Optimization problems in graphs with locational uncertainty

Marin Bougeret, Jérémy Omer, Michael Poss

▶ **To cite this version:**

Marin Bougeret, Jérémy Omer, Michael Poss. Optimization problems in graphs with locational uncertainty. 2022. hal-03331166v3

HAL Id: hal-03331166
https://hal.science/hal-03331166v3

Preprint submitted on 4 May 2022 (v3), last revised 20 Dec 2023 (v5)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Optimization problems in graphs with locational uncertainty

Marin Bougeret
LIRMM, University of Montpellier, CNRS, France, marin.bougeret@lirmm.fr
Jérémy Omer
IRMAR, INSA de Rennes, Rennes, France, jeremy.omer@insa-rennes.fr
Michael Poss
LIRMM, University of Montpellier, CNRS, France, michael.poss@lirmm.fr

Many discrete optimization problems amount to selecting a feasible subgraph of least weight. We consider in this paper the context of spatial graphs where the positions of the vertices are uncertain and belong to known uncertainty sets. The objective is to minimize the sum of the distances in the chosen subgraph for the worst positions of the vertices in their uncertainty sets. We first prove that these problems are $\mathcal{NP}$-hard even when the feasible subgraphs consist either of all spanning trees or of all $s-t$ paths. Given this hardness, we propose an exact solution algorithm combining integer programming formulations with a cutting plane algorithm, identifying the cases where the separation problem can be solved efficiently. We also propose a conservative approximation and show its equivalence to the affine decision rule approximation in the context of Euclidean distances. We compare our algorithms to three deterministic reformulations on instances inspired by the scientific literature for the Steiner tree problem and a facility location problem.

Key words: combinatorial optimization, robust optimization, $\mathcal{NP}$-hardness, cutting plane algorithms, dynamic programming

---

1. Introduction

Research in combinatorial optimization has provided efficient algorithms to solve many complex discrete decision problems, providing exact or near-optimal solutions in reasonable amounts of time. The applications are countless, ranging from logistics (network design, facility location, ...) to scheduling. Many of these applications amount to selecting a subgraph of a ground graph $\mathbb{G} = (\mathbb{V}, \mathbb{E})$ among a family of feasible subgraphs $\mathcal{G}$ and that minimizes its total weight. Among those, we focus on spatial graphs on a given metric space $(\mathcal{M}, d)$, where each vertex i is assigned a position $u_i \in \mathcal{M}$. Denoting by $E[G]$ the set of edges of G, the cost of graph $G \in \mathcal{G}$ is given by $\sum_{\{i,j\}\in E[G]} d(u_i, u_j)$, leading to the combinatorial optimization problem

$$\min_{G\in\mathcal{G}} \sum_{\{i,j\}\in E[G]} d(u_i, u_j). \tag{Π}$$

Problem Π encompasses many applications, such as network design and facility location. These are typically subject to data uncertainty, be it because of the duration of the decision process, measurement errors, or simply lack of information. One successful framework that has emerged to address uncertainty is robust optimization (Ben-Tal and Nemirovski 1998), where the uncertain parameters are modeled with convex sets such as polytopes, or with finite sets of points. In this field, combinatorial robust optimization focuses on discrete robust optimization problems (Bertsimas and Sim 2003, Buchheim and Kurtz 2018, Kasperski and Zieliński 2016, Kouvelis and Yu 2013). We enter this framework by considering the model where the positions of the vertices are subject to uncertainty, therefore impacting the distances among the vertices. The resulting problem thus seeks to find the feasible subgraph that minimizes its worst-case sum of distances. Formally, we introduce for each vertex $i \in \mathbb{V}$ the set of possible locations as the uncertainty set $\mathcal{U}_i \subseteq \mathcal{M}$ of cardinality $\sigma_i = |\mathcal{U}_i|$. Using the notations $u = (u_1, \ldots, u_{|\mathbb{V}|})$ and $\mathcal{U} = \times_{i \in \mathbb{V}} \mathcal{U}_i$, the general problem considered in this paper can be cast as

$$\min_{G \in \mathcal{G}} \max_{u \in \mathcal{U}} \sum_{\{i,j\} \in E[G]} d(u_i, u_j). \tag{ROBUST-Π}$$

We also devote a particular attention to computing the objective function of ROBUST-Π, namely

$$\max_{u \in \mathcal{U}} \sum_{\{i,j\} \in E[G]} d(u_i, u_j). \tag{EVAL-C}$$

As an illustration, the following two applications fall into the context of problem ROBUST-Π.

Example 1 (Subway network design)**.** *Designing and expanding a subway network forms an important optimization problem faced by large cities. The new lines should efficiently cover dense city areas while interacting well with the existing transportation lines. A key aspect of this problem amounts to locating the new subway stations. In addition to the technical considerations inherent to any construction, these also involve political considerations as local officials are never happy to let their citizens face the inconvenience of heavy civil engineering. This political lever is particularly complex in cities like Brussels having multiple local governments that must all agree before the stations can finally be constructed. As the overall process takes years, facing local government changes, the exact locations of the metro stations typically evolve between the first draft and the final implementation. Now, the exact locations of the stations impact the lengths of the resulting lines, the construction cost of which is typically proportional to their lengths (Gutiérrez-Jarpa et al. 2013). The cost of digging the new*

lines can therefore be modeled as a network design problem with locational uncertainty on the position of the vertices, usually including additional technical and environmental constraints.

Example 2 (Strategic facility location)**.** *A production company wishes to expand its activities in a new region, locating additional facilities. We consider the strategic level where the company may only choose approximate locations, as the exact locations will be known later, after all technical and legal considerations have been studied. Furthermore, the assignment of facilities to clients also needs to be considered in advance as each client may request different products, which require different installations at the facilities. As always in such facility location problems, the distances between the future clients and facilities lead to significant transportation costs that need to be kept as low as possible. In this particular case, these distances depend on locations that are uncertain at the time planning decisions are made. Importantly, the distances are provided by the underlying road network (Melkote and Daskin 2001), which yields a graph-induced metric* $(\mathcal{M}, d)$.

Traditionally, robust optimization problems with an objective function that is concave in the uncertain parameters are reformulated as monolithic models using conic duality (Ben-Tal and Nemirovski 1998). These techniques do not readily extend to function $d(u_i, u_j)$ as the latter is non-concave in general. Actually, for Euclidean metric spaces based on the vector space $\mathbb{R}^\ell, \ell \in \mathbb{Z}^+$, $d(u_i, u_j) = \|u_i - u_j\|_2$ is convex in u_i and u_j. Function $\|u_i - u_j\|_2$ is closely related to the second-order cone (SOC) constraints considered by Zhen et al. (2021) for robust problems with polyhedral uncertainty sets. Zhen et al. (2021) linearize such robust SOC constraints by introducing adjustable variables, turning the problem into an adjustable robust optimization problem that can be tackled exactly (Ayoub and Poss 2016, Zhen et al. 2018, Zeng and Zhao 2013) or approximately using affine decision rules (Ben-Tal et al. 2004) or finite adaptability approaches (Bertsimas and Dunning 2016, Hanasusanto et al. 2015, Postek and Den Hertog 2016, Subramanyam et al. 2019), among others.

Interestingly, the approach of Zhen et al. (2021), extended in Roos et al. (2018) to more general convex functions, makes no particular assumption on the feasibility set of the decision variables, herein represented by the set $\mathcal{G}$. A second work closely related to ROBUST-Π is that of Citovsky et al. (2017), who rely on computational geometry techniques to provide constant-factor approximation algorithms in the special case where $\mathcal{G}$ contains all Hamiltonian cycles of $\mathbb{G}$. They propose in particular to solve a deterministic counterpart

of ROBUST-Π where the uncertain distances are replaced by the maximum pairwise distances $d_{ij}^{max} = \max_{u_i \in \mathcal{U}_i, u_j \in \mathcal{U}_j} d(u_i, u_j)$, for each $(i, j) \in \mathbb{V}^2, i \neq j$.

To summarize, we see that while Zhen et al. (2021) provide valuable tools for addressing problems defined in Euclidean metric spaces considering uncertainty polytopes, their approaches cannot be used for graph-induced metric spaces, such as those mentioned in Example 2. On the other hand, Citovsky et al. (2017) focused on the case where $\mathcal{G}$ contains all Hamiltonian cycles of $\mathbb{G}$. The main purpose of the present paper is thus to provide a more general solution algorithm that is valid for any set $\mathcal{G}$ and metric space $(\mathcal{M}, d)$. We only assume that $\mathcal{U}$ is finite, encompassing the two cases mentioned above. Specifically, when $(\mathcal{M}, d)$ is a Euclidean space, the distance is a convex function implying that maximizing over $\mathcal{U}$ is equivalent to maximizing over the polytope $\text{conv}(\mathcal{U}) = \times_{i \in \mathbb{V}} \text{conv}(\mathcal{U}_i)$. Then, in the case of graph-induced metrics, the set $\mathcal{M}$ is the set of nodes of a finite graph, meaning that each $\mathcal{U}_i \subseteq \mathcal{M}$ must be finite as well.

In this context, we can summarize our contributions as follows:

- We prove that ROBUST-Π is $\mathcal{NP}$-hard even when $\mathcal{G}$ consists of all $s-t$ paths and $(\mathcal{M}, d)$ is the one-dimensional Euclidean metric space or when $\mathcal{G}$ consists of all spanning trees of $\mathbb{G}$. These results illustrate how the nature of ROBUST-Π fundamentally differs from the classical min-max robust problem with cost uncertainty, which is known to be polynomially solvable whenever the costs lie in independent uncertainty sets (Aissi et al. 2009).

- We provide a general cutting-plane algorithm for ROBUST-Π that relies on integer programming formulations for $\mathcal{G}$. We further show that the separation problem EVAL-C is $\mathcal{NP}$-hard and provide two algorithms for computing EVAL-C. One is based on integer programming formulations while the other one relies on a dynamic programming algorithm that involves the threewidth of G.

- We leverage the above dynamic programming to provide a compact formulation for the problem when $\mathcal{G}$ contains only stars (or unions of stars). We can, in theory, extend that idea to trees, albeit presenting poor numerical performance.

- We propose a conservative approximation of the problem that uncouples $\mathcal{U}$ into its projections $\mathcal{U}_i$, $i \in \mathbb{V}$. In the case of Euclidean metric spaces, this approximation leads

to mixed-integer second-order conic reformulations, and turns out to be equivalent to the affine decision rule reformulation proposed by Zhen et al. (2021).

- We compare the exact cutting plane algorithm numerically with the above conservative approximation and simple deterministic reformulations. The benchmark is composed of two families of instances. The first family includes Steiner tree instances that illustrate subway network design. The second one is composed of strategic facility location instances.

The rest of the paper is structured as follows. Section 2 studies the hardness of EVAL-C and ROBUST-Π. In Section 3, we develop the exact solution algorithm. The latter involves a dynamic programming algorithm for trees, generalized to graphs of bounded threewidth in the appendix. Section 4 details the conservative reformulation. In Section 5, we present our numerical experiments. The appendix details the extension of the dynamic programming to graphs with bounded threewidth (Appendices A and B), the details of the compact formulations (Appendix C) for trees, and the equivalence between our conservative reformulation and that of Zhen et al. (2021) (Appendix D).

2. Hardness results

We study in this section the complexity of the optimization problems ROBUST-Π and EVAL-C. For the statements and proofs of the complexity results presented in this paper, we start by providing more formal definitions of the inputs of the optimization problems we consider.

2.1 Problem definitions

We are interested in the class $\mathcal{S}$ of deterministic combinatorial optimization problems that can be formulated as $\min_{G\in\mathcal{G}} c(u, G)$, where

$$c(u, G) = \sum_{\{i,j\}\in E[G]} d(u_i, u_j).$$

Any $\Pi \in \mathcal{S}$ represents a specific problem, such as the shortest path or the minimum spanning tree problem.

An instance of problem Π is defined by its input $I = (\mathbb{G}, \alpha, u, D)$, where $\mathbb{G}$ is the aforementioned graph, α contains problem-specific additional input, $u \in \mathcal{M}^n$ denotes the position of the nodes, and D is a (symmetric) matrix that provides the distance between each pair

$u, u' \in \bigcup_{i \in V[\mathbb{G}]} \{u_i\}$. Thus, $\mathcal{G}$ contains the subgraphs of $\mathbb{G}$ that satisfy the constraints specific to Π for input I. As an example, for the Shortest Path problem (SP), the additional input $\alpha = \{s, t\}$ consists of the origin and destination nodes, so the objective is to find an $s - t$ path P minimizing $c(u, P)$. In the Minimum Spanning Tree problem (MST), $\alpha = \emptyset$, and the objective is to find a spanning tree T minimizing $c(u, T)$.

For any problem $\Pi \in \mathcal{S}$, we define its robust counterpart ROBUST-Π. The input of an instance of ROBUST-Π is $I = (\mathbb{G}, \alpha, \mathcal{U}, \mathcal{D})$ where $\mathbb{G}$ and α are as before, $\mathcal{U}_i \subseteq \mathcal{M}$ contains the possible positions for node i, and $\mathcal{D}$ is a (symmetric) matrix recording the distance between each pair $u, u' \in \bigcup_{i \in V[\mathbb{G}]} \mathcal{U}_i$. The objective of ROBUST-Π is to find $G \in \mathcal{G}$ that minimizes $\max_{u \in \mathcal{U}} c(u, G)$. Observe that ROBUST-$\Pi$ is a generalization of Π as it corresponds to the case where $\sigma_i = 1$ for any i.

