Optimization problems in graphs with locational uncertainty: Electronic Companion

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

EC.1. Definitions related to parameterized complexity and treewidth

EC.1.1. 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.

EC.1.2. 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.

EC.2. 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 $X \subseteq V$.

EC.2.1. Definition of the auxiliary problem

In this section we consider that we are given a fixed input of ADVERSARIAL, 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.

EC.2.2. 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. We obtain the $\Rightarrow$ direction 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, $\textsc{opt}(w, f) = \textsc{opt}(w^L, f) + \textsc{opt}(w^R, f) - d^{(w,f)}$, 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 $c(u, G^w) = \textsc{opt}(w, f)$. 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 $\textsc{opt}(w, f) \leq \textsc{opt}(w^L, f) + \textsc{opt}(w^R, f) - d^{(w,f)}$.

Let us now turn to the other inequality. Let u^L such that $c(u^L, G^{w_L}) = \textsc{opt}(w^L, f)$, u^R such that $c(u^R, G^{w_R}) = \textsc{opt}(w^R, f)$, 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 follows from induction and using Lemma 2 that $A(w, f) = \textsc{opt}(w, f)$.

EC.2.3. 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 easily verified.

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 $A(w', f_{|X^{w'}}) + d^{(i,w,f)}$. Using Lemma 4 and induction, we obtain that $A(w, f) = \text{OPT}(w, f)$.

EC.2.4. 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 $f^{(i,x)}(i) = x$. We obtain the following lemma.

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 follows by induction and using Lemma 6 that $A(w, f) = \text{OPT}(w, f)$.

EC.2.5. Putting pieces together

Theorem 1. ADVERSARIAL/$tw + \sigma$ *is* $\mathcal{FPT}$*. More precisely, we can compute an optimal solution of* ADVERSARIAL *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 ADVERSARIAL, we start (see Section EC.1.1.) 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 $\mathcal{O}(N\sigma^{t+1})$. Plugging the corresponding values, we get the claimed running time. □

EC.3. 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 feasible solution 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 $\mathbb{E}^{bidir}$. Similarly, we introduce the incoming and outgoing stars of 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.

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

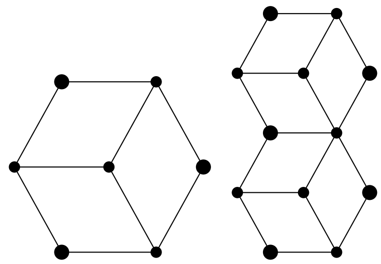
$$\begin{aligned}
\min \quad & \omega \\
\text{s.t.} \quad & \omega \geq z_r^k, \quad \forall k \in [\sigma_r] && \text{(EC.1)} \\
& z_i^k \geq \sum_{j \in \delta^+(i)} x_{ij} \max_{\ell \in [\sigma_j]} \left(d(u_i^k, u_j^\ell) + z_j^\ell\right), \quad \forall i \in V, k \in [\sigma_j] && \text{(EC.2)} \\
& x \in \mathcal{X}, z \geq 0. && \text{(EC.3)}
\end{aligned}$$

To linearize the maxima in the right-hand-side of (EC.2), 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 (EC.2) with

$$z_i^k \geq \sum_{j \in \delta^+(i)} x_{ij} Z_{ij}^k, \quad \forall i \in V \setminus T_0, k, \ell \in [\sigma_j]. \tag{EC.4}$$



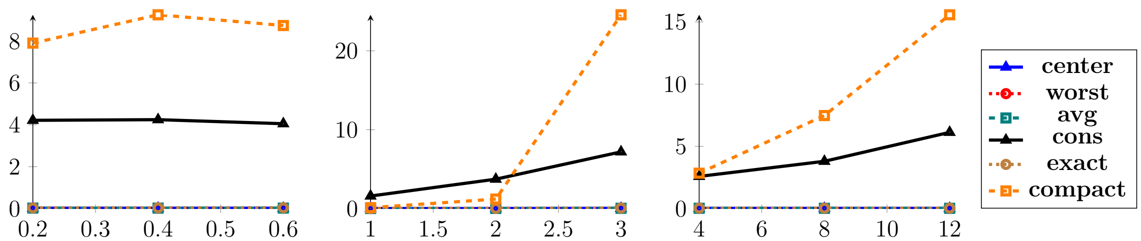
(a) format(1) (b) format(2)

Figure 1: Small instances inspired by the format instance from SteinLib, T contains the larger nodes.

The right-hand-side of constraints (EC.4) 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 1 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 2: STP: Average solution times in seconds on instances format(κ) for each algorithm when varying one of the parameters.

The results presented on Figure 2 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.

EC.4. 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

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

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

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

is equivalent to

$$\forall u \in \operatorname{conv}(\mathcal{U}) : \sum_{(i,j)\in\vec{\mathbb{E}}} x_{ij} \max_{w_{ij}\in\mathcal{W}^p} w_{ij}^T(u_i - u_j) \leq \omega \tag{EC.5}$$

$$\Leftrightarrow \forall w \in \mathcal{W}, u \in \operatorname{conv}(\mathcal{U}) : \sum_{(i,j)\in\vec{\mathbb{E}}} x_{ij} w_{ij}^T(u_i - u_j) \leq \omega \tag{EC.6}$$

$$\Leftrightarrow \forall w \in \mathcal{W} : \max\left\{ \sum_{(i,j)\in\vec{\mathbb{E}}} x_{ij} w_{ij}^T \left(\sum_{k=1}^{\sigma_i} \lambda_i^k u_i^k - \sum_{\ell=1}^{\sigma_j} \lambda_j^\ell u_j^\ell \right) \;\middle|\; \sum_{k=1}^{\sigma_i} \lambda_i^k = 1, \forall i \in \mathbb{V},\ \lambda \geq 0 \right\} \leq \omega \tag{EC.7}$$

$$\Leftrightarrow \forall w \in \mathcal{W} : \min\left\{ \sum_{i\in\mathbb{V}} \mu_i \;\middle|\; \mu_i \geq \left(\sum_{(i,j)\in\vec{\mathbb{E}}} x_{ij} w_{ij}^T - \sum_{(j,i)\in\vec{\mathbb{E}}} x_{ji} w_{ji}^T \right) u_i^k, \forall i \in \mathbb{V}, k \in [\sigma_i] \right\} \leq \omega. \tag{EC.8}$$

