





JAAP PEDERSEN<sup>1</sup>  
NIELS LINDNER<sup>2</sup>  
DANIEL REHFELDT<sup>3</sup>  
THORSTEN KOCH<sup>4</sup>

# Comparing Branching Rules for the Quota Steiner Tree Problem with Interference

---

<sup>1</sup>  0000-0003-4047-0042  
<sup>2</sup>  0000-0002-8337-4387  
<sup>3</sup>  0000-0002-2877-074X  
<sup>4</sup>  0000-0002-1967-0077

Zuse Institute Berlin  
Takustr. 7  
14195 Berlin  
Germany

Telephone: +49 30 84185-0  
Telefax: +49 30 84185-125

E-mail: [bibliothek@zib.de](mailto:bibliothek@zib.de)  
URL: <http://www.zib.de>

ZIB-Report (Print) ISSN 1438-0064  
ZIB-Report (Internet) ISSN 2192-7782

# Comparing Branching Rules for the Quota Steiner Tree Problem with Interference

Jaap Pedersen<sup>[0000-0003-4047-004]</sup>, Niels Lindner<sup>[0000-0002-8337-4387]</sup>,  
Daniel Rehfeldt<sup>[0000-0002-2877-074X]</sup> and Thorsten Koch<sup>[0000-0002-1967-0077]</sup>

**Abstract** Branching decisions play a crucial role in branch-and-bound algorithms for solving combinatorial optimization problems. In this paper, we investigate several branching rules applied to the Quota Steiner Tree Problem with Interference (QSTPI). The Quota Steiner Tree Problem (QSTP) generalizes the classical Steiner Tree Problem (STP) in graphs by seeking a minimum-cost tree that connects a subset of profit-associated vertices to meet a given quota. The extended version, QSTPI, introduces interference among vertices: Selecting certain vertices simultaneously reduces their individual contributions to the overall profit. This problem arises, for example, in positioning and connecting wind turbines, where turbines possibly shadow other turbines, reducing their energy yield. While exact solvers for standard STP-related problems often rely heavily on reduction techniques and cutting-plane methods – rarely generating large branch-and-bound trees – experiments reveal that large instances of QSTPI require significantly more branching to compute provably optimal solutions. In contrast to branching on variables, we utilize the combinatorial structure of the QSTPI by branching on the graph’s vertices. We adapt classical and problem-specific branching rules and present a comprehensive computational study comparing the effectiveness of these branching strategies.

---

Jaap Pedersen  
Zuse Institute Berlin, Germany e-mail: [pedersen@zib.de](mailto:pedersen@zib.de)

Niels Lindner  
Freie Universität Berlin and Zuse Institute Berlin, Germany

Daniel Rehfeldt  
IVU Traffic Technologies, Berlin, Germany

Thorsten Koch  
Technische Universität Berlin and Zuse Institute Berlin, Germany

## 1 Introduction

In this paper, we investigate several branching rules for the Quota Steiner Tree Problem with Interference (QSTPI) introduced in [1]. The original Quota Steiner Tree Problem (QSTP) generalizes the classical Steiner Tree Problem (STP) in graphs by seeking a minimum-cost tree that connects a subset of profit-associated vertices to meet a given quota [2]. In the QSTPI, interference between the chosen vertices may decrease their individual contribution to the total profit. Such a scenario occurs, for example, in optimizing placement and connections of wind turbines, where turbines possibly shadow other turbines, reducing their energy output.

## 2 Problem formulation

Let  $G = (V, E)$  be an undirected graph, whose edge set  $E$  is associated with costs  $c : E \rightarrow \mathbb{R}_{\geq 0}$  and whose vertex set  $V$  is partitioned into a set of *fixed terminals*  $T_f$ , a set of *potential terminals*  $T_p$  associated with *quota profits*  $q : T_p \rightarrow \mathbb{R}_{> 0}$ , and a set of additional *Steiner nodes*. Moreover, a *quota*  $Q \in \mathbb{R}_{> 0}$  is given. For every  $i \neq j \in T_p$ , let  $I_{ij} \in \mathbb{R}_{\geq 0}$  be the interference experienced by  $j$  if  $i$  is chosen, reducing the quota profit  $q_j$  by  $I_{ij}$ . Finally, let  $d_{ij} \in \mathbb{R}_{\geq 0}$  be the distance between two vertices in  $T_p$  and let  $D_{min}$  be a minimum distance. The QSTPI aims to find a tree  $S = (V', E') \subseteq G$  that contains  $T_f$ , minimizes the costs  $C(S) := \sum_{(i,j) \in E'} c_{ij}$ , guarantees that  $d_{ij} \geq D_{min}$  for all  $i, j \in V' \cap T_p$  with  $i \neq j$ , and fulfills the quota taking the induced interference into account, i.e.,  $Q(S) := \sum_{i \in T_p \cap V'} \left( q_i - \sum_{j \in T_p \cap V' \setminus \{i\}} I_{ji} \right) \geq Q$ .