We are also interested in the problem EVAL-C, which corresponds to computing $c(G) = \max_{u \in \mathcal{U}} c(u, G)$ for a given graph G. An input of EVAL-C is $I = (G, \mathcal{U}, \mathcal{D})$ where G, $\mathcal{U}$ and $\mathcal{D}$ are as before.

For all these optimization problems, we define OPT(I) as the optimal solution cost of instance I. When clear from the context, we may also use the shorter notation OPT.

2.2 Problem ROBUST-Π

Let Π be the assignment problem. Any instance of Π involves a partition of $\mathbb{V}$ in two sets of equal size, denoted $\mathbb{V}^1$ and $\mathbb{V}^2$, such that any $G \in \mathcal{G}$ contains exactly $n/2$ edges covering $\mathbb{V}^1$ and $\mathbb{V}^2$. Because the edges belonging to any $G \in \mathcal{G}$ are disjoint, we have that

$$c(G) = \sum_{\{i,j\} \in E[G]} \max_{u_i \in \mathcal{U}_i, u_j \in \mathcal{U}_j} d(u_i, u_j) = \sum_{\{i,j\} \in E[G]} d_{ij}^{max}.$$

Hence, ROBUST-Π can be solved in polynomial time by solving the deterministic assignment problem defined by the partition $\mathbb{V}^1 \cup \mathbb{V}^2$ and the vector of weights d^{max}. In spite of this easy example, we show in this section that ROBUST-Π can in general not be reduced to Π as ROBUST-Π is typically harder than Π.

We illustrate the hardness of ROBUST-Π by focusing on the two polynomially solvable problems Π mentioned in Section 2.1, namely the shortest path problem SP and the minimum spanning tree problem MST. These problems have been largely studied in the robust combinatorial optimization literature under cost uncertainty (e.g. Kasperski and Zieliński

(2009), Yaman et al. (2001)), namely

$$\min_{G\in\mathcal{G}} \max_{\kappa\in\mathcal{K}} \sum_{\{i,j\}\in E[G]} \kappa_{ij}, \tag{1}$$

where $\mathcal{K}$ is a given uncertainty set included in the positive orthant. It is folklore (e.g., Aissi et al. (2009)) that when $\mathcal{K}$ is the Cartesian product of intervals, $\mathcal{K} = \times_{e\in E[G]}[\underline{\kappa}_e, \overline{\kappa}_e]$ for $\underline{\kappa}_e \leq \overline{\kappa}_e$, problem (1) can be reformulated as $\min_{G\in\mathcal{G}} \sum_{\{i,j\}\in E[G]} \overline{c}_{ij}$ making the robust problems as easy as their nominal counterparts. The result below shows that such is not the case for ROBUST-Π, as SP turns $\mathcal{NP}$-hard even in the simple case where each $\mathcal{U}_i$ is a subset of $\mathbb{R}$. Notice that in the 1-dimensional Euclidean space, the convexity of $d(u_i, u_j) = |u_i - u_j|$ implies that $\mathcal{U}_i$ is equivalent to a segment $[\underline{u}_i, \overline{u}_i]$, for some $\underline{u}_i \leq \overline{u}_i$.

Proposition 1. ROBUST-*SP is $\mathcal{NP}$-hard in the weak sense, even when $(\mathcal{M}, d)$ is the 1-dimensional Euclidean space.*

Proof. Given a set of integers $\{a_1, \ldots, a_n\}$, with $A = \sum_{i=1}^n a_i$, the $\mathcal{NP}$-complete decision problem partition asks for a subset $S \subset \{1, \ldots, n\}$ such that $\sum_{i\in S} a_i = A/2$. Let $K > 0$ be a large enough integer. The reduction considers the graph $\mathbb{G}$ with $2n + 2$ vertices and $4n$ edges as illustrated Figure 1; the regions $\mathcal{U}_i$ are translated away from vertex o for visibility. Specifically, our reduction locates vertices s and t at 0 while $\mathcal{U}_i = [-u_i^-, u_i^+]$ for each vertex i different from s and t. The definition of u^+ and u^- alternates along the vertices $v_i, v_{i+1}, v_{i+2}, \ldots$ and similarly for vertices w_i: for each $i = 2k + 1$, we define $u^+_{v_i} = K + a_i$, $u^-_{v_i} = K + \frac{A}{n} - a_i$, $u^+_{w_i} = K$ and $u^-_{w_i} = K + \frac{A}{n}$, while for each $i = 2k$, we define $u^+_{v_i} = K + \frac{A}{n} - a_i$, $u^-_{v_i} = K + a_i$, $u^+_{w_i} = K + \frac{A}{n}$ and $u^-_{w_i} = K$.

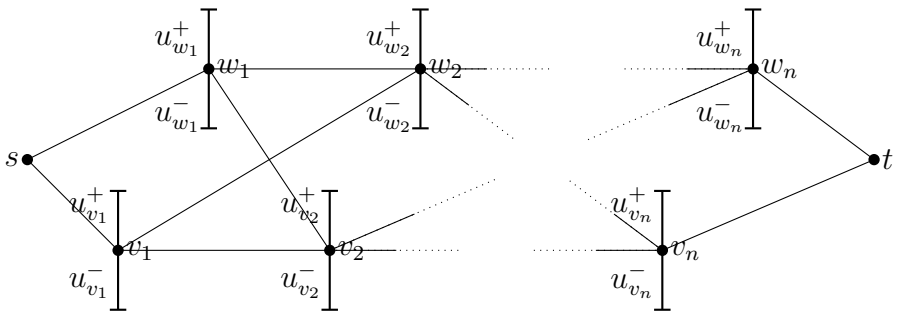


Figure 1: Reduction from partition when $\mathcal{U}$ is a Cartesian product of segments.

We first show that for K large enough, the worst-case $u \in \mathcal{U}$ for any path P from s to t alternates from the top of an interval to the bottom of the subsequent interval along the

path. To prove this, notice that for any vertex $v \in \mathbb{V} \setminus \{s,t\}$, $u_v^- \in [K-A, K+A]$ and $u_v^+ \in [K-A, K+A]$ and the same holds for any vertex w. Hence, if u alternates for the entire path, the resulting cost is not smaller than $c = 2n(K-A)$. On the contrary, if u misses one alternation, its cost cannot be greater than $c' = 2(n-1)(K+A) + 2A$. Hence, taking $K > 2nA$ ensures $c > c'$.

The reduction works as follow. Let $S \subseteq \{1,\dots,n\}$ be a subset of integers and $\overline{S}$ its complement. We associate to S the path P_S from s to t that contains v_i for each $i \in S$ and w_i for each $i \in \overline{S}$. From the above, only two scenarios in $\mathcal{U}$ must be considered in the worst-case and each vertex $i \in \{w_1, v_1, \dots, w_n, v_n\}$ contributes to the total length with either $2u_i^+$ or $2u_i^-$, depending on the scenario considered. We have

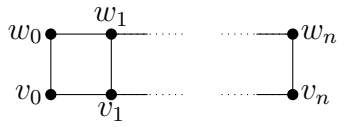
$$\begin{aligned} c(P_S) &= \max_{u\in\mathcal{U}} \sum_{\{i,j\}\in P_S} \|u_i - u_j\|_2 \\ &= 2\max\left(nK + \sum_{i\in S} a_i, n(K+\frac{A}{n}) - \sum_{i\in S} a_i\right) \\ &= 2\max\left(nK + \sum_{i\in S} a_i, A + nK - \sum_{i\in S} a_i\right) \\ &= 2nK + 2\max\left(\sum_{i\in S} a_i, \sum_{i\in\overline{S}} a_i\right). \end{aligned}$$

Hence, there exists a path $P_S \in \mathcal{G}$ with minimum cost of $2nK + A$ if and only if there exists a set S such that $\sum_{i\in S} a_i = \sum_{i\in\overline{S}} a_i = A/2$. □

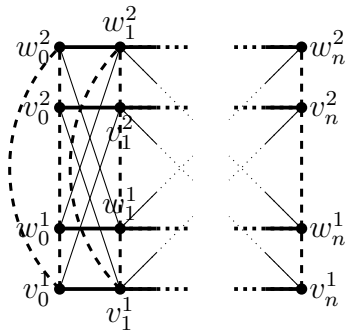
For MST we can prove the hardness of the problem only for a more general metric space.

Proposition 2. ROBUST-*MST is $\mathcal{NP}$-hard in the weak sense.*

Proof. We consider the same partition problem as in the proof of Proposition 1. Now $\mathbb{G}$ contains the $2n+2$ vertices and $3n+1$ edges depicted on Figure 2a. We consider the metric space $(\mathcal{M}, d)$ induced by the weighted graph $G_\mathcal{M} = (V_\mathcal{M}, E_\mathcal{M}, \omega)$ depicted on Figure 2b. Let $K > 0$ be a number large enough. The dashed edges and thin edges have their weights equal to K and $2K$, respectively, while $\omega_{v_{i-1}^1 v_i^1} = 3K + a_i$, $\omega_{v_{i-1}^2 v_i^2} = 3K + \frac{A}{n} - a_i$, $\omega_{w_{i-1}^1 w_i^1} = 3K$, and $\omega_{w_{i-1}^2 w_i^2} = 3K + \frac{A}{n}$ for each $i = 1,\dots,n$. The weight vector ω satisfies the triangle inequalities, so the metric d induced on $V_\mathcal{M}$ by the shortest paths in $G_\mathcal{M}$ satisfies $d_{ij} = \omega_{ij}$ for each $\{i,j\} \in E_\mathcal{M}$. Finally, we define $\mathcal{U}_{v_i} = \{v_i^1, v_i^2\}$ and $\mathcal{U}_{w_i} = \{w_i^1, w_i^2\}$ for $i = 0,\dots,n$.



(a) G



(b) $G_{\mathcal{M}}$.

Figure 2: Graphs used in the reduction for the minimum spanning tree problem.

We first observe that the cost of a vertical edge $\{v_i, w_i\}$ is equal to K for all positions of $(v_i, w_i) \in \mathcal{U}_{v_i} \times \mathcal{U}_{w_i}$. Let us consider any tree T in $\mathbb{G}$ that contains n^v vertical edges and n^h horizontal edges, where $n \leq n^h \leq 2n$. For K large enough, we claim that the worst-case $u \in \mathcal{U}$ locates all vertices either in the bottom layer of $G_{\mathcal{M}}$ that consists of vertices v_i^1 and w_i^1 for $i = 0, \ldots, n$, or in the top layer that consists of the remaining vertices. To prove the claim, notice that the weight of any horizontal edge in $G_{\mathcal{M}}$ is comprised between $3K - A$ and $3K + A$, while the weight of any diagonal edge is $2K$. Hence, if u locates all its vertices either in the bottom or in the top layer, the resulting cost is not smaller than $c = n^h(3K - A) + n^v K$. On the contrary, if u alternates at least once between the layers, its cost cannot be greater than $c' = (n^h - 1)(3K + A) + 2K + n^v K$. Hence, taking $K > (4n - 1)A \geq (2n^h - 1)A$ ensures $c > c'$, proving the claim.

We prove next that for K large enough, any optimal tree T in $\mathbb{G}$ must contain $n+1$ vertical edges and n horizontal ones. Following the above claim, the cost of a horizontal edge $\{v_i, v_{i+1}\}$ or $\{w_i, w_{i+1}\}$ for a worst-case $u \in \mathcal{U}$ is comprised between $3K - A$ and $3K + A$. Hence, any tree T with $n^h \in \{n+1, \ldots, 2n\}$ horizontal edges costs at least $c = n^h(3K - A) + n^v K$ while any tree having $n^h - 1$ horizontal edges costs at most $c = (n^h - 1)(3K + A) + (n^v + 1)K$. Hence, taking $K > 2nA \geq n^h A$ ensures $c > c'$, proving $n^h = n$ in any optimal solution.

As in the proof of Proposition 1, we let $S \subseteq \{1, \ldots, n\}$ be a subset of integers and $\overline{S}$ its complement. We associate to S the tree T_S that contains $\{v_{i-1}, v_i\}$ for each $i \in S$ and

$\{w_{i-1}, w_i\}$ for each $i \in \overline{S}$. Following the claim above, only two scenarios in $\mathcal{U}$ must be considered, and following again the reasoning used in the proof of Proposition 1, we have

$$c(T_S) = \max\left(3nK + \sum_{i\in S} a_i, 3nK + A - \sum_{i\in S} a_i\right) = 3nK + \max\left(\sum_{i\in S} a_i, \sum_{i\in\overline{S}} a_i\right).$$

Hence, there exists a spanning tree $T_S \in \mathcal{G}$ with minimum cost of $3nK + A/2$ if and only if there exists a set S such that $\sum_{i\in S} a_i = \sum_{i\in\overline{S}} a_i = A/2$. □

We detail in the remark below how, for any positive integer ℓ, the metric space $(\mathcal{M}, d)$ used in the proof of Proposition 2 cannot be embedded isometrically into $\mathbb{R}^\ell$. As a consequence, the hardness of ROBUST-MST in Euclidean spaces remains an open problem.