Observe that the left-hand side of (EC.8) 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

$$\mu_i(w) = \mu_i^0 + \sum_{(i',j')\in\vec{\mathbb{E}}} \mu_{i,i'j'}^T w_{i'j'}, \tag{EC.9}$$

where $\mu_i^0 \in \mathbb{R}$ and $\mu_{i,i'j'} \in \mathbb{R}^p$. Replacing (19) by (EC.8) with μ substituted with the right-hand side of (EC.9), 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

$$\min \quad \omega \tag{EC.10}$$

$$\text{s.t.} \quad \omega \geq \sum_{i \in \mathbb{V}} \mu_i^0 + \sum_{(i,j) \in \vec{\mathbb{E}}} \left\| \mu_{i,ij} + \mu_{j,ij} + \sum_{i' \neq i,j} \mu_{i',ij} \right\|_2 \tag{EC.11}$$

$$\mu_i^0 \geq \sum_{(i,j) \in \vec{\mathbb{E}}} \|x_{ij} u_i^k - \mu_{i,ij}\|_2 + \sum_{(j,i) \in \vec{\mathbb{E}}} \|x_{ji} u_i^k + \mu_{i,ji}\|_2 + \sum_{(i',j') \in \vec{\mathbb{E}}: i \neq i',j'} \|\mu_{i,i'j'}\|_2, \quad \forall i \in \mathbb{V}, k \in [\sigma_i] \tag{EC.12}$$

$$x \in \mathcal{X}. \tag{EC.13}$$

We discuss next how we can substantially reduce the number of affine multipliers in (EC.9), and consequently, in problem (EC.10)–(EC.13). Let $(\tilde{\omega}, \tilde{x}, \tilde{\mu}^0, \tilde{\mu})$ denote an optimal solution to (EC.10)–(EC.13). Observe that the optimal solution cost is equal to $\tilde{\omega} = \max_{u \in \mathcal{U}} \tilde{\omega}(u)$ where

$$\begin{aligned}
\tilde{\omega}(u) &= \sum_{i \in \mathbb{V}} \left(\sum_{(i,j) \in \vec{\mathbb{E}}} \|\tilde{x}_{ij} u_i - \tilde{\mu}_{i,ij}\|_2 + \sum_{(j,i) \in \vec{\mathbb{E}}} \|\tilde{x}_{ji} u_i + \tilde{\mu}_{i,ji}\|_2 + \sum_{(i',j') \in \vec{\mathbb{E}}: i \neq i',j'} \|\tilde{\mu}_{i,i'j'}\|_2 \right) \\
&\qquad + \sum_{(i,j) \in \vec{\mathbb{E}}} \left\| \tilde{\mu}_{i,ij} + \tilde{\mu}_{j,ij} + \sum_{i' \neq i,j} \tilde{\mu}_{i',ij} \right\|_2 \qquad \text{(EC.14)} \\
&= \sum_{(i,j) \in \vec{\mathbb{E}}} \left(\|\tilde{x}_{ij} u_i - \tilde{\mu}_{i,ij}\|_2 + \|\tilde{x}_{ij} u_j + \tilde{\mu}_{j,ij}\|_2 + \sum_{i' \neq i,j} \|\tilde{\mu}_{i',ij}\|_2 + \left\| \tilde{\mu}_{i,ij} + \tilde{\mu}_{j,ij} + \sum_{i' \neq i,j} \tilde{\mu}_{i',ij} \right\|_2 \right) \qquad \text{(EC.15)}
\end{aligned}$$

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 (EC.15) 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

$$\tilde{\omega}'(u) \leq \tilde{\omega}(u), \tag{EC.16}$$

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 (EC.15).

Observe that for each $u \in \mathcal{U}$

$$\tilde{\omega}''(u) = \sum_{(i,j)\in\vec{\mathbb{E}}} \left(\|\tilde{x}_{ij}u_i - \tilde{\mu}''_{i,ij}\|_2 + \|\tilde{x}_{ij}u_j + \tilde{\mu}''_{j,ij}\|_2 + \|\tilde{\mu}''_{i,ij} + \tilde{\mu}''_{j,ij}\|_2 \right) \tag{EC.17}$$

$$= \sum_{(i,j)\in\vec{\mathbb{E}}} \left(\|\tilde{x}_{ij}u_i - \tilde{\mu}'_{i,ij}\|_2 + \|\tilde{x}_{ij}u_j - \tilde{\mu}'_{i,ij}\|_2 \right) \le \tilde{\omega}'(u). \tag{EC.18}$$

From (EC.16) and (EC.18), 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 (EC.10)–(EC.13). Thus, renaming the variables $\mu_{i,ij}$ as μ_{ij}, formulation (EC.10)–(EC.13) becomes

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

and the equivalence with (14)–(17) 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.

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.

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

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.

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.

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.

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

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.

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.

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* .