groupe de voyageurs avec et sans communication. Enfin, nous nous penchons sur la competitivité des stratégies déterministes pour certaines familles de graphes. Deux stratégies, avec un ratio inférieur à $2k + 1$ sont proposées: une pour les graphes cordaux avec poids uniformes et l'autre pour les graphes où la taille de la plus grande coupe minimale séparant s et t est au plus k.

Title : A guide book for the traveller on graphs full of blockages

Keywords : multi-terminal cuts, parameterized complexity, Canadian Traveller Problem, competitive analysis

Abstract : We study NP-hard problems on graphs with blockages seen as models of networks which are exposed to risk of failures.
We treat cut problems via the parameterized complexity framework. The cutset size p is taken as a parameter. Given a set of sources $\{s_1, \ldots, s_k\}$ and a target t, we propose an algorithm which builds a small edge cut of size p separating at least r sources from t. This NP-complete problem is called Partial One-Target Cut. It belongs to the family of multiterminal cut problems. Our algorithm is fixed-parameter tractable (FPT) as its execution takes $2^{O(p^2)} n^{O(1)}$. We prove that the vertex version of this problem, which imposes cuts to contain vertices instead of edges, is W[1]-hard. Then, we design an FPT algorithm which counts the minimum vertex (S, T)-cuts of an undirected graph in time $2^{O(p \log p)} n^{O(1)}$.
We provide numerous results on the competitive ratio of both deterministic and randomized strategies for the Canadian Traveller Problem. The optimal ratio obtained for the deterministic strategies on general graphs is $2k+1$, where k is a given upper bound on the number of blockages. We show that randomized strategies which do not use memory cannot improve the bound $2k + 1$. In addition, we discuss the tightness of lower bounds on the competitiveness of randomized strategies. The distance competitive ratio for a group of travellers possibly equipped with telecommunication devices is studied. Eventually, a strategy dedicated to equal-weight chordal graphs is proposed while another one is built for graphs with small maximum (s, t)-cuts. Both strategies outperform the ratio $2k + 1$.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France

université PARIS-SACLAY