Remark 1. *The graph $G_\mathcal{M}$ described in the above proof cannot be embedded isometrically into an Euclidean space, as can be seen by considering the triangle $w_0^1\, w_1^1\, w_0^2$ and the fourth point v_0^2. The sides of the triangle have length $d(w_0^1, w_1^1) = 3K$, $d(w_1^1, w_0^2) = 2K$, and $d(w_0^2, w_0^1) = 2K$. Hence, since $d(v_0^2, w_0^1) = d(v_0^2, w_0^2) = K$, any isometric embedding maps v_0^2 to the midpoint of segment $w_0^1\, w_0^2$, so its Euclidean distance to w_1^1 must be $\sqrt{\frac{11}{2}}K$. This is in contradiction with $d(v_0^2, w_1^1) = \min(\omega_{v_0^2 w_0^1} + \omega_{w_0^1 w_1^1}, \omega_{v_0^2 v_1^2} + \omega_{v_1^2 w_1^1}) = \min(4K, 4K + \frac{A}{n} - a_1)$. The above illustrates that when $\mathcal{G}$ contains all spanning trees of $\mathbb{G}$, the complexity of* ROBUST-Π *is still open when one considers only Euclidean metric spaces.*

2.3 Problem EVAL-C

We now turn to the difficulty of computing the objective function EVAL-C. Our first result (Proposition 3 below) is that EVAL-C is hard, even when the metric space is reduced to two points, or the input graph is a clique. For this, we consider particularly simple metric spaces, and rely on a reduction from problem MAX-CUT. We recall that MAX-CUT is a famous problem in combinatorial optimization that, given any input graph G, seeks a partition $\{V_1, V_2\}$ of $V[G]$ such that $|\{e \in E[G] : |e \cap V_1| = 1\}|$ is maximized.

Proposition 3. *Even when $|\mathcal{M}| = 2$, there is no $\mathcal{PTAS}$ for* EVAL-C *unless $\mathcal{P} = \mathcal{NP}$.*

Proof. Let us denote the objective function of MAX-CUT as $c^{\text{MAX-CUT}}(V_1, V_2) = |\{e \in E[G] \mid |e \cap V_1| = 1\}|$. Further, we denote by $\text{OPT}^{\text{MAX-CUT}}(G)$ the value of an optimal solution for graph G. Given an input graph G of MAX-CUT, we define $\mathcal{M} = \{0, 1\}$ and I as the graph G itself, $\mathcal{U}_i = \mathcal{M}$ for any $i \in V[G]$, and the distance d by $d(x, y) = |x - y|$.

Given a solution $\{V_1, V_2\}$ (which is a partition) of MAX-CUT, we define $u_i = 0$ if $i \in V_1$, and 1 if $i \in V_2$. This implies $c(u, I) = c^{\text{MAX-CUT}}(V_1, V_2)$. For the reverse direction, given a solution u of EVAL-C, we define $V_1 = \{i \mid u_i = 0\}$ and $V_2 = V[G] \setminus V_1$, and we also have $c(u, I) = c^{\text{MAX-CUT}}(V_1, V_2)$.

The above immediatly implies that there is an S-reduction (see for instane Crescenzi (1997)) from MAX-CUT to EVAL-C, proving the result. □

Our second result assumes instead that G is a clique, and relies on a reduction from problem MAX-DIVERSITY. Given a ground set X, a natural number k and a diversity function $div : 2^X \to \mathbb{R}$, MAX-DIVERSITY seeks to output a subset $S \subseteq X$ of size k that maximizes $div(S)$. The special case of interest here, proved $\mathcal{NP}$-hard in Cevallos et al. (2018), considers that the ground set X is the three-dimensional Euclidean space, and that $div(S) = \sum_{i,j \in X} \|i - j\|_2$. We obtain immediately the following result.

Proposition 4. *Even when restricted to input graphs that are cliques, to metric space defined by squared Euclidean distance in the three-dimensional Euclidean space, and to instances where $\mathcal{U}_i = \mathcal{M}$ for all i,* EVAL-C *is $\mathcal{NP}$-hard.*

Let us now turn to parameterized complexity, and let tw be the treewidth of G, see Appendix A.1 for the formal definition of treewidth. Informally, tw measures the thickness of a tree structure defining G. In particular, $tw(G) = 1$ for any tree G. As we show in the next section that computing EVAL-C is polynomial on trees, a natural question is to determine if we can extend this result by proving that EVAL-C/tw admits an $\mathcal{FPT}$ algorithm (where EVAL-C/tw denotes problem EVAL-C parameterized by tw, as defined in Appendix A.2), meaning an algorithm running in $f(tw) \cdot |I|^c$ for some computable function f and constant c. The following proposition implies that it is very unlikely, and thus places EVAL-C with the few problems that are not $\mathcal{FPT}$ by treewidth.

Proposition 5. EVAL-C/tw *is $\mathcal{W}[1]$-hard.*

Proof. Given a graph G, and a set of integers (called colors) $L(i)$ for any $i \in V[G]$, problem LIST-COL aims at deciding whether we can find a color $f(i) \in L(i)$ for any $i \in V[G]$ such that for any edge $\{i, j\} \in E[G]$, $f(i) \neq f(j)$. It is known Fellows et al. (2011) that LIST-COL/tw is $\mathcal{W}[1]$-hard. Let us now prove that there is a parameterized reduction from LIST-COL/tw to EVAL-C/tw, which implies (see Appendix A.2) that EVAL-C/tw is $\mathcal{W}[1]$-hard.

Given a graph G a list of colors $L(i)$ for any $i \in V[G]$, we define $\mathcal{M} = \bigcup_{i \in V[G]} L(i)$, and $d(c_1, c_2) = 0$ is $c_1 = c_2$, and 1 otherwise. We define the uncertainty set of G as follows: for any i, we let $\mathcal{U}_i = L(i)$. It is now straightforward to verify that we have a YES-instance of LIST-COL if and only if $c(G) = m$. As the reduction can be computed in polynomial time, and the graph (and thus its treewidth) is unchanged, this provides a parameterized reduction, and we get the desired result. □

3. Exact solution of ROBUST-Π

A popular type of algorithms solving exactly difficult robust optimization problems replaces the large uncertainty set by an approximation of small cardinality, leading to a relaxation of the original problem. Then, these algorithms iterate between solving integer programming formulations for the robust problem with small uncertainty set, and checking the optimality of the solution for the relaxation by solving an adversarial separation problem. This process leads to cutting plane algorithms (e.g., Bertsimas et al. (2016), Fischetti and Monaci (2012), Naoum-Sawaya and Buchheim (2016)). Such algorithms involve frequent calls to computing the objective function EVAL-C, so we start this section by studying how to solve this problem. Then, we detail in Section 3.2 the overall cutting plane algorithm for ROBUST-Π.

3.1 Solving the cost evaluation problem

Given the hardness results from the previous section, we propose two approaches to computing EVAL-C that have non-polynomial running times in general. The first approach relies on an integer programming formulation. For each $i \in \mathbb{V}$ and $k \in \{1, \dots, \sigma_i\}$, binary variable y_i^k takes value 1 if and only if vertex i is located at position u_i^k. Therefore, EVAL-C is equal to

$$\begin{aligned} \max \quad & \sum_{\{i,j\} \in E[G]} \sum_{k=1}^{\sigma_i} \sum_{\ell=1}^{\sigma_j} d(u_i^k, u_j^\ell) y_i^k y_j^\ell \\ \text{s.t.} \quad & \sum_{k=1}^{\sigma_i} y_i^k = 1, \quad \forall i \in V[G] \\ & y_i \in \{0,1\}^{\sigma_i}, \quad \forall i \in V[G] \end{aligned}$$

which can be linearized using classical techniques.

It is also possible to compute EVAL-C efficiently whenever G has small treewidth $tw(G)$ using a dynamic programming algorithm. Let us detail the algorithm whenever G is a tree

rooted at vertex r, which we assume oriented from r to its leaves L. We denote by $D(i)$ the set that contains the direct descendants of i, which is empty if i is a leaf. Let $\text{OPT}(i, u_i)$ be the maximum value obtained for the subtree starting at i given that node i is located at u_i. We obtain the following recursion:

$$\text{OPT}(i, u_i) = \begin{cases} \sum\limits_{j \in D(i)} \max\limits_{u_j \in \mathcal{U}_j} d(u_i, u_j) + \text{OPT}(j, u_j), & i \in V[G] \setminus L \\ 0, & i \in L \end{cases} \tag{2}$$

and the optimal solution cost is given by $\max_{u_r \in \mathcal{U}_r} \text{OPT}(r, u_r)$. Dynamic programming recursion (2) will be used in our numerical experiments, which involve trees and stars.

Recall that $tw = tw(G)$ and let us further denote $\sigma = \max_{i \in V[G]} \sigma_i$. Using dynamic programming on a weel-chosen tree decomposition of G (see Appendix A.1 for the definition), one can readily extend the above idea to any graph of bounded treewidth, leading to Theorem 1, whose proof is deferred to Appendix B. We point out that according to Proposition 5 we cannot (unless $\mathcal{W}[1] = \mathcal{FPT}$) remove the dependency in σ to get for example a $\mathcal{O}(poly(n) \times f(tw))$, and this holds for any computable function f.

Theorem 1. EVAL-C$/tw + \sigma$ *is* $\mathcal{FPT}$. *More precisely, we can compute an optimal solution of* EVAL-C *in time* $\mathcal{O}(n \times tw \times \sigma^{\mathcal{O}(tw)})$.

3.2 Cutting plane algorithm for the robust problem

Now that we have depicted numerical methods for computing EVAL-C, we wish to make the extra step towards the exact solution of the complete problem, ROBUST-Π. For this, we design an exact solution algorithm that generates scenarios of $\mathcal{U}$ on the fly in the course of a branch-and-cut algorithm. We thus choose to associate to each $G \in \mathcal{G}$ the binary vector $x^G \in \{0,1\}^m$ such that $x^G_{ij} = 1$ if and only if $\{i, j\} \in E[G]$. As a result, the counterpart of $\mathcal{G}$ is the set of binary vectors $\mathcal{X} = \{x^G, G \in \mathcal{G}\}$. With these notations, ROBUST-Π can be reformulated as

$$\min_{x \in \mathcal{X}} \max_{u \in \mathcal{U}} \sum_{\{i,j\} \in \mathbb{E}} x_{ij} d(u_i, u_j). \tag{3}$$

Let $\widetilde{\mathcal{U}}$ be a finite subset of $\mathcal{U}$. An exact algorithm for (3), described in Algorithm 1, relies on the following relaxed formulation

$$\min \left\{ \omega \;\middle|\; \omega \geq \sum_{\{i,j\} \in \mathbb{E}} x_{ij} d(u_i, u_j),\ \forall u \in \widetilde{\mathcal{U}}, x \in \mathcal{X} \right\}. \tag{4}$$

Algorithm 1 describes an iterative cutting-plane implementation, alternating between the solution of the relaxed master problem (4) and the adversarial separation problem EVAL-C. Practical implementation of these algorithms typically rely instead on branch-and-cut algorithms, where the adversarial separation problem is solved at each integer node of the branch-and-bound-tree.

Algorithm 1: Cutting-plane algorithm for ROBUST-Π

repeat
  Let $(\tilde{\omega}, \tilde{x})$ be an optimal solution of (4)
  Let G be the graph induced by $\tilde{x}$
  Compute $c(G) = \max_{u \in \mathcal{U}} \sum_{\{i,j\} \in E[G]} d(\tilde{u}_i, \tilde{u}_j)$ and let $\tilde{u}$ be a maximizer
  if $c(G) > \tilde{\omega}$ **then** $\widetilde{\mathcal{U}} \leftarrow \widetilde{\mathcal{U}} \cup \{\tilde{u}\}$
until $c(G) \leq \tilde{\omega}$
return G

3.3 Compact formulation for stars

Depending on the structure of the elements of $\mathcal{G}$, the dynamic programming recursion (2) naturally leads to a compact formulation for the problem. We detail next this idea for the case where any $G \in \mathcal{G}$ is a union of disjoint stars rooted at the elements of a known set $R \subseteq \mathbb{V}$. For each $r \in R$ and $u_r^k \in \mathcal{U}_r$, let us introduce the optimization variable z_r^k to model $\text{OPT}(r, u_r^k)$, the cost of the star rooted at r given that $u_r = u_r^k$. Let $N(i)$ be the set of neighbours of any node $i \in \mathbb{V}$. Plugging variables z and x into (2) and noticing that any node connected to r must be a leaf, we immediately obtain

$$z_r^k = \sum_{j \in N(r)} x_{rj} \max_{u_j \in \mathcal{U}_j} d(u_r^k, u_j). \tag{5}$$

Notice that the maximization appearing in the right-hand-side of (5) does not involve optimization variables, se we can define the constant $d_{rkj}^{max} = \max_{u_j \in \mathcal{U}_j} d(u_r^k, u_j)$. Then, introducing z_r as the worst-case cost of the star rooted at r, we have

$$z_r = \max_{k \in [\sigma_r]} z_r^k = \max_{k \in [\sigma_r]} \sum_{j \in N(r)} x_{rj} d_{rkj}^{max}. \tag{6}$$

Overall, we wish to minimize the sum of z_r over all $r \in R$. Reformulating (6) through an epigraphic reformulating, we obtain

$$\begin{aligned} \min \quad & \omega \\ \text{s.t.} \quad & \omega \geq \sum_{r \in R} z_r \\ & z_r \geq \sum_{j \in N(r)} x_{rj} d^{max}_{rkj}, \quad \forall u_r \in \mathcal{U}_r, k \in [\sigma_r] \\ & x \in \mathcal{X}, z \geq 0 \end{aligned}$$

The above construction can, in theory, be extended to trees. However, in that case the recurrence relations lead to products between variables, which turns out to be inefficient numerically. See Appendix C for details.

