The Multi-Stop Station Location Problem: Exact Approaches

Erik Mühmer*[1], Miriam Ganz[1], Marco E. Lübbecke[1], and Felix J. L. Willamowski[1]

[1]RWTH Aachen University, Chair of Operations Research, Kackertstr. 7, D-52072 Aachen, Germany

Abstract

The multi-stop station location problem (MSLP) is motivated by infrastructure planning for range-limited transportation services such as electric intercity buses. We study this foundational infrastructure-siting and routing problem in which a fixed set of ordered trips must be made feasible under vehicle-range limits by installing shared stations and, if necessary, allowing detours between consecutive planned stops. We present an arc-based mixed-integer linear program (MILP) and two pattern-based reformulations, one based on complete trip paths and one based on trip segment paths. The latter yields shortest path pricing problems, and both are combined in a nested approach. For the pattern-based reformulations, we develop branch-price-and-cut algorithms with a problem-specific primal heuristic, branching strategies, and, for the segment-based formulation, additional cutting planes. On synthetic benchmarks derived from TSPLIB, the pattern-based approaches are more scalable than the compact MILP, and the nested approach performs best overall, solving 82.4% of all instances to optimality within the time limit. On a GTFS-based intercity bus instance with 1644 nodes and 2408 trips, the nested approach proves optimality in 40.06 hours. These results show that exact methods can solve the MSLP beyond toy instances on public-data transportation networks.

Keywords: Multi-Stop Station Location · Transportation Infrastructure Planning · Charging Station Placement · Branch-Price-and-Cut

1 Introduction

The *(directed) multi-stop station location* problem (MSLP), introduced by Willamowski, Ganz, and Mühmer (2020), is a NP-hard optimization problem on a graph. It consists of a network design (or location) and a pathfinding (or routing) component. The task is to enable *trips*, given by sequences of nodes (*stops*), respecting maximum ranges, and minimizing cost. For each trip the maximum distance (*range*) that can be traveled through

*Corresponding author; e-mail: muehmer@or.rwth-aachen.de

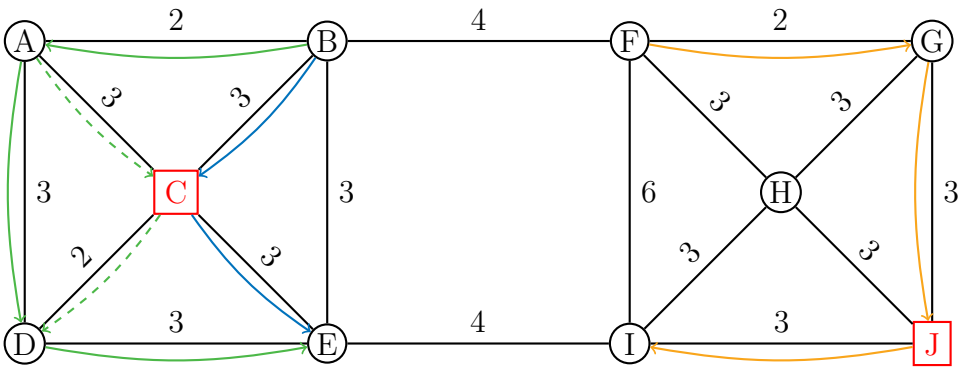


Figure 1: Depicted is a MSLP instance with 10 nodes and 3 trips. The first trip (green) starts at node B and has to visit A, D, and E. The second trip (blue) starts at node B and has to visit C and E. The third trip (orange) starts at node F and has to visit G, J, and I. These given nodes, the *stops*, have to be visited by the respective trips in the given order, and (possibly several) intermediate nodes, the *stations*, between two stops may be added. Detours from the shortest connection between consecutive stops are not only allowed, but may be necessary for feasibility. The ranges for all trips are equal to 5 and a station can be installed at every node. A feasible solution contains the red squared nodes that represent the installed stations, and the dashed lines that indicate a detour that is necessary to visit a station.

the graph without visiting special nodes (*stations*) is bounded. The feasibility of trips with respect to the range can be ensured by visiting (potentially several) stations between any two stops of a trip, if necessary. This may incur a (potentially considerable) detour. At stations, the traveled distances are reset, that is, the full range is available again. It is part of the problem to decide which stations to install. Once installed, a station can be visited by any trip. The total installation cost and trip costs for traversing edges must be minimized. Figure 1 shows an instance of the MSLP.

An important motivation for studying the MSLP is the siting of charging infrastructure for a given set of intercity trips operated by electric vehicles (EVs). In such services, the planning problem is not merely to make isolated origin-destination movements feasible. Instead, each service consists of an ordered sequence of planned stops that have to be visited in a prescribed order, while limited vehicle range may force intermediate charging visits between consecutive stops. The infrastructure decision is shared across all trips, because once installed, a station can be used by the entire service network. This creates a joint transportation network design and routing problem in which the operator has to trade off infrastructure cost against the detours needed to maintain service feasibility. While the range of EVs without recharging may be sufficient for intracity travel, long distances are more challenging. For example, an intercity bus service that wants to electrify its fleet needs to ensure that all planned trips can be realized with respect to the ranges of the EVs. In particular, it is desirable to install charging infrastructure in a cost-effective manner while limiting charging-related detours.

There is abundant research on locating charging or alternative fuel stations. However, the MSLP combines three structural features that are central to our setting: predefined ordered multi-stop trips, shared station-siting decisions across trips, and detours between consecutive planned stops. As we contrast below, this structure is not captured directly by origin-destination refueling models, flow-covering formulations, or relay-type network design models.

In this paper, we make three main contributions. First, we introduce an arc-based mixed-integer linear program (MILP) as a compact mathematical formulation of the MSLP and derive two pattern-based reformulations, one that assigns complete paths to trips and one that assigns paths to parts (*segments*) of a trip. The latter leads to efficiently solvable shortest path pricing problems, and we also combine both reformulations in a nested approach. Second, we develop branch-price-and-cut algorithms for all pattern-based formulations. These algorithms include a problem-specific primal heuristic, traditional and problem-specific branching strategies and, for the second pattern-based formulation, additional problem-specific cutting planes. Third, computational experiments on synthetically generated instances compare the performance of the proposed formulations, and a large real-world instance shows that the nested approach can solve a practically relevant case to proven optimality. Our implementation relies on SCIP (Bestuzheva et al., 2021).

In this paper, we deliberately adopt a foundational scope. Many practically motivated optimization problems like facility location or electric vehicle routing were first studied in a clean core form before richer operational variants were addressed. We follow the same approach by isolating the shared infrastructure-routing core of the MSLP and developing rigorous and computationally viable solution methods for that fundamental problem. Accordingly, the present model abstracts from several operational features, including charging times, timetable consistency, passenger-demand feedback, and limited charging resources at stations. The assumptions we do retain, such as full recharging, linear driving costs, and the absence of station-capacity constraints, are common in the literature (see Section 1.1 for details).

The remainder of this paper is organized as follows. In Section 1.1, we contrast the related literature against our work. In Section 2, we formally define the MSLP and introduce an arc-based and two pattern-based formulations. The algorithmic components we use to solve the pattern-based formulations with branch-price-and-cut are described in Section 3. In Section 4, we present our experiments and discuss the results. Section 5 concludes our work.

1.1 Related Work

Planning problems related to electromobility or alternative fuels have received considerable attention in the past two decades. On the one hand, the limited ranges and the scarcity of the necessary refueling infrastructure, such as charging stations, have been incorporated into existing and well-studied problems (e.g., vehicle routing). On the other hand, new problems and models have emerged and been studied. In the following, we focus on related work that locates stations in a network to ensure feasibility with respect to length, distance, or similar constraints.

For a better overview and to highlight differences to our work, we group related literature into categories based on the interpretation of trips.

1. Approaches that do not consider *any predefined trips* at all. For a given network, stations are placed such that all nodes of certain node sets are reachable from each other with respect to the limited range. These coverage based approaches typically enforce connectivity in a graph with respect to range limitations. The objective function varies, but typically takes into account the infrastructure cost. Example works in this category are by Funke, Nusser, and Storandt, 2015; Kınay, Gzara, and

Alumur, 2021; Storandt and Funke, 2013; Y.-W. Wang and Chuah-Chih Lin, 2009; Y.-W. Wang and C.-R. Wang, 2010; You and Hsieh, 2014.

2. Literature dealing with *origin-destination pairs* that have to be connected with respect to range limitations. The first problem of this category is the flow-refueling location problem introduced by Kuby and Lim (2005). For given origin-destination pairs with assigned flow volumes one locates refueling stations on a network such that the total (refueled) flow volume is maximized. Each flow is assumed to follow a *fixed shortest path* from origin to destination. This problem received much attention in the literature and several extensions and adaptations have been studied (Capar and Kuby, 2012; Kuby and Lim, 2007; Kuby, Lines, et al., 2009; Lim and Kuby, 2010; S. A. MirHassani and Ebrazi, 2013; Rose et al., 2020; Scheiper, Schiffer, and Walther, 2019; Upchurch and Kuby, 2010; Upchurch, Kuby, and Lim, 2009; Vries and Duijzer, 2017). Kim and Kuby (2012) propose an extension that allows flows to deviate from the assigned shortest path to refuel. The authors propose a MILP that requires precomputation of possible detour paths. Since this precomputation step is very expensive, Kim and Kuby (2013) also propose heuristic approaches. Again, many alternative approaches as well as extensions were studied (Göpfert and Bock, 2019; Hosseini, S. MirHassani, and Hooshmand, 2017; Huang, S. Li, and Qian, 2015; Cheng-Chang Lin and Chuan-Chih Lin, 2018; Nordlund, Tassiulas, and Lange, 2023; Yıldız, Arslan, and Karaşan, 2016). Yıldız, Arslan, and Karaşan (2016) present a branch-and-price approach to an extension that respects a distance threshold that limits the length of a deviation path. Subsequently, Göpfert and Bock (2019) present a branch-and-cut approach to the problem and compare the performance with the results of Yıldız, Arslan, and Karaşan (2016).

   Another class of related problems, not motivated by alternative fuel infrastructure design, are network design problems that aim to locate special vertices or arcs that, for example, refresh a signal. Cabral et al. (2007) introduce the network design problem with relays. Given an undirected graph with edge costs and lengths, one must select edges and locate relays at vertices such that given origin-destination pairs are connected by a path that does not violate a length constraint on the travel distance of a signal without passing through a relay. The sum of the edge and relay costs is minimized, where each selected edge and relay just have to be paid once. Many variations of the problem have been studied, e.g., in the context of optical networks (S. Chen, Ljubić, and Raghavan, 2010; S. Chen, Ljubić, and Raghavan, 2015; Hartstein, Shalom, and Zaks, 2016), or in the presence of multiple commodities and other additional constraints (Kabadurmus and Smith, 2016; X. Li et al., 2017; Zheng et al., 2017). While many heuristic approaches exist, recent work also presents exact approaches based on branch-and-cut and branch-and-price (Leitner et al., 2019; Leitner et al., 2020; X. Li et al., 2017; Nigam and Agarwal, 2014; Yıldız, Karaşan, and Yaman, 2018). In addition, there is more related work that is not directly linked to the already mentioned problems, like the work of Kınay, Gzara, and Alumur (2021), who consider origin-destination round trips and minimize the total required charging in addition to the infrastructure cost.

3. *Vehicle routing* literature that additionally considers refueling and locating refueling stations. For an extensive overview we refer to the work of Schiffer, Schneider, et al. (2019). Location routing problems combine the routing of vehicles and the placement of stations (for refueling at intermediate stops). In recent years, many works

dealing with location routing problems and solution approaches were published (Almouhanna et al., 2020; Calik et al., 2018; Y. Chen et al., 2021; Hof, Schneider, and Goeke, 2017; Schiffer and Walther, 2017; Worley, Klabjan, and Sweda, 2012; Yang and Sun, 2015).

4. Literature considering *predefined trips with more than two stops* (origin and destination). This literature is often motivated by (fast-)charging electric buses. Trips are predefined as ordered lists of stops, at each of which a charger could be placed. In this context, deviations from the already existing routes are not allowed (Dinç Yalçin et al., 2023; Esmaeilnejad, Kattan, and Wirasinghe, 2023; He, Song, and Liu, 2025; Kunith, Mendelevitch, and Goehlich, 2017; Tzamakos, Iliopoulou, and Kepaptsoglou, 2023). In some cases, even charging is only allowed when no passengers are transported (Alamatsaz, Quesnel, and Eicker, 2024; Stumpe, 2024). Most recently, a closely related work studies a continuous station location problem with intermediate stops along trips, but with partly different assumptions (Nayeem et al., 2025). On the one hand, the authors also assume full recharging, linear driving costs, and no capacity constraints at stations. On the other hand, they consider repeatedly driven roundtrips whose length is bounded by the driving range of the vehicles, i.e., intra-city traffic is the main application scenario. This assumption clearly separates their work from ours, as it suffices that each trip is covered by at least one station.

Most related work falls into the first two categories. However, the approaches tackling those kinds of problems are not suited for the MSLP. The instance depicted in Figure 1 clarifies the differences. For the entire graph a coverage approach would install at least three stations (I to G requires H or J, and A to G requires B and F or E and I), whereas the MSLP just requires two stations. Furthermore, for the subgraph with the nodes $\{\mathrm{A}, \mathrm{B}, \mathrm{C}, \mathrm{D}, \mathrm{E}\}$ no station is required for coverage (but the MSLP requires one). As a solution of a coverage based approach is also feasible for an instance considering origin-destination pairs, this example also separates the MSLP from such problems. Literature of the third category deals with problems that share the typical vehicle routing assumptions. In contrast to those problems, the MSLP does not consider depots or vehicles that have to be assigned to customers, i.e., in the context of the MSLP customers are already assigned to vehicles (in a fixed order) and can be visited multiple times. Hence, the routing part of the MSLP differs as it allows only routing to stations between two stops. Category four is most related, but typically does not allow detours.

The general MSLP was introduced by Willamowski, Ganz, and Mühmer (2020). The authors discuss theoretical results and propose an approximation algorithm to solve the MSLP. We integrate the idea of their approximation algorithm into our approaches. To the best of our knowledge, ours are the first exact approaches to solve a problem like the MSLP that considers trips with predefined multiple stops (and not only origin-destination pairs) and allows for deviations to visit additional charging stations.

2 Formulations

Before introducing mathematical formulations to capture the MSLP, we first formally define the problem (Willamowski, Ganz, and Mühmer, 2020). The directed multi-stop station location problem is given by a directed graph $G = (V, E)$, edge costs $c^E : E \rightarrow$

$\mathbb{Q}_{\geq 0}$, edge lengths $\ell : E \to \mathbb{Q}_{\geq 0}$, a set $F \subseteq V$ of nodes at which stations can be placed, station costs $c^F : F \to \mathbb{Q}_{\geq 0}$, and a set of trips T. Each trip $t \in T$ is given by a sequence of stops $\mathcal{S}_t = (v_1, \ldots, v_{m_t}) \in V^{m_t}$ for $m_t \in \mathbb{N}_{\geq 2}$ and its associated maximum range $b_t \in \mathbb{Q}_{\geq 0}$. Let $\mathcal{S}_t[i]$ with $i \in [m_t] := \{1, \ldots, m_t\}$ denote the i-th element of $\mathcal{S}_t$. For convenience, we use the abbreviation $t[i] := \mathcal{S}_t[i]$. Each trip t consists of segments $S_t := [m_t - 1]$, which are implicitly given by the stops, i.e., the start and the end of a segment correspond to two consecutive stops of the trip. The goal is to select stations $F^* \subseteq F$ and for each trip $t \in T$ a path

$$\mathcal{P}_t = (v_1, f_1^1, \ldots, f_1^{k_1}, v_2, f_2^1, \ldots, f_2^{k_2}, \ldots, v_{m_t-1}, f_{m_t-1}^1, \ldots, f_{m_t-1}^{k_{m_t-1}}, v_{m_t}) \tag{1}$$

where $v_i = t[i]$ for all $i \in [m_t]$, each $k_s \in \mathbb{N}_{\geq 0}$ denotes the number of station visits in segment $s \in S_t$, and $f_s^k \in F^*$ for all $s \in S_t$ and $k \in \{1, \ldots, k_s\}$. The path $\mathcal{P}_t$ contains exactly the ordered stops of t and additional stops at stations. Between two consecutive stops of t the path $\mathcal{P}_t$ must only visit stations $f \in F^*$ that are used to reset the remaining range to b_t (i.e., in the context of electromobility, we assume full charging). Since each $\mathcal{P}_t$ has to obey the range, the lengths of the paths between the first stop and the first station stop, any two consecutive station stops (there could be planned stops of t in between), the last station stop and the last stop, or, if $\mathcal{P}_t = \mathcal{S}_t$, between the first stop and the last stop, must be within b_t. In the following, we assume that G is a complete directed auxiliary graph on V, where each edge $(u, v) \in E$ represents a precomputed route from u to v in the underlying transportation network. Accordingly, $c^E(u, v)$ denotes the cost of that route, whereas $\ell(u, v)$ denotes its length with respect to range consumption. Moreover, we assume that c^E satisfies the triangle inequality, which ensures that it is not advantageous to visit a station only to reduce the path cost. The objective is to minimize the sum of the station costs and the path costs

$$\sum_{f \in F^*} c^F(f) + \sum_{t \in T} c^E(\mathcal{P}_t) \quad .$$

Additionally, we introduce useful definitions for the multi-stop station location problem. Often we need to refer to specific edges depending on the trip and its segments. Therefore, we distinguish different types of edges. We can enter a segment, i.e., we move from a stop of a trip (the start of the segment) to a station node. If we are in a segment, we can move to other stations or leave the segment again by moving to the next stop of the trip. Moreover, we can omit any additional station stops in a segment by moving directly from the start to the end of the segment. Thus, for a trip $t \in T$ and a segment $s \in S_t$, we define

- $\hat{\ell}_{t,s,f} := \ell(t[s], f)\ \forall f \in F$ (the length of the entering edge (v, f) connecting the start node of s and station f),
- $\bar{\ell}_{t,s} := \ell(t[s], t[s+1])$ (the length of the traversing edge (v_1, v_2) connecting the start node and the end node of s), and
- $\tilde{\ell}_{t,s,f} := \ell(f, t[s+1])\ \forall f \in F$ (the length of the leaving edge (f, v) connecting station f and the end node of s).

Analogously, concerning the edge costs, we define for a trip $t \in T$ and a segment $s \in S_t$

- $\hat{c}_{t,s,f} := c^E(t[s], f)\ \forall f \in F$,

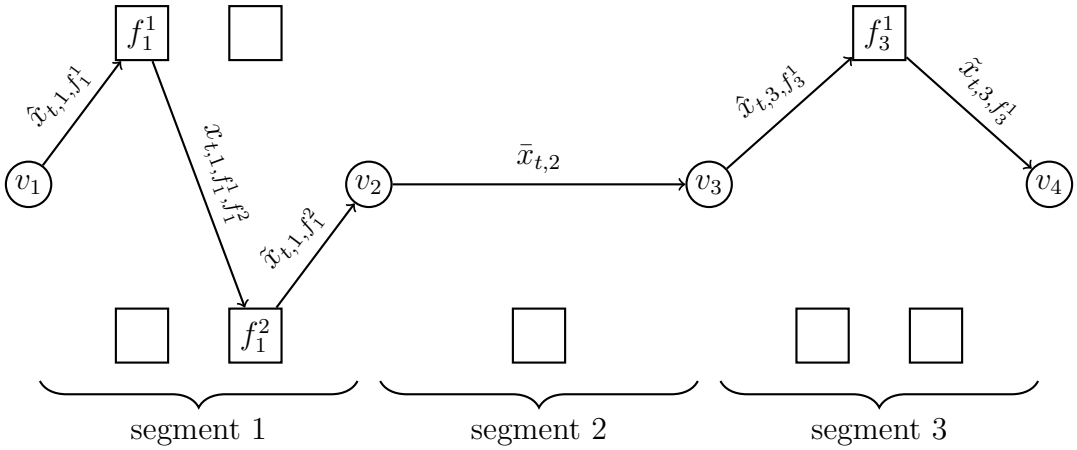


Figure 2: Nomenclature for variables and cost/length parameters on arcs: A path $\mathcal{P}_t$ for trip t with three segments, with different numbers of stations visited on each (possibly none). The caret accent signifies entering a segment (usually with less remaining range than b_t), the tilde accent is for leaving a segment, and the bar accent stands for just passing through ("shortcut") the segment without any visit to a station. All arc variables/parameters without an accent represent direct/shortest connections between stations, of which there can be many per segment.

