Solving the bi-objective prize-collecting Steiner tree problem with the ϵ-constraint method

Markus Leitner [a,1] Ivana Ljubić [b,2] Markus Sinnl [b,3]

[a] *Institute of Computer Graphics and Algorithms, Vienna University of Technology, Vienna, Austria*

[b] *Department of Statistics and Operations Research, University of Vienna, Vienna, Austria*

Abstract

In this paper, we study the bi-objective prize-collecting Steiner tree problem, whose goal is to find a subtree that minimizes the edge costs for building that tree, and, at the same time, to maximize the collected node revenues. We propose to solve the problem using an ϵ-constraint algorithm. This is an iterative mixed-integer-programming framework that identifies one solution for every point on the Pareto front. In this framework, a branch-and-cut approach for the single-objective variant of the problem is enhanced with warm-start procedures that are used to (i) generate feasible solutions, (ii) generate violated cutting planes, and (iii) guide the branching process. Standard benchmark instances from the literature are used to assess the efficacy of our method.

Keywords: bi-objective combinatorial optimization, Steiner tree problem, ϵ-constraint method

[1] Supported by the Austrian Science Fund (FWF) under grant I892-N23. Email: leitner@ads.tuwien.ac.at

[2] Supported by the APART Fellowship of the Austrian Academy of Sciences. Email: ivana.ljubic@univie.ac.at

[3] Email: markus.sinnl@univie.ac.at

1 Introduction. Problem definition

Many problems arising in the design of telecommunication, district heating, or water distribution networks, in forestry planning or even in image processing and system biology can be modeled as variants of the prize-collecting Steiner tree problem (PCSTP) (see e.g., [1] for a recent survey). The problem is defined as follows: Given an undirected graph $G = (V, E)$, with node revenues $r : V \to \mathbb{N}_0$, and edge costs $c : E \to \mathbb{N}$, find a subtree $G' = (V', E'), V' \subseteq V, E' \subseteq E$, such that $\sum_{v \in V'} r_v - \sum_{e \in E'} c_e$ is maximized. If node revenues and edge costs are measured using the same (e.g., monetary) units, this objective function corresponds to the net-worth maximization problem. However, difficulties arise if the "node revenues" are used to model some other aspects that cannot be so easily expressed using the monetary units. For instance, in wildlife corridor design (see [4]), the goal is to determine a subset of land parcels that connect areas of biological significance for a given species. In this case, nodes correspond to land parcels and their revenues correspond to habitats' suitability. Typically, in such applications, the problem is modeled as a budget-constrained prize-collecting Steiner tree problem (see [4]), where the budget limits are provided by decision makers, and the collected revenues are maximized subject to the given budget. From the perspective of a decision maker it would be preferable to consider both objectives, namely, the cost minimization and the revenue maximization, as part of the optimization process. In that case, a decision maker would be able to easily perform a sensitivity analysis regarding the variations of the budget and the corresponding deviations from the total revenue. Therefore, we propose to solve the PCSTP as a bi-objective optimization problem. In this work we study an iterative mixed-integer-programming method known as the ϵ-constraint method for solving the bi-objective PCSTP (BOPCSTP).

Problem formulation

The set of all feasible subtrees will be modeled using a rooted Steiner arborescence model originally proposed in [6], see also [7]. Let T be the set of *terminal nodes*, (i.e., nodes with positive revenue), and $V \setminus T$ be the set of potential *Steiner nodes* (i.e., nodes with zero revenue). We create a directed graph $(V \cup \{0\}, A)$ with a root 0 as follows: $A := \{(i, j), (j, i) \mid \forall e = \{i, j\} \in E\} \cup \{(0, t) \mid \forall t \in T\}$, and set $c_{ij} = c_{ji} = c_e$ and $c_{0t} = 0$, for all $t \in T$. Using binary variables x_{ij}, for all $(i, j) \in A$ to indicate whether an arc is part of a solution and binary variables y_v, for all $v \in V$ to indicate if a vertex v is part of a solution, the following inequalities are used to model the set of all feasible

subarborescences of G rooted at 0:

$$\mathcal{P} = \{(x,y) \in \{0,1\}^{|A|+|V|} \mid x(\delta^-(i)) = y_i, \forall i \in V, \;\; x(\delta^+(0)) = 1, \\ x(\delta^-(W)) \geq y_k, \forall W \subseteq V, k \in W\},$$

where $\delta^+(W)$ and $\delta^-(W)$ denote the outgoing and incoming cutset, resp., for any $W \subseteq V$, and $x(A') := \sum_{(i,j)\in A'} x_{ij}$ for each $A' \subseteq A$. The BOPCSTP can now be defined as follows:

$$(\min \sum_{a\in A} c_a x_a, \max \sum_{v\in V} r_v y_v) \text{ subject to } (x,y) \in \mathcal{P}. \tag{1}$$