4. Conservative approximation

We introduce next a (conservative) approximation of ROBUST-Π that leads to compact formulations. Let us introduce an additional optimization variable $\mu_e \in \mathcal{M}$ for each $e \in \mathbb{E}$, and consider the following optimization problem

$$\min_{\substack{x \in \mathcal{X} \\ \mu \in \mathcal{M}^{|\mathbb{E}|}}} \max_{u \in \mathcal{U}} \sum_{\{i,j\} \in \mathbb{E}} x_{ij}(d(u_i, \mu_{ij}) + d(\mu_{ij}, u_j)). \tag{CONS-Π}$$

Remark 2. *Due to the triangle inequalities, the optimal solution cost of* CONS-Π *is not smaller than the optimal solution cost of* ROBUST-Π*, so* CONS-Π *is a conservative approximation of* ROBUST-Π*.*

We show next how to reformulate CONS-Π as a discrete optimization problem featuring a polynomial number of variables. Noticing that

$$\sum_{\{i,j\} \in \mathbb{E}} x_{ij}(d(u_i, \mu_{ij}) + d(\mu_{ij}, u_j)) = \sum_{i \in \mathbb{V}} \sum_{\{i,j\} \in \mathbb{E}} x_{ij} d(u_i, \mu_{ij}),$$

we obtain

$$\max_{u \in \mathcal{U}} \sum_{\{i,j\} \in \mathbb{E}} x_{ij}(d(u_i, \mu_{ij}) + d(\mu_{ij}, u_j)) = \max_{u \in \mathcal{U}} \sum_{i \in \mathbb{V}} \sum_{\{i,j\} \in \mathbb{E}} x_{ij} d(u_i, \mu_{ij}) = \sum_{i \in \mathbb{V}} \max_{u_i \in \mathcal{U}_i} \sum_{\{i,j\} \in \mathbb{E}} x_{ij} d(u_i, \mu_{ij}).$$

Thus, we can introduce an additional variable d_i for each node $i \in \mathbb{V}$, so CONS-Π can be reformulated as

$$\min \quad \sum_{i \in \mathbb{V}} d_i \tag{7}$$

$$\text{s.t.} \quad d_i \geq \sum_{\{i,j\} \in \mathbb{E}} x_{ij} d(u_i, \mu_{ij}), \quad \forall i \in V, u_i \in \mathcal{U}_i \tag{8}$$

$$x \in \mathcal{X},\ \mu \in \mathcal{M}^{|\mathbb{E}|}. \tag{9}$$

The interest of the above reformulation is that the uncertainty sets $\mathcal{U}_i$, $i \in \mathbb{V}$, appear in distinct constraints, so (8) contains $\sum_{i \in \mathbb{V}} \sigma_i$ constraints, which is significantly smaller than the $\prod_{i \in \mathbb{V}} \sigma_i$ elements in the global uncertainty set $\mathcal{U}$. In practice, the numerical difficulty of problem (7)–(9) typically depends on the considered metric space $(\mathcal{M}, d)$ and feasibility set $\mathcal{X}$. For instance, using ad-hoc pre-processing rules, we may be able to reduce the domain of each variable μ_{ij} to a small subset of $\mathcal{M}_{ij} \subset \mathcal{M}$. These rules may not even need to be exact as problem (7)–(9) is only a conservative approximation of the original problem ROBUST-Π.

In what follows, we further develop the case where $(\mathcal{M}, d)$ is the p-dimensional Euclidean space so the distance $d(u_i, u_j) = \|u_i - u_j\|_2$ is now well-defined for any $u_i, u_j \in \mathbb{R}^p$. We can leverage this to relax the discrete restriction $\mu \in \mathcal{M}^{|\mathbb{E}|}$ to $\mu \in \mathbb{R}^{p \times |\mathbb{E}|}$, obtaining

$$\min \quad \sum_{i \in \mathbb{V}} d_i \tag{10}$$

$$\text{s.t.} \quad d_i \geq \sum_{\{i,j\} \in \mathbb{E}} x_{ij} \|u_i - \mu_{ij}\|_2, \quad \forall i \in \mathbb{V}, u_i \in \mathcal{U}_i \tag{11}$$

$$x \in \mathcal{X},\ \mu \in \mathcal{M}^{|\mathbb{E}|}. \tag{12}$$

The non-linearities in constraints (11) can be avoided by replacing $x_{ij}\|u_i - \mu_{ij}\|_2$ with $\|x_{ij}u_i - \mu_{ij}\|_2$: if $x_{ij} = 1$, both expressions coincide; otherwise, $x_{ij} = 0$, and setting $\mu_{ij} = 0$ also yields equality. Introducing additional variables to isolate each norm into a unique constraint, we finally obtain the following mixed-integer second-order cone programming formulation

$$\min \quad \sum_{i \in \mathbb{V}} d_i \tag{13}$$

$$\text{s.t.} \quad d_i \geq \sum_{e \in \mathbb{E}} \sum_{i \in e} \nu_{i,e}^k \tag{14}$$

$$\nu_{i,e}^k \geq \|x_e u_i^k - \mu_e\|_2, \quad \forall e \in \mathbb{E}, i \in e, k \in [\sigma_i] \tag{15}$$

$$x \in \mathcal{X},\ \mu \in \mathcal{M}^{|\mathbb{E}|}. \tag{16}$$

We conclude this section by mentioning that (13)–(16) can be alternatively obtained by following the approach proposed in Zhen et al. (2021). Specifically, let us recall the epigraphic reformulation of ROBUST-Π

$$\min \quad \omega \tag{17}$$

$$\text{s.t.} \quad \omega \geq \sum_{\{i,j\}\in\mathbb{E}} x_{ij}\|u_i - u_j\|_2, \quad \forall u \in \mathcal{U} \tag{18}$$

$$x \in \mathcal{X}. \tag{19}$$

Then, we detail in Appendix D how each constraint (18) can be reformulated by introducing recourse variables which, approximated through affine decision rules, leads exactly to (13)–(16). This connection underlines that the difference between the optimal solution costs of ROBUST-Π and CONS-Π can be interpreted as the suboptimality of affine decision rules for approximating two-stage robust optimization. It also suggests that stronger conservative approximations could be obtained by using more expressive decision rules, such as the lifted affine decision rules proposed by de Ruiter and Ben-Tal (2017).

5. Computational experiments

In this section, we compare numerically the exact algorithm from Section 3, denoted **exact** hereafter, with three heuristic algorithms that solve deterministic counterparts of ROBUST-Π. Namely, each of these heuristics considers a symmetric function $\hat{d} : V \times V \to \mathbb{R}_+$ and returns the optimal solution of $\min_{G\in\mathcal{G}} \sum_{\{i,j\}\in E[G]} \hat{d}_{ij}$. Three such functions $\hat{d}$ are considered:

worst: $\hat{d}_{ij} = \max\limits_{u_i\in\mathcal{U}_i, u_j\in\mathcal{U}_j} d(u_i, u_j)$, as suggested by Citovsky et al. (2017).

center: $\hat{d}_{ij} = d(\beta_i, \beta_j)$, where β_i is any *geometric median* of $\mathcal{U}_i$, defined as

$$\beta_i \in \arg\min_{u\in\mathcal{M}} \sum_{u'\in\mathcal{U}_i} d(u, u').$$

avg: $\hat{d}_{ij} = \frac{1}{\sigma_i\sigma_j} \sum\limits_{u_i\in\mathcal{U}_i, u_j\in\mathcal{U}_j} d(u_i, u_j)$.

We also include in our numerical assessment the conservative approximation depicted in Section 4, and denoted **cons**. We compare these algorithms on the applications mentioned in the introduction: a subway network design problem, modeled as a Steiner tree problem (STP), and a simple plant location problem (SPL). Since the applications involve stars and

trees, the separation problems of **exact** can be solved using the dynamic programming recurrence presented in (2).

The purpose of our experiments is two-fold. First and foremost, we wish to assess the numerical efficiency of the exact solution algorithm in terms of solution times. Second, we measure the approximation ratios obtained by the heuristic algorithms, by comparing the cost of their solutions to the optimal solution costs.

The algorithms have been coded in Julia (Bezanson et al. 2012), using JuMP (Dunning et al. 2017) to interface the mixed integer linear programming (MILP) solver CPLEX. They have been carried out on a processor Intel(R) Core(TM) i7-10510U CPU1.80GHz using up to 4 threads in parallel and with a total running time limit of 2 hours. The source code of every algorithm is publicly available at `https://github.com/mjposs/locational_uncertainty`.

5.1 Steiner tree problem

We consider the problem of expanding the subway network of a city, modeled as a Euclidean Steiner tree problem. The compulsory points model the future stops of the subway, while the other points model the possible knickpoints of the lines. We thus consider an undirected graph $\mathbb{G} = (\mathbb{V}, \mathbb{E})$ where $T \subseteq \mathbb{V}$ denotes the set of compulsory vertices; we consider an arbitrary root $r \in T$ and set $T_0 = T \setminus \{r\}$. Set $\mathcal{G}$ thus contains all trees of $\mathbb{G}$ that cover the vertices of T. Sets $\mathcal{U}_i \subseteq \mathbb{R}^2$ model the possible locations for the vertices, which we assume to be polyhedral sets, and we assume that the distance $d(u_i, u_j) = \|u_i - u_j\|_2$ is the Euclidean distance.

We consider the classical disaggregated MILP formulation for the problem involving two sets of variables (Magnanti and Wong 1984). For each undirected edge $\{i, j\} \in \mathbb{E}$, binary variable x_{ij} takes value 1 if and only if the edge is used. Then, for each $t \in T_0$ and $e = \{i, j\} \in \mathbb{E}$, the fractional variable f_{ij}^t decides how much flow related to t is sent on the directed arc (i, j). Let $\mathbb{E}^{bidir}$ be the set of directed edges obtained from E by including the two opposite edges (i, j) and (j, i) for each undirected edge $\{i, j\} \in \mathbb{E}$. Defining the incoming and outgoing stars at node i as $\delta^-(i) = \left\{ j \;\middle|\; (j, i) \in \mathbb{E}^{bidir} \right\}$ and $\delta^+(i) = \left\{ j \;\middle|\; (i, j) \in \mathbb{E}^{bidir} \right\}$, respectively, and the balance of vertex i as $b_i^t = 0$ for $i \in T_0 \setminus \{t\}$, $b_r^t = -1$ and $b_t^t = 1$, we

obtain

$$\begin{aligned}
\min \left(\max_{u \in \mathcal{U}} \sum_{\{i,j\} \in \mathbb{E}} x_{ij} \|u_i - u_j\|_2 \right) \\
\text{s.t.} \sum_{(j,i) \in \delta^-(i)} f^t_{ji} - \sum_{(i,j) \in \delta^+(i)} f^t_{ij} = b^t_i, \quad \forall i \in \mathbb{V}, t \in T_0 \\
f^t_{ij} + f^t_{ji} \leq x_{ij}, \quad \forall \{i,j\} \in \mathbb{E}, t \in T_0 \\
f \geq 0, \ x \text{ binary}
\end{aligned}$$

5.1.1 Instances

We assess the different solution algorithms on the instances P6E with 100 vertices and 5 terminals (p619, p620, and 621) that are publicly available at `http://steinlib.zib.de/testset.php`. Each of these instances has 180 edges. The position of the vertices, denoted $\bar{u}_i$ hereafter, are not available in the data files P6E, so we estimate them using a variant of the MDS-MAP algorithm from Shang et al. (2003). Specifically, we apply classical multidimensional scaling (MDS) from the Julia package MultivariateStats (see `https://github.com/JuliaStats/MultivariateStats.jl`) to compute the positions $\bar{u}$ from the distances, completing the distance matrix with the shortest path values. The uncertainty sets $\mathcal{U}_i, i \in \mathbb{V}$ are then computed randomly based on two parameters: Δ that scales the diameter of each set $\mathcal{U}_i$, and σ the common number of elements of all $\mathcal{U}_i, i \in \mathbb{V}$. To be more precise, we first compute the average distance among pairs of points in $\mathbb{V}$, $\bar{d} = \sum_{i<j} \frac{\|\bar{u}_i - \bar{u}_j\|}{n(n-1)/2}$. For each $i \in \mathbb{V}$, we then uniformly draw one random value in $\rho_i \in [0, \Delta \cdot \bar{d}]$ and define the circle C_i of center $\bar{u}_i$ and radius ρ_i. Then, we take σ equidistant points on C_i, yielding

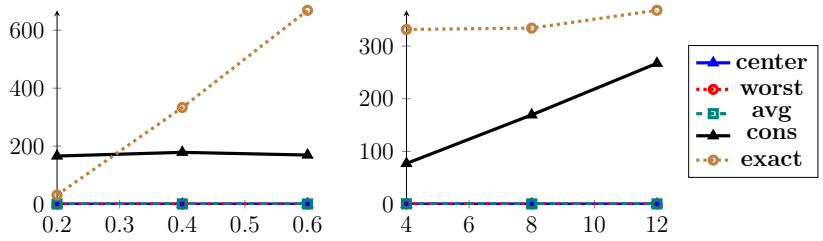
$$\mathcal{U}_i = \left\{ \left(\bar{u}_{i1} + \rho_i \cos\left(\frac{2k\pi}{\sigma}\right), \bar{u}_{i2} + \rho_i \sin\left(\frac{2k\pi}{\sigma}\right) \right), k = 1, \ldots, \sigma \right\}.$$

Following the above procedure, we create 5 random instances for each P6E instance and choice of parameters, yielding 135 instances in total.