- $\bar{c}_{t,s} := c^E(t[s], t[s+1])$, and
- $\tilde{c}_{t,s,f} := c^E(f, t[s+1])\ \forall f \in F$.

Accents on the variables in the following compact formulation (and later pricing problems) follow the same semantics, see also Figure 2.

2.1 Arc-based Formulation

The MSLP can be described mathematically by an arc-based compact formulation, which allows us to solve an MSLP instance as a mixed-integer program (MILP) using a standard MILP solver. We call this approach $\mathcal{A}_{\text{A}}$. Let $F_{t,v} := \{f \mid f \in F, v \neq f, \ell(v,f) \leq b_t\}$ be the set of all stations that are directly reachable from $v \in V$ with respect to the range b_t of trip $t \in T$. Thus, reachability in $F_{t,v}$ is defined with respect to the route lengths ℓ, not with respect to adjacency in the underlying transportation network. We use different types of x-variables to model the possible arcs of each trip, see Figure 2. The binary variable $x_{t,s,i,j}$ indicates whether station j is visited directly after station i within segment s of trip t. Similarly, the binary variables $\hat{x}_{t,s,i}$ and $\tilde{x}_{t,s,i}$ indicate whether a station i is visited as first or last station in a segment s of trip t, respectively. The binary ("shortcut") variable $\bar{x}_{t,s}$ represents that no station is visited at all (set to 1) on segment s of trip t. Additionally, the binary variable y_f indicates whether a station is installed and the continuous variable $r_{t,s}$ is used to keep track of the remaining range when leaving segment s of trip t. We model the directed MSLP as follows:

$$\min \quad \sum_{f \in F} c^F(f) y_f + \sum_{\substack{t \in T, \\ s \in S_t}} \left(\bar{c}_{t,s} \bar{x}_{t,s} + \sum_{i \in F} \left(\hat{c}_{t,s,i} \hat{x}_{t,s,i} + \tilde{c}_{t,s,i} \tilde{x}_{t,s,i} + \sum_{j \in F_{t,i}} c^E(i,j) x_{t,s,i,j} \right) \right) \tag{2}$$

$$\text{s.t.} \quad \hat{x}_{t,s,j} + \sum_{\substack{i \in F: \\ j \in F_{t,i}}} x_{t,s,i,j} = \tilde{x}_{t,s,j} + \sum_{i \in F_{t,j}} x_{t,s,j,i} \qquad \forall t \in T,\ \forall s \in S_t,\ \forall j \in F \tag{3}$$

$$\bar{x}_{t,s} + \sum_{f \in F} \hat{x}_{t,s,f} = 1 \qquad \forall t \in T,\ \forall s \in S_t \tag{4}$$

$$\hat{x}_{t,s,f} + \sum_{\substack{i \in F: \\ f \in F_{t,i}}} x_{t,s,i,f} \leq y_f \qquad \forall t \in T,\ \forall s \in S_t,\ \forall f \in F \tag{5}$$

$$\bar{\ell}_{t,s} \bar{x}_{t,s} + \sum_{f \in F} \hat{\ell}_{t,s,f} \hat{x}_{t,s,f} \leq \begin{cases} b_t, & \text{if } s = 1 \\ r_{t,s-1}, & \text{else} \end{cases} \qquad \forall t \in T,\ \forall s \in S_t \tag{6}$$

$$\bar{\ell}_{t,s} \bar{x}_{t,s} + \sum_{f \in F} \tilde{\ell}_{t,s,f} \tilde{x}_{t,s,f} \leq b_t - r_{t,s} \qquad \forall t \in T,\ \forall s \in S_t \tag{7}$$

$$r_{t,s} + (b_t + \bar{\ell}_{t,s}) \bar{x}_{t,s} \leq \begin{cases} 2b_t, & \text{if } s = 1 \\ b_t + r_{t,s-1}, & \text{else} \end{cases} \qquad \forall t \in T,\ \forall s \in S_t \tag{8}$$

$$x_{t,s,i,j} \in \{0,1\} \qquad \forall t \in T,\ \forall s \in S_t, i \in F, j \in F_{t,i} \tag{9}$$

$$\bar{x}_{t,s} \in \{0,1\} \qquad \forall t \in T,\ \forall s \in S_t \tag{10}$$

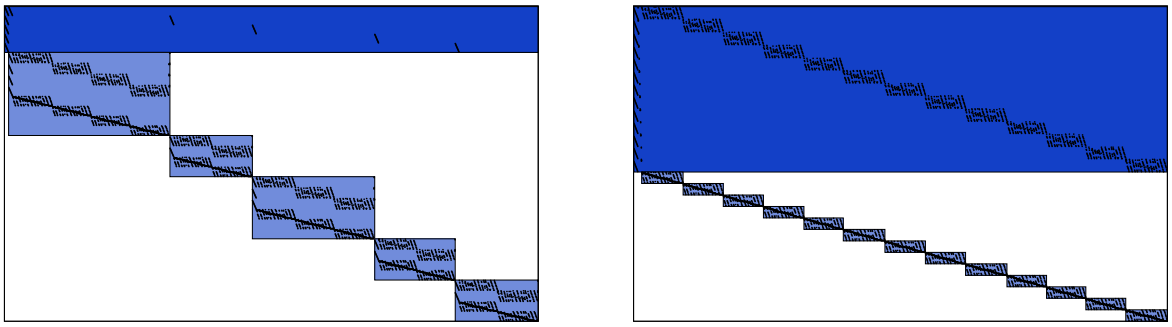
$$\hat{x}_{t,s,i} \in \{0,1\} \qquad \forall t \in T,\ \forall s \in S_t,\ \forall i \in F \tag{11}$$

$$\tilde{x}_{t,s,i} \in \{0,1\} \qquad \forall t \in T,\ \forall s \in S_t,\ \forall i \in F \tag{12}$$

$$y_f \in \{0,1\} \qquad \forall f \in F \tag{13}$$

$$r_{t,s} \geq 0 \qquad \forall t \in T,\ \forall s \in S_t \tag{14}$$

The objective function minimizes the cost of installing new stations and the path cost of all trips. For each segment of a trip, the flow-conservation constraints (3) ensure that a station node is entered and left if it is visited. The trip constraints (4) ensure that each trip is completely planned and that no segment is omitted. The station-linking constraints (5) ensure that a station must be selected if it is visited in at least one segment of a trip. In (5), the left-hand side collects all ways in which station f can be entered within segment s, either directly from the segment start via $\hat{x}_{t,s,f}$ or from another station via an arc $x_{t,s,i,f}$. The remaining constraints implement the range restrictions. In a segment we only have the option of either going directly from the start node of the segment to the end node of the segment, or visiting one or more stations from the start node before reaching the end node of the segment. The x-variables are defined only for the edges for which $\ell(i,j) \leq b_t$ where $i, j \in F$. This ensures that the range restrictions for edges connecting two stations are not violated. Constraints (6) and (7) limit the available range at the end of each segment of a trip. More specifically, (6) ensures that the remaining range after the previous segment is large enough to enter the next segment. Likewise, (7) ensures that the remaining range at the end of a segment takes into account the last step needed to reach the end of the segment. Moreover, (8) links the remaining-range variable of a segment to that of the previous segment if no station is visited.



(a) trip patterns (b) segment patterns

Figure 3: The two exploited structures of a MSLP instance are visualized by the rearranged coefficient matrices of the arc-based formulation. Black dots correspond to non-zero entries. The dark blue boxes represent the master problems and the light blue boxes represent the pricing problems.

2.2 Pattern-based Formulations

The arc-based formulation allows us to solve instances of the multi-stop station location problem using a MILP solver. However, for large instances, the size of the formulation grows rapidly because it uses $O(N^2M)$ variables and $O(NM)$ constraints with $N = |V|$ and $M = \sum_{t \in T} |S_t|$. Thus, we propose multiple decompositions that result in pattern-based formulations. These are then solved with branch-price-and-cut. In what follows, we assume the reader to be familiar with column generation and branch-price-and-cut (Desrosiers et al., 2026).

2.2.1 Trip Patterns

The first formulation ("trip-pattern-based," denoted by $\mathcal{A}_{\mathrm{T}}$) decomposes the problem by its trips, as visualized in Figure 3a. It can be viewed as a slightly modified Dantzig-Wolfe reformulation (Dantzig and Wolfe, 1960) of the arc-based formulation with one block per trip. The pricing problems handle the detailed path construction for an entire trip, whereas the master problem contains trip-covering and station-linking constraints. That is, the master problem only decides which trip patterns and which stations are selected, while the pricing problem ensures that the corresponding path visits the stops in the correct order and satisfies the range restrictions. For trip $t \in T$, denote by P_t the set of all valid paths of the form (1), i.e., all stops are visited in the correct order and the ranges are not violated (by visiting stations if necessary). We call $p \in P_t$ a pattern for trip t. The cost c_p of p is the sum of the costs of all edges used on the path. The master problem is based on binary variables that reflect whether a pattern is selected or not. Since there is a huge number of such variables, we use column generation to solve its linear relaxation. The restricted master problem with only a (small) subset of variables is alternately solved with several pricing problems that generate further pattern variables. While the master problem consists of only two different types of constraints, the pricing problems are much more involved.

Master Problem The master problem of the trip-pattern-based formulation has to select a pattern for each trip and compute the total costs. This is realized with pattern

variables $\lambda_p \in \{0,1\}$ with $p \in P_t$ and station variables $y_f \in \{0,1\}$ with $f \in F$, which we already know from the arc-based formulation. We denote the set of stations visited by a pattern p by F_p.

$$\min \quad \sum_{f \in F} c^F(f) y_f + \sum_{\substack{t \in T, \\ p \in P_t}} c_p \lambda_p \tag{15}$$

$$\text{s.t.} \quad \sum_{p \in P_t} \lambda_p \geq 1 \qquad \forall t \in T \qquad [\pi_t \geq 0] \tag{16}$$

$$\sum_{\substack{p \in P_t: \\ f \in F_p}} \lambda_p \leq y_f \qquad \forall t \in T,\ \forall f \in F \qquad [\pi_{t,f} \leq 0] \tag{17}$$

$$\lambda_p \in \{0,1\} \qquad \forall t \in T,\ \forall p \in P_t \tag{18}$$

$$y_f \in \{0,1\} \qquad \forall f \in F \tag{19}$$

$$\tag{20}$$

The objective function minimizes the costs of all selected patterns and the costs of the selected stations. The trip-covering constraints (16) ensure that each trip is covered by a selected pattern, and the station-linking constraints (17) push the station variables to 1 if station f is visited by a selected pattern of trip t. We write (16) in covering form. Since all pattern costs are non-negative, any optimal solution satisfies it at equality, and we exploit this fact instead of explicitly enforcing equality. For constraints containing pattern variables, the corresponding dual variables are given (in brackets) as we need them to formulate the pricing problems.

Pricing Problems The role of the pricing problem(s) is to identify pattern variables with negative reduced cost. As mentioned earlier, each pattern variable corresponds to a specific path for a trip. We use one pricing problem for each trip $t \in T$:

$$\min \quad -\pi_t - \sum_{f \in F} \pi_{t,f} y_f + \sum_{s \in S_t} \left(\bar{c}_{t,s} \bar{x}_{t,s} + \sum_{f \in F} \left(\hat{c}_{t,s,f} \hat{x}_{t,s,f} + \tilde{c}_{t,s,f} \tilde{x}_{t,s,f} + \sum_{f' \in F_{t,f}} c^E(f,f') x_{t,s,f,f'} \right) \right) \tag{21}$$

$$\text{s.t.} \quad \hat{x}_{t,s,f} + \sum_{\substack{f' \in F: \\ f \in F_{t,f'}}} x_{t,s,f',f} = \tilde{x}_{t,s,f} + \sum_{f' \in F_{t,f}} x_{t,s,f,f'} \qquad \forall s \in S_t,\ \forall f \in F \tag{22}$$

$$\bar{x}_{t,s} + \sum_{f \in F} \hat{x}_{t,s,f} = 1 \qquad \forall s \in S_t \tag{23}$$

$$\hat{x}_{t,s,f} + \sum_{\substack{f' \in F: \\ f \in F_{t,f'}}} x_{t,s,f',f} \leq y_f \qquad \forall s \in S_t,\ \forall f \in F \tag{24}$$

$$\bar{\ell}_{t,s} \bar{x}_{t,s} + \sum_{f \in F} \hat{\ell}_{t,s,f} \hat{x}_{t,s,f} \leq \begin{cases} b_t, & \text{if } s = 1 \\ r_{t,s-1}, & \text{else} \end{cases} \qquad \forall s \in S_t \tag{25}$$

$$\bar{\ell}_{t,s} \bar{x}_{t,s} + \sum_{f \in F} \tilde{\ell}_{t,s,f} \tilde{x}_{t,s,f} \leq b_t - r_{t,s} \qquad \forall s \in S_t \tag{26}$$

$$r_{t,s} + (b_t + \bar{\ell}_{t,s}) \bar{x}_{t,s} \leq \begin{cases} 2b_t, & \text{if } s = 1 \\ b_t + r_{t,s-1}, & \text{else} \end{cases} \qquad \forall s \in S_t \tag{27}$$

$$x_{t,s,i,j} \in \{0,1\} \qquad \forall i \in F, j \in F_{t,i} \tag{28}$$

$$\bar{x}_{t,s} \in \{0,1\} \qquad \forall s \in S_t \tag{29}$$

$$\hat{x}_{t,s,i} \in \{0,1\} \qquad \forall i \in F \tag{30}$$

$$\tilde{x}_{t,s,i} \in \{0,1\} \qquad \forall i \in F \tag{31}$$

$$y_f \in \{0,1\} \qquad \forall f \in F \tag{32}$$

$$r_{t,s} \geq 0 \qquad \forall s \in S_t \tag{33}$$

For a fixed single trip t, constraints (22)–(27) correspond to constraints (3)–(8) of the arc-based formulation. In particular, (24) uses the same incoming-arc convention as (5) of the compact formulation. A solution to the pricing problem in x-variables represents a valid path of the form (1), which allows us to compute its cost in the objective function and, therefore, the reduced cost of the corresponding pattern variable. Note that the pricing problem contains more information about the path than is needed in (17) of the master problem, namely the information about which stations are visited.

2.2.2 Segment Patterns

The second formulation ("segment-pattern-based," referred to as $\mathcal{A}_{\text{S}}$) decomposes the problem by its segments. It can be seen as a Dantzig-Wolfe reformulation of the arc-based formulation with one block per segment. The pricing problems handle the path construction within a segment, whereas the master problem retains the station-linking and range-coupling constraints and explicitly keeps the $\bar{x}$-variables (10). Figure 3b visualizes the exploited structure. This formulation uses binary pattern variables that represent paths through a specific segment using at least one station. As with the previous formulation, this would result in too many variables, and we use column generation. We experiment with this model because the resulting pricing problems can be solved very efficiently. The segment-pattern-based formulation moves the path construction for a single

segment into the pricing problem and keeps the coupling decisions in the master problem. In particular, the master problem contains the station variables, the shortcut variables, and the constraints that link the remaining range across consecutive segments of a trip. As a consequence, the pricing problems are much simpler than in the trip-pattern-based formulation and can be solved as shortest path problems.

Master Problem The master problem of the segment-pattern-based formulation is more involved, since it must ensure that the patterns selected for the segments of a trip are compatible with respect to the range. The main difference is that the segment-pattern-based formulation uses patterns $p \in P_{t,s}$ that describe a (valid) path through a segment $s \in S_t$ of a trip $t \in T$ with cost c_p. For a given pattern p, the new parameters $\hat{\ell}_p \in \mathbb{Q}_{\geq 0}$ and $\tilde{\ell}_p \in \mathbb{Q}_{\geq 0}$ specify the length of the edge from the start of the segment (a stop) to the first station and from the last station to the end of the segment (the next stop), respectively. That is, for a pattern p belonging to a segment s of trip t, we have $\hat{\ell}_p = \hat{\ell}_{t,s,i}$ and $\tilde{\ell}_p = \tilde{\ell}_{t,s,j}$ where $i \in F$ is the first station visited and $j \in F$ is the last station visited according to pattern p. Moreover, variables and constraints responsible for finding a path through a segment in the arc-based formulation are removed.

$$\min \quad \sum_{f \in F} c^F(f) y_f + \sum_{\substack{t \in T, \\ s \in S_t}} \left(\bar{c}_{t,s} \bar{x}_{t,s} + \sum_{p \in P_{t,s}} c_p \lambda_p \right) \tag{34}$$

$$\text{s.t.} \quad \bar{x}_{t,s} + \sum_{p \in P_{t,s}} \lambda_p \geq 1 \qquad \forall t \in T,\ \forall s \in S_t \qquad [\pi_{t,s} \geq 0] \tag{35}$$

$$\sum_{\substack{p \in P_{t,s}: \\ f \in F_p}} \lambda_p \leq y_f \qquad \forall t \in T,\ \forall s \in S_t,\ \forall f \in F \qquad [\pi_{t,s,f} \leq 0] \tag{36}$$

$$\bar{\ell}_{t,s} \bar{x}_{t,s} + \sum_{p \in P_{t,s}} \hat{\ell}_p \lambda_p \leq \begin{cases} b_t, & \text{if } s = 1 \\ r_{t,s-1}, & \text{else} \end{cases} \qquad \forall t \in T,\ \forall s \in S_t \qquad [\hat{\pi}_{t,s} \leq 0] \tag{37}$$

$$\bar{\ell}_{t,s} \bar{x}_{t,s} + \sum_{p \in P_{t,s}} \tilde{\ell}_p \lambda_p \leq b_t - r_{t,s} \qquad \forall t \in T,\ \forall s \in S_t \qquad [\tilde{\pi}_{t,s} \leq 0] \tag{38}$$

$$r_{t,s} + (b_t + \bar{\ell}_{t,s}) \bar{x}_{t,s} \leq \begin{cases} 2b_t, & \text{if } s = 1 \\ b_t + r_{t,s-1}, & \text{else} \end{cases} \qquad \forall t \in T,\ \forall s \in S_t \tag{39}$$

$$\lambda_p \in \{0,1\} \qquad \forall t \in T,\ \forall s \in S_t,\ \forall p \in P_{t,s} \tag{40}$$

$$\bar{x}_{t,s} \in \{0,1\} \qquad \forall t \in T,\ \forall s \in S_t \tag{41}$$

$$y_f \in \{0,1\} \qquad \forall f \in F \tag{42}$$

$$r_{t,s} \geq 0 \qquad \forall t \in T,\ \forall s \in S_t \tag{43}$$

Constraints (35)–(39) correspond, respectively, to constraints (4)–(8) of the arc-based formulation. As in the trip-pattern-based formulation, we write (35) in covering form. Since all corresponding pattern and shortcut costs are non-negative, any optimal solution satisfies it at equality. For constraints containing pattern variables the corresponding dual variables are given in brackets.

