INTERDICTION OF MINIMUM SPANNING TREES AND OTHER MATROID BASES

Noah Weninger* and Ricardo Fukasawa
Department of Combinatorics & Optimization, University of Waterloo
*nweninger@uwaterloo.ca

1. Introduction

Bilevel programming is a generalization of mixed integer programming where a subset of the variables can be constrained to be optimal for a secondary optimization problem.

$$\begin{aligned} \max \quad & v^\top x + w^\top y \\ \text{s.t.} \quad & Ax + By \le \beta \\ & x \in \mathbb{Z}^r \times \mathbb{R}^s \\ & y \in \arg\min\left\{ f^\top y \mid Cx + Dy \le \delta,\ y \in \mathbb{Z}^p \times \mathbb{R}^q \right\} \end{aligned}$$

It is often interpreted as a 2-round 2-player game, where the **Leader** selects the x variables, and the **Follower** selects the y variables.

An **interdiction problem** is a bilevel programming problem, which is typically of the form

$$\max_{X \in \mathcal{U}} \min_{\substack{Y \in \mathcal{L} \\ X \cap Y = \emptyset}} w(Y) \qquad \text{(Notation: } w(Y) = \textstyle\sum_{i \in Y} w_i)$$

where:

- $w_1, w_2, \dots, w_m \in \mathbb{Z}$ are the **weights**. We assume $w_1 < w_2 < \cdots < w_m$ for simplicity.
- $c_1, c_2, \dots, c_m \in \mathbb{Z}_{\ge 0}$ are the **costs** and $B \in \mathbb{Z}_{\ge 0}$ is the **budget**.
- $\mathcal{U} = \{X \subseteq \{1, \dots, m\} \mid c(X) \le B\}$ is the Leader's feasible region.
- $\mathcal{L} \subseteq 2^{\{1,\dots,m\}}$ is the set of feasible solutions to some combinatorial optimization problem.
- We assume that for every $X \in \mathcal{U}$, there exists some $Y \in \mathcal{L}$ such that $X \cap Y = \emptyset$. (★)

$\Longrightarrow$ The Leader can be seen as an adversary who is trying to make the solution to the Follower's problem as bad as possible by **interdicting** (deleting) some parts of the structure.

2. Problem statement

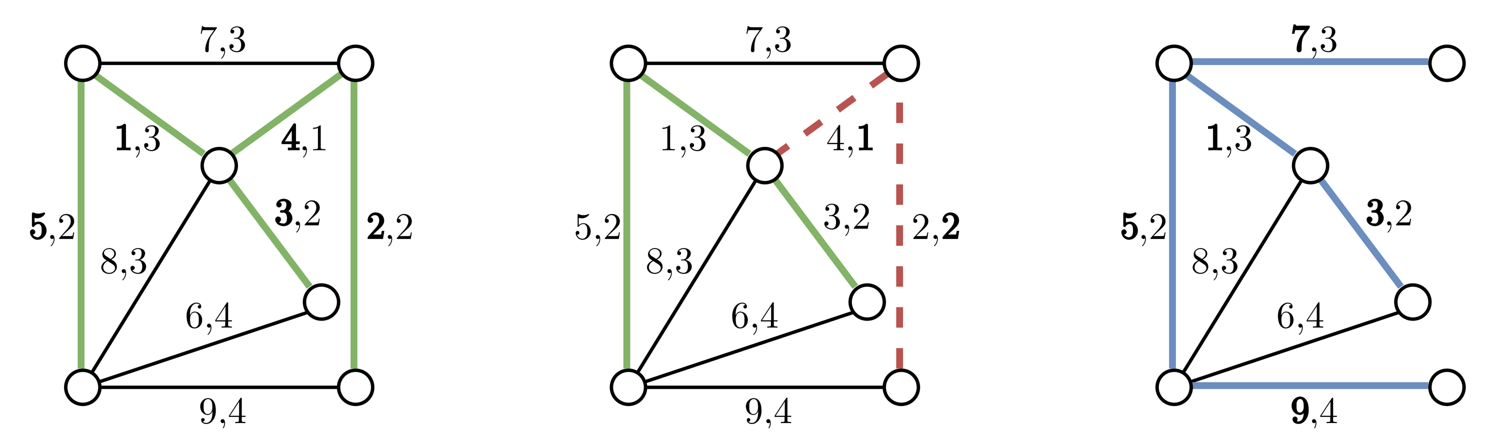
We define **minimum spanning tree (MST) interdiction** as the interdiction problem with