5.1.2 Results

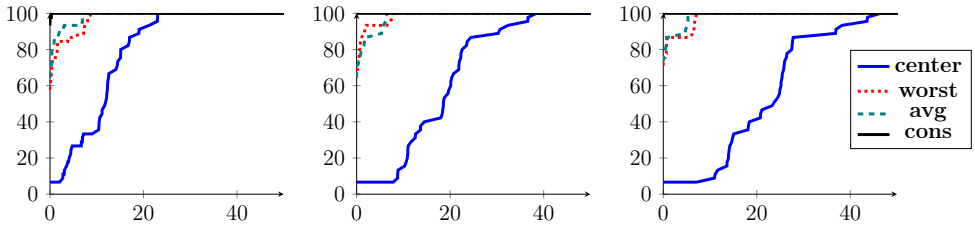
Figure 3 reports the average solution times, illustrating the impact of the dimension of the diameters of the uncertainty sets, represented by Δ, and the number of elements in each set, given by σ. The figure illustrates that, unsurprisingly, the three deterministic counterparts are solved much faster than **cons** and **exact**. More interesting is the fact that **exact** is faster than the heuristic algorithm **cons** when Δ is small. However, the difficulty of solving

exact grows rapidly with the value of Δ. Notice also that three instances based on p621, corresponding to $\Delta = 0.6$, could not be solved to exact optimality within two hours, ending with optimality gaps of 1%, 5%, and 7%, respectively. Hence, the value reported on Figure 3a for $\Delta = 0.6$ is actually a lower bound for the true average value.



(a) Varying Δ (b) Varying σ

Figure 3: STP: Average solution times in seconds on instances P6E for each algorithm when varying one of the parameters.



(a) P6E, $\Delta = 0.2$ (b) P6E, $\Delta = 0.4$ (c) P6E, $\Delta = 0.6$

Figure 4: STP: % of instances for which the additional relative cost is less than x.

Figure 4 reports the cumulative distributions of the cost increase for each of the four heuristic algorithms, relatively to the cost of the exact solution. Formally, let $z(H)$ denote the cost of the solution returned by $H \in \{\textbf{center}, \textbf{worst}, \textbf{avg}, \textbf{cons}\}$ and z^* denote the optimal solution cost. For each $H \in \{\textbf{center}, \textbf{worst}, \textbf{avg}, \textbf{cons}\}$, the corresponding curve reports

$$f(x) = 100 \frac{\#\{\text{instances for which } z(H) \leq (1+x) \cdot z^*\}}{\#\{\text{all instances}\}}. \tag{20}$$

These results show that **worst**, **avg** and **cons** provide solutions with values very close to the optimal one, with **cons** being the best of the three, always leading to the optimal solution. In contrast, the quality of **center** becomes rather poor as Δ increases, ranging up to an extra cost 50% for some of these instances, and with nearly half of the instances with $\Delta = 0.6$ having an extra cost of at least 20%.

Remark 3. *The optimality of the solutions returned by* ***cons*** *that is displayed on Figure 4 means that optimizing along function*

$$c^{\boldsymbol{cons}}(x) = \min_{\mu} \max_{u \in \mathcal{U}} \sum_{\{i,j\} \in \mathbb{E}} x_{ij}(d(u_i, \mu_{ij}) + d(\mu_{ij}, u_j))$$

returns the same optimal solution as optimizing along the true objective

$$c(x) = \max_{u \in \mathcal{U}} \sum_{\{i,j\} \in \mathbb{E}} x_{ij} d(u_i, u_j).$$

It does not mean, however, that the conservative approximation (or, equivalently, the affine decision rule approximation, ass discussed in Appendix D) is exact in this case. Specifically, looking at the detailed results reveals that $c^{\boldsymbol{cons}}(x^*) > c(x^*)$ *for the optimal solution* x^* *returned by* ***cons****.*

5.2 Simple plant location

We consider a strategic facility location problem where the exact location of the facility may be perturbed due to local political and technical considerations, while the exact position of the clients themselves is subject to uncertainty (Correia and da Gama 2015). The distances between the facilities and the clients are computed from the shortest path distance on a weighted graph that represents the underlying road network. The problem can then be modeled with the weighted graph $\mathbb{G} = (\mathbb{V}, \mathbb{E}, l)$, the vertices of which represent the possible locations for the facilities and clients, while each edge and its weight represent the existence of a road between two vertices together with its length. The metric is induced by graph $\mathbb{G}$, so $\mathcal{M} = \mathbb{V}$ and $d(u, v)$ is equal to the shortest path between u and v for every $u, v \in V$.

Let $I \subseteq \mathbb{V}$ and $J \subseteq \mathbb{V}$ represent the set of clients and possible locations for the facilities. We consider the problem of choosing p facilities among J and assigning every client to its closest facility so as to minimize the total assignment cost. For each $j \in J$, let y_j be a binary variable indicating whether a facility is located at j, and for each $i \in I, j \in J$, let $x_{ij} \in \{0, 1\}$ indicate whether client i is assigned to facility j. The robust problem can then be formulated

as

$$\begin{aligned}
\min & \left(\max_{u \in \mathcal{U}} \sum_{i \in I, j \in J} x_{ij} d(u_i, u_j) \right) \\
\text{s.t.} & \sum_{j \in J} x_{ij} = 1, \quad \forall i \in I \\
& x_{ij} \leq y_j, \quad \forall i \in I, j \in J \\
& \sum_{j \in J} y_j = p \\
& x, y \text{ binary}
\end{aligned}$$

5.2.1 Instances

We construct the graph $\mathbb{G} = (\mathbb{V}, \mathbb{E}, l)$ as follows. For each vertex i, we generate its position u_i uniformly in the square $[0, 1]^2$ and we select edges so that the resulting graph is planar and connected and shorter edges are more likely to appear. This procedure allows to mimic real transportation networks (Daskin 1993). More precisely, we first compute a minimum cost spanning tree based on the weights $\{w_{ij} = \|u_i - u_j\|_2^{-2}\}$ to ensure the graph is connected. Then, we iteratively select $m - n + 1$ additional edges following the probability distribution $Prob_{ij} = \frac{w_{ij}}{\sum_{\{i',j'\}} w_{i'j'}}$ for each $i \neq j \in \mathbb{V}$ while ensuring the resulting graph is planar. The length l_{ij} of each edge $\{i, j\} \in \mathbb{E}$ is then given by $\|u_i - u_j\|_2$ and the the distance between every pair of vertices is given by the shortest path between them in G. For each $i \in \mathbb{V}$, we define $\mathcal{U}_i$ as the σ vertices that are closest to i. Finally, I is defined as a random subset of $\mathbb{V}$ such that $\mathcal{U}_i \cap \mathcal{U}_{i'} = \emptyset$ for each $i, i' \in I$ and $J = \mathbb{V} \setminus I$. Following the above procedure, we create 2 random instances for each choice of parameters n, m, σ and $|I|$, leading to 486 instances.

5.2.2 Results

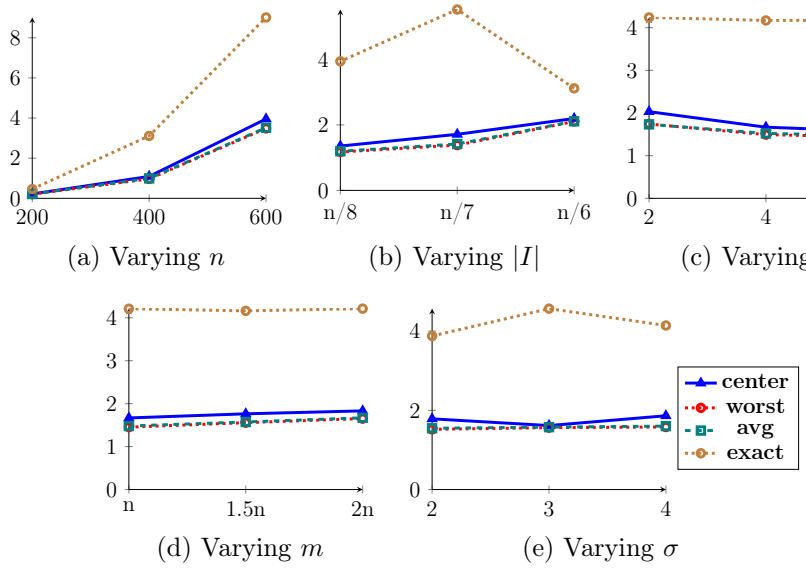
Figure 5 reports the average solution times, showing that **exact** is able to solve every instance to optimality within a few seconds, being roughly twice slower than the heuristic algorithms. The figure further underlines that n is the parameter having the strongest impact on the solution time. This was expected given that larger values for n imply more elements in I and J, and therefore, larger models. The charts presented for the remaining 4 parameters do not lead to clear conclusions.

Heuristic **cons** is not included in the comparison because its efficiency strongly depends on the definition of sets $\mathcal{M}_{ij}$, as discussed in Section 4. While defining $\mathcal{M}_{ij} = \mathcal{M}$ is likely

to be intractable, it would lead to the tightest bounds. To obtain a good trade-off between quality and time, one should come up with ad-hoc sets $\mathcal{M}_{ij} \subset \mathcal{M}$ obtained through heuristics that would be tailored to the specific instances used. This is beyond the scope of the current paper, which aims at proposing general methods rather than ad-hoc algorithms for specific data sets.

Then, following again formula (20), Figure 6 reports the cumulative distributions of the cost increase of each of the three deterministic heuristics, relatively to the cost of the exact solution. The results focus only on the parameters having an impact on the resuting costs, namely n and σ. They illustrate that **avg** is the best approximation, followed closely by **worst** and **center**. They also show that **center** behaves worse for small instances and those having larger uncertainty sets.

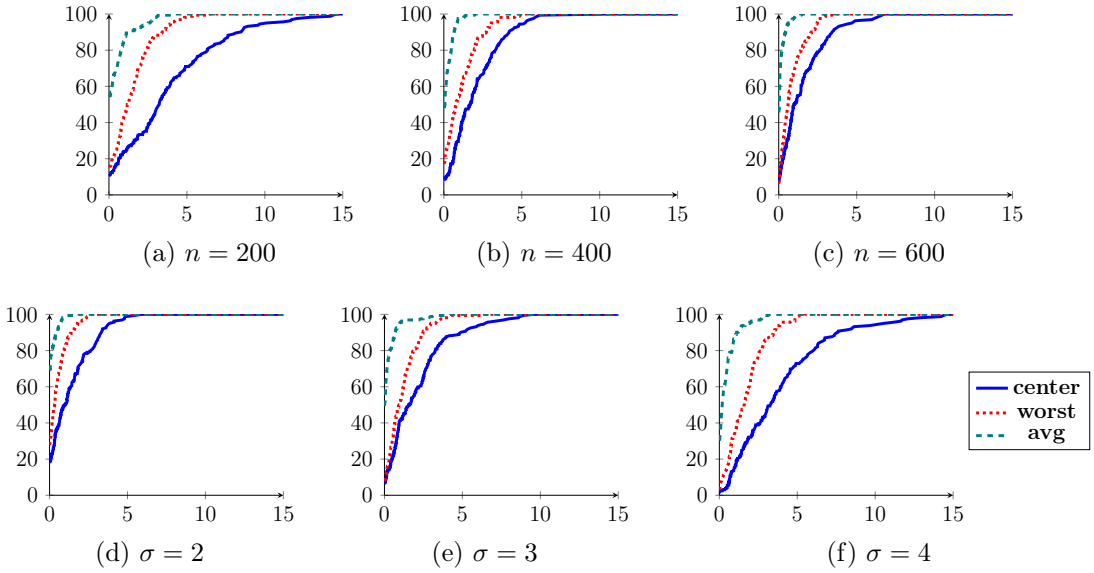
Overall, these results illustrate that given the quick solution times of **exact** due to the compact formulation presented in Section 3.3, heuristic algorithms do not seem necessary for obtaining good solutions to this problem. Yet, if one wishes to reduce further the solution times, **avg** should be preferred over **worst** and **center**.



(a) Varying n (b) Varying $|I|$ (c) Varying p

(d) Varying m (e) Varying σ

Figure 5: SPL: Average solution times in seconds for each algorithm when varying one of the parameters.



(a) $n = 200$ (b) $n = 400$ (c) $n = 600$

(d) $\sigma = 2$ (e) $\sigma = 3$ (f) $\sigma = 4$

Figure 6: SPL: For each heuristic algorithm, the curve plots (20), the cumulative distribution of the % of instances for which the returned solution has an additional (relative) cost less than the value of the abscissa.