To optimize such an objective vector, we turn to the concept of Pareto optimality [5]. In bi-objective optimization, a solution is Pareto optimal, if there is no other feasible solution which is better in one objective and not worse in the other. The corresponding objective vector is called non-dominated. If all other feasible solutions are better in at most one of the objectives, the solution is called weakly Pareto optimal. The set of all Pareto optimal solutions is called efficient set and the set of all non-dominated vectors is called Pareto front. Note that the size of the Pareto front is at most as big as the size of the efficient set, since one point on the Pareto front can be achieved by more than one Pareto optimal solution. Our goal for the BOPCSTP is to find one solution for every point on the Pareto front.

2 ϵ-constraint method for the BOPCSTP

To find one solution for every point on the Pareto front, we use the ϵ-constraint method which is popular for solving bi-objective combinatorial optimization problems, see e.g., [3,8]. The method works by repeatedly solving ϵ-constraint problems, which are obtained by adding one of the objectives as constraint. In our approach, the second objective is relaxed and added as constraint. Notice that the same solution will be optimal for the maximization problem of the second objective and for the minimization problem $\min \sum_{v\in V} r_v(1-y_v)$. This follows from the fact that the difference between the two objective values is $\sum_{v\in V} r_v$, which is a constant, and in the rest of the paper we will assume that the second objective function is given in minimization form. Hence, for a given $\epsilon \geq 0$, we obtain the following ϵ-constraint problem $P(\epsilon)$ for the BOPCSTP:

$$P(\epsilon): \qquad \min\{\sum_{a\in A} c_a x_a \mid (x,y) \in \mathcal{P}, \; \sum_{v\in V} r_v(1-y_v) \leq \epsilon\}$$

Every optimal solution for a given value of ϵ is weakly Pareto optimal [5]. In the ϵ-constraint method, this fact, together with the integrality of the input, is used to systematically find one solution for every point on the Pareto front by decreasing ϵ by Δ, where Δ is the greatest common divisor of all p_v, $v \in T$. Since we chose the second objective as ϵ-constraint, the starting boundary point is the Pareto optimal point with the minimum possible value for the first objective. This point corresponds to a single-terminal solution without edges and with the terminal that has the maximal node revenue, i.e., $\epsilon = \sum_{v \in V} r_v - r^*$, where r^* is the maximal node revenue (ties are broken randomly). The value of ϵ is then decreased by Δ in each iteration. The algorithm terminates when the Pareto optimal point with the minimum possible value of the second objective is reached. This point corresponds to a solution which is the minimum cost Steiner tree connecting all terminals, i.e., $\epsilon = 0$. An overview of the algorithm is given in Algorithm 1. The Pareto optimal solutions are kept in the set *Sol*.

Algorithm 1 ϵ-constraint method for the BOPCSTP

```
Sol ← ∅;   r* = max_{v∈V} r_v;   ε ← Σ_{v∈V} r_v − r*
while ε ≥ 0 do
    (x*, y*) ← argmin P(ε)
    Sol ← Sol ∪ (x*, y*)
    ε ← Σ_{v∈V} r_v(1 − y*_v) − Δ
Remove weakly Pareto optimal points from Sol
```

The weakly Pareto optimal solutions found by the algorithm have the same cost but different revenue, since we are only minimizing the first objective. There are different ways to deal with this. Since preliminary tests showed that the number of discovered weakly Pareto optimal solutions is relatively small in our instances, we simply remove them in a postprocessing phase.

3 Solution framework and acceleration features

The proposed ϵ-constraint method heavily depends on the choice of the mixed-integer-programming formulation $\mathcal{P}$ that is used to model feasible solutions, and on the solution method associated with it. Our implementation relies on the branch-and-cut approach from [7] for the single-objective PCSTP. In this section, we describe three acceleration features built in our solution framework that are enhancements of the approach from [7] with respect to the bi-objective optimization.

Our algorithm exploits information gained during the iterative ϵ-constraint framework (called "solution process" in the following): (i) to construct heuristic starting solutions, (ii) to define branching priorities, and (iii) to speed-up the generation of violated cutting planes. In the following, solving a single ILP by branch-and-cut will be denoted with "iteration".