Pricing Problems Each pattern variable corresponds to a particular path through a segment of a trip that visits at least one station. We use a pricing problem for each segment $s \in S_t$ and each trip $t \in T$:

$$\min \quad -\pi_{t,s} + \sum_{i \in F} \Big((\hat{c}_{t,s,i} - \pi_{t,s,i} - \hat{\pi}_{t,s}\hat{\ell}_{t,s,i})\hat{x}_{t,s,i} + (\tilde{c}_{t,s,i} - \tilde{\pi}_{t,s}\tilde{\ell}_{t,s,i})\tilde{x}_{t,s,i} \Big) \tag{44}$$
$$+ \sum_{i \in F} \sum_{j \in F_{t,i}} (c^E(i,j) - \pi_{t,s,j}) x_{t,s,i,j}$$

$$\text{s.t.} \quad \hat{x}_{t,s,j} + \sum_{\substack{i \in F: \\ j \in F_{t,i}}} x_{t,s,i,j} = \tilde{x}_{t,s,j} + \sum_{i \in F_{t,j}} x_{t,s,j,i} \qquad \forall j \in F \tag{45}$$

$$\sum_{f \in F} \hat{x}_{t,s,f} = 1 \tag{46}$$

$$x_{t,s,i,j} \in \{0,1\} \qquad \forall i \in F, j \in F_{t,i} \tag{47}$$

$$\hat{x}_{t,s,i} \in \{0,1\} \qquad \forall i \in F \tag{48}$$

$$\tilde{x}_{t,s,i} \in \{0,1\} \qquad \forall i \in F \tag{49}$$

As before, the x-variables represent a valid segment pattern, and the objective function (44) minimizes the reduced cost of the corresponding pattern variable. Here, (45) corresponds to (3) of the arc-based formulation. In addition, (46), corresponding to (4), ensures that the computed path leaves the first node of the segment. Note that every pricing problem is a shortest path problem and can be solved efficiently. Therefore, we create a graph $G_s(V_s, E_s)$ to solve the pricing problem of segment s. The node set V_s contains all possible stations F, an artificial source node, and an artificial sink node. The source node corresponds to the start node of the segment (a stop) and the sink node corresponds to the end node of the segment (the next stop). The source has only outgoing edges to station nodes and the sink has only incoming edges from station nodes. Furthermore, $E_s \subseteq E$ contains all edges between possible stations that do not violate the range. As a result, all paths between two stations in G_s are valid with respect to the range. A feasible solution to a pricing problem is a path through the modified graph starting at the source and ending at the sink. We define the edge lengths such that the length of a path from the source to the sink is equal to the non-constant part of the objective function. By the sign restrictions of the dual variables given above, all resulting edge lengths are non-negative. Hence, the pricing problem can be solved as a shortest path problem and, in particular, negative cycles do not occur. We obtain the reduced cost of the corresponding pattern by subtracting $\pi_{t,s}$ from the length of the path. Figure 4 depicts a small example.

2.2.3 Nested Decomposition Approach

As mentioned in Section 2.2.1, the trip-pattern-based formulation $\mathcal{A}_{\mathrm{T}}$ uses pricing problems that solve the MSLP with respect to a trip and a modified objective function. Hence, we can again exploit the segment structure within such a pricing problem by decomposing it into segments, referred to as $\mathcal{A}_{\mathrm{N}}$. Figure 5 visualizes this nested decomposition. More precisely, $\mathcal{A}_{\mathrm{N}}$ applies the segment-pattern-based formulation $\mathcal{A}_{\mathrm{S}}$ to the single-trip pricing problem of $\mathcal{A}_{\mathrm{T}}$. In the resulting inner problem, the cost of a charging station $f \in F$ is replaced by the negative value of the dual variable $\pi_{t,f}$ of (17), while the reduced cost of the resulting trip pattern is obtained by subtracting π_t, the dual variable of (16), from the inner objective value.

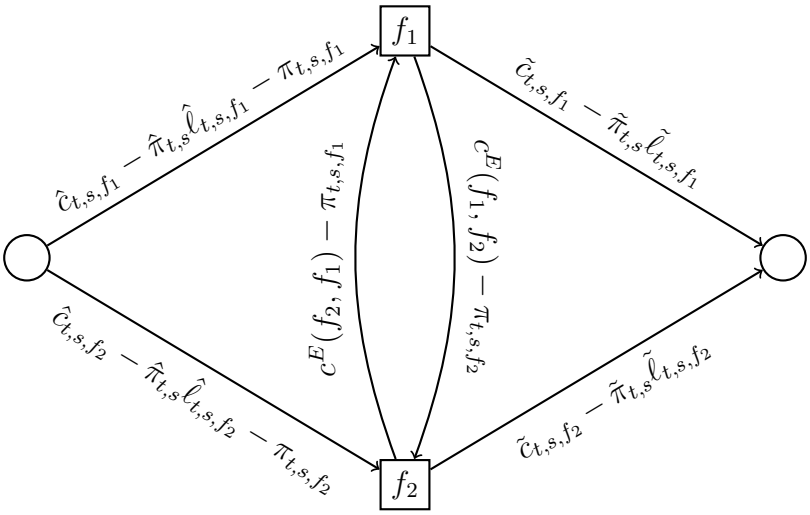


Figure 4: A pricing problem modeled as a shortest path problem. The left circle represents the start node and the right circle represents the end node of the considered paths.

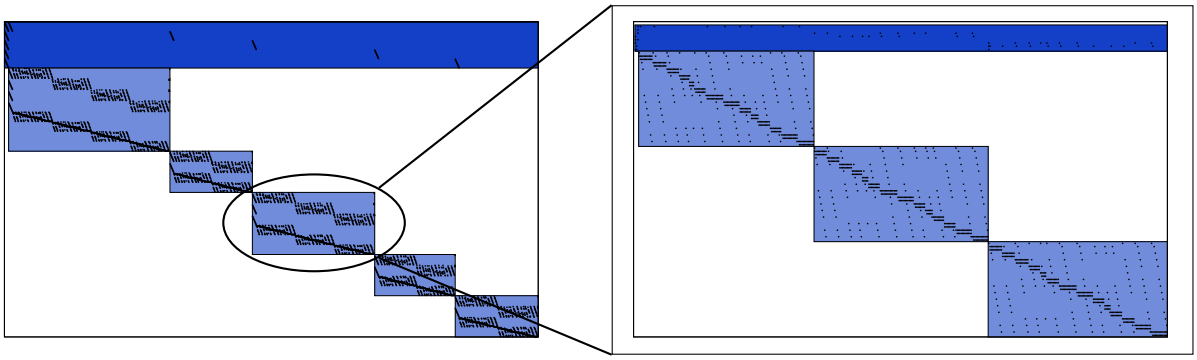


Figure 5: The nested structure of a MSLP instance is visualized using re-arranged coefficient matrices. Black dots correspond to non-zero entries, the dark blue boxes represent the master problems, and the light blue boxes represent the pricing problems. On the left is the structure exploited by the outer trip-pattern-based formulation, and on the right is the structure of a single pricing problem exploited by the inner segment-pattern-based formulation.

3 Algorithmic Components

While our arc-based formulation (Section 2.1) can be solved with a standard MILP solver, the pattern-based formulations presented in Sections 2.2.1–2.2.3 require branch-and-price algorithms.

3.1 Primal Heuristic

We use a problem specific primal heuristic that works on the original problem. The main idea is to transform the problem into a shortest path problem for each trip by constructing a new graph using a given set of allowed stations. If the constructed graphs are connected, a shortest feasible path is computed for each trip. We have adapted the construction used by the approximation algorithm presented by Willamowski, Ganz, and Mühmer (2020). Instead of using the actual station costs, we set all station costs to 0. To prevent the heuristic from repeatedly computing the same solution, we restrict the set of stations

based on the current fractional solution to the restricted master problem. Note that for a given fixed set of stations (i.e., F^* is known in advance), the primal heuristic computes an optimal solution if a feasible solution exists: the given stations can be used at no additional cost, and by construction a shortest path is computed for each trip such that the range is not violated. Thus, the path costs of the trips are minimum with respect to the provided stations. From the heuristic's solution, a solution to the master problem is created. Since the heuristic works on the original problem, it may find solutions that cannot be represented by the current variables of the restricted master problem. In this case, we generate them and add them to the restricted master problem.

3.2 Pricing

Each of our pattern-based formulations comes with their own (restricted) master and pricing problems. The restricted master problem in the current node of the branch-and-price tree can be infeasible, e.g., at the root node (if there is no trivial solution) or after branching. In this case, Farkas pricing (Achterberg, 2007) is applied, and standard reduced cost pricing otherwise.

3.2.1 Farkas Pricing

Exploiting the structure of our problem, we use a tailored Farkas pricing. We apply the primal heuristic described in Section 3.1 since it produces a feasible solution if and only if the problem instance is feasible. We distinguish two cases. At start, if the restricted master problem does not contain any priced variables yet, we run the algorithm with multiple configurations to generate a set of initial patterns. We use static station costs (set to 0 and to the actual costs) as well as dynamic station costs, which are iteratively set to 0, see Willamowski, Ganz, and Mühmer, 2020. For each subsequent call to the Farkas pricing procedure, the primal heuristic is run with all stations that are not disabled (e.g., by branching) and costs set to 0. After that, we know that either the current node is infeasible, or which patterns we can add to make the restricted master problem feasible.

3.2.2 Reduced Cost Pricing

Depending on the used pattern-based formulation, the reduced cost pricing procedure differs in whether the primal heuristic (see Section 3.1) is called. For $\mathcal{A}_{\mathrm{T}}$ and for the outer master problem of $\mathcal{A}_{\mathrm{N}}$, we call the heuristic at each pricing iteration at the root node and once when a new branching node is processed. For $\mathcal{A}_{\mathrm{S}}$ and for single-trip segment-pricing instances, including the inner solver of $\mathcal{A}_{\mathrm{N}}$, we invoke it at most once per node after the initialization phase. We distinguish these cases because preliminary experiments showed that the segment-pricing setting typically results in larger trees and benefits from heuristic solutions found during branching. In contrast, for $\mathcal{A}_{\mathrm{T}}$ and for the outer master problem of $\mathcal{A}_{\mathrm{N}}$, more frequent heuristic calls at child nodes did not justify their additional cost. For all formulations, we select the allowed stations provided to the heuristic by evaluating the current solution to the restricted master LP. All stations whose variables have a value of at least $\tau \in \mathbb{Q}_{\geq 0}$ in the current LP solution can be selected by the heuristic. Preliminary tests have shown that invoking the heuristic with the threshold $\tau = 0.5$ yields comparable results to other thresholds or even multiple invocations with different thresholds. If the heuristic is successful (i.e., the heuristic found a valid solution) and we need new pattern variables, they are added to the restricted master problem and

the primal bound is updated if necessary. Moreover, if all station variables are fixed, we do not need to price new variables since the heuristic computes an optimal solution for the specific set of stations. Otherwise, we price new variables by solving the pricing problems using the current dual information. Then, the priced variables are added to the restricted master problem and we update the current lower bound by computing the Lagrangian lower bound using the results of the pricing problems and the current objective value of the LP relaxation.

Outline of the main pricing procedure:

1. Run the heuristic with the allowed edges and all stations that currently have a solution value of at least τ. For $\mathcal{A}_{\mathrm{T}}$ and for the outer master problem of $\mathcal{A}_{\mathrm{N}}$, do this at each pricing iteration at the root node and once when a new branching node is processed. For $\mathcal{A}_{\mathrm{S}}$, including the inner solver of $\mathcal{A}_{\mathrm{N}}$, do this at most once per node after the initialization phase.

2. Check whether all stations are fixed (e.g., due to branching).

   - If so, no pricing is required. The result of the heuristic is processed and the node is marked as solved.

3. Solve the pricing problems parameterized with the current dual information.

4. Update the current lower bound based on this iteration.

3.3 Cutting Planes

We use cutting planes in the master problem of $\mathcal{A}_{\mathrm{S}}$ that are similar to knapsack cover cuts (Achterberg, 2007). The cuts exploit (fractional) paths that would violate the range limitation in the integer case. Figure 6 shows an example of such a situation. We assume that (for a fixed trip t) we have found a fractional path starting at segment s_1 using variable λ_{p_1} and ending at segment s_2 using variable λ_{p_2} with

$$\bar{\ell}_{t,s_1,s_2} = \sum_{\substack{s' \in S_t: \\ s_1 < s' < s_2}} \bar{\ell}_{t,s'} \ .$$

If

$$\tilde{\ell}_{p_1} + \hat{\ell}_{p_2} + \bar{\ell}_{t,s_1,s_2} > b_t$$

and

$$\lambda_{p_1} + \lambda_{p_2} + \sum_{\substack{s' \in S_t: \\ s_1 < s' < s_2}} \bar{x}_{t,s'} > |\{s' \in S_t \mid s_1 \leq s' \leq s_2\}| - 1 \ ,$$

then (50) is a violated valid inequality:

$$\lambda_{p_1} + \lambda_{p_2} + \sum_{\substack{s' \in S_t: \\ s_1 < s' < s_2}} \bar{x}_{t,s'} \leq |\{s' \in S_t \mid s_1 \leq s' \leq s_2\}| - 1 \ . \tag{50}$$

In addition to λ_{p_1} and λ_{p_2}, we can include other pattern variables belonging to s_1 and s_2 in the inequality if the corresponding patterns would also violate the range. Hence, we can lift (50) to (51) by choosing the bounds b_1 and b_2 such that $b_1 + b_2 + \bar{\ell}_{t,s_1,s_2} \geq b_t$:

$$\sum_{\substack{p \in P_{t,s_1}: \\ \tilde{\ell}_p > b_1}} \lambda_p + \sum_{\substack{p \in P_{t,s_2}: \\ \hat{\ell}_p > b_2}} \lambda_p + \sum_{\substack{s' \in S_t: \\ s_1 < s' < s_2}} \bar{x}_{t,s'} \leq |\{s' \in S_t \mid s_1 \leq s' \leq s_2\}| - 1 \ . \tag{51}$$

Finally, a path can start or end with a shortcut instead of a pattern. Let

$$X^{b_1,b_2}_{t,s_1,s_2} = \{\bar{x}_{t,s'} \mid s' \in S_t,\ s_1 < s' < s_2\} \cup \{\bar{x}_{t,s_1} \mid \bar{\ell}_{t,s_1} > b_1\} \cup \{\bar{x}_{t,s_2} \mid \bar{\ell}_{t,s_2} > b_2\}$$

denote the set of all shortcut variables relevant for this extension. We can then again lift (51) resulting in (52)

$$\sum_{\substack{p \in P_{t,s_1}: \\ \tilde{\ell}_p > b_1}} \lambda_p + \sum_{\substack{p \in P_{t,s_2}: \\ \hat{\ell}_p > b_2}} \lambda_p + \sum_{\bar{x} \in X^{b_1,b_2}_{t,s_1,s_2}} \bar{x} \leq |\{s' \in S_t \mid s_1 \leq s' \leq s_2\}| - 1 \ . \tag{52}$$

These inequalities are valid for the MSLP and do not cut any feasible integer solution because for every feasible integer solution there exists a segment $s \in S_t$ with $s_1 \leq s \leq s_2$ for which all related variables are zero since otherwise the range bound would be violated. Thus, the sum of the variable values of any presented inequality is at most $|\{s' \in S_t \mid s_1 \leq s' \leq s_2\}| - 1$. Since these cuts are added to the master problem, the pricing problems must handle the new dual values. Let π_r denote the associated dual value of cut r, which is added as a new row to the master problem. We need to add

$$-\sum_{\substack{i \in F: \\ \tilde{\ell}_{t,s_1,i} > b_1}} \pi_r \tilde{x}_{t,s_1,i}$$

to the objective function of the pricing problem of s_1 and

$$-\sum_{\substack{i \in F: \\ \hat{\ell}_{t,s_2,i} > b_2}} \pi_r \hat{x}_{t,s_2,i}$$

to the objective function of the pricing problem of s_2 (see Section 2.2.2). Therefore, in the graph used to solve a pricing problem, we must adapt the lengths of the edges entering the sink or leaving the source if they violate the maximum range b_1 or b_2, respectively. This modification cannot lead to negative distances in the graph because $\pi_r \leq 0$. Thus, we do not need to worry about negative lengths when solving a pricing problem even in the presence of these cuts.

Note that these valid inequalities can also be obtained by using an idea described by Wolsey (1990). This is achieved by aggregating the range-coupling constraints (37), (38), and (39). For a trip $t \in T$ and two segments $s_1,\ s_2 \in S_t$ with $s_1 \leq s_2$, we obtain a knapsack constraint by combining the inequality of (38) corresponding to segment s_1, the inequalities of (39) corresponding to all segments $s' \in S_t$ with $s_1 < s' < s_2$, and the inequality of (37) corresponding to segment s_2. Because of (35), any sum of pattern and shortcut variables of a segment is bounded from above by 1. Thus, we can apply the idea described by Wolsey (1990) to the obtained knapsack constraint with respect to the upper bounds implied by (35).

Furthermore, the inequalities can also be translated to the arc-based formulation. By replacing the pattern variables in (52), we get (53).

$$\sum_{\substack{f \in F: \\ \tilde{\ell}_{t,s_1,f} > b_1}} \tilde{x}_{t,s_1,f} + \sum_{\substack{f \in F: \\ \hat{\ell}_{t,s_2,f} > b_2}} \hat{x}_{t,s_2,f} + \sum_{\bar{x} \in X^{b_1,b_2}_{t,s_1,s_2}} \bar{x} \leq |\{s' \in S_t \mid s_1 \leq s' \leq s_2\}| - 1 \tag{53}$$

Thus, we can separate the cuts while solving the arc-based formulation as well as while solving the pricing problems of $\mathcal{A}_{\mathrm{T}}$.

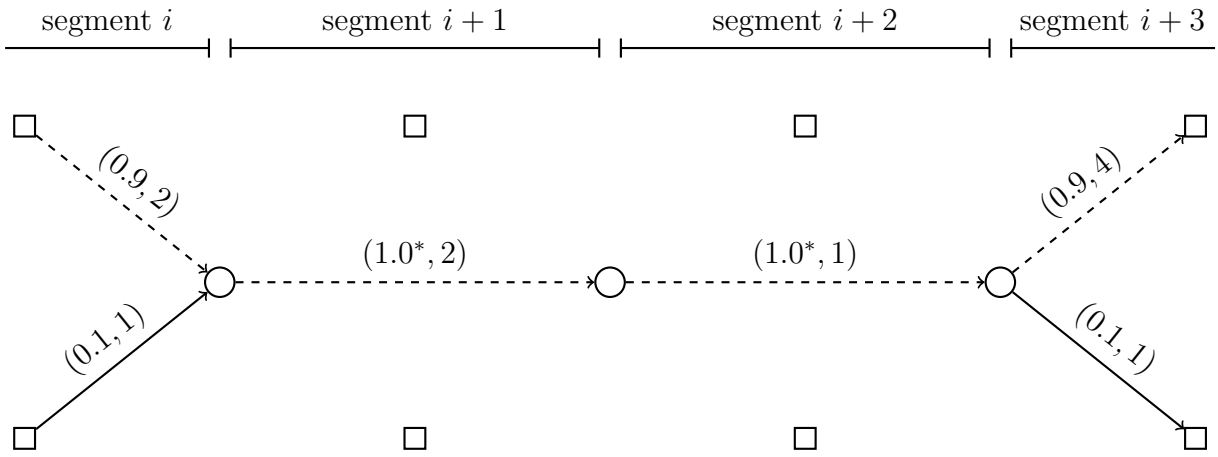


Figure 6: A part of a fractional solution is visualized. The graph shown is not the underlying instance graph. The circles represent stops of a trip (two consecutive stops belong to a segment as start and end). The rectangles are copies of station nodes. For each segment all possible stations are available (here we have 2 stations). In this example, b_t is equal to 8. The edges show the (fractional) path used by the current solution. A tuple (a, b) corresponds to the (fractional) value a of the solution variable and to the length b of the edge. The dashed edges belong to a path that would be invalid in the integer case.