6. Concluding remarks

This paper has been devoted to the study of general combinatorial optimization problems defined in spatial graphs with locational uncertainty, thus encompassing applications arising in transportation and facility location, among others. After proving the $\mathcal{NP}$-hardness of these problems, we have developed an exact solution algorithm based on scenario generation. The bottleneck of this algorithm lies in the separation problem, so we have studied in depth the complexity of that problem, also proposing an integer programming formulation. We have also proposed a conservative approximation that turns out to be equivalent to the affine decision rules approximation by Zhen et al. (2021) in the case of Euclidean distances. We have compared these algorithms numerically to different deterministic approximations on Steiner tree and location instances inspired by the scientific literature.

Our results illustrate that the exact algorithms are fast, being able to solve in reasonable amounts of time instances of realistic sizes. They also illustrate that the deterministic reformulations based on average or worst-case distances provide very good solutions in short amounts, offering interesting alternatives whenever an exact solution cannot be computed in an acceptable time.

References

Aissi H, Bazgan C, Vanderpooten D (2009) Min-max and min-max regret versions of combinatorial optimization problems: A survey. *Eur. J. Oper. Res.* 197(2):427–438, URL http://dx.doi.org/10.1016/j.ejor.2008.09.012.

Ayoub J, Poss M (2016) Decomposition for adjustable robust linear optimization subject to uncertainty polytope. *Comput. Manag. Science* 13(2):219–239, URL http://dx.doi.org/10.1007/s10287-016-0249-2.

Ben-Tal A, Goryashko AP, Guslitzer E, Nemirovski A (2004) Adjustable robust solutions of uncertain linear programs. *Math. Program.* 99(2):351–376, URL http://dx.doi.org/10.1007/s10107-003-0454-y.

Ben-Tal A, Nemirovski A (1998) Robust convex optimization. *Mathematics of Operations Research* 23(4):769–805.

Bertsimas D, Dunning I (2016) Multistage robust mixed-integer optimization with adaptive partitions. *Oper. Res.* 64(4):980–998, URL http://dx.doi.org/10.1287/opre.2016.1515.

Bertsimas D, Dunning I, Lubin M (2016) Reformulation versus cutting-planes for robust optimization. *Comput. Manag. Science* 13(2):195–217.

Bertsimas D, Sim M (2003) Robust discrete optimization and network flows. *Mathematical Programming* 98(1-3):49–71.

Bezanson J, Karpinski S, Shah VB, Edelman A (2012) Julia: A fast dynamic language for technical computing. *arXiv preprint arXiv:1209.5145* .

Bodlaender HL, Drange PG, Dregi MS, Fomin FV, Lokshtanov D, Pilipczuk M (2013) A o(cˆk n) 5-approximation algorithm for treewidth. *CoRR* abs/1304.6321, URL http://arxiv.org/abs/1304.6321.

Buchheim C, Kurtz J (2018) Robust combinatorial optimization under convex and discrete cost uncertainty. *EURO J. Computational Optimization* 6(3):211–238, URL http://dx.doi.org/10.1007/s13675-018-0103-0.

Cevallos A, Eisenbrand F, Morell S (2018) Diversity maximization in doubling metrics. *CoRR* abs/1809.09521, URL http://arxiv.org/abs/1809.09521.

Citovsky G, Mayer T, Mitchell JSB (2017) TSP With Locational Uncertainty: The Adversarial Model. Aronov B, Katz MJ, eds., *33rd International Symposium on Computational Geometry (SoCG 2017)*, volume 77 of *Leibniz International Proceedings in Informatics (LIPIcs)*, 32:1–32:16 (Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik), ISBN 978-3-95977-038-5, ISSN 1868-8969, URL http://dx.doi.org/10.4230/LIPIcs.SoCG.2017.32.

Correia I, da Gama FS (2015) Facility location under uncertainty. *Location science*, 177–203 (Springer).

Crescenzi P (1997) A short guide to approximation preserving reductions. *Proceedings of Computational Complexity. Twelfth Annual IEEE Conference*, 262–273, URL http://dx.doi.org/10.1109/CCC.1997.612321.

Cygan M, Fomin FV, Kowalik L, Lokshtanov D, Marx D, Pilipczuk M, Pilipczuk M, Saurabh S (2015) *Parameterized Algorithms* (Springer), URL http://dx.doi.org/10.1007/978-3-319-21275-3.

Daskin M (1993) Genrand2: A random network generator. *Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL, USA* .

de Ruiter F, Ben-Tal A (2017) Tractable nonlinear decision rules for robust optimization. Technical report, Working paper.

Downey RG, Fellows MR (2013) *Fundamentals of Parameterized Complexity*. Texts in Computer Science (Springer), URL http://dx.doi.org/10.1007/978-1-4471-5559-1.

Dunning I, Huchette J, Lubin M (2017) Jump: A modeling language for mathematical optimization. *SIAM Review* 59(2):295–320, URL http://dx.doi.org/10.1137/15M1020575.

Fellows MR, Fomin FV, Lokshtanov D, Rosamond F, Saurabh S, Szeider S, Thomassen C (2011) On the complexity of some colorful problems parameterized by treewidth. *Information and Computation* 209(2):143–153.

Fischetti M, Monaci M (2012) Cutting plane versus compact formulations for uncertain (integer) linear programs. *Math. Program. Comput.* 4(3):239–273.

Gutiérrez-Jarpa G, Obreque C, Laporte G, Marianov V (2013) Rapid transit network design for optimal cost and origin–destination demand capture. *Computers & Operations Research* 40(12):3000–3009, ISSN 0305-0548, URL http://dx.doi.org/https://doi.org/10.1016/j.cor.2013.06.013.

Hanasusanto GA, Kuhn D, Wiesemann W (2015) K-adaptability in two-stage robust binary programming. *Operations Research* 63(4):877–891.

Kasperski A, Zieliński P (2009) On the approximability of minmax (regret) network optimization problems. *Information Processing Letters* 109(5):262–266.

Kasperski A, Zieliński P (2016) Robust discrete optimization under discrete and interval uncertainty: A survey. *Robustness analysis in decision aiding, optimization, and analytics*, 113–143 (Springer).

Kloks T (1994) *Treewidth, Computations and Approximations*, volume 842 of *Lecture Notes in Computer Science* (Springer-Verlag), URL http://dx.doi.org/10.1007/BFb0045375.

Kouvelis P, Yu G (2013) *Robust discrete optimization and its applications*, volume 14 (Springer Science & Business Media).

Magnanti TL, Wong RT (1984) Network design and transportation planning: Models and algorithms. *Transportation science* 18(1):1–55.

Melkote S, Daskin MS (2001) An integrated model of facility location and transportation network design. *Transportation Research Part A: Policy and Practice* 35(6):515–538, ISSN 0965-8564, URL http://dx.doi.org/https://doi.org/10.1016/S0965-8564(00)00005-7.

Naoum-Sawaya J, Buchheim C (2016) Robust critical node selection by benders decomposition. *INFORMS J. Comput.* 28(1):162–174, URL http://dx.doi.org/10.1287/ijoc.2015.0671.

Postek K, Den Hertog D (2016) Multistage adjustable robust mixed-integer optimization via iterative splitting of the uncertainty set. *INFORMS Journal on Computing* 28(3):553–574.

Roos E, Den Hertog D, Ben-Tal A, de Ruiter FJ, Zhen J (2018) Approximation of hard uncertain convex inequalities. *Optimization Online URL http://www.optimization-online.org/DB_HTML/2018/06/6679. html* .

Shang Y, Ruml W, Zhang Y, Fromherz MP (2003) Localization from mere connectivity. *Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*, 201–212.

Subramanyam A, Gounaris CE, Wiesemann W (2019) K-adaptability in two-stage mixed-integer robust optimization. *Mathematical Programming Computation* 1–32.

Yaman H, Karasan OE, Pinar MÇ (2001) The robust spanning tree problem with interval data. *Oper. Res. Lett.* 29(1):31–40, URL http://dx.doi.org/10.1016/S0167-6377(01)00078-5.

Zeng B, Zhao L (2013) Solving two-stage robust optimization problems using a column-and-constraint generation method. *Operations Research Letters* 41(5):457–461.

Zhen J, de Ruiter FJ, Roos E, den Hertog D (2021) Robust optimization for models with uncertain second-order cone and semidefinite programming constraints. *INFORMS Journal on Computing* .

Zhen J, Hertog DD, Sim M (2018) Adjustable robust optimization via fourier-motzkin elimination. *Operations Research* 66(4):1086–1100, URL `http://dx.doi.org/10.1287/opre.2017.1714`.

A. Definitions related to parameterized complexity and treewidth

A.1 Tree decompositions and treewidth.

A *tree decomposition* of a graph $G = (V, E)$ is a pair $\mathcal{D} = (T, \mathcal{B})$, where T is a tree and $\mathcal{B} = \{X^w \mid w \in V[T]\}$ is a collection of subsets of V, called *bags*, such that:

- $\bigcup_{w \in V[T]} X^w = V$,
- for every edge $\{i, j\} \in E$, there is a $w \in V[T]$ such that $\{i, j\} \subseteq X^w$, and
- for every $\{x, y, z\} \subseteq V[T]$ such that z lies on the unique path between x and y in T, $X^x \cap X^y \subseteq X^z$.

We call the vertices of T *vertices* of $\mathcal{D}$ and the sets in $\mathcal{B}$ *bags* of $\mathcal{D}$. The *width* of a tree decomposition $\mathcal{D} = (T, \mathcal{B})$ is $\max_{w \in V[T]} |X^w| - 1$. The *treewidth* of a graph G, denoted by $tw(G)$, is the smallest integer t such that there exists a tree decomposition of G of width at most t. Let us now recall the definition of a *nice tree decomposition*, which will make the presentation of the algorithm used to proof Theorem 1 much simpler.

Let $\mathcal{D} = (T, \mathcal{B})$ be a rooted tree decomposition of G (meaning that T has a special vertex r called the *root*). As T is rooted, we naturally define an ancestor relation among bags, and say that $X^{w'}$ is a *descendant* of X^w if the vertex set of the unique simple path in T from r to w' contains w. In particular, every vertex w is a descendant of itself. For every $w \in V[T]$, we define $G^w = G[\bigcup\{X^{w'} \mid X^{w'} \text{ is a descendant of } X^w \text{ in } T\}]$.

Such a rooted decomposition is called a *nice tree decomposition* of G if the following conditions hold:

- $X^r = \emptyset$.

- Every vertex of T has at most two children in T.
- For every leaf $\ell \in V[T]$, $X^\ell = \emptyset$. Each such vertex ℓ is called a *leaf vertex*.
- If $w \in V[T]$ has exactly one child w', then either
  - $X^w = X^{w'} \cup \{i\}$ for some $i \notin X^{w'}$. Each such vertex is called an *introduce vertex*.
  - $X^w = X^{w'} \setminus \{i\}$ for some $i \in X^{w'}$. Each such vertex is called a *forget vertex*.
- If $w \in V[T]$ has exactly two children w_L and w_R, then $X^w = X^{w_L} = X^{w_R}$. Each such vertex w is called a *join vertex*.

We recall that one of the key property of such a nice decomposition is that for any $w \in V[T]$, X^w is a separator of G. This implies in particular that, in a join vertex, there is no edge $\{i, j\} \in G^w$ such that $i \in V[G^{w_L}] \setminus X^w$ and $j \in V[G^{w_R}] \setminus X^w$.

Given a tree decomposition of a graph G of width t and x vertices, it is possible to transform it in polynomial time into a *nice* one of width t and xt vertices (Kloks 1994). Moreover, it is possible (Bodlaender et al. (2013)) to compute a tree decomposition of width $tw' = \mathcal{O}(tw(G))$ and $\mathcal{O}(n)$ vertices in time $\mathcal{O}(c^{tw(G)}n)$, where $n = |V|$. By using these two results, we can compute in time $\mathcal{O}(c^{tw(G)}n)$ a nice tree decomposition of width $\mathcal{O}(tw(G))$ with $\mathcal{O}(tw(G)n)$ vertices.

A.2 Parameterized complexity

We refer the reader to Downey and Fellows (2013), Cygan et al. (2015) for basic background on parameterized complexity, and we recall here only some basic definitions. A *parameterized problem* is a language $L \subseteq \Sigma^* \times \mathbb{N}$, where Σ is some fixed alphabet. For an instance $I = (x, k) \in \Sigma^* \times \mathbb{N}$, k is called the *parameter*. Given a classical (non-parameterized) decision problem $L_c \subseteq \Sigma^*$ and a function $\kappa : \Sigma^* \to \mathbb{N}$, we denote by $L_c/\kappa = \{(x, \kappa(x)\} \mid x \in L_c\}$ the associated parameterized problem.

A parameterized problem L is *fixed-parameter tractable* ($\mathcal{FPT}$) if there exists an algorithm $\mathcal{A}$, a computable function f, and a constant c such that given an instance $I = (x, k)$, $\mathcal{A}$ (called an $\mathcal{FPT}$ *algorithm*) correctly decides whether $I \in L$ in time bounded by $f(k) \cdot |I|^c$. For instance, the Vertex Cover problem parameterized by the size of the solution is $\mathcal{FPT}$.