Constructing feasible solutions
Potential starting solutions are kept in a list L, which is indexed by the first objective and contains only non-dominated solutions. L is initialized with the non-optimal incumbent solutions of the first iteration. In the following iterations, L is populated in two ways: The first way consists of adding, for the Pareto optimal solution computed in the previous iteration, a terminal t not in this solution with minimum connection costs. If there are several terminals with the same minimum connection costs, one with the highest revenue is chosen. Moreover, non-optimal incumbent solutions are also added to L. The starting solution for the current iteration is chosen as follows: Let c^* be the cost of the Pareto optimal solution of the previous iteration and let Δ_c be the greatest common divisor of all c_e, $e \in E$. We define $c' := c^* + \Delta_c$. If the entry $L[c']$ exists and is a feasible solution, we use it, otherwise we continue increasing c' repeatedly by Δ_c until a feasible solution is found, or at most hundred entries have proceeded.

Guiding the branching process
For guiding the branching process, we consider two strategies in which different priorities are associated with variables. In the basic branching strategy (BBS), no information from the ϵ-constraint framework is used, i.e., branching is guided as for the single-objective PCSTP: the highest branching priority is associated with terminal-variables, followed by the priorities for variables of Steiner nodes, followed by the priorities for arc-variables. In the advanced branching strategy (ABS), we collect the information gained in the previous iterations to guide the branching process. In the first iteration, branching priorities are initialized as above. Every time a terminal or Steiner node occurs in an optimal solution of an iteration, the branching priority of the corresponding variable is increased by one for the following iteration.

Detecting violated cutting planes
The idea of cut pools for bi-objective MIP approaches is to store cuts from previous iterations and reuse them [9]. We implemented a cut pool for directed cutset constraints in the following way: During the first branch-and-cut iter-

ation, all violated cuts detected by the maximum-flow algorithm are stored in the cut pool. In each following iteration, violated cutset constraints are then separated as follows: (i) check, if there exist violated cuts in the cut pool created in the previous iteration (ii) add all these cuts and delete them from the current cut pool, in case no such cuts exist, call the maximum-flow separation routine. In either case, all detected violated cuts are inserted in the new cut pool that is used in the next iteration.

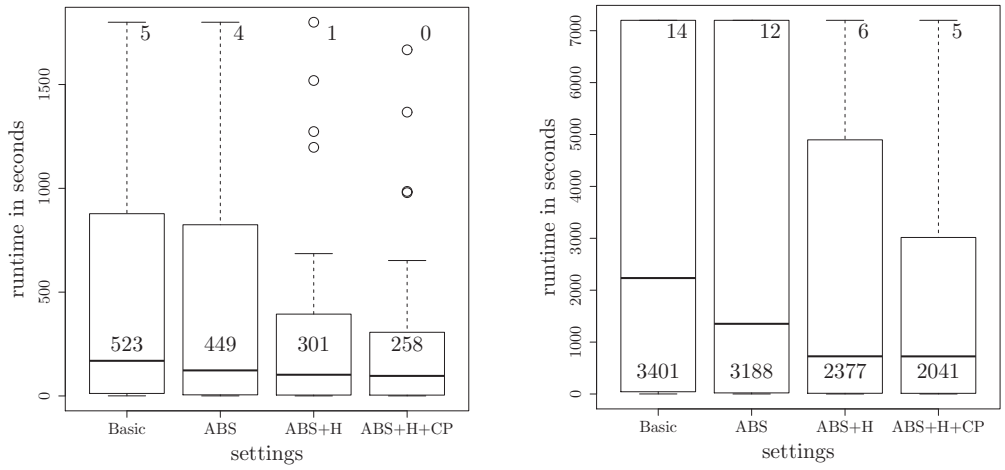
4 Computational results

The computational results have been obtained using a Intel Xeon X5500 with 2.67Ghz and 24GB RAM and CPLEX 12.4 as solver for the ILPs. As test instances we used the prize-collecting variant of the Steiner instance sets C and D from the OR-library [2]. Both the C and D sets contain 40 graphs divided into two groups, denoted by I and II. Instances in C have 500 nodes and between 625 and 12 500 edges, instances in D have 1 000 nodes and between 1 250 and 25 000 edges. The node revenues for terminals in the instances of type I are between 1 and 10 and in instances of type II between 1 and 100.

We tested the following four settings: "Basic" is a basic setting in which single-objective ϵ-constraint PCSTP is repeatedly solved, without taking the advantage of the bi-objective framework and learning from the previous iterations (i.e., initialization heuristics and cut pools are turned off and *BBS* is the branching strategy). Setting "ABS" is the Basic setting enhanced by the advanced branching strategy (*ABS*) and without initialization heuristics and cut pools. The setting "ABS+H" uses the initialization heuristics in addition to the *ABS* branching, and finally, in the setting "ABS+H+CP" all three bi-objective acceleration features are turned on (branching, heuristics and cut pools).