We separate cuts for each trip at each tree node by a single-pass greedy scan over consecutive segments. The procedure maintains one current fractional path. For each segment, it inspects shortcut variables and positively valued pattern variables of the current LP solution. If the current path can be extended by a shortcut, the shortcut is used. Otherwise, the current segment closes the current candidate path and starts a new one. Whenever the accumulated length exceeds the trip range, we derive a candidate cut and check explicitly whether it is violated. If the current segment is represented by a shortcut, we discard the first segment of the current fractional path and continue with the remaining path, so that the previous second segment becomes the new first segment of the updated path. This is a heuristic separation procedure and may miss violated cuts.

3.4 Branching

In order to obtain integer solutions, we have implemented multiple branching strategies. At every tree node we decide which one should be used. This is done by static rules and a pseudo-cost branching scheme.

3.4.1 Branching on Station Variables

The first type of branching candidates is used by $\mathcal{A}_{\mathrm{T}}$, $\mathcal{A}_{\mathrm{S}}$, and $\mathcal{A}_{\mathrm{N}}$. All fractional station variables y_f with $f \in F$ are branching candidates. When such a candidate is selected, we branch directly on the corresponding variable. Since these are binary variables, the branching decisions correspond to selecting or forbidding the station f, i.e., the variable is fixed to 1 or 0 in the master problem. In addition, the pricing problems can no longer use this station if it is forbidden. This can be achieved by fixing the corresponding variables (of the pricing problems) to 0 or by removing the corresponding node from the graph so that it cannot be selected by the shortest path algorithm.

3.4.2 Branching on Shortcut Variables

In addition to fractional station variables, $\mathcal{A}_{\mathrm{S}}$ may encounter fractional shortcut variables $\bar{x}_{t,s}$ with $t \in T$, $s \in S_t$. Therefore, we also use these variables as branching candidates. When such a candidate is selected, we branch directly on the corresponding variable. Since these are binary variables, the branching decisions correspond to forbidding any station or enforcing at least one station in the segment s, i.e., the variable is fixed to 1 or 0 in the master problem. The pricing problem of the corresponding segment does not need to be called anymore if the shortcut variable is fixed to 1 (and all pattern variables of this segment can be fixed to 0).

3.4.3 Branching on Distances

The last branching candidates, used exclusively by $\mathcal{A}_{\mathrm{S}}$, refer to fractional sums of pattern variables λ_p with $p \in P_{t,s}$, $t \in T$, and $s \in S_t$. They are closely related to the cuts described in Section 3.3. We look for a fractional path (for a fixed trip t) that would violate the range in the integer case. Such a path starts with leaving a segment $s_1 \in S_t$ and ends with entering a segment $s_2 \in S_t$ with $s_1 < s_2$. All segments between s_1 and s_2 must not visit any station, i.e., for all segments $s' \in S_t$ with $s_1 < s' < s_2$ the value of $\bar{x}_{t,s}$ must be 1. We assume that we have found such a path starting with pattern $p_1 \in P_{t,s_1}$ and ending with pattern $p_2 \in P_{t,s_2}$. Let

$$\bar{\ell}_{t,s_1,s_2} = \sum_{\substack{s' \in S_t: \\ s_1 < s' < s_2}} \bar{\ell}_{t,s'}$$

denote the length of the segments between s_1 and s_2. The branching is realized by creating three child nodes, each covering another restriction that would invalidate the path. We calculate two length bounds (54) and (55) to define the two sets $B_1 = \{\lambda_p \mid p \in P_{t,s_1} \wedge \tilde{\ell}_p > b_1\}$ and $B_2 = \{\lambda_p \mid p \in P_{t,s_2} \wedge \hat{\ell}_p > b_2\}$ such that $\lambda_{p_1} \in B_1$, $\lambda_{p_2} \in B_2$, and $B_1 \cap B_2 = \emptyset$,

$$b_1 = \tilde{\ell}_{p_1} - \frac{\tilde{\ell}_{p_1}}{\tilde{\ell}_{p_1} + \hat{\ell}_{p_2}} \left(\tilde{\ell}_{p_1} + \hat{\ell}_{p_2} + \bar{\ell}_{t,s_1,s_2} - b_t \right) \ , \tag{54}$$

$$b_2 = b_t - b_1 - \bar{\ell}_{t,s_1,s_2} \ . \tag{55}$$

In addition, we define the set $B_3 = \{\bar{x}_{t,s'} \mid s_1 < s' < s_2\}$, which contains all shortcut variables between the first and second segments (whose values are equal to 1). The first branching node fixes the variables of B_1 to zero, the second of B_2, and the third enforces that not all shortcut variables of B_3 are equal to 1. In case that all shortcut variables of B_3 are fixed to 1, the third node can be omitted as it would be infeasible and can be pruned. Figure 7 depicts the branching decision (in the master problem). No feasible solution is excluded because we have $b_t = b_1 + b_2 + \bar{\ell}_{t,s_1,s_2}$ and, thus, solutions that do not belong to any branch would violate the range. In the pricing problem, we need to ensure that we do not generate patterns that violate these bounds. Therefore, we remove all edges from the graph that do not satisfy the branching decision, i.e., all starting or ending edges that are too long. As a result, we can still use our shortest path approach to generate new patterns. Since the violated cutting planes are not separated exactly, it can happen that we find a violated cutting plane while examining the distance branching candidates. In such a case, we add this cutting plane and perform no branching.

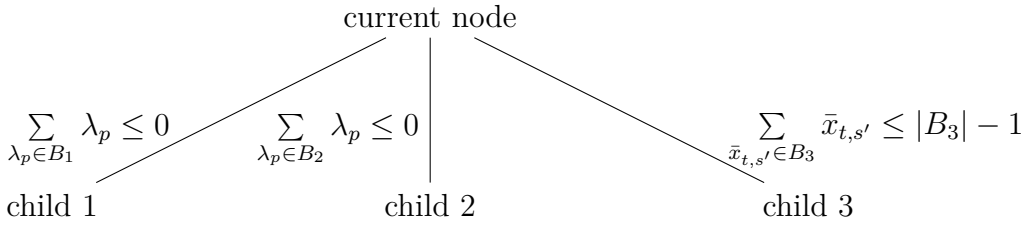


Figure 7: The nodes created with respect to a distance branching candidate are shown. The constraints of the first and second branch can be enforced by fixing the affected pattern variables to 0. The third branch is realized by a linear constraint that is added to the master problem.

3.4.4 Completeness of the Branching Scheme

For $\mathcal{A}_{\mathrm{T}}$ and for the outer master problem of $\mathcal{A}_{\mathrm{N}}$, it is sufficient to branch on station variables. Assume that we have an LP-feasible solution in which all station variables are integral. Then, for each trip $t \in T$, every positively valued pattern variable uses only stations whose station variable is equal to 1. If, for some trip t, more than one pattern variable is fractional, all positively valued pattern variables for that trip must have the same objective coefficient. Otherwise, we could improve the LP solution by shifting weight from a more expensive feasible tour to a cheaper one. Hence, for each such trip, we can set one positively valued pattern variable to 1 and all other positively valued pattern variables of that trip to 0 without destroying feasibility or changing the objective value. Repeating this argument trip by trip yields an integral solution with the same objective value.

For $\mathcal{A}_{\mathrm{S}}$ and for the inner solver of $\mathcal{A}_{\mathrm{N}}$, station branching alone is not sufficient because fractional pattern and shortcut variables can induce a fractional path that would violate the range in the integer case, as illustrated in Figure 6. Whenever such a path exists, we either separate a violated cut, apply distance branching, or branch on a shortcut variable. Consequently, if no branching candidate is available, all station and shortcut variables are integral and no fractional path violating the range remains. Hence, every remaining fractional path is feasible in the integer case. By LP optimality, all positively valued feasible paths of a trip must then have the same cost. Otherwise, we could improve the LP solution by shifting weight to a cheaper path. Therefore, we may choose one such path for each trip without worsening the objective value.

3.4.5 Selection of a Branching Candidate

At each node that requires branching, we must decide which branching candidate to use. All approaches rely on a branching score calculated using strong branching (Achterberg, 2007) and pseudo-costs (Achterberg, 2007).

For $\mathcal{A}_{\mathrm{T}}$ and for the master problem of $\mathcal{A}_{\mathrm{N}}$, only fractional station variables are considered as branching candidates.

For $\mathcal{A}_{\mathrm{S}}$, we use a hierarchical selection process if the current instance consists of multiple trips. We first consider station variable branching candidates. If no fractional station variable is available, we consider distance branching candidates. If neither a station nor a distance branching candidate is available, we consider shortcut branching candidates.

For single-trip instances, $\mathcal{A}_{\mathrm{S}}$ (including the inner solver of $\mathcal{A}_{\mathrm{N}}$) uses a modified selection

process of branching candidates. We collapse the hierarchy and evaluate all branching candidates at once. The final choice among the evaluated candidates is based on the same branching score. Preliminary experiments indicated that this variant yields better performance for such single-trip instances.

We prefer station branching in the multi-trip setting because we observed that fixing station variables helps the primal heuristic (see Section 3.1) and may affect multiple trips. As a result, the overall solving process can benefit from a strong primal bound during branching.

To select a branching candidate, we rely on pseudo-cost values (Achterberg, 2007). For all branching candidates, pseudo-cost values are maintained that are used to predict the objective gains if we branch on a candidate. Let $j \in \{1, 2\}$ be a branch's direction (first or second branch). Then, the pseudo-cost value $\psi_{h,j}$ is the normalized objective gain we expect when we branch on candidate h. For a branch, we know the fractional part $\delta_{h,j}$ of the solution values restricted by that branch. For example, if we branch on a station variable with value 0.4, the fractional parts are 0.4 and 0.6 (first branch and second branch). Using these values, we can calculate the predicted objective gain of a branch

$$g_{h,j} = \psi_{h,j} \cdot \delta_{h,j} \quad . \tag{56}$$

Whenever we have to decide between branching candidates, we calculate the branching score

$$\sigma_h = g_{h,1} \cdot g_{h,2} \tag{57}$$

for each branching candidate h. We choose a candidate with the highest branching score. After branching we need to update the pseudo-cost values. The experienced objective value gain is used to update the pseudo-cost values by calculating

$$\psi_{h,j}^{\text{new}} = \psi_{h,j} + \frac{1}{k}\left(\frac{z_{\text{lp}}^{\text{child}} - z_{\text{lp}}^{\text{parent}}}{\delta_{h,j}} - \psi_{h,j}\right), \tag{58}$$

where k is the number of branchings performed on branching candidate h and $z_{\text{lp}}^{\text{child}}$ and $z_{\text{lp}}^{\text{parent}}$ are the objective values of the master problem's relaxation of the child and parent nodes, respectively. If we have branched only a few times (or not at all) on a branching candidate, we have no reliable pseudo-cost values available (all are initially zero). Then, strong branching is used to compute the objective gains. If we have branched at least five times on that candidate, we use the pseudo-cost values to predict the objective gains.

3.5 Preprocessing

In a preprocessing step we reduce the size of the original problem instance by removing inactive members of the station-linking constraints (5), (17), and (36). In other words, if a station assignment cannot be part of an optimal solution, we can forbid it. This preprocessing exploits the triangle inequality with respect to c^E. We identify such assignments by using a solution provided by the heuristic, which is run at the beginning of the solving process. The preprocessing considers each trip separately. The idea is that assigning a station to a distant segment can cause such high detour costs that the path of a known solution for that trip dominates all possible paths that use this assignment, even if no other trip uses any of the selected stations. I.e., if the minimum cost of a path visiting a station in a particular segment is larger than the sum of the installation costs of the used stations and the cost of a path used in another valid solution with respect to

a single trip, we can forbid that station for this segment. In the trip-based formulation, if this is true for all segments of a trip, we can forbid the station for the entire trip. In the segment-based formulation, we forbid the corresponding segment-station assignments. We use this preprocessing throughout the computational study.

We exclude all variables that correspond to edges that violate the range of the trip. Furthermore, we do not have to include the corresponding inequalities of (6), (8), (37), and (39) for the first segments.

3.6 Implementation

We solve the arc-based formulation (2)–(14) using the C++ interface of Gurobi 9.5.2 (Gurobi Optimization, LLC., 2023). Our branch-price-and-cut implementation is based on SCIP 8.0.2 (Bestuzheva et al., 2021), which provides a C/C++ API. The solver was built on Debian 11 using g++ 10.2.1.

In addition to the model reductions from Section 3.5, we exploit presolving routines provided by Gurobi and SCIP to strengthen the formulations before the main solution process starts. For example, redundant variables, such as the range variables assigned to the last segment of a trip, are removed.

For the arc-based approach, we register the primal heuristic and the problem-specific cuts through Gurobi callbacks. The heuristic is executed once at every tree node, and cut separation is performed whenever Gurobi has solved the LP relaxation to optimality.

For the branch-and-price approaches, we use SCIP because it provides a state-of-the-art branch-price-and-cut framework that allows us to focus on the problem-specific algorithmic components. In particular, we implement custom plugins for pricing, branching, and cut separation. Moreover, we use the Boost Graph Library (Siek, Lee, and Lumsdaine, 2001) (version 1.75) for the shortest-path computations arising in the pricing problems and in the heuristic. Parts of the implementation (mentioned below) are parallelized using OpenMP (OpenMP Architecture Review Board, 2018).

The SCIP-based implementation is centered around a SCIP object that manages the restricted master problem, that is, we do not maintain separate SCIP structures for the original problem. We include the default plugins but disable SCIP's default separation routines. The main component is the pricer, which performs Farkas pricing and reduced cost pricing and generates new variables by solving the corresponding pricing problems through designated SCIP callbacks.

For branching, we register a branching rule that is executed before SCIP's default branching rules. We use SCIP's strong branching and pseudo-cost machinery to evaluate branching candidates by means of SCIP's branching score. Once a candidate has been selected, the branching rule creates the corresponding child nodes and attaches branching constraints to them. These constraints are enforced by a dedicated constraint handler, which updates variable fixings and the pricing problems whenever branching constraints become active or inactive.

We also implement a custom separator and disable SCIP's default separators. The separator searches for violated problem-specific cuts, works on multiple trips in parallel, and is called at every node of the search tree.

The implementation of $\mathcal{A}_{\mathrm{S}}$ maintains explicit data structures for all pricing problems. Each pricing problem is represented by a graph, and a shortest-path algorithm is used to solve it. By default, these graphs are cached and modified only as needed during pricing. For large instances, caching can be disabled to reduce memory consumption. Since each

shortest-path computation returns only one path, each pricing problem generates at most one new variable per call. Because we solve pricing problems in parallel, we add new variables to the master problem in a deterministic order so that runtime variations do not affect the overall solution process. For $\mathcal{A}_{\mathrm{T}}$, the pricing problems are solved by Gurobi using the implementation of $\mathcal{A}_{\mathrm{A}}$, that is, the arc-based formulation together with the primal heuristic and the cuts. Likewise, $\mathcal{A}_{\mathrm{N}}$ uses $\mathcal{A}_{\mathrm{S}}$ as the solver for its pricing problems. For $\mathcal{A}_{\mathrm{T}}$ and $\mathcal{A}_{\mathrm{N}}$, we rebuild the pricing models in each pricing iteration instead of reusing them across iterations. This keeps preprocessing and the removal of unnecessary variables aligned with the current cost vector. Within $\mathcal{A}_{\mathrm{N}}$, each inner segment-pricing solve still uses the cached graph representation.

The actual heuristic computations are triggered from the pricing routines because the heuristic may produce solutions that are not representable by the current restricted master problem. In such cases, the missing variables are generated and added before the solution is processed by the master problem. The heuristic is also registered as a SCIP plugin, but its callback is used only to submit previously computed solutions to SCIP. If static station costs are used, the heuristic can process multiple trips in parallel.

4 Experiments

We experimentally compare the performance of the arc-based formulation and the pattern-based formulations solved with branch-price-and-cut. For $\mathcal{A}_{\mathrm{S}}$, the problem-specific cuts are treated as an integral part of the final solver. Preliminary runs without these cuts led to a severe degradation in performance, so we do not report this variant separately. First, we present the instance sets and the platforms used. We then evaluate and discuss the results.

4.1 Platform and Instances

For our experiments, we generated random instances based on instances of the TSPLIB (Reinelt, 1991), excluding instances with more than 500 nodes. We processed the TSP instances using the Python library tsplib95 (version 0.7.1). Each instance inherits the graph of the corresponding TSP instance and the cost of an edge is equal to its length, so $\ell = c^E$. A station can be placed at any node (i.e., $F = V$). Let $c_{\mathrm{trips}} = \sum_{t \in T} \sum_{s \in S_t} \bar{c}_{t,s}$ be the total cost of all trips without detours. Then, we balance the trip and station costs such that the cost of building 20% of the stations is equal to c_{trips} (i.e., 20% of the stations are worth the same as a detour that doubles the trip cost). Hence, we uniformly set the installation cost of all stations $f \in F$ to

$$c^F(f) = 5\frac{c_{\mathrm{trips}}}{|F|} \quad .$$

We have created two instance sets, I_1 and I_2. Both instance sets contain the same number of instances with the same sizes (in terms of number of nodes and number of trips): for each TSP instance we created instances with 10, 20, 30, and 40 trips. In addition, for both instance sets and for each trip, the number of stops was chosen uniformly at random from $[2, \sqrt{|V|}]$. For instances of I_1, a stop of a trip was chosen uniformly at random from the set of all nodes without its direct predecessor. In contrast, for instances of I_2, a (new) stop of a trip was chosen based on a normal distribution and based on the distances to the last

three (already chosen) stops. Let ℓ_{trips} be the total length of all trips, i.e., $\ell_{\text{trips}} = c_{\text{trips}}$. We use the same range

$$b_t = \left\lfloor \frac{2}{3} \frac{\ell_{\text{trips}}}{|T|} \right\rfloor$$

for all trips $t \in T$, which ensures that a trip of average length cannot be completed without visiting at least one station. We ran these experiments on Debian 11 computing nodes equipped with two Intel Xeon L5630 processors (providing 16 logical processors in total) and 128 GB of DDR3 RAM. We enforced a time limit of one hour. All reported branch-price-and-cut runs use the preprocessing described in Section 3.5.

Moreover, we created one large instance based on real-world data. For this purpose, we used public GTFS data (MobilityData, 2019), containing public transportation schedules including stops and routes. We decided to use data of an intercity bus service. The data contain trips offered in Europe from January to March 2020. Since the raw data are not directly suitable for our purpose, we preprocess them as follows. We restrict attention to bus trips and remove trips of other types. First, we aggregate stops with the same postal code and choose as representative the stop that occurs most frequently in the data. Second, we merge representative stops whose routed distance in both directions is below 10 kilometers, again keeping the more frequently used representative. If clustering creates consecutive identical stops within a trip, we remove duplicates and discard trips that collapse to a single stop. We also remove trips that are contained as consecutive subsequences of other trips. Since the range of electric buses is roughly between 90 and 550 kilometers (Automotive World Ltd, 2021; Perumal, Lusby, and Larsen, 2022), we set the range to 250 kilometers and remove trips whose total routed length, rounded down to kilometers, does not exceed this value. After preprocessing, the final instance contains 1644 nodes, 2408 trips, and 17405 segments. Furthermore, all nodes are possible locations for stations. We computed directed routes between all remaining nodes with travel time as objective. The length ℓ of an edge corresponds to the routed distance in kilometers and the cost c^E to the routed travel time in minutes. The underlying route lengths and route times are rounded down to kilometers and minutes, respectively. Hence, the instance is a discretized approximation of the routed network. Minor numerical deviations from the triangle inequality may therefore remain. Furthermore, we balanced station cost against trip cost by setting the cost of one station to 15% of the average travel time of all trips, i.e.,

$$c^F(f) = \left\lfloor 0.15 \frac{c_{\text{trips}}}{|T|} \right\rfloor = 65$$

for all stations $f \in F$. Thus, if the travel time of a trip would increase by more than this value, it is beneficial to install a proper station (which does not imply a detour). We ran this instance (without a time limit) on a Debian 11 workstation equipped with an Intel i7-8700 processor (providing 12 logical processors in total) and 32 GB of DDR4 RAM.