Within parameterized problems, the $\mathcal{W}$-hierarchy may be seen as the parameterized equivalent to the class $\mathcal{NP}$ of classical decision problems. Without entering into details

(see Downey and Fellows (2013), Cygan et al. (2015) for the formal definitions), a parameterized problem being $\mathcal{W}[1]$-hard can be seen as a strong evidence that this problem is *not* $\mathcal{FPT}$. The canonical example of $\mathcal{W}[1]$-hard problem is Independent Set parameterized by the size of the solution.

The most common way to transfer $\mathcal{W}[1]$-hardness is via parameterized reductions. A *parameterized reduction* from a parameterized problem L_1 to a parameterized problem L_2 is an algorithm that, given an instance (x, k) of L_1, outputs an instance (x', k') of L_2 such that

- (x, k) is a yes-instance of L_1 if and only if (x', k') is a yes-instance of L_2,
- $k' \leq g(k)$ for some computable function g, and
- the running time is bounded by $f(k) \cdot |x|^{\mathcal{O}(1)}$ for some computable function f.

If L_1 is $\mathcal{W}[1]$-hard and there is a parameterized reduction from L_1 to L_2, then L_2 is $\mathcal{W}[1]$-hard as well.

B. Computing the objective function on small treewidth graphs

Throughout this section, we consider the graph $G = (V, E)$ and denote by $u_{|X}$ the vector u restricted to components u_i such that $i \in X$, for any $X \subseteq V$.

B.1 Definition of the auxiliary problem

In this section we consider that we are given a fixed input of eval-c, and a nice tree decomposition $\mathcal{D} = (T, \mathcal{B})$ of G. Given $w \in V[T]$, we denote $\mathcal{U}^w = \times_{i \in V[G^w]} \mathcal{U}_i$. Let us define the following maximization problem Π. An input of Π is a pair (w, f) where $w \in V[T]$, and f is a function from X^w to $\mathcal{M}$ such that for any $i \in X^w$, $f(i) \in \mathcal{U}_i$. An output is a vector $u \in \mathcal{U}^w$ such that for any $i \in X^w, u_i = f(i)$, which we denote by $u \vdash (w, f)$. The objective is to maximize $c(u, G^w)$. We denote by $\textsc{opt}(w, f)$ the optimal value for instance (w, f). As usual in DP algorithms, to simplify the presentation we will define an algorithm A that given an input (w, f) only computes the value $\textsc{opt}(w, f)$. This algorithm could be easily modified to get an associated optimal solution.

B.2 Join case

Let w be a join vertex with children w^L and w^R. Given two vectors $u^L \in \mathcal{U}^{w^L}$ and $u^R \in \mathcal{U}^{w^R}$, such that for any $i \in X^w, u^L_i = u^R_i$, we define $u = u^L \diamond u^R$ by $u_i = u^L_i$ for any $i \in V[G^{w^L}]$, and $u_i = u^R_i$ for any $i \in V[G^{w^R}]$. Observe that u is well defined as for $i \in X^w$, $u^L_i = u^R_i$.

Lemma 1. *Let (w, f) be an input of Π such that w is a join vertex with children w^L and w^R. For any $u \in \mathcal{U}^w$, $u \vdash (w, f)$ if and only if there exists u^L, u^R such that the following conditions hold:*

- $u^L \vdash (w^L, f)$
- $u^R \vdash (w^R, f)$
- $u = u^L \diamond u^R$

Proof. The $\Rightarrow$ direction is immediate by defining $u^L = u_{|V[G^{w_L}]}$ (resp. $u^R = u_{|V[G^{w_R}]}$). In the $\Leftarrow$ direction, observe that $u^L \diamond u^R$ is well defined as for any $i \in X^w, u^L_i = u^R_i = f(i)$, and $u \vdash (w, f)$ is also immediate. □

Lemma 2. *Let (w, f) be an input of Π such that w is a join vertex with children w^L and w^R. Then,* $\text{OPT}(w, f) = \text{OPT}(w^L, f) + \text{OPT}(w^R, f) - d^{(w,f)}$, *where* $d^{(w,f)} = \sum_{i,j \in X^w, \{i,j\} \in E[G]} d(f(i), f(j))$.

Proof. Let us start with the $\leq$ inequality. Let u such that $c(u, G^w) = \text{OPT}(w, f)$. Let u^L and u^R as defined by Lemma 1. Observe that $c(u, G^w) = c(u, G^{w_L}) + c(u, G^{w_R}) - d^{(w,f)}$ as edges inside X^w are counted twice in the first two terms. We have $c(u, G^{w_L}) = c(u^L, G^{w_L})$, and $c(u^L, G^{w_L}) \leq OPT(w^L, f)$ as $u^L \vdash (w^L, f)$, and same properties hold for the right side. This implies $\text{OPT}(w, f) \leq \text{OPT}(w^L, f) + \text{OPT}(w^R, f) - d^{(w,f)}$.

Let us now turn to the other inequality. Let u^L such that $c(u^L, G^{w_L}) = \text{OPT}(w^L, f)$, u^R such that $c(u^R, G^{w_R}) = \text{OPT}(w^R, f)$, and $u = u^L \diamond u^R$. According to Lemma 1, $u \vdash (w, f)$, and again $c(u, G^w) = c(u^L, G^{w_L}) + c(u^R, G^{w_R}) - d^{(w,f)}$, implying the desired inequality. □

We are now ready to define the DP algorithm A in the join case. Given an input (w, f) of Π such that w is a join vertex with children w^L and w^R, $A(w, f)$ returns $A(w^L, f) + A(w^R, f) - d^{(w,f)}$. It is immediate by induction and using Lemma 2 that $A(w, f) = \text{OPT}(w, f)$.

B.3 Introduce case

Given any input (w, f) of Π and $X \subseteq X^w$, we denote by $f_{|X}$ function f restricted to X. The following two lemmas are immediate.

Lemma 3. *Let (w, f) be an input of Π such that w is an introduce vertex with children w'. Let i be such that $X^w = X^{w'} \cup \{i\}$. For any $u \in \mathcal{U}^w$, $u \vdash (w, f)$ if and only if the following conditions hold:*

- $u_{|V[G^{w'}]} \vdash (w', f_{|X^{w'}})$
- $u_i = f(i)$

Lemma 4. *Let (w, f) be an input of Π such that w is an introduce vertex with children w'. Let i be such that $X^w = X^{w'} \cup \{i\}$. Then, $\text{OPT}(w, f) = \text{OPT}(w', f_{|X^{w'}}) + d^{(i,w,f)}$, where $d^{(i,w,f)} = \sum_{j \in X^w, \{i,j\} \in E[G]} d(f(i), f(j))$.*

We are now ready to define the DP algorithm A in the introduce case. Given an input (w, f) of Π such that w is an introduce vertex with children w', where $X^w = X^{w'} \cup \{i\}$, $A(w, f)$ returns $A(w', f_{|X^{w'}}) + d^{(i,w,f)}$. It is immediate by induction and using Lemma 4 that $A(w, f) = \text{OPT}(w, f)$.

B.4 Forget case

Let (w, f) be an input of Π such that w is a forget vertex with children w'. Let i such that $X^{w'} = X^w \cup \{i\}$. For any $x \in \mathcal{M}$, we denote $f^{(i,x)}$ the function from $X^{w'}$ to $\mathcal{M}$ such that $f^{(i,x)}(j) = f(j)$ for any $j \neq i$, and $f^{(i,x)}(i) = x$. The following Lemma is immediate.

Lemma 5. *Let (w, f) be an input of Π such that w is a forget vertex with children w'. Let i be such that $X^{w'} = X^w \cup \{i\}$. For any $u \in \mathcal{U}^w$, $u \vdash (w, f)$ if and only if $u \vdash (w', f^{(i,u_i)})$.*

Lemma 6. *Let (w, f) be an input of Π such that w is a forget vertex with children w'. Let i be such that $X^{w'} = X^w \cup \{i\}$. Then, $\text{OPT}(w, f) = \max_{x \in \mathcal{U}_i} \text{OPT}(w', f^{(i,x)})$.*

Proof. Observe first that $G^w = G^{w'}$. Let us start with the $\leq$ inequality. Let u such that $c(u, G^w) = \text{OPT}(w, f)$. Notice that $c(u, G^w) = c(u, G^{w'})$. By Lemma 5, $u \vdash (w', f^{(i,u_i)})$, implying $c(u, G^{w'}) \leq \text{OPT}(w', f^{(i,u_i)}) \leq \max_{x \in \mathcal{U}_i} \text{OPT}(w', f^{(i,x)})$.

Let us now turn to the other inequality. Let $x^* \in \mathcal{U}_i$ maximizing the right side. Let u such that $c(u, G^{w'}) = \text{OPT}(w', f^{(i,x^*)})$. Notice that as $u \vdash (w', f^{(i,x^*)}), u_i = x^*$, and thus

$u \vdash (w', f^{(i,u_i)})$). According to Lemma 5, $u \vdash (w, f)$, implying that $c(u, G^{w'}) = c(u, G^w) \leq \text{OPT}(w, f)$. □

We are now ready to define the DP algorithm A in the forget case. Given an input (w, f) of Π such that w is a forget vertex with children w', where $X^{w'} = X^w \cup \{i\}$, $A(w, f)$ returns $\max_{x \in \mathcal{U}_i} A(w', f^{(i,x)})$. It is immediate by induction and using Lemma 6 that $A(w, f) = \text{OPT}(w, f)$.

B.5 Putting pieces together

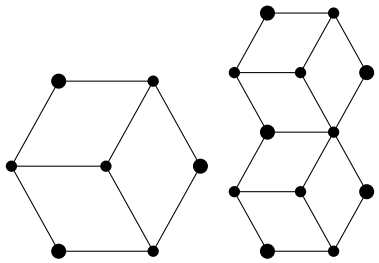
Theorem 2. EVAL-C/$tw + \sigma$ *is* $\mathcal{FPT}$. *More precisely, we can compute an optimal solution of* EVAL-C *in time* $\mathcal{O}(ntw\sigma^{\mathcal{O}(tw)})$, *where* $n = |V|$, $tw = tw(G)$, *and* $\sigma = \max_{i \in V} \sigma_i$.

Proof. Given an input $(\mathcal{M}, d, G, \mathcal{U})$ of EVAL-C, we start (see Appendix A.1) by computing in time $\mathcal{O}(c^{tw(G)}n)$ a nice tree decomposition of width $\mathcal{O}(tw(G))$ with $N = \mathcal{O}(ntw(G))$ vertices. Remember that this nice tree decomposition is rooted on a vertex r such that $X^r = \emptyset$. Then, we output $A(r, \emptyset)$. Notice that as $X^r = \emptyset$, the second parameter (the function from X^r to $\mathcal{M}$) is defined nowhere and denoted $\emptyset$. As A solves Π optimaly, we have $A(r, \emptyset) = \text{OPT}(r, \emptyset)$. Moreover, as $G^r = G$, we have $\text{OPT}(r, \emptyset) = c(G)$.

Let us now consider the running time of A. Given a tree decomposition with N vertices (in the tree of bags) and of width t, the size of the DP table is $\mathcal{O}(N\sigma^t)$, the time to compute one entry is dominated by the forget case where the branching is in $\mathcal{O}(\sigma)$, implying a running time in $\mathcal{O}(N\sigma^{t+1})$. Pluging the corresponding values, we get the claimed running time. □

C. Compact formulation for the Steiner Tree Problem

We extend next the construction from Section 3.3 to trees, albeit this involves logical constraints. We consider more particularly the case of the Steiner Tree Problem where a set of terminals $T \subseteq V$ is given and any $G \in \mathcal{G}$ is a Steiner tree connecting the terminals of T. We further assume that r is a given arbitrary root in T and that any $x \in \mathcal{X}$ describes an directed tree from r to the set of terminals $T \setminus \{r\}$. In particular, this involves that the edges are directed, so variables x_{ij} and x_{ji} now denote the two directed edges (i, j) and (j, i) obtained from $\{i, j\}$, leading to the directed set of edges E^{dir}. Similarly, we introduce the incoming and outgoing stars of i as $\delta^-(i) = \left\{ j \;\middle|\; (j, i) \in E^{dir} \right\}$ and $\delta^+(i) = \left\{ j \;\middle|\; (i, j) \in E^{dir} \right\}$.



(a) format(1) (b) format(2)

Figure 7: Small instances inspired by the format instance from SteinLib, T contains the larger nodes.