The QSPTI can be formulated as a mixed-integer program (MIP) by transforming the original graph into a directed graph  $D = (V, A)$ , replacing each edge by a pair of antiparallel arcs. For a subset of vertices  $W \subseteq V$ , we denote by  $\delta^+(W)$  the set of outgoing arcs and by  $\delta^-(W)$  the set of incoming arcs. Following the transformation in [3] and [1], in which for every potential terminal  $i$  a new fixed terminal  $i'$  is added to  $T_f$ , and for each newly-added fixed terminal  $i'$  a directed arc  $(i, i')$  of zero costs and a directed arc  $(r, i')$  of zero costs are added to  $A$ , where  $r \in T_f$  is an arbitrary root node (see Fig. 1), the MIP formulation of the QSTPI is given as:

$$\min \quad c^T x \quad (1)$$

$$\text{s.t.} \quad x(\delta^-(W)) \geq 1 \quad \forall W \subset V, r \notin W, |W \cap T_f| \geq 1 \quad (2)$$

$$\sum_{i \in T_p} q_i x_{r, i'} + I_{tot} \leq \sum_{i \in T_p} q_i - Q \quad (3)$$

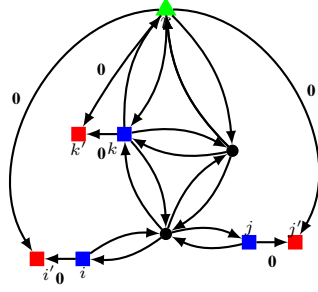
$$I_{tot} \geq \sum_{i \in T_p} \sum_{j \neq i \in T_p} I_{i,j} m_{i,j} \quad (4)$$

$$m_{i,j} \geq x_{i, i'} + x_{j, j'} - 1 \quad \forall i < j \in T_p, \quad (5)$$

$$x_{i, i'} + x_{j, j'} \leq 1 \quad \forall i, j \in T_p, i \neq j, d_{ij} < D_{min} \quad (6)$$

$$x_{i,j} \in \{0, 1\} \quad \forall (i, j) \in A, \quad m_{i,j} \geq 0 \quad \forall i < j \in T_p. \quad (7)$$

Constraints (2) ensure connectivity and that the fixed terminals are included. The quota constraint respecting the total interference is described by (3). Constraints (4)-(5) represent the linearization of the interference loss  $I_{ij}x_{i,i'}x_{j,j'}$  when choosing the potential terminals  $i$  and  $j$ . Constraints (6) guarantees that no two potential terminals are chosen that are too close to each other.



**Fig. 1:** QSTPI instance. Root node in green; Fixed terminals  $T_f$  in red; Potential terminals  $T_p$  in blue; Steiner nodes in black.

### 3 Vertex-based branching

Branching is one of the key elements in current state-of-the-art branch-and-bound (B&B) and branch-and-cut algorithms. Typically, two decisions are made: How to split the problem into subproblems (branching), and which subproblem to select (node selection). In this paper, we focus on branching strategies. For a comprehensive overview, the reader is referred to, e.g., [4]. Most state-of-the-art general MIP solvers commonly branch on variables, as it only requires changing the bounds of a variable without any further knowledge of the problem structure. Considering STP-related problems, it has proven to be effective to devise a branching strategy based on the vertices of the original graph  $G$  instead of branching on a single MIP variable [5]: The branching decision incorporates then the much stronger decision of including (removing) a vertex of  $G$  into (from) the solution. Effectively, branching on a vertex implies the following constraints for general STP-related problems:

$$\text{Including vertex } i: \sum_{a \in \delta^-(i)} x_a = 1 \quad (8)$$

$$\text{Excluding vertex } i: \sum_{a \in \delta^-(i)} x_a + \sum_{a \in \delta^+(i)} x_a = 0. \quad (9)$$