4.2 Results and Discussion

In the following, we present and discuss the results of the conducted experiments. For the synthetic benchmarks, we focus on three questions: which exact approach is computationally strongest overall, which approach yields the best bounds when the time limit is reached, and how the branching behavior of $\mathcal{A}_{\text{S}}$ and $\mathcal{A}_{\text{N}}$ differs. The GTFS-based case study serves a different purpose: it illustrates how the strongest approach behaves on

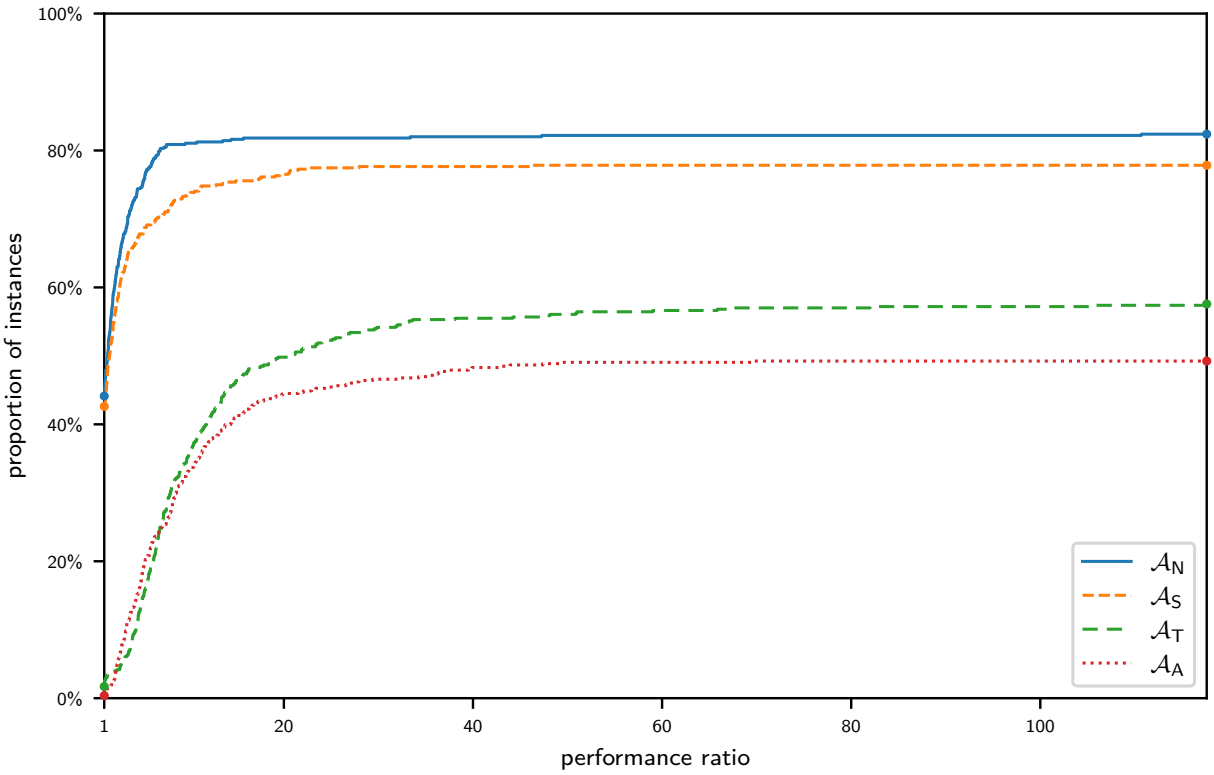


Figure 8: Performance profile with respect to the solving times. The x-axis is truncated for better readability. The right dot of each line corresponds to the proportion of instances solved to optimality within the time limit.

a large transportation instance derived from public data. Appendix A reports the per-instance solving times and gaps in Tables A.1–A.4 and the performance profiles for each instance set and size category in Figure A.1. In the following, all box plots show the first, second, and third quartiles of the data. Moreover, the whiskers in a box plot visualize the range of all data points that fall within the range of the first quartile minus 1.5 times the interquartile range and the third quartile plus 1.5 times the interquartile range. Outliers are not shown unless otherwise stated. Figure 8 shows the performance profiles (Dolan and Moré, 2002) with respect to the solving time, Figure 9 shows the proportions of instances solved to optimality within the time limit for each synthetic instance family and trip count, and Figure 10 summarizes the remaining relative gaps after an approach terminated. The relative gap is defined as

$$gap = \frac{\bar{z} - \underline{z}}{\underline{z}} \ ,$$

where $\bar{z}$ is the best found primal bound and $\underline{z}$ is the best computed dual bound. If instances are infeasible and the solver detected this within the time limit, the gap is recorded as 0%.

The aggregated performance profiles in Figure 8 show that $\mathcal{A}_N$ and $\mathcal{A}_S$ dominate the other approaches. Overall, $\mathcal{A}_N$ is the fastest approach on the largest fraction of instances (44.1%), followed by $\mathcal{A}_S$ (42.6%), and it solves 82.4% of all instances to optimality within the time limit, compared with 77.8% for $\mathcal{A}_S$. Figure 9 shows, however, that this overall comparison depends strongly on the instance family: $\mathcal{A}_S$ is consistently stronger on the uniformly generated instances I_1, where it solves between 78.8% and 100.0% of the instances for each trip count, whereas $\mathcal{A}_N$ is consistently stronger on the more clustered

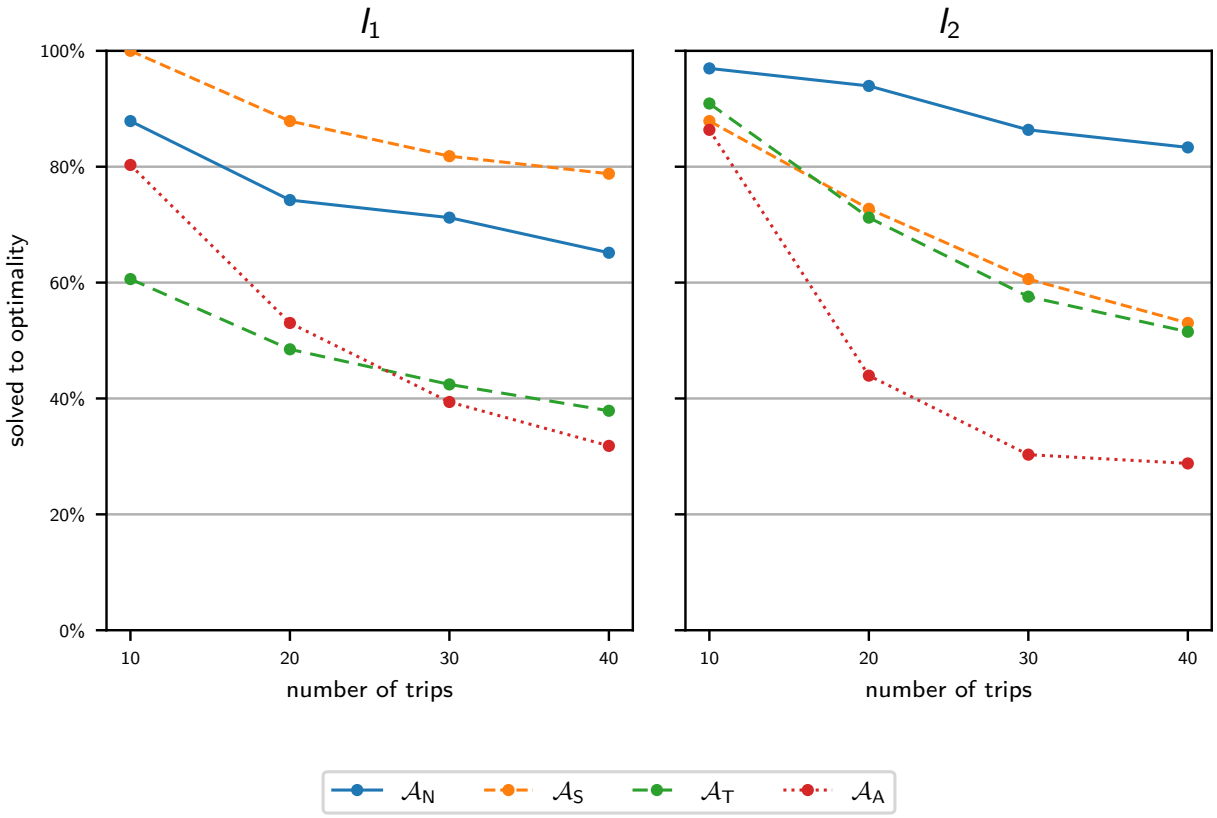


Figure 9: Proportions of synthetic instances solved to optimality within the time limit. The two plots correspond to the instance families I_1 and I_2, respectively.

instances I_2, where it solves between 83.3% and 97.0% of the instances. Thus, the nested approach performs best overall, while the segment-pattern-based formulation remains highly competitive on I_1. The picture is similar if we look at the remaining gaps when the approach terminates (either optimal or when the time limit is reached). Figure 10 summarizes these remaining gaps over all synthetic instances. Both $\mathcal{A}_N$ and $\mathcal{A}_S$ terminate with markedly smaller gaps than $\mathcal{A}_T$ and $\mathcal{A}_A$. As in the timing results, the appendix plots show that $\mathcal{A}_S$ has an advantage on I_1, whereas $\mathcal{A}_N$ is stronger on I_2. Moreover, $\mathcal{A}_T$ typically terminates with a smaller remaining relative gap compared to $\mathcal{A}_A$. This is consistent with the fact that $\mathcal{A}_T$ manages to solve more instances to optimality within the time limit than $\mathcal{A}_A$. Note that the maximum remaining gap achieved by $\mathcal{A}_A$ is infinite, since it is unable to compute the required bounds for some instances.

Furthermore, we want to provide some insights into the branching behavior of $\mathcal{A}_S$, $\mathcal{A}_T$, and $\mathcal{A}_N$. Figure 11 shows that $\mathcal{A}_S$ requires branching on 96.4% of all synthetic instances, whereas the branching incidence is only 8.7% for $\mathcal{A}_T$ and 14.8% for $\mathcal{A}_N$. Conditional on branching, the median number of created tree nodes is 921 for $\mathcal{A}_S$ but only 2 for both $\mathcal{A}_T$ and $\mathcal{A}_N$. Hence, the main branching burden is specific to $\mathcal{A}_S$, while the stronger outer reformulation used by $\mathcal{A}_T$ and $\mathcal{A}_N$ usually suffices to prove optimality without branching. Since $\mathcal{A}_T$ and $\mathcal{A}_N$ branch only rarely, the additional computational advantage of $\mathcal{A}_N$ over $\mathcal{A}_T$ is primarily due to the nested pricing approach rather than a markedly different branching behavior. Nevertheless, $\mathcal{A}_T$ has a lower branching incidence and requires fewer branching nodes on average, indicating that it produces columns that are better suited to reach integrality. Since both approaches rely on the same master problem, this leaves room for improving $\mathcal{A}_N$.

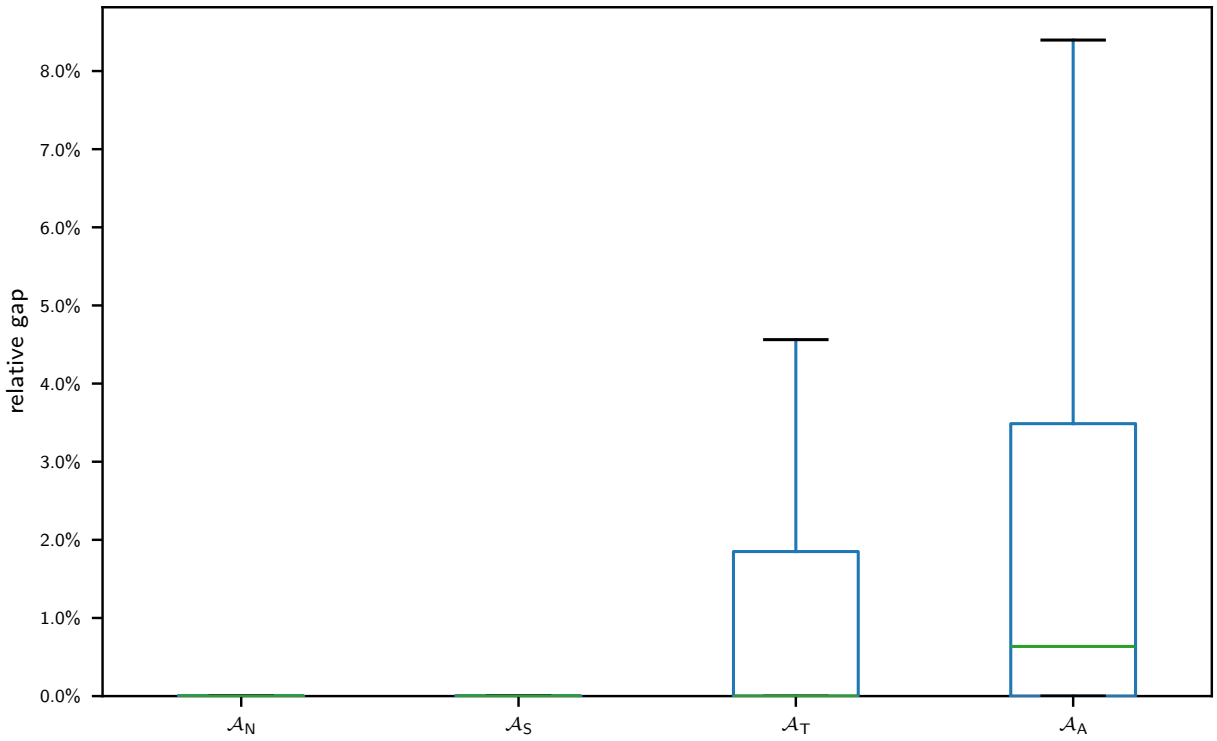
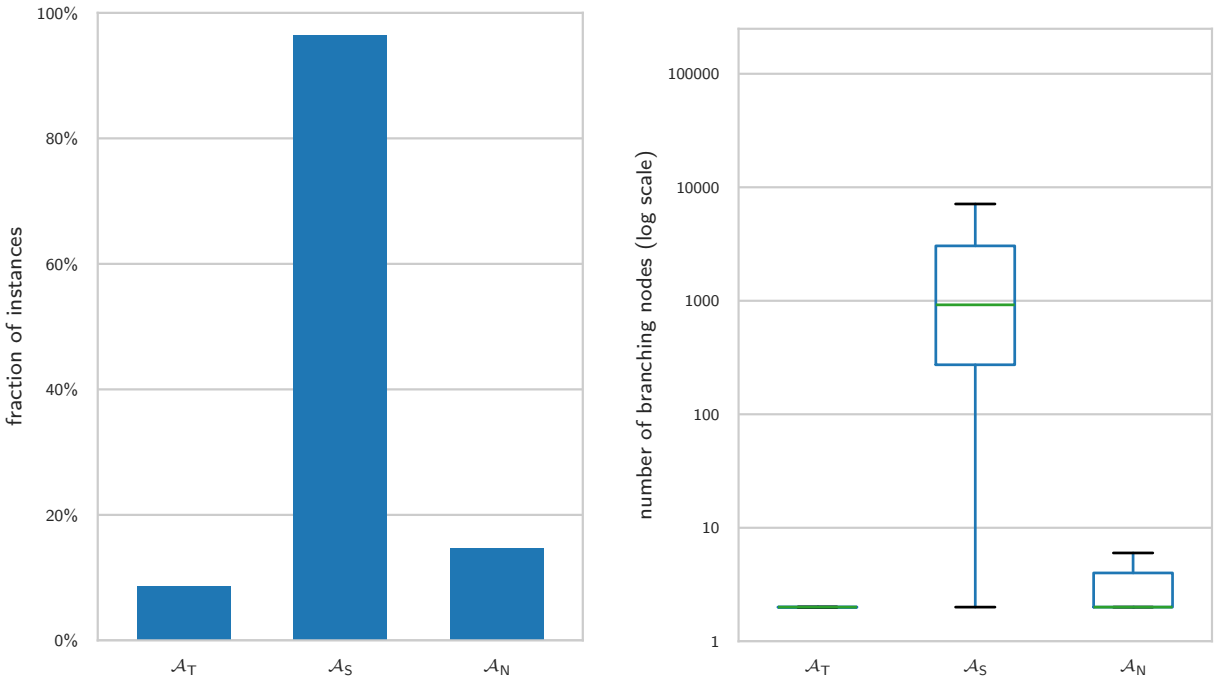


Figure 10: Remaining relative gaps after the approaches terminated, aggregated over all synthetic instances.

Finally, we evaluate our approaches on a real-world instance. Preliminary experiments with large real-world instances showed that $\mathcal{A}_N$ performs best among all our presented approaches, which is consistent with the results presented earlier. In particular, the memory requirements of $\mathcal{A}_A$ are so large that we were not able to run it on large real-world instances. On such instances, even the restricted master problem of $\mathcal{A}_S$ dominates the runtime, so that the advantage of the fast pricing problems vanishes. Therefore, we only evaluate $\mathcal{A}_N$ on the real-world instance. The solving process terminated after 40.06 hours with an optimal and feasible solution. The found optimal solution and two illustrative high-detour trips are visualized in Figure 12. A total of 297 stations are selected in the solution. Table 1 reports the solve time, the number of selected stations, and selected summary statistics of the resulting travel-time increases.

Figure 13 summarizes these travel-time changes over all trips. The left subfigure shows the empirical cumulative distribution function (ECDF) of the relative travel-time increase compared with the input trip times, and the right subfigure shows the ordered tail of the 50 largest additional travel times. On average, the travel time increases by about 1.2%, and the median increase is about 0.2%. The mean additional travel time is 6.0 minutes and the median is 1.0 minute. Even the 90th and 95th percentiles of the additional travel time are only 15.0 and 24.7 minutes, respectively. Together, these diagnostics show both the concentration of small changes and the magnitude of the worst affected trips.