Then, extending the optimization variable z to any node in V, we obtain the following formulation

$$\begin{aligned}
\min \quad & \omega \\
\text{s.t.} \quad & \omega \geq z_r^k, \quad \forall k \in [\sigma_r] && (21)\\
& z_i^k \geq \sum_{j \in \delta^+(i)} x_{ij} \max_{\ell \in [\sigma_j]} \left(d(u_i^k, u_j^\ell) + z_j^\ell \right), \quad \forall i \in V, k \in [\sigma_j] && (22)\\
& x \in \mathcal{X}, z \geq 0. && (23)
\end{aligned}$$

To linearize the maxima in the right-hand-side of (22), we introduce variables Z_{ij}^k such that

$$Z_{ij}^k \geq d(u_i^k, u_j^\ell) + z_j^\ell, \quad \forall \ell \in [\sigma_j]$$

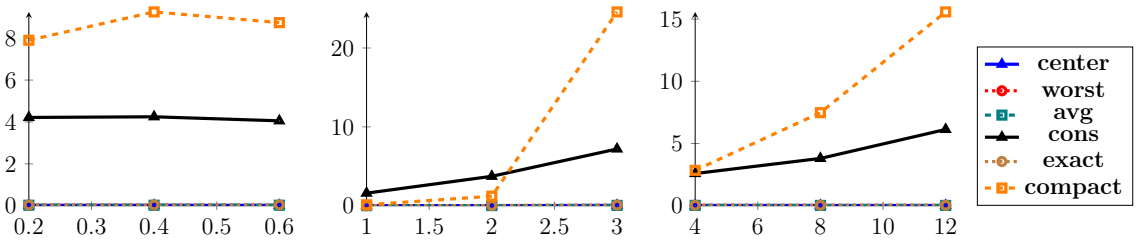
and replace (22) with

$$z_i^k \geq \sum_{j \in \delta^+(i)} x_{ij} Z_{ij}^k, \quad \forall i \in V \setminus T_0, k, \ell \in [\sigma_j]. \tag{24}$$

The right-hand-side of constraints (24) can be further linearized with the help of additional variables X_{ij}^k and logical constraints

$$x_{ij} = 1 \implies X_{ij}^k \geq Z_{ij}^k.$$

We illustrate and compare the above formulation on small artificial instances built upon the *format* instance which includes 7 vertices and 9 edges (the instance is available at `http://steinlib.zib.de/format.php`). To get larger instances from the *format* instance, we remove the central terminal and add layered copies of the instance. Figure 7 depicts the original structure of the *format* instance and that obtained by adding one copy. We denote as format(κ) the instance with κ copies of the original graph.



(a) Varying Δ (b) Varying κ (c) Varying σ

Figure 8: STP: Average solution times in seconds on instances format(κ) for each algorithm when varying one of the parameters.

The results presented on Figure 8 underline that **compact** can hardly solve large instances, as the solution times increase significantly with κ. They also illustrate that **cons** is much slower than **exact** on these small artificial instances.

D. Connection with the affine decision rules approximation from Zhen et al. (2021)

Notice first that, due to the convexity of the norm, the constraint

$$\forall u \in \mathcal{U} : \sum_{\{i,j\}\in\mathbb{E}} x_{ij}\|u_i - u_j\|_2 \leq \omega$$

is equivalent to

$$\forall u \in \text{conv}(\mathcal{U}) : \sum_{\{i,j\}\in\mathbb{E}} x_{ij}\|u_i - u_j\|_2 \leq \omega.$$

Next, let us denote the unit ball of dimension p by $\mathcal{W}^p$, as well as $\mathcal{W} = \times_{e\in\mathbb{E}}\mathcal{W}^p$. Let us also direct arbitrarily every edge in $\mathbb{E}$, leading to the set of directed edges $\vec{\mathbb{E}}$. Following the same idea as (Zhen et al. 2021, Theorem 1), we obtain that the constraint

$$\forall u \in \text{conv}(\mathcal{U}) : \sum_{(i,j)\in\vec{\mathbb{E}}} x_{ij}\|u_i - u_j\|_2 \leq \omega$$

is equivalent to

$$\forall u \in \mathrm{conv}(\mathcal{U}) : \sum_{(i,j)\in\vec{\mathbb{E}}} x_{ij} \max_{w_{ij}\in\mathcal{W}^p} w_{ij}^T(u_i - u_j) \leq \omega \tag{25}$$

$$\Leftrightarrow \forall w \in \mathcal{W}, u \in \mathrm{conv}(\mathcal{U}) : \sum_{(i,j)\in\vec{\mathbb{E}}} x_{ij} w_{ij}^T(u_i - u_j) \leq \omega \tag{26}$$

$$\Leftrightarrow \forall w \in \mathcal{W} : \max \left\{ \sum_{(i,j)\in\vec{\mathbb{E}}} x_{ij} w_{ij}^T \left(\sum_{k=1}^{\sigma_i} \lambda_i^k u_i^k - \sum_{\ell=1}^{\sigma_j} \lambda_j^\ell u_j^\ell \right) \;\middle|\; \sum_{k=1}^{\sigma_i} \lambda_i^k = 1, \forall i \in \mathbb{V},\, \lambda \geq 0 \right\} \leq \omega \tag{27}$$

$$\Leftrightarrow \forall w \in \mathcal{W} : \min \left\{ \sum_{i\in\mathbb{V}} \mu_i \;\middle|\; \mu_i \geq \left(\sum_{(i,j)\in\vec{\mathbb{E}}} x_{ij} w_{ij}^T - \sum_{(j,i)\in\vec{\mathbb{E}}} x_{ji} w_{ji}^T \right) u_i^k, \forall i \in \mathbb{V}, k \in [\sigma_i] \right\} \leq \omega. \tag{28}$$

Observe that the left-hand side of (28) can be interpreted as a two-stage robust optimization problem without first-stage variables, with μ playing the role of the second-stage variables, and with w representing the uncertain parameters. This type of models being notoriously difficult to solve to optimality, we follow (Zhen et al. 2021, Lemma 1) and seek a heuristic solution by considering second-stage variables μ that can be expressed as affine decision rules

$$\mu_i(w) = \mu_i^0 + \sum_{(i',j')\in\vec{\mathbb{E}}} \mu_{i,i'j'}^T w_{i'j'}, \tag{29}$$

where $\mu_i^0 \in \mathbb{R}$ and $\mu_{i,i'j'} \in \mathbb{R}^p$. Replacing (18) by (28) with μ substituted with the right-hand side of (29), we obtain

$$\begin{aligned}
\min \quad & \omega \\
\text{s.t.} \quad & \omega \geq \sum_{i\in\mathbb{V}} \left(\mu_i^0 + \sum_{(i',j')\in\vec{\mathbb{E}}} \mu_{i,i'j'}^T w_{i'j'} \right), \quad \forall w \in \mathcal{W} \\
& \mu_i^0 + \sum_{(i',j')\in\vec{\mathbb{E}}} \mu_{i,i'j'}^T w_{i'j'} \geq \left(\sum_{(i,j)\in\vec{\mathbb{E}}} x_{ij} w_{ij}^T - \sum_{(j,i)\in\vec{\mathbb{E}}} x_{ji} w_{ji}^T \right) u_i^k, \quad \forall i \in \mathbb{V}, k \in [\sigma_i], w \in \mathcal{W} \\
& x \in \mathcal{X}.
\end{aligned}$$

Dualizing the robust counstraints with respect to $w \in \mathcal{W}$ yields

$$\min \quad \omega \tag{30}$$

$$\text{s.t.} \quad \omega \geq \sum_{i\in\mathbb{V}} \mu_i^0 + \sum_{(i,j)\in\vec{\mathbb{E}}} \left\| \mu_{i,ij} + \mu_{j,ij} + \sum_{i'\neq i,j} \mu_{i',ij} \right\|_2 \tag{31}$$

$$\mu_i^0 \geq \sum_{(i,j)\in\vec{\mathbb{E}}} \|x_{ij}u_i^k - \mu_{i,ij}\|_2 + \sum_{(j,i)\in\vec{\mathbb{E}}} \|x_{ji}u_i^k + \mu_{i,ji}\|_2 + \sum_{(i',j')\in\vec{\mathbb{E}}:i\neq i',j'} \|\mu_{i,i'j'}\|_2, \quad \forall i \in \mathbb{V}, k \in [\sigma_i] \tag{32}$$

$$x \in \mathcal{X}. \tag{33}$$

We discuss next how we can substantially reduce the number of affine multipliers in (29), and consequently, in problem (30)–(33). Let $(\tilde{\omega}, \tilde{x}, \tilde{\mu}^0, \tilde{\mu})$ denote an optimal solution to (30)–(33). Observe that the optimal solution cost is equal to $\tilde{\omega} = \max_{u\in\mathcal{U}} \tilde{\omega}(u)$ where

$$\tilde{\omega}(u) = \sum_{i\in\mathbb{V}} \left(\sum_{(i,j)\in\vec{\mathbb{E}}} \|\tilde{x}_{ij}u_i - \tilde{\mu}_{i,ij}\|_2 + \sum_{(j,i)\in\vec{\mathbb{E}}} \|\tilde{x}_{ji}u_i + \tilde{\mu}_{i,ji}\|_2 + \sum_{(i',j')\in\vec{\mathbb{E}}:i\neq i',j'} \|\tilde{\mu}_{i,i'j'}\|_2 \right) + \sum_{(i,j)\in\vec{\mathbb{E}}} \left\| \tilde{\mu}_{i,ij} + \tilde{\mu}_{j,ij} + \sum_{i'\neq i,j} \tilde{\mu}_{i',ij} \right\|_2 \tag{34}$$

$$= \sum_{(i,j)\in\vec{\mathbb{E}}} \left(\|\tilde{x}_{ij}u_i - \tilde{\mu}_{i,ij}\|_2 + \|\tilde{x}_{ij}u_j + \tilde{\mu}_{j,ij}\|_2 + \sum_{i'\neq i,j} \|\tilde{\mu}_{i',ij}\|_2 + \left\| \tilde{\mu}_{i,ij} + \tilde{\mu}_{j,ij} + \sum_{i'\neq i,j} \tilde{\mu}_{i',ij} \right\|_2 \right) \tag{35}$$

We are going to define a sequence of two new solutions, $\tilde{\mu}'$ and $\tilde{\mu}''$, such that the corresponding values $\tilde{\omega}'(u)$ and $\tilde{\omega}''(u)$ are not greater than $\tilde{\omega}(u)$ for each $u \in \mathcal{U}$. First, we define $\tilde{\mu}'$ by setting $\tilde{\mu}'_{i',ij} = 0$ for all $(i,j) \in \vec{\mathbb{E}}$ such that $i' \notin \{i,j\}$ and $\tilde{\mu}'_{i',ij} = \tilde{\mu}_{i',ij}$ otherwise, and let $\tilde{\omega}'(u)$ be the right-hand side of (35) with μ replaced by μ'. Observe that

$$\begin{aligned} \sum_{i'\neq i,j} \|\tilde{\mu}'_{i',ij}\|_2 + \left\| \tilde{\mu}'_{i,ij} + \tilde{\mu}'_{j,ij} + \sum_{i'\neq i,j} \tilde{\mu}'_{i',ij} \right\|_2 &= \left\| \tilde{\mu}'_{i,ij} + \tilde{\mu}'_{j,ij} \right\|_2 \\ &= \left\| \tilde{\mu}_{i,ij} + \tilde{\mu}_{j,ij} \right\|_2 \\ &\leq \sum_{i'\neq i,j} \|\tilde{\mu}_{i',ij}\|_2 + \left\| \tilde{\mu}_{i,ij} + \tilde{\mu}_{j,ij} + \sum_{i'\neq i,j} \tilde{\mu}_{i',ij} \right\|_2, \end{aligned}$$

which implies that

$$\tilde{\omega}'(u) \leq \tilde{\omega}(u), \tag{36}$$

for each $u \in \mathcal{U}$. Second, we define another solution $\tilde{\mu}''$ such that $\tilde{\mu}''_{i,ij} = \tilde{\mu}'_{i,ij}$ and $\tilde{\mu}''_{j,ij} = -\tilde{\mu}'_{i,ij}, \forall (i,j) \in \vec{\mathbb{E}}$. Then, we denote as $\tilde{\omega}''(u)$ the corresponding right-hand side of (35). Observe that for each $u \in \mathcal{U}$

$$\tilde{\omega}''(u) = \sum_{(i,j)\in\vec{\mathbb{E}}} \left(\|\tilde{x}_{ij}u_i - \tilde{\mu}''_{i,ij}\|_2 + \|\tilde{x}_{ij}u_j + \tilde{\mu}''_{j,ij}\|_2 + \|\tilde{\mu}''_{i,ij} + \tilde{\mu}''_{j,ij}\|_2 \right) \tag{37}$$

$$= \sum_{(i,j)\in\vec{\mathbb{E}}} \left(\|\tilde{x}_{ij}u_i - \tilde{\mu}'_{i,ij}\|_2 + \|\tilde{x}_{ij}u_j - \tilde{\mu}'_{i,ij}\|_2 \right) \leq \tilde{\omega}'(u). \tag{38}$$

From (36) and (38), we see that we can set $\tilde{\mu}_{i',ij} = 0$ for all $(i,j) \in \vec{\mathbb{E}}$ such that $i' \notin \{i,j\}$ and $\tilde{\mu}_{i,ij} = -\tilde{\mu}_{j,ij}, \forall (i,j) \in \vec{\mathbb{E}}$ without deteriorating the quality of the solution returned by (30)–(33). Thus, renaming the variables $\mu_{i,ij}$ as μ_{ij}, formulation (30)–(33) becomes

$$\begin{aligned}
\min \quad & \omega \\
\text{s.t.} \quad & \omega \geq \sum_{i\in\mathbb{V}} \mu_i^0 \\
& \mu_i^0 \geq \sum_{(i,j)\in\vec{\mathbb{E}}} \|x_{ij}u_i^k - \mu_{ij}\|_2 + \sum_{(j,i)\in\vec{\mathbb{E}}} \|x_{ji}u_i^k - \mu_{ji}\|_2, \quad \forall i \in \mathbb{V}, u \in \mathcal{U} \\
& x \in \mathcal{X},
\end{aligned}$$

and the equivalence with (13)–(16) follows by removing the dummy variable ω, introducing artificial variables to separate the norms into individual second-order cone constraints, and renaming μ_i^0 as d_i.