Figures 1a and 1b compare the four settings by showing boxplots of the running times for the 40 instances of the set C and D, respectively. The numbers shown in the boxes are the average running times over 40 instances, and the numbers shown above the boxes explain in how many out of 40 cases, we were not able to discover the complete Pareto front within the given timelimit. The timelimit was set to 1 800 and 7 200 seconds, for the set C and D, respectively. Regarding the set C, only with the setting "ABS+H+CP", the complete Pareto front of all 40 instances could be discovered, moreover the results also indicate that this is the fastest among all approaches. Instance C5-II turned out to be the most difficult one and only the last setting has finished within the timelimit for this instance. This instance also has the biggest

Pareto front by far, with 1 462 points on it, the second biggest front is from instance C5-I and has size 1 056. The size of the Pareto front is clearly one of the factors that heavily influences the running time. Further factors that determine the running time are the number of terminals and the density of instances, as this is the case for the single-objective PCSTP. Not using the three acceleration features of our bi-objective framework clearly worsens the overall performance. The instances of group D turned out to be much more difficult to solve, and even with the fastest setting, for five instances (D5-II, D10-II, D15-II, D19-II and D20-II) we were not able to explore the complete Pareto front within two hours. Among all instances solved within the time-limit, instance D5-I has the biggest Pareto front with 2 160 points in it. Again, the runtime does not only depend on the size of the Pareto front, e.g., with the last setting, instance D4-II with a Pareto front of size 1 860 is solved in 3 130 seconds, while instance D19-II, with a Pareto front of size 537 needs 5 924 seconds. Instances with five or ten terminals are easy to solve, however, the runtime is up to five times higher than the one needed for instances of set C with five or ten terminals.



(a) Instance set C

(b) Instance set D

Fig. 1. Runtimes for the four settings of our bi-objective framework.

5 Conclusions and future work

In this paper, we introduced the bi-objective PCSTP and showed how to transform a single-objective branch-and-cut algorithm to an ϵ-constraint algorithm to find one solution for every point on the Pareto front. Moreover, three

acceleration techniques that exploit information obtained during the solution process are presented. Computational results show that these techniques are crucial to discover the Pareto front for larger instances. The largest Pareto front discovered has a size of 2 160, the largest graph, for which the whole Pareto front has been found, has 1 000 nodes and 25 000 edges.

In a forthcoming paper, more potential acceleration techniques for the ϵ-constraint method will be explored, the method will be compared to other solution algorithms for bi-objective combinatorial optimization problems and the computational results will be analyzed in more detail.

References

[1] Álvarez-Miranda, E., I. Ljubić and P. Toth, *Exact approaches for solving robust prize-collecting steiner tree problems*, submitted (2012).

[2] Beasley, J. E., *OR-Library: Distributing Test Problems by Electronic Mail* (1990).

[3] Bérubé, J.-F., M. Gendreau and J.-Y. Potvin, *An exact ϵ -constraint method for bi-objective combinatorial optimization problems: Application to the Traveling Salesman Problem with Profits*, European Journal of Operational Research **194** (2009), pp. 39–50.

[4] Dilkina, B. and C. Gomes, *Solving connected subgraph problems in wildlife conservation*, in: *CPAIOR 2010*, Springer LNCS **6140**, 2010, pp. 102–116.

[5] Ehrgott, M. and M. M. Wiecek, *Mutiobjective Programming*, in: F. S. Hillier, editor, *Multiple Criteria Decision Analysis: State of the Art Surveys*, International Series in Operations Research & Management Science **78**, Springer New York, 2005 pp. 667–708.

[6] Fischetti, M., *Facets of two Steiner arborescence polyhedra.*, Mathematical Programming **51** (1991), pp. 401–419.

[7] Ljubić, I., R. Weiskircher, U. Pferschy, G. W. Klau, P. Mutzel and M. Fischetti, *An Algorithmic Framework for the Exact Solution of the Prize-Collecting Steiner Tree Problem*, Mathematical Programming **105** (2006), pp. 427–449.

[8] Neumayer, P. and D. Schweigert, *Three algorithms for bicriteria integer linear programs*, OR Spectrum **16** (1994), pp. 267–276.

[9] Riera-Ledesma, J. and J. Salazar-González, *The biobjective travelling purchaser problem*, European Journal of Operational Research **160** (2005), pp. 599–613.