By including the vertex  $i$ , the vertex  $i$  has exactly one incoming arc, see (8). By excluding the vertex, no arc can enter or leave the vertex, see (9). Considering the special structure of the QSTPI, the following additional implications arise by branching on  $i \in T_p$ :

$$\text{Including vertex: } x_{i,i'} = 1 \wedge x_{r,i'} = 0 \quad (10)$$

$$\sum_{a \in \delta^-(j)} x_a + \sum_{a \in \delta^+(j)} x_a = 0 \quad \forall j \neq i \in T_p, d_{ij} < D_{\min} \quad (11)$$

$$\text{Excluding vertex: } x_{i,i'} = 0 \wedge x_{r,i'} = 1 \quad (12)$$

By including a vertex  $i$ , all potential terminals  $j \in T_p$  which are too close to the vertex  $i$  ( $d_{ij} < D_{\min}$ ) are immediately excluded by setting all their incoming and outgoing arcs to zero, see (11).

### 3.1 Branching strategies

In the following, let  $H$  be the current problem in the B&B algorithm and  $H^+$  and  $H^-$  be the up and down child of  $H$  which are created by including and excluding a vertex, respectively. A typical goal of branching strategies is to choose variables which have the most impact on dual bounds. Let  $\Delta^+ := \bar{c}_{H^+} - \bar{c}_H$  and  $\Delta^- := \bar{c}_{H^-} - \bar{c}_H$  be the upgain and the downgain of a branching decision or the estimate of that decision. These values are usually projected on a single score value  $s$ . Commonly-used and problem-independent branching strategies are, for example, *infeasible branching* (maxinfeas), *full strong branching* (fullstrong), *pseudocosts with strong branching initialization* (pscst), or its generalization *reliability branching* (relpscsts), see, e.g., [4].

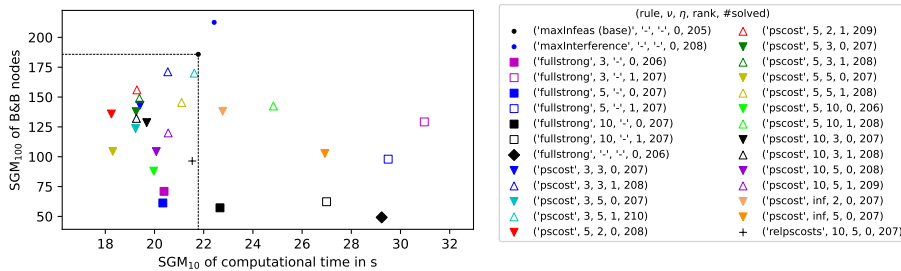
The current state-of-the-art STP-solver SCIP-JACK [5] branches on the most infeasible vertex. Let  $f_i := \sum_{a \in \delta^-(i)} \bar{x}_a$  be the inflow of vertex  $i$  for the optimal solution  $\bar{x}$  of the LP-relaxation of the current subproblem  $H$ . The *infeasibility* score  $s_i$  of a vertex is determined by  $s_i = |f_i - 0.5|$ . Strong branching is performed on the  $\nu$  most promising vertices (see below) calculating the score by  $s_i = \max\{\Delta^+, \varepsilon\} \cdot \max\{\Delta^-, \varepsilon\}$  with  $\varepsilon = 10^{-6}$ . Introducing  $\varepsilon$  is necessary to avoid that scores are set to zero if one of  $\Delta^-$  and  $\Delta^+$  is zero. Considering pseudocosts with strong branching initialization, strong branching is performed as before until a B&B-tree depth of  $\eta$ . Reaching a tree depth of  $\eta$ , an average up- and downgain is calculated on all non-initialized vertices. After the initialization phase, the score of a vertex  $i$  is determined by  $s_i = (1 - \mu) \min\{(1 - f_i)\Delta^+, f_i\Delta^-\} + \mu \max\{(1 - f_i)\Delta^+, f_i\Delta^-\}$  using SCIP's value of  $\mu = \frac{1}{6}$ , which surprisingly yields better results than using the commonly-used product scoring function shown above. In reliability branching, a vertex is called reliable if it has been branched on at least  $\eta_{rel}$  times. If a vertex is not reliable, its score is calculated by strong branching, incrementing its counter by one. Again, this is only done on the  $\nu$  best vertices. If a vertex is reliable, its score is determined by its pseudocost as before.

Considering the QSTPI, we propose the following problem-specific branching strategies that we call *most interference branching*. Given an optimal solution  $\bar{x}$  of the LP-relaxation at a subproblem  $H$ , a score for each vertex  $i \in T_p$  is computed based on its associated interference by  $s_i = \sum_{j \neq i \in T_p} I_{ij} \bar{x}_{j,j'} \bar{x}_{i,i'}$ . As the interference is one of the key problem-specific attributes, branching on the vertex that causes the most interference is likely to induce a significant change in the LP solution, and thus in the dual bounds of the problem.