The real-world instance also yields useful planning diagnostics. As Figure 13 shows, most trips exhibit only small increases in travel time, while only a small tail incurs substantial additional travel time. These trips merit closer inspection in a planning study. The two right-hand subfigures of Figure 12 show two illustrative high-detour trips. The upper subfigure shows the most extreme case, which almost doubles its travel time and incurs an additional 331 minutes. This suggests that the current candidate-station set



(a) branching incidence (b) node counts if branching occurs

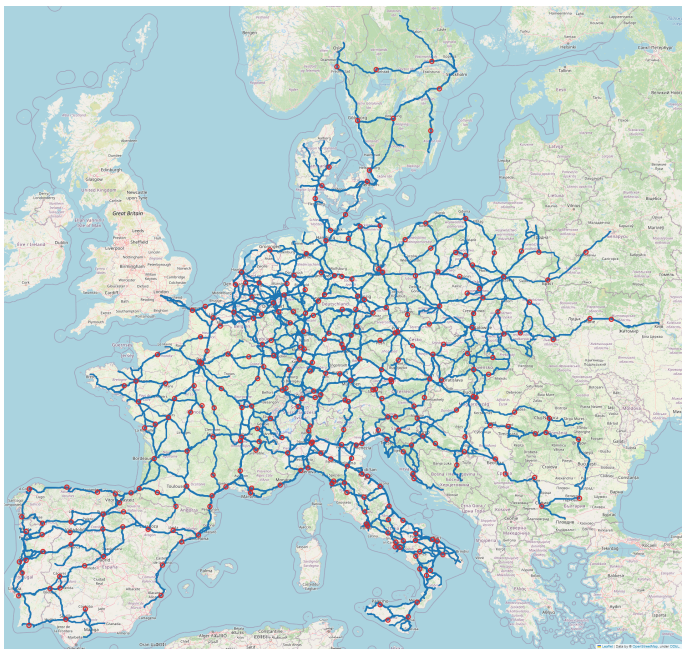
Figure 11: Branching incidence and branching-node counts on the synthetic instances. The right subfigure uses a logarithmic y-axis and considers only instances for which branching occurs.

leaves poor charging opportunities in that part of the network, so the trip has to be rerouted between distant stations. The lower subfigure is representative of a broader pattern: several of the next-largest detours share essentially the same detour corridor. Hence, adding candidate stations in that sparse area would not only improve one isolated route but several of the most affected trips simultaneously. Table 2 summarizes the seven repeated detour areas with the largest cumulative additional travel time over all segments. We define a detour area by a set of inserted off-route charging stops used by segments in the solution, and name it after the region of these stops. Detour areas are merged if one is a subset of another. The table reports the number of affected segments, the cumulative additional travel time, the share of the total additional travel time over all segments, and the maximum additional travel time caused by the detour area. The listed seven areas affect 361 trips and account for 44.1% of the total additional travel time. The table shows both frequent moderate detours and smaller groups with much larger maxima.

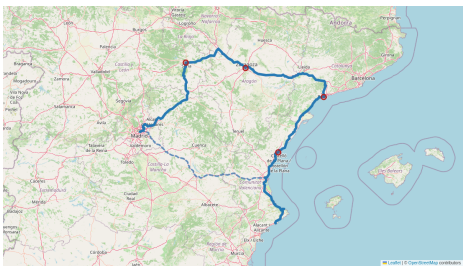
This observation also supports our decision not to impose an explicit detour bound. Large detours are rarely attractive in the optimum, but a strict bound could render such an instance infeasible and would hide where the candidate-station set is too sparse.

5 Conclusion

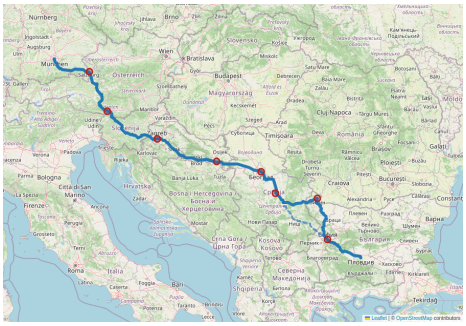
In this paper, we study exact approaches to the (directed) multi-stop station location problem. First, we present an arc-based mixed-integer formulation that can be solved with any MILP solver. Then, we propose several pattern-based formulations solved by branch-price-and-cut. Specifically, we propose a pattern-based formulation that assigns a complete path to each trip and another formulation that assigns paths to the segments of a



(a) overall solution

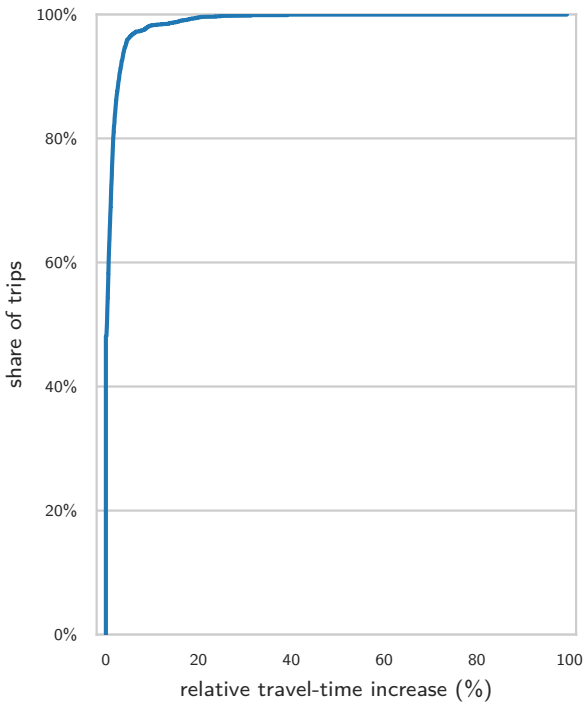


(b) most extreme detour

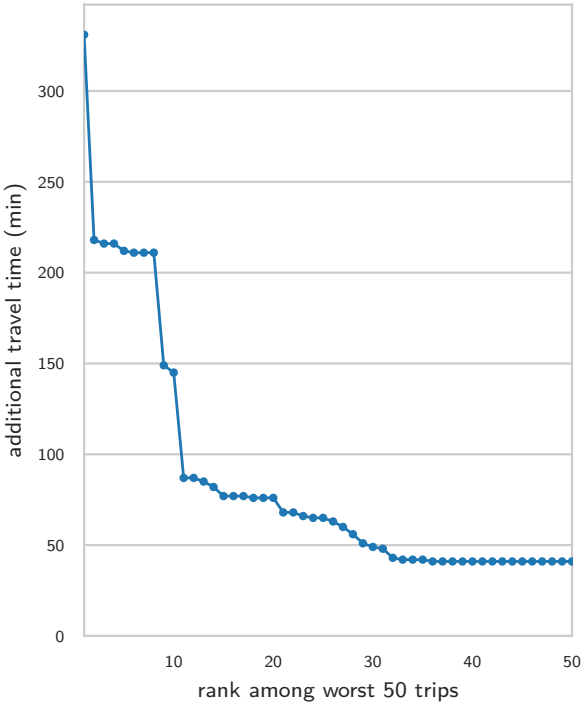


(c) shared-corridor detour

Figure 12: The best found solution (left) and two illustrative high-detour trips (right) are visualized. In the two right-hand subfigures, dashed lines indicate the original routes and solid lines the detoured routes in the solution. Trips are represented by blue lines and charging stations by red dots. The visualization relies on data licensed under the Open Data Commons Open Database License (ODbL) by the OpenStreetMap Foundation.



(a) ECDF of relative increases



(b) tail of additional travel times

Figure 13: GTFS travel-time diagnostics. The left subfigure shows the empirical cumulative distribution function (ECDF) of the relative travel-time increase, in percent. The right subfigure shows the 50 largest additional travel times, in minutes.

Table 1: Summary statistics for the GTFS-based case study.

Statistic	Value
Solve time (h)	40.06
Selected stations	297
Mean relative increase (%)	1.21
Median relative increase (%)	0.22
Mean additional travel time (min)	6.03
Median additional travel time (min)	1.00
90th percentile additional travel time (min)	15.00
95th percentile additional travel time (min)	24.65
Maximum additional travel time (min)	331.00

Table 2: Detour areas with the largest cumulative additional travel time over all segments.

Detour area	Segments	Sum add. time (min)	Share of total (%)	Max add. time (min)
Mecklenburg-Vorpommern	41	1566.0	10.8	41.0
Central Serbia	7	1495.0	10.3	218.0
Budapest	74	949.0	6.5	31.0
Bourgogne	83	775.0	5.3	18.0
Campania	140	735.0	5.0	15.0
Banskobystricky	6	459.0	3.2	77.0
Castille and Leon	10	448.0	3.1	77.0

trip. Our approaches use a problem-specific primal heuristic and branching strategies, and for the segment-pattern-based formulation also problem-specific cuts. We conduct experiments using randomly generated instances based on TSPLIB instances. Our experiments show that the pattern-based formulations clearly outperform the arc-based formulation. Among them, the segment-pattern-based formulation is strongest on the uniformly generated instance family, whereas the nested approach performs best overall and is clearly strongest on the clustered family. Moreover, we create a large instance using real-world data of an intercity bus service to illustrate that the nested approach also scales to a large public-data transportation instance. In fact, the nested branch-price-and-cut approach is able to solve this instance to optimality in less than two days.

The primary purpose of our paper is to introduce a sound exact solution methodology for a new combinatorial optimization problem that is motivated by, among others, infrastructure planning for electromobility. We therefore deliberately adopt a foundational scope. Nevertheless, our work naturally suggests several extensions of different difficulty. Some seem accessible with comparatively small changes to the model or even only the data. For example, one might want to weight or prioritize trips individually or impose capacity limits at stations. Other extensions require more substantial changes but still seem within reach with the presented setup, such as upper bounding the extra costs caused by detours. Extensions that introduce an explicit timing component, for example to respect charging profiles, would change the character of the problem more fundamentally. Yet, we hope to have provided a solid foundation to build on in order to answer such questions in the future.

CRediT authorship contribution statement

Erik Mühmer: Conceptualization, data curation, formal analysis, investigation, methodology, software, validation, visualization, writing - original draft, writing - review and editing.

Miriam Ganz: Conceptualization, methodology, writing - original draft, writing - review and editing.

Marco E. Lübbecke: Conceptualization, methodology, supervision, writing - review and editing.

Felix J. L. Willamowski: Conceptualization, methodology, writing - review and editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Funding

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Data availability

The source code, synthetic benchmark instances, and manuscript evaluation outputs underlying this article will be made publicly available at `https://github.com/erik-m/mslp-exact/` upon publication. The GTFS-based case study is derived from publicly available timetable data together with routing information based on public map data. Because the legal status of redistributing the fully processed GTFS-derived instance is still being clarified with respect to third-party data terms, those derived files are not included in the present public package. Additional derived materials may be provided by the corresponding author upon reasonable request, subject to third-party data restrictions.

Declaration of generative AI and AI-assisted technologies in the manuscript preparation process

During the preparation of this work the authors used GitHub Copilot (GPT-5.4) and Anthropic Claude Opus 4.8 to support revision planning, improve wording and readability, and for updates to evaluation scripts used to generate manuscript figures and tables. Moreover, they used this tool to check for missed important related literature and to discuss and improve figures and tables. After using it, the authors reviewed and edited the content as needed and take full responsibility for the content of the published article.

Particularly, the first version of the manuscript and the source code was created without the above mentioned tools.

References

Achterberg, Tobias (2007). "Constraint Integer Programming". PhD thesis. Technical University of Berlin.

Alamatsaz, Kayhan, Frédéric Quesnel, and Ursula Eicker (2024). *Joint optimization of electric bus scheduling and fast charging infrastructure location planning*. Les Cahiers du GERAD G–2024–28. GERAD.

Almouhanna, Abdullah, Carlos L. Quintero-Araujo, Javier Panadero, Angel A. Juan, Banafsheh Khosravi, and Djamila Ouelhadj (2020). "The location routing problem using electric vehicles with constrained distance". In: *Computers & Operations Research* 115, 104864. DOI: 10.1016/j.cor.2019.104864.

Automotive World Ltd (2021). *MAN sets the standard for range: Fully electric bus breaks the 550 kilometre barrier*. URL: https://www.automotiveworld.com/news-releases/man-sets-the-standard-for-range-fully-electric-bus-breaks-the-550-kilometre-barrier/ (visited on 04/03/2023).

Bestuzheva, Ksenia, Mathieu Besançon, Wei-Kun Chen, Antonia Chmiela, Tim Donkiewicz, Jasper van Doornmalen, Leon Eifler, Oliver Gaul, Gerald Gamrath, Ambros Gleixner, Leona Gottwald, Christoph Graczyk, Katrin Halbig, Alexander Hoen, Christopher Hojny, Rolf van der Hulst, Thorsten Koch, Marco Lübbecke, Stephen J. Maher, Frederic Matter, Erik Mühmer, Benjamin Müller, Marc E. Pfetsch, Daniel Rehfeldt, Steffan Schlein, Franziska Schlösser, Felipe Serrano, Yuji Shinano, Boro Sofranac, Mark Turner, Stefan Vigerske, Fabian Wegscheider, Philipp Wellner, Dieter Weninger, and Jakob Witzig (2021). *The SCIP optimization suite 8.0*. arXiv. DOI: 10.48550/ARXIV.2112.08872.

Cabral, Edgar Alberto, Erhan Erkut, Gilbert Laporte, and Raymond A. Patterson (2007). "The network design problem with relays". In: *European Journal of Operational Research* 180.2, pp. 834–844. DOI: 10.1016/j.ejor.2006.04.030.

Calik, Hatice, Ammar Oulamara, Caroline Prodhon, and Said Salhi (2018). "The electric location-routing problem: formulation and Benders decomposition approach". In: *Computers & Operations Research* 131, 105251. DOI: 10.1016/j.cor.2021.105251.

Capar, Ismail and Michael Kuby (2012). "An efficient formulation of the flow refueling location model for alternative-fuel stations". In: *IIE Transactions* 44.8, pp. 622–636. DOI: 10.1080/0740817X.2011.635175.

Chen, Si, Ivana Ljubić, and S. Raghavan (2010). "The regenerator location problem". In: *Networks* 55.3, pp. 205–220. DOI: 10.1002/net.20366.

— (2015). "The generalized regenerator location problem". In: *INFORMS Journal on Computing* 27.2, pp. 204–220. DOI: 10.1287/ijoc.2014.0621.

Chen, Yanru, Decheng Li, Zongcheng Zhang, M. I. M. Wahab, and Yangsheng Jiang (2021). "Solving the battery swap station location-routing problem with a mixed fleet of electric and conventional vehicles using a heuristic branch-and-price algorithm with an adaptive selection scheme". In: *Expert Systems with Applications* 186, p. 115683. DOI: 10.1016/j.eswa.2021.115683.

Dantzig, George B. and Philip Wolfe (1960). "Decomposition principle for linear programs". In: *Operations Research* 8.1, pp. 101–111. DOI: 10.1287/opre.8.1.101.

Desrosiers, J., M. Lübbecke, G. Desaulniers, and J.B. Gauthier (2026). *Branch-and-Price*. Springer Cham. DOI: 10.1007/978-3-031-96917-1.

Dinç Yalçin, Gülçin, Ece Doğan, Ahmed Yasin Kaya, and Enes Can Oğuz (2023). "A mathematical model for the location of charging station for electric buses". In: *Endüstri Mühendisliği* 34.2, pp. 184–200. DOI: 10.46465/endustrimuhendisligi.1165292.

Dolan, Elizabeth D. and Jorge J. Moré (2002). "Benchmarking optimization software with performance profiles". In: *Mathematical Programming* 91, pp. 201–213. DOI: 10.1007/s101070100263.

Esmaeilnejad, Seyedshahab, Lina Kattan, and S. Chan Wirasinghe (2023). "Optimal charging station locations and durations for a transit route with battery-electric buses: a two-stage stochastic programming approach with consideration of weather conditions". In: *Transportation Research Part C: Emerging Technologies* 156, p. 104327. DOI: 10.1016/j.trc.2023.104327.

Funke, Stefan, Andre Nusser, and Sabine Storandt (2015). "Placement of loading stations for electric vehicles: no detours necessary!" In: *Journal of Artificial Intelligence Research* 53, pp. 633–658. DOI: 10.1613/jair.4688.

Göpfert, Paul and Stefan Bock (2019). "A branch&cut approach to recharging and refueling infrastructure planning". In: *European Journal of Operational Research* 279.3, pp. 808–823. DOI: 10.1016/j.ejor.2019.06.031.

Gurobi Optimization, LLC. (2023). *Gurobi Optimizer*. URL: https://www.gurobi.com/solutions/gurobi-optimizer/ (visited on 04/03/2023).

Hartstein, Itamar, Mordechai Shalom, and Shmuel Zaks (2016). "On the complexity of the regenerator location problem treewidth and other parameters". In: *Discrete Applied Mathematics* 199, pp. 199–225. DOI: 10.1016/j.dam.2015.01.036.

He, Yi, Ziqi Song, and Zhaocai Liu (2025). "Fast-charging station deployment for battery electric bus systems considering electricity demand charges". In: *Sustainable Cities and Society* 48, 101530 (). DOI: 10.1016/j.scs.2019.101530. (Visited on 01/30/2025).

Hof, Julian, Michael Schneider, and Dominik Goeke (2017). "Solving the battery swap station location-routing problem with capacitated electric vehicles using an AVNS algorithm for vehicle-routing problems with intermediate stops". In: *Transportation Research Part B: Methodological* 97, pp. 102–112. DOI: 10.1016/j.trb.2016.11.009.

Hosseini, M., S.A. MirHassani, and F. Hooshmand (2017). "Deviation-flow refueling location problem with capacitated facilities: model and algorithm". In: *Transportation Research Part D: Transport and Environment* 54, pp. 269–281. DOI: 10.1016/j.trd.2017.05.015.

Huang, Yongxi, Shengyin Li, and Zhen Sean Qian (2015). "Optimal deployment of alternative fueling stations on transportation networks considering deviation paths". In: *Networks and Spatial Economics* 15.1, pp. 183–204. DOI: 10.1007/s11067-014-9275-1.

Kabadurmus, Ozgur and Alice E. Smith (2016). "Multi-commodity k-splittable survivable network design problems with relays". In: *Telecommunication Systems* 62.1, pp. 123–133. DOI: 10.1007/s11235-015-0067-9.

Kim, Jong-Geun and Michael Kuby (2012). "The deviation-flow refueling location model for optimizing a network of refueling stations". In: *International Journal of Hydrogen Energy* 37.6, pp. 5406–5420. DOI: 10.1016/j.ijhydene.2011.08.108.

— (2013). "A network transformation heuristic approach for the deviation flow refueling location model". In: *Computers & Operations Research* 40.4, pp. 1122–1131. DOI: 10.1016/j.cor.2012.10.021.

Kınay, Ömer Burak, Fatma Gzara, and Sibel A Alumur (2021). "Full cover charging station location problem with routing". In: *Transportation Research Part B: Methodological* 144, pp. 1–22. DOI: 10.1016/j.trb.2020.12.001.

Kuby, Michael and Seow Lim (2005). "The flow-refueling location problem for alternative-fuel vehicles". In: *Socio-Economic Planning Sciences* 39.2, pp. 125–145. DOI: 10.1016/j.seps.2004.03.001.

— (2007). "Location of alternative-fuel stations using the flow-refueling location model and dispersion of candidate sites on arcs". In: *Networks and Spatial Economics* 7.2, pp. 129–152. DOI: 10.1007/s11067-006-9003-6.

Kuby, Michael, Lee Lines, Ronald Schultz, Zhixiao Xie, Jong-Geun Kim, and Seow Lim (2009). "Optimization of hydrogen stations in Florida using the flow-refueling location model". In: *International Journal of Hydrogen Energy* 34.15, pp. 6045–6064. DOI: 10.1016/j.ijhydene.2009.05.050.

Kunith, Alexander, Roman Mendelevitch, and Dietmar Goehlich (2017). "Electrification of a city bus network—an optimization model for cost-effective placing of charging infrastructure and battery sizing of fast-charging electric bus systems". In: *International Journal of Sustainable Transportation* 11.10, pp. 707–720. DOI: 10.1080/15568318.2017.1310962.

Leitner, Markus, Ivana Ljubić, Martin Riedler, and Mario Ruthmair (2019). "Exact approaches for network design problems with relays". In: *INFORMS Journal on Computing* 31.1, pp. 171–192. DOI: 10.1287/ijoc.2018.0820.

— (2020). "Exact approaches for the directed network design problem with relays". In: *Omega* 91, 102005. DOI: 10.1016/j.omega.2018.11.014.

Li, Xiangyong, Shaochong Lin, Peng Tian, and Y.P. Aneja (2017). "Models and column generation approach for the resource-constrained minimum cost path problem with relays". In: *Omega* 66, pp. 79–90. DOI: 10.1016/j.omega.2016.01.012.

Lim, Seow and Michael Kuby (2010). "Heuristic algorithms for siting alternative-fuel stations using the flow-refueling location model". In: *European Journal of Operational Research* 204.1, pp. 51–61. DOI: 10.1016/j.ejor.2009.09.032.