$$\mathcal{L} = \{Y \subseteq E \mid Y \text{ is a spanning tree of } G\}$$

where $G = (V, E)$ is a graph with m edges. MST interdiction is strongly NP-hard [2].

Example. Edges are labeled with w_e, c_e.

1. The original graph G. The original MST of G has weight **15**.	2. Edges are interdicted. For budget $B = 4$, this is the optimal X; it has cost **3**	3. G after interdiction. The new MST of $G \setminus X$ has weight **25**.

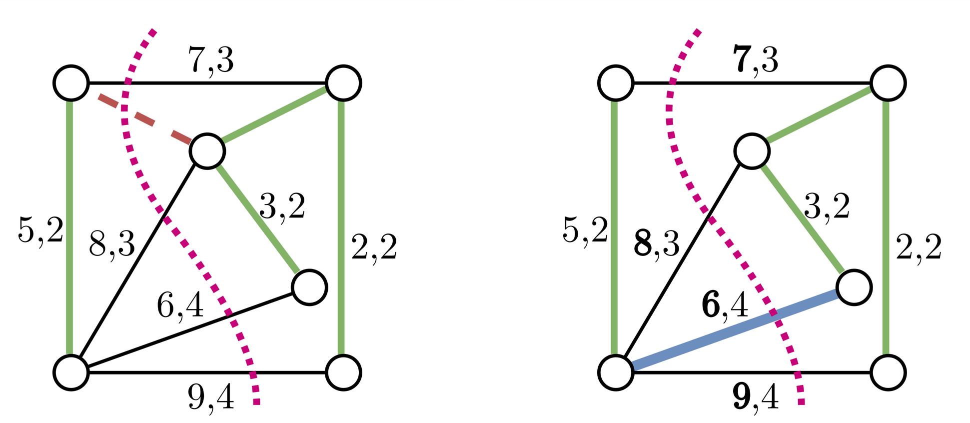


By (★), B cannot be larger than 5, because the global minimum cut has cost 6.

3. Incrementally interdicting edges

Suppose we add some edges to X one by one, in order of increasing weight.
To follow along, consider: **what would Kruskal's algorithm do?**

Every time one edge is interdicted (added to X), the current MST is cut into two components.	To reconnect the MST, we only need to add one edge. The replacement edge is the lowest weight edge crossing the cut.