## 4 Computational Study

We have implemented all the strategies described in §3.1 in SCIP-JACK using SCIP 8.0.1 [6] with CPLEX 12.10 [7] as LP solver. When fullstrong is performed on the  $\nu$  most promising vertices we sort the vertices by 1) most infeasible inflow or 2) most interference. The computations are performed on the small- and medium-sized instances presented in [1]. Instances that are solved at the root node are filtered out resulting in 277 instances in total. All computational experiments are executed single-threaded in a non-exclusive mode on a cluster with Intel XeonGold 6342 CPUs running at 2.8 GHz, where four CPUs and 64 GB of RAM are reserved for each run, with a time limit of 21 600 seconds.

Fig. 2 shows the shifted geometric mean (SGM) of time (shift=10) and B&B nodes (shift=100) of the instances that are solved by the standard most infeasible strategy (205 in total) for the examined settings. Additionally, the legend shows the number of instances solved by each setting. Choosing the vertex with the highest interference contribution instead of the most infeasible one, both the computing time and the number of nodes increase. However, three more instances are solved. As expected, full strong branching reduces the node count by over 70 % number of nodes, but computing time is over 30 % higher. Using only a subset of vertices to perform full strong is helpful if they are chosen by their infeasibility and not by their interference. Almost all settings for the pseudocost branching yield better result in terms of computing time and node count than the most infeasible branching. Although choosing the vertices based on their infeasibility gives better SGMs for time and nodes, sorting by interference solves at least an instance more with ('pscost', 5, 2, 1) solving the most instances (210).



**Fig. 2:** Shifted geometric mean (SGM) for computational time (SGM<sub>10</sub>) vs. B&B nodes (SGM<sub>100</sub>) for all tested settings. The dotted lines mark the threshold with respect to the baseline rule *most infeasible branching*. Additional to the settings description, i.e., *rule*, number most promising vertices that are used in strong branching  $\nu$ , the tree depth (reliabilty parameter in case of *relpscost*)  $\eta$ , and the ranking strategy (0: most infeasible; 1: most interference), the legend also states the number of solved instances.

## 5 Conclusion

In this paper, we investigate several branching rules on the QSTPI in the current state-of-the-art STP solver SCIP-JACK making substantial improvements with respect to both node count and computing time. We introduce the problem-specific concept of maximum interference for ranking the vertices which helps to solve a few instances more within the timelimit. In this study, we focused on the effectiveness with regard to the QSTPI. In the future, comprehensive computational experiments should be performed for general, yet unsolved STP-related instances.

## Acknowledgements

The work for this article has been conducted in the Research Campus MODAL funded by the German Federal Ministry of Education and Research (BMBF) (fund numbers 05M14ZAM, 05M20ZBM, 05M2025).

## References

1. Pedersen, J., Lindner, N., Rehfeldt, D. & Koch, T. *Integrated Wind Farm Design: Optimizing Turbine Placement and Cable Routing with Wake Effects* 2025. arXiv: 2501.07203 [math.OC]. <https://arxiv.org/abs/2501.07203>.
2. Johnson, D. S., Minkoff, M. & Phillips, S. *The Prize Collecting Steiner Tree Problem: Theory and Practice* in *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms* (Society for Industrial and Applied Mathematics, USA, Feb. 2000), 760–769. ISBN: 978-0-89871-453-1.
3. Pedersen, J., Weinand, J. M., Syranidou, C. & Rehfeldt, D. An Efficient Solver for Large-Scale Onshore Wind Farm Siting Including Cable Routing. *European Journal of Operational Research* **317**, 616–630 (May 2024).
4. Achterberg, T., Koch, T. & Martin, A. Branching rules revisited. *Operations Research Letters* **33**, 42–54 (2005).
5. Rehfeldt, D. *Faster Algorithms for Steiner Tree and Related Problems: From Theory to Practice* PhD thesis (Technische Universität Berlin, 2021). <https://opus4.kobv.de/opus4-zib/frontdoor/index/index/docId/8514>.
6. Bestuzheva, K. *et al.* Enabling Research through the SCIP Optimization Suite 8.0. *ACM Transactions on Mathematical Software* (Mar. 2023).
7. IBM ILOG. *V12.10: User's Manual for CPLEX* 2022. <https://www.ibm.com/analytics/cplex-optimizer>.