Lin, Cheng-Chang and Chuan-Chih Lin (2018). "The p-center flow-refueling facility location problem". In: *Transportation Research Part B: Methodological* 118, pp. 124–142. DOI: 10.1016/j.trb.2018.10.008.

MirHassani, S. A. and R. Ebrazi (2013). "A flexible reformulation of the refueling station location problem". In: *Transportation Science* 47.4, pp. 617–628. DOI: 10.1287/trsc.1120.0430.

MobilityData (2019). *General Transit Feed Specification Reference*. URL: https://gtfs.org/reference/static (visited on 04/03/2023).

Nayeem, Moddassir Khan, Fuhad Ahmed Opu, Omar Abbaas, and Sara Abu-Aridah (2025). *Finite dominating sets for the refueling station location problem in fleet operations*. Version Number: 1. DOI: 10.48550/ARXIV.2509.11441. URL: https://arxiv.org/abs/2509.11441 (visited on 04/02/2026).

Nigam, Ashutosh and Yogesh K. Agarwal (2014). "Optimal relay node placement in delay constrained wireless sensor network design". In: *European Journal of Operational Research* 233.1, pp. 220–233. DOI: 10.1016/j.ejor.2013.08.031.

Nordlund, Nicholas, Leandros Tassiulas, and Jan-Hendrik Lange (2023). *Optimization methods for the capacitated refueling station location problem with routing*. arXiv. DOI: 10.48550/arXiv.2310.05569.

OpenMP Architecture Review Board (2018). *OpenMP Application Programming Interface*. URL: https://www.openmp.org/wp-content/uploads/OpenMP-API-Specification-5.0.pdf (visited on 04/03/2023).

Perumal, Shyam S.G., Richard M. Lusby, and Jesper Larsen (2022). "Electric bus planning & scheduling: a review of related problems and methodologies". In: *European Journal of Operational Research* 301.2, pp. 395–413. DOI: 10.1016/j.ejor.2021.10.058.

Reinelt, Gerhard (1991). "TSPLIB—A traveling salesman problem library". In: *ORSA Journal on Computing* 3.4, pp. 376–384. DOI: 10.1287/ijoc.3.4.376.

Rose, Philipp K., Rizqi Nugroho, Till Gnann, Patrick Plötz, Martin Wietschel, and Melanie Reuter-Oppermann (2020). "Optimal development of alternative fuel station networks considering node capacity restrictions". In: *Transportation Research Part D: Transport and Environment* 78, p. 102189. DOI: 10.1016/j.trd.2019.11.018.

Scheiper, Barbara, Maximilian Schiffer, and Grit Walther (2019). "The flow refueling location problem with load flow control". In: *Omega* 83, pp. 50–69. DOI: 10.1016/j.omega.2018.02.003.

Schiffer, Maximilian, Michael Schneider, Grit Walther, and Gilbert Laporte (2019). "Vehicle routing and location routing with intermediate stops: a review". In: *Transportation Science* 53.2, pp. 319–343. DOI: 10.1287/trsc.2018.0836.

Schiffer, Maximilian and Grit Walther (2017). "The electric location routing problem with time windows and partial recharging". In: *European Journal of Operational Research* 260.3, pp. 995–1013. DOI: 10.1016/j.ejor.2017.01.011.

Siek, Jeremy, Lie-Quan Lee, and Andrew Lumsdaine (2001). *The Boost Graph Library (BGL)*. URL: https://www.boost.org/doc/libs/1_75_0/libs/graph/doc/index.html (visited on 04/03/2023).

Storandt, Sabine and Stefan Funke (2013). "Enabling e-mobility: facility location for battery loading stations". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 27.1, pp. 1341–1347. DOI: 10.1609/aaai.v27i1.8478.

Stumpe, Miriam (2024). "A new mathematical formulation for the simultaneous optimization of charging infrastructure and vehicle schedules for electric bus systems". In: *Transportation Research Procedia* 78, pp. 402–409. DOI: 10.1016/j.trpro.2024.02.051.

Tzamakos, Dionysios, Christina Iliopoulou, and Konstantinos Kepaptsoglou (2023). "Electric bus charging station location optimization considering queues". In: *International Journal of Transportation Science and Technology* 12.1, pp. 291–300. DOI: 10.1016/j.ijtst.2022.02.007.

Upchurch, Christopher and Michael Kuby (2010). "Comparing the p-median and flow-refueling models for locating alternative-fuel stations". In: *Journal of Transport Geography* 18.6, pp. 750–758. DOI: 10.1016/j.jtrangeo.2010.06.015.

Upchurch, Christopher, Michael Kuby, and Seow Lim (2009). "A model for location of capacitated alternative-fuel stations". In: *Geographical Analysis* 41.1, pp. 85–106. DOI: 10.1111/j.1538-4632.2009.00744.x.

Vries, Harwin de and Evelot Duijzer (2017). "Incorporating driving range variability in network design for refueling facilities". In: *Omega* 69, pp. 102–114. DOI: 10.1016/j.omega.2016.08.005.

Wang, Ying-Wei and Chuah-Chih Lin (2009). "Locating road-vehicle refueling stations". In: *Transportation Research Part E: Logistics and Transportation Review* 45.5, pp. 821–829. DOI: 10.1016/j.tre.2009.03.002.

Wang, Ying-Wei and Chuan-Ren Wang (2010). "Locating passenger vehicle refueling stations". In: *Transportation Research Part E: Logistics and Transportation Review* 46.5, pp. 791–801. DOI: 10.1016/j.tre.2009.12.001.

Willamowski, Felix J. L., Miriam Ganz, and Erik Mühmer (2020). *The multi-stop station location problem*. repORt 2020–60. Lehrstuhl für Operations Research, RWTH Aachen University.

Wolsey, Laurence A. (1990). "Valid inequalities for 0–1 knapsacks and MIPs with generalised upper bound constraints". In: *Discrete Applied Mathematics* 29.2, pp. 251–261. DOI: 10.1016/0166-218X(90)90148-6.

Worley, Owen, Diego Klabjan, and Timothy M. Sweda (2012). "Simultaneous vehicle routing and charging station siting for commercial electric vehicles". In: *2012 IEEE International Electric Vehicle Conference*. IEEE, pp. 1–3.

Yang, Jun and Hao Sun (2015). "Battery swap station location-routing problem with capacitated electric vehicles". In: *Computers & Operations Research* 55, pp. 217–232. DOI: 10.1016/j.cor.2014.07.003.

Yıldız, Barış, Okan Arslan, and Oya Ekin Karaşan (2016). "A branch and price approach for routing and refueling station location model". In: *European Journal of Operational Research* 248.3, pp. 815–826. DOI: 10.1016/j.ejor.2015.05.021.

Yıldız, Barış, Oya Ekin Karaşan, and Hande Yaman (2018). "Branch-and-price approaches for the network design problem with relays". In: *Computers and Operations Research* 92, pp. 155–169. DOI: 10.1016/j.cor.2018.01.004.

You, Peng-Sheng and Yi-Chih Hsieh (2014). "A hybrid heuristic approach to the problem of the location of vehicle charging stations". In: *Computers & Industrial Engineering* 70, pp. 195–204. DOI: 10.1016/j.cie.2014.02.001.

Zheng, Hong, Xiaozheng He, Yongfu Li, and Srinivas Peeta (2017). "Traffic equilibrium and charging facility locations for electric vehicles". In: *Networks and Spatial Economics* 17.2, pp. 435–457. DOI: 10.1007/s11067-016-9332-z.

A Appendix

Table A.1: Achieved times (in seconds) for each instance of I_1 and approach.

Instances	10 trips				20 trips				30 trips				40 trips			
	$\mathcal{A}_N$	$\mathcal{A}_S$	$\mathcal{A}_T$	$\mathcal{A}_A$	$\mathcal{A}_N$	$\mathcal{A}_S$	$\mathcal{A}_T$	$\mathcal{A}_A$	$\mathcal{A}_N$	$\mathcal{A}_S$	$\mathcal{A}_T$	$\mathcal{A}_A$	$\mathcal{A}_N$	$\mathcal{A}_S$	$\mathcal{A}_T$	$\mathcal{A}_A$
a280	1579	**301**	tl	2446	tl	**1131**	tl	tl	tl	**1808**	tl	tl	tl	**2724**	tl	tl
att48	9	**1**	60	7	41	**36**	246	80	**32**	38	130	211	**106**	148	380	492
bayg29	4	**1**	8	**1**	**6**	11	25	17	**6**	15	20	19	**13**	32	47	35
bays29	3	**1**	6	6	**2**	**2**	11	5	**3**	20	15	32	**2**	40	15	85
berlin52	**7**	10	68	62	**13**	61	164	361	376	**74**	661	278	**120**	146	761	959
bier127	562	**106**	tl	2439	951	**427**	tl	tl	1798	**691**	tl	tl	1862	**1299**	tl	tl
brazil58	9	**6**	159	60	72	**41**	412	285	**70**	191	649	605	168	**94**	1348	465
brg180	345	**10**	tl	152	284	**42**	1417	347	503	**91**	tl	531	621	**65**	901	534
burma14	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	1	**0**	**0**	**0**	**0**
ch130	177	**60**	620	628	436	**119**	2866	1277	644	**185**	tl	1941	1220	**421**	tl	tl
ch150	229	**67**	tl	966	2357	**830**	tl	tl	2058	**453**	tl	tl	**1438**	1769	tl	tl
d198	1530	**301**	tl	tl	tl	tl	tl	tl	tl	**3030**	tl	tl	tl	**1931**	tl	tl
d493	tl	**1473**	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl
dantzig42	5	**3**	89	15	**8**	13	93	35	**16**	27	110	135	**37**	89	196	80
eil101	96	**13**	746	295	135	**83**	1936	1286	484	**131**	tl	tl	331	**201**	3269	tl
eil51	21	**15**	241	36	40	**25**	288	123	**26**	29	258	140	66	**63**	412	282
eil76	35	**14**	629	116	81	**38**	411	349	**90**	121	1369	996	**109**	198	2310	2087
fl417	tl	**1849**	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl
fri26	**1**	**1**	4	8	6	**4**	22	11	**5**	12	20	13	**4**	5	32	32
gil262	2386	**1085**	tl	tl	3267	**1372**	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl
gr120	60	**23**	1190	393	252	**162**	tl	1844	947	**272**	tl	tl	3114	**1481**	tl	2646
gr137	171	**44**	785	104	680	**243**	3587	tl	1020	**442**	tl	2351	2924	**1730**	tl	tl
gr17	**0**	**0**	**0**	**0**	**0**	**0**	2	1	1	**0**	2	1	**1**	**1**	6	4
gr202	1346	**260**	tl	2626	tl	**757**	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl
gr21	**0**	**0**	**0**	**0**	**1**	**1**	6	2	**2**	7	6	8	**2**	8	6	8
gr229	662	**206**	2615	2387	tl	**2294**	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl
gr24	**1**	**1**	2	2	**1**	4	9	11	**1**	6	4	12	**4**	11	18	14
gr431	tl	**2416**	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl
gr48	5	**2**	30	18	110	**8**	244	62	**9**	12	164	95	**29**	41	228	641
gr96	75	**14**	957	87	109	**48**	322	596	269	**139**	1240	tl	**403**	914	tl	tl
hk48	7	**3**	40	29	**15**	46	83	53	61	**54**	307	118	**65**	127	186	165
kroA100	74	**15**	481	143	74	**52**	679	985	2924	**186**	3561	tl	629	**269**	tl	tl
kroA150	1999	**42**	tl	319	980	**705**	tl	tl	830	**401**	tl	tl	1611	**692**	tl	tl
kroA200	544	**128**	2620	1188	1245	**196**	tl	2075	2820	**801**	tl	tl	tl	**1752**	tl	tl
kroB100	54	**13**	643	218	361	**72**	900	615	468	**229**	tl	tl	430	**272**	3125	tl
kroB150	324	**77**	2460	tl	612	**181**	tl	1483	2726	**449**	tl	tl	1271	**1111**	tl	tl
kroB200	1655	**122**	tl	1801	2945	**435**	tl	tl	tl	**878**	tl	tl	tl	**3239**	tl	tl
kroC100	101	**25**	541	341	423	**79**	tl	1958	279	**152**	3446	2405	980	**515**	tl	tl
kroD100	60	**10**	152	173	197	**60**	1266	686	270	**164**	1378	978	717	**478**	tl	tl
kroE100	117	**27**	494	225	193	**181**	1288	2055	137	**82**	2719	759	378	**335**	3586	tl
lin105	1241	**11**	1179	65	203	**150**	1338	1794	305	**161**	2985	tl	1603	**245**	tl	tl
lin318	tl	**951**	tl	tl	tl	**3419**	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl
linhp318	1272	**699**	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl
pcb442	tl	**1107**	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl
pr107	713	**98**	tl	801	469	**357**	tl	tl	1500	**456**	tl	tl	**1188**	tl	tl	tl
pr124	259	**24**	1583	347	1450	**501**	tl	tl	**576**	881	tl	tl	**489**	1016	tl	tl
pr136	161	**46**	1114	522	tl	**406**	tl	tl	976	**717**	tl	tl	839	**482**	tl	tl
pr144	303	**210**	2323	2767	tl	**768**	tl	tl	397	**318**	tl	tl	tl	**1874**	tl	tl
pr152	215	**45**	933	259	437	**132**	2884	1204	1478	**731**	tl	tl	1119	**636**	tl	tl
pr226	tl	**2607**	tl	tl	tl	**2275**	tl	tl	tl	**1948**	tl	tl	tl	tl	tl	tl
pr264	1784	**826**	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl
pr299	tl	**231**	tl	1758	tl	**798**	tl	tl	tl	**2233**	tl	tl	tl	**2924**	tl	tl
pr439	3169	**791**	tl	3056	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl
pr76	55	**12**	355	60	**57**	84	756	610	**94**	185	1434	tl	**232**	307	2272	3473
rat195	542	**78**	tl	358	1903	**1188**	tl	tl	2878	**467**	tl	tl	tl	**1507**	tl	tl
rat99	40	**11**	288	154	109	**100**	1349	1144	178	**170**	2528	1044	485	**319**	2894	tl
rd100	224	**59**	tl	876	421	**133**	tl	tl	414	**137**	2232	tl	443	**160**	tl	tl
rd400	tl	**536**	tl	tl	tl	**2515**	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl
si175	1340	**193**	tl	2570	2200	**631**	tl	tl	2478	**988**	tl	tl	tl	**2019**	tl	tl
st70	14	**9**	237	81	145	**59**	914	280	389	**73**	1211	1443	**188**	208	2130	tl
swiss42	20	**7**	95	20	**15**	35	101	68	**24**	67	189	101	**59**	125	165	129
ts225	896	**359**	tl	tl	2380	**537**	tl	tl	tl	**1132**	tl	tl	tl	**1368**	tl	tl
tsp225	445	**113**	tl	975	2981	**482**	tl	tl	tl	**3271**	tl	tl	tl	**3378**	tl	tl
u159	144	**38**	1445	641	371	**194**	tl	tl	1848	**311**	tl	tl	1462	**483**	tl	tl
ulysses16	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
ulysses22	2	**1**	10	3	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

tl: time limit reached without proving optimality. Bold: shortest time per instance and number of trips.

Table A.2: Achieved gaps (as percentages) for each instance of I_1 and approach.

Instances	10 trips				20 trips				30 trips				40 trips			
	$\mathcal{A}_N$	$\mathcal{A}_S$	$\mathcal{A}_T$	$\mathcal{A}_A$	$\mathcal{A}_N$	$\mathcal{A}_S$	$\mathcal{A}_T$	$\mathcal{A}_A$	$\mathcal{A}_N$	$\mathcal{A}_S$	$\mathcal{A}_T$	$\mathcal{A}_A$	$\mathcal{A}_N$	$\mathcal{A}_S$	$\mathcal{A}_T$	$\mathcal{A}_A$
a280	**0.0**	**0.0**	2.1	**0.0**	0.1	**0.0**	3.6	2.1	0.8	**0.0**	3.7	3.0	1.8	**0.0**	3.7	–
att48	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
bayg29	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
bays29	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
berlin52	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
bier127	**0.0**	**0.0**	1.0	**0.0**	**0.0**	**0.0**	0.3	4.6	**0.0**	**0.0**	0.5	2.1	**0.0**	**0.0**	5.8	3.4
brazil58	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
brg180	**0.0**	**0.0**	0.1	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	2.0	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
burma14	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
ch130	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	0.3	**0.0**	**0.0**	**0.0**	0.1	2.2
ch150	**0.0**	**0.0**	0.2	**0.0**	**0.0**	**0.0**	1.1	2.8	**0.0**	**0.0**	0.3	2.4	**0.0**	**0.0**	1.4	3.0
d198	**0.0**	**0.0**	1.7	0.9	**0.0**	0.8	4.2	2.1	1.0	**0.0**	6.8	2.5	1.4	**0.0**	4.9	3.2
d493	1.2	**0.0**	1.2	1.8	2.0	**1.1**	2.1	–	3.9	**1.5**	5.2	–	3.5	**1.1**	3.6	–
dantzig42	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
eil101	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	1.0	1.5	**0.0**	**0.0**	**0.0**	1.8
eil51	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
eil76	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
fl417	1.3	**0.0**	2.7	1.5	2.4	**0.8**	5.3	–	2.4	**0.5**	2.7	–	2.6	**0.6**	2.6	–
fri26	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
gil262	**0.0**	**0.0**	2.2	2.2	**0.0**	**0.0**	4.2	1.9	3.5	**0.8**	5.0	2.0	4.2	**0.5**	5.9	3.0
gr120	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	1.3	2.0	**0.0**	**0.0**	0.8	**0.0**
gr137	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	1.1	**0.0**	**0.0**	0.5	**0.0**	**0.0**	**0.0**	5.6	3.3
gr17	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
gr202	**0.0**	**0.0**	1.8	**0.0**	0.7	**0.0**	9.0	3.3	3.2	**0.9**	6.2	2.3	2.8	**2.0**	6.4	3.7
gr21	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
gr229	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	4.8	2.4	2.3	**1.3**	4.0	2.8	4.2	**2.7**	7.8	7.3
gr24	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
gr431	1.0	**0.0**	2.9	0.6	1.7	**0.5**	1.9	–	**2.6**	3.3	5.0	–	3.2	**1.0**	3.3	–
gr48	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
gr96	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	2.2	**0.0**	**0.0**	0.1	4.4
hk48	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
kroA100	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	3.0	**0.0**	**0.0**	**0.0**	2.2
kroA150	**0.0**	**0.0**	0.4	**0.0**	**0.0**	**0.0**	0.8	3.2	**0.0**	**0.0**	1.5	1.0	**0.0**	**0.0**	1.3	1.3
kroA200	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	0.1	**0.0**	**0.0**	**0.0**	2.2	2.9	2.2	**0.0**	3.4	2.7
kroB100	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	0.1	2.2	**0.0**	**0.0**	**0.0**	0.9
kroB150	**0.0**	**0.0**	**0.0**	1.1	**0.0**	**0.0**	0.3	**0.0**	**0.0**	**0.0**	1.6	2.2	**0.0**	**0.0**	4.3	1.4
kroB200	**0.0**	**0.0**	0.2	**0.0**	**0.0**	**0.0**	2.0	0.4	0.7	**0.0**	3.1	2.5	**0.0**	**0.0**	8.3	2.8
kroC100	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	0.2	4.1
kroD100	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	1.1	2.5
kroE100	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	1.9
lin105	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	1.0	**0.0**	**0.0**	**0.0**	0.8
lin318	0.7	**0.0**	1.1	1.1	0.8	**0.0**	2.8	2.7	**1.2**	1.3	3.8	–	2.9	**1.3**	7.0	–
linhp318	**0.0**	**0.0**	0.5	1.5	1.3	**0.9**	4.2	1.9	5.9	**0.7**	3.3	–	**1.5**	1.6	3.4	–
pcb442	0.9	**0.0**	2.3	1.4	1.4	**0.4**	1.6	–	3.1	**1.3**	3.2	–	2.6	**1.3**	5.3	–
pr107	**0.0**	**0.0**	1.1	**0.0**	**0.0**	**0.0**	2.0	2.9	**0.0**	**0.0**	4.9	3.3	**0.0**	3.6	10.2	4.0
pr124	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	4.7	3.7	**0.0**	**0.0**	0.2	3.4	**0.0**	**0.0**	1.2	2.4
pr136	**0.0**	**0.0**	**0.0**	**0.0**	0.5	**0.0**	1.0	2.8	**0.0**	**0.0**	0.3	2.7	**0.0**	**0.0**	6.7	1.8
pr144	**0.0**	**0.0**	**0.0**	**0.0**	0.2	**0.0**	2.1	3.4	**0.0**	**0.0**	**0.0**	1.6	0.8	**0.0**	2.0	3.4
pr152	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	3.2	1.8	**0.0**	**0.0**	1.7	1.7
pr226	0.3	**0.0**	3.1	2.8	0.2	**0.0**	3.5	2.4	**0.0**	**0.0**	7.1	4.0	3.6	**2.2**	5.9	2.6
pr264	**0.0**	**0.0**	0.6	1.9	3.7	**0.7**	5.0	2.4	8.8	**1.9**	9.5	4.1	3.2	**0.8**	3.6	5.0
pr299	0.4	**0.0**	1.3	**0.0**	2.0	**0.0**	2.6	0.9	1.6	**0.0**	3.6	2.8	5.1	**0.0**	5.5	4.2
pr439	**0.0**	**0.0**	2.0	**0.0**	1.6	**1.0**	2.5	–	3.5	**1.5**	3.9	–	3.7	**1.6**	5.0	–
pr76	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	1.0	**0.0**	**0.0**	**0.0**	**0.0**
rat195	**0.0**	**0.0**	0.9	**0.0**	**0.0**	**0.0**	4.3	2.9	**0.0**	**0.0**	2.7	0.4	0.3	**0.0**	3.5	3.0
rat99	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	3.1
rd100	**0.0**	**0.0**	1.4	**0.0**	**0.0**	**0.0**	0.5	2.0	**0.0**	**0.0**	**0.0**	1.1	**0.0**	**0.0**	**0.0**	1.8
rd400	0.1	**0.0**	2.6	0.6	1.5	**0.0**	1.9	–	2.5	**1.3**	4.6	–	2.3	**1.1**	2.5	–
si175	**0.0**	**0.0**	1.4	**0.0**	**0.0**	**0.0**	2.1	2.7	**0.0**	**0.0**	2.7	2.9	0.1	**0.0**	2.5	2.5
st70	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	1.9
swiss42	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
ts225	**0.0**	**0.0**	0.2	2.0	**0.0**	**0.0**	0.7	1.2	0.1	**0.0**	4.1	2.6	0.6	**0.0**	1.9	4.3
tsp225	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	2.5	0.6	**0.0**	**0.0**	4.7	2.3	5.0	**0.0**	7.3	2.6
u159	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	1.3	**0.0**	**0.0**	0.8	1.4	**0.0**	**0.0**	1.2	1.2
ulysses16	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
ulysses22	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**