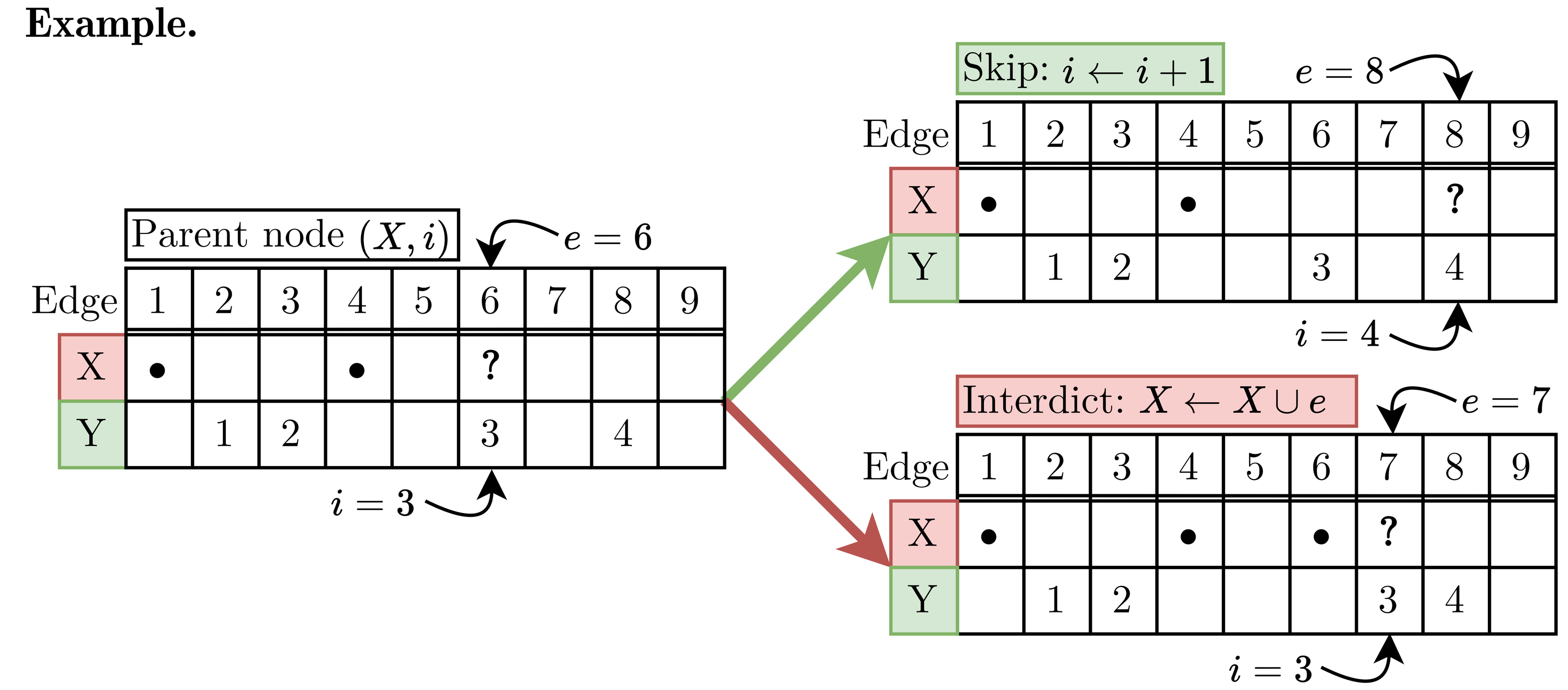
Note:

- By (★), a replacement edge always exists given that $X \in \mathcal{U}$.
- The replacement edge has larger weight than the interdicted edge.
- If the interdicted edge is *not* in the MST, the MST does not change; we should never interdict such an edge in an optimal solution.

4. Branch-and-bound

We solve the problem with branch-and-bound. Nodes are identified by a pair (X, i); from this we derive the current MST Y, and e, the edge to branch on. A **dynamic data structure** is used to quickly update Y and e as X and i change.

Example.



Motivated by our prior work on knapsack interdiction [4], we want an upper bound $f(e, c(X))$ on how much $w(Y)$ can increase by in *any child node* of (X, i). Then, if $f(e, c(X)) + w(Y)$ is at most the current lower bound weight $w(Y^*)$, we can prune (X, i).

Theorem 1.

Let $k = \max\{|X| : X \in \mathcal{U}\}$ and let n be the number of vertices in G. Even without pruning, the worst-case running time of the algorithm is

$$\tilde{O}\left(\binom{n+k}{\min\{n,k\}} \right) = \tilde{O}(\min\{(5.44\,n/k)^k, (5.44\,k/n)^n\}).$$

The previous best was $\tilde{O}(n^k)$. In terms of n and k, our algorithm is asymptotically faster than the previous best, up to polylog factors of n (polylog factors are hidden by $\tilde{O}$).

5. Upper bounds

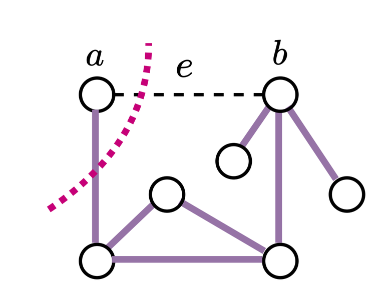
Key Question: What is an upper bound on the increase in MST weight when we add any set Z to X where $Z \subseteq \{e, \dots, m\}$ and $Z \cup X \in \mathcal{U}$, given that we only know that $c(X) = u$ and X contains no edges of weight $\ge w_e$? Call this bound $f(e, u)$.

Suppose we could answer this question when $Z = \{e\}$; call this bound $\delta(e, u)$. Then, we can 'integrate' it using **dynamic programming**, treating $\delta(e, u)$ as knapsack profit values.

$$f(e,u) = \begin{cases} 0 & \text{if } e > m, \\ f(e+1, u) & \text{if } i \le m \text{ and } c_e > B - u, \\ \max\{f(e+1,u), f(e+1, u+c_i) + \delta(e,u)\} & \text{otherwise.} \end{cases}$$

$\Longrightarrow$ We can **precompute** $f(e, u)$ and use the table of values to upper bound and prune nodes. Now, let's handle the $Z = \{e\}$ case, i.e., define $\delta(e, u)$.

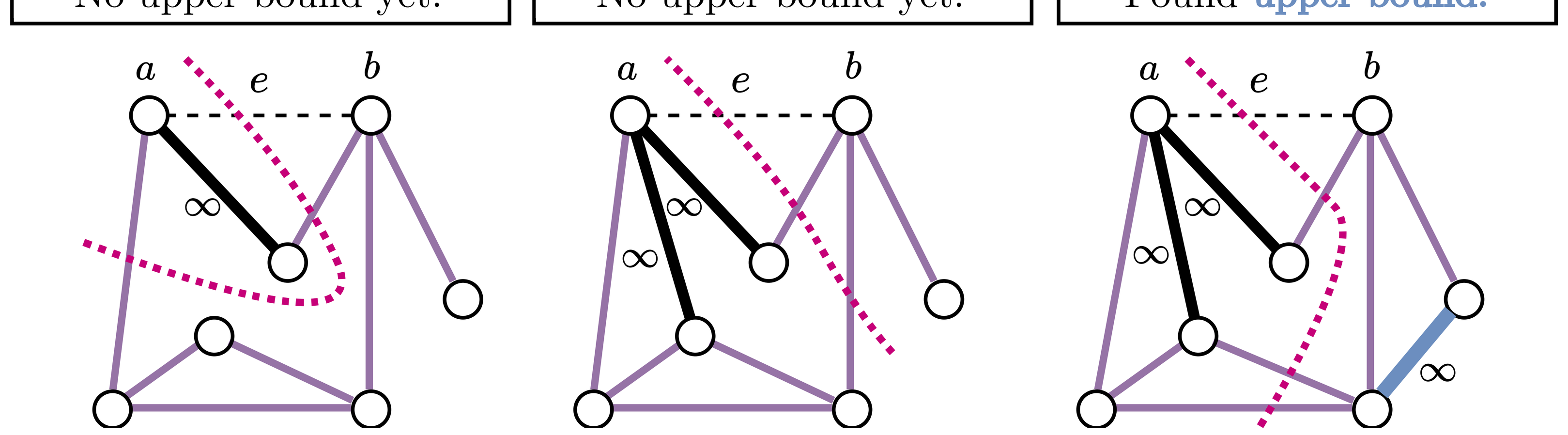
Consider only edges of weight $< w_e$. We know X is a subset of these edges, but not what X is precisely. **We only know** $\mathbf{c(X)}$.



Observe that e **cannot** be in the MST if the **min cost** a-b **cut** has cost $> c(X)$. If this happens, set $\delta(e, u) = 0$.

Suppose otherwise, and add edges of weight $> w_e$ incrementally by weight, with cost ∞. When the **min** a-b **cut** has cost $> c(X)$, the edge we just added is an upper bound.

1. Min cut has cost $\le c(X)$. No upper bound yet.	2. Min cut has cost $\le c(X)$. No upper bound yet.	3. Min cut has cost $> c(X)$. Found upper bound!



Let e' be the upper bound edge we found in the above procedure, and set $\delta(e, u) = w_{e'} - w_e$.

6. Computational results

We compare our solver to the computational results reported for the best known solvers, published in two previous papers. We implemented our solver in C++ and released it under an open-source license, along with all problem instances. All running times are in seconds.

Our solver is...

~30000x faster

vs the previous MIP-based solver [3].

		Our solver		Best from [3]	
n	$\tilde{m}$	Time	Opt%	Time	Opt%
40	78	**0.003**	**100**	0.04	100
40	118	**0.008**	**100**	0.26	100
80	160	**0.012**	**100**	0.49	100
80	240	**0.051**	**100**	1,943.23	75
80	305	**0.112**	**100**	1,469.49	83
160	318	**0.044**	**100**	97.28	100
160	476	**0.088**	**100**	3,829.95	48
160	624	**0.155**	**100**	6,066.43	20
200	398	**0.057**	**100**	280.76	100
200	596	**0.13**	**100**	3,757.74	48
200	784	**0.231**	**100**	7,200	0

Note: Each row corresponds to instances with n vertices and roughly $\tilde{m}$ edges. These instances are of a problem variant, but effectively, the budget B is large.

~100x faster

vs previous branch-and-bound solver [1].

	Our solver	Best from [1]
n	Time	Time
20	**0.027**	0.864
25	**0.049**	1.318
30	**0.036**	1.261
50	**0.405**	27.863
75	**1.848**	220.188
100	**4.475**	688.838
200	**5.286**	572.557
300	**40.097**	1,793.46
400	**89.085**	7,265.85

Note: These instances all have unit costs, and a budget B between 3 and 9. They are grouped by the number of vertices n; all graphs are complete graphs. Both solvers solved all instances.

We also randomly generated a variety of new instances to determine what qualities make an instance difficult to solve for our algorithm. We found that:

- The hardest instances have high density graphs and large budget relative to the costs.
- The magnitude of the costs and weights has little effect on difficulty.
- A few instances with only 25 vertices could not be solved within 1 hour.

7. Extensions

MST interdiction is a special case of **matroid interdiction** over a matroid M, in which

$$\mathcal{L} = \{Y \subseteq \{1, \dots, m\} \mid Y \text{ is a basis of } M\}.$$

- To solve a matroid interdiction problem on an arbitrary matroid, we can use the algorithm from Section 4 but with a modified dynamic data structure and upper bound. Depending on the matroid, this may be slower by a factor of $O(m)$ compared to Theorem 1.
- The bound $f(e, u)$ works for any matroid, as long as we can compute $\delta(e, u)$ for that matroid. We give a simple, weak definition of $\delta(e, u)$ which works for any matroid.
- For **partition matroids** we show that an exact $\delta(e, u)$ can be computed efficiently, and that $f(e, u)$ actually yields an exact solution in this case—no branch-and-bound is needed! As a consequence we show partition matroid interdiction is only weakly NP-hard.
- Some results extend to the variant where the leader forces elements to be **included** in the basis. We reduce this to matroid interdiction by taking the dual of the matroid.
- All of our results extend to the **min-max** variant. We can reduce between these two variants by negating the weights.

8. References

[1] Cristina Bazgan, Sonia Toubaline, and Daniel Vanderpooten. Efficient determination of the k most vital edges for the minimum spanning tree problem. *Computers & Operations Research*, 39(11):2888–2898, 2012.

[2] Greg N Frederickson and Roberto Solis-Oba. Increasing the weight of minimum spanning trees. *Journal of Algorithms*, 33(2):244–266, 1999.

[3] Ningji Wei, Jose L Walteros, and Foad Mahdavi Pajouh. Integer programming formulations for minimum spanning tree interdiction. *INFORMS Journal on Computing*, 33(4):1461–1480, 2021.

[4] Noah Weninger and Ricardo Fukasawa. A fast combinatorial algorithm for the bilevel knapsack problem with interdiction constraints. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 438–452. Springer, 2023.

LaTeX TikZposter