–: no primal bound found within the time limit. Bold: smallest gap per instance and number of trips.

Table A.3: Achieved times (in seconds) for each instance of I_2 and approach.

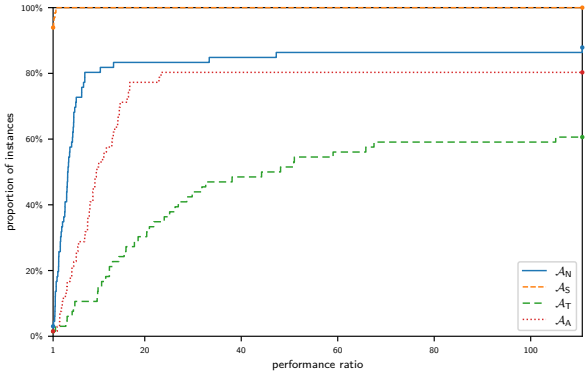
Instances	10 trips				20 trips				30 trips				40 trips			
	$\mathcal{A}_N$	$\mathcal{A}_S$	$\mathcal{A}_T$	$\mathcal{A}_A$	$\mathcal{A}_N$	$\mathcal{A}_S$	$\mathcal{A}_T$	$\mathcal{A}_A$	$\mathcal{A}_N$	$\mathcal{A}_S$	$\mathcal{A}_T$	$\mathcal{A}_A$	$\mathcal{A}_N$	$\mathcal{A}_S$	$\mathcal{A}_T$	$\mathcal{A}_A$
a280	**305**	552	2445	1015	**762**	tl	3331	tl	**2652**	tl	tl	tl	**3409**	tl	tl	tl
att48	**3**	4	47	13	**7**	14	77	25	**8**	61	58	226	**16**	126	81	246
bayg29	2	**1**	6	3	**2**	5	9	8	**9**	33	39	33	**7**	18	19	34
bays29	**1**	**1**	4	4	**2**	8	10	13	**5**	13	23	21	**2**	18	20	25
berlin52	6	**2**	30	9	**14**	31	90	595	**32**	144	218	359	**19**	260	129	768
bier127	321	**54**	2570	443	245	**147**	2012	1635	**700**	tl	tl	tl	391	**314**	tl	tl
brazil58	9	**5**	124	29	**15**	48	166	93	**37**	80	225	497	**35**	178	312	967
brg180	126	**35**	1027	551	193	**100**	1124	tl	**557**	1080	2986	tl	**721**	1900	tl	tl
burma14	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
ch130	**43**	53	127	186	**156**	371	1123	tl	**311**	3144	tl	tl	**254**	tl	1594	tl
ch150	**42**	64	233	218	**119**	2434	1243	tl	**295**	3310	tl	tl	**333**	tl	tl	tl
d198	263	**146**	2338	1446	**768**	tl	tl	tl	**3094**	tl	tl	tl	**1709**	tl	tl	tl
d493	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl
dantzig42	1	**0**	22	2	**14**	33	89	36	**19**	50	87	118	**17**	86	107	144
eil101	**56**	100	463	477	**60**	369	1726	tl	**121**	426	1466	tl	**444**	3430	tl	tl
eil51	22	**5**	66	30	**14**	49	114	151	**32**	81	194	424	**29**	378	234	1045
eil76	**3**	7	31	61	**52**	54	396	463	**128**	291	729	tl	**133**	2595	1287	tl
fl417	**903**	tl	tl	2386	**697**	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl
fri26	3	**1**	8	5	**2**	3	11	9	**2**	5	8	15	**3**	22	24	26
gil262	**240**	tl	1802	tl	**1527**	tl	tl	tl	tl	tl	tl	tl	**1359**	tl	tl	tl
gr120	29	**11**	286	130	**45**	180	303	1770	**84**	344	550	tl	**539**	1511	tl	tl
gr137	228	**84**	1856	570	252	**207**	1719	3309	**237**	793	1097	tl	**1139**	1570	tl	tl
gr17	**0**	**0**	1	**0**	1	**0**	7	1	1	**0**	1	**0**	1	**0**	1	1
gr202	295	**76**	2004	943	1670	**940**	tl	tl	**2895**	tl	tl	tl	tl	tl	tl	tl
gr21	**0**	**0**	1	**0**	**1**	**1**	4	3	**1**	3	8	4	**2**	14	8	14
gr229	**1769**	2243	tl	tl	**2306**	tl	tl	tl	**2363**	tl	tl	tl	**2495**	tl	tl	tl
gr24	**1**	**1**	4	3	**0**	**0**	1	1	**0**	3	2	3	**2**	7	8	15
gr431	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl
gr48	10	**5**	126	27	**12**	23	151	95	**10**	94	58	193	**20**	420	209	738
gr96	41	**19**	234	81	**102**	294	1234	tl	**147**	481	tl	tl	**94**	873	1041	tl
hk48	4	**3**	38	19	19	**17**	162	57	**79**	136	305	390	**25**	276	124	338
kroA100	66	**52**	618	257	260	**96**	1320	653	**105**	660	1401	tl	**167**	2927	702	tl
kroA150	**38**	110	241	125	**254**	838	669	tl	**399**	2561	2659	tl	**320**	tl	2254	tl
kroA200	**91**	1020	637	2677	**346**	367	2043	tl	**1850**	tl	tl	tl	**1070**	tl	tl	tl
kroB100	30	**21**	153	184	**49**	155	478	1581	**75**	1283	999	tl	**98**	512	998	tl
kroB150	**81**	1417	455	2791	**571**	765	tl	tl	**305**	tl	1726	tl	**620**	tl	tl	tl
kroB200	461	**155**	2999	2236	**508**	tl	3246	tl	**894**	tl	tl	tl	**987**	tl	tl	tl
kroC100	17	**9**	204	86	**44**	56	606	952	**72**	370	321	tl	**230**	888	1561	tl
kroD100	94	**15**	210	93	**113**	927	878	tl	**109**	511	606	tl	**255**	tl	2236	tl
kroE100	**67**	152	308	174	**81**	852	761	tl	**69**	537	526	tl	**103**	tl	750	tl
lin105	35	**33**	217	161	**139**	759	1778	tl	**135**	596	1028	tl	**310**	998	tl	tl
lin318	**370**	2716	2375	2741	**1072**	tl	tl	tl	**3016**	tl	tl	tl	**2683**	tl	tl	tl
linhp318	**425**	2835	1953	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl
pcb442	**626**	tl	1947	tl	**1878**	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl
pr107	**0**	**0**	**0**	8	78	**41**	300	180	**1**	**1**	**1**	26	**1**	**1**	**1**	44
pr124	**212**	409	716	1241	**147**	3169	1328	tl	**434**	tl	tl	tl	**132**	tl	2305	tl
pr136	**39**	49	214	377	232	**119**	554	1398	**208**	1347	2020	tl	**281**	2336	1371	tl
pr144	**84**	1900	336	tl	**132**	1225	598	tl	**358**	2879	1775	tl	**328**	tl	2394	tl
pr152	**40**	1114	534	80	**133**	598	1312	tl	**1420**	tl	tl	tl	tl	tl	tl	tl
pr226	**56**	tl	425	288	**215**	tl	tl	tl	**2234**	tl	tl	tl	**999**	tl	tl	tl
pr264	**235**	506	1222	501	**524**	tl	tl	tl	tl	tl	tl	tl	**2302**	tl	tl	tl
pr299	**477**	2108	2089	1049	**2038**	tl	tl	tl	**2729**	tl	tl	tl	tl	tl	tl	tl
pr439	**1525**	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl
pr76	158	**26**	3074	73	**51**	107	724	1853	**58**	106	285	948	**488**	2677	2597	tl
rat195	**86**	672	629	682	**192**	268	1277	tl	**1973**	tl	tl	tl	**1385**	tl	tl	tl
rat99	**27**	32	167	189	**48**	248	553	tl	**59**	830	1105	tl	719	**353**	2341	tl
rd100	**19**	82	134	490	**37**	141	568	1359	**70**	3234	568	tl	**157**	2340	1189	tl
rd400	**1305**	tl	tl	tl	**2417**	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl	tl
si175	**150**	254	413	902	**509**	tl	3519	tl	**810**	tl	tl	tl	**880**	tl	tl	tl
st70	38	**29**	309	102	**59**	68	343	1082	**53**	89	552	919	**83**	909	547	3099
swiss42	**6**	**6**	50	32	**6**	20	50	44	**16**	32	75	65	**8**	151	64	550
ts225	241	**141**	1025	777	**710**	1386	tl	tl	**1042**	tl	tl	tl	**2365**	tl	tl	tl
tsp225	**316**	795	1553	827	**1839**	tl	tl	tl	**1851**	tl	tl	tl	tl	tl	tl	tl
u159	**50**	130	613	612	**359**	1245	tl	tl	**422**	3543	2122	tl	**403**	1707	tl	tl
ulysses16	**0**	**0**	1	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
ulysses22	**0**	**0**	2	1	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

tl: time limit reached without proving optimality. Bold: shortest time per instance and number of trips.

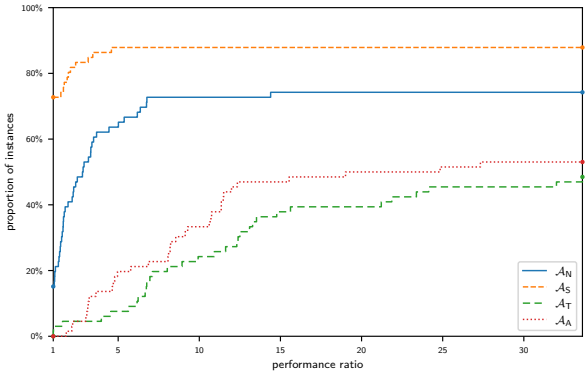
Table A.4: Achieved gaps (as percentages) for each instance of I_2 and approach.

Instances	10 trips				20 trips				30 trips				40 trips			
	$\mathcal{A}_N$	$\mathcal{A}_S$	$\mathcal{A}_T$	$\mathcal{A}_A$	$\mathcal{A}_N$	$\mathcal{A}_S$	$\mathcal{A}_T$	$\mathcal{A}_A$	$\mathcal{A}_N$	$\mathcal{A}_S$	$\mathcal{A}_T$	$\mathcal{A}_A$	$\mathcal{A}_N$	$\mathcal{A}_S$	$\mathcal{A}_T$	$\mathcal{A}_A$
a280	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	1.8	**0.0**	3.4	**0.0**	3.5	4.6	5.2	**0.0**	3.1	3.3	9.1
att48	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
bayg29	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
bays29	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
berlin52	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
bier127	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	2.5	1.1	6.8	**0.0**	**0.0**	0.2	0.8
brazil58	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
brg180	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	1.7	**0.0**	**0.0**	**0.0**	6.0	**0.0**	**0.0**	0.6	5.5
burma14	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
ch130	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	4.4	**0.0**	**0.0**	**0.0**	4.9	**0.0**	1.5	**0.0**	4.1
ch150	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	5.2	**0.0**	**0.0**	0.6	6.4	**0.0**	3.8	**0.0**	5.7
d198	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	3.7	0.5	3.4	**0.0**	2.2	3.4	3.8	**0.0**	4.1	4.2	4.4
d493	**0.5**	0.6	1.6	1.6	1.3	**1.1**	2.8	–	2.6	**1.5**	3.4	–	2.8	**1.7**	3.1	–
dantzig42	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
eil101	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	6.5	**0.0**	**0.0**	**0.0**	3.1	**0.0**	**0.0**	0.3	6.6
eil51	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
eil76	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	3.2	**0.0**	**0.0**	**0.0**	6.3
fl417	**0.0**	1.3	0.3	**0.0**	**0.0**	1.4	1.9	2.5	**0.1**	2.4	5.7	2.9	**0.1**	4.0	6.5	4.9
fri26	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
gil262	**0.0**	2.0	**0.0**	1.7	**0.0**	2.2	1.8	5.3	**0.2**	2.6	3.8	4.3	**0.0**	3.5	4.2	7.0
gr120	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	2.8	**0.0**	**0.0**	0.5	5.0
gr137	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	3.4	**0.0**	**0.0**	3.1	3.1
gr17	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
gr202	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	3.2	1.9	**0.0**	1.3	4.1	3.0	**0.4**	1.8	5.3	2.8
gr21	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
gr229	**0.0**	**0.0**	0.7	2.8	**0.0**	2.3	2.5	5.0	**0.0**	1.2	1.1	1.5	**0.0**	2.6	2.2	4.7
gr24	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
gr431	**0.6**	0.8	1.9	1.9	**1.1**	1.7	2.4	2.7	3.6	**0.6**	4.4	–	3.3	**2.6**	5.7	–
gr48	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
gr96	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	3.1	**0.0**	**0.0**	**0.0**	3.9	**0.0**	**0.0**	**0.0**	5.7
hk48	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
kroA100	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	3.0	**0.0**	**0.0**	**0.0**	4.8
kroA150	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	4.0	**0.0**	**0.0**	**0.0**	4.3	**0.0**	4.4	**0.0**	6.0
kroA200	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	2.3	**0.0**	2.2	2.9	7.2	**0.0**	3.7	1.0	9.7
kroB100	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	5.7	**0.0**	**0.0**	**0.0**	5.6
kroB150	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	0.1	3.6	**0.0**	2.2	**0.0**	5.2	**0.0**	2.1	0.2	5.5
kroB200	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	2.1	**0.0**	3.9	**0.0**	1.5	0.1	4.0	**0.0**	3.0	0.3	5.7
kroC100	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	3.2	**0.0**	**0.0**	**0.0**	6.1
kroD100	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	3.8	**0.0**	**0.0**	**0.0**	3.7	**0.0**	2.9	**0.0**	5.2
kroE100	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	2.5	**0.0**	**0.0**	**0.0**	3.7	**0.0**	2.1	**0.0**	5.9
lin105	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	1.6	**0.0**	**0.0**	**0.0**	4.2	**0.0**	**0.0**	0.1	3.8
lin318	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	3.0	0.5	5.7	**0.0**	4.0	2.5	8.3	**0.0**	3.3	3.9	7.3
linhp318	**0.0**	**0.0**	**0.0**	1.2	**0.4**	2.6	1.2	4.5	**0.1**	4.5	5.9	8.3	**0.1**	3.2	2.8	9.5
pcb442	**0.0**	1.0	**0.0**	1.3	**0.0**	3.0	2.8	3.4	**0.5**	3.2	4.3	5.8	**3.2**	3.5	5.8	–
pr107	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
pr124	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	4.0	**0.0**	3.7	**0.0**	6.7	**0.0**	4.4	**0.0**	8.3
pr136	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	4.8	**0.0**	**0.0**	**0.0**	5.0
pr144	**0.0**	**0.0**	**0.0**	0.8	**0.0**	**0.0**	**0.0**	5.2	**0.0**	**0.0**	**0.0**	5.8	**0.0**	3.3	**0.0**	7.5
pr152	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	1.6	**0.0**	2.6	5.8	6.1	**0.2**	5.3	0.8	10.8
pr226	**0.0**	1.2	**0.0**	**0.0**	**0.0**	4.1	2.4	5.4	**0.0**	3.0	2.3	4.4	**0.0**	3.9	6.1	6.7
pr264	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	1.8	0.2	2.9	**0.2**	1.5	4.0	3.2	**0.0**	3.3	10.6	5.5
pr299	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	1.3	0.7	2.5	**0.0**	1.8	4.6	4.0	**2.4**	**2.4**	8.1	6.3
pr439	**0.0**	1.0	0.2	1.0	**0.3**	2.1	2.9	3.6	**0.9**	2.3	2.7	207.6	**1.8**	3.0	4.1	–
pr76	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	5.2
rat195	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	1.7	**0.0**	2.3	4.3	4.4	**0.0**	2.3	1.5	5.0
rat99	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	2.3	**0.0**	**0.0**	**0.0**	4.2	**0.0**	**0.0**	**0.0**	2.7
rd100	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	6.0	**0.0**	**0.0**	**0.0**	5.7
rd400	**0.0**	1.4	0.1	0.8	**0.0**	2.2	1.2	3.3	**1.3**	2.6	4.5	5.6	**2.6**	3.6	6.0	–
si175	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	3.8	**0.0**	8.4	**0.0**	5.1	1.5	10.9	**0.0**	3.7	3.8	9.6
st70	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
swiss42	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
ts225	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	1.8	**0.0**	2.4	5.1	5.2	**0.0**	2.1	6.1	6.0
tsp225	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	2.5	1.8	4.8	**0.0**	3.0	6.3	4.9	**0.2**	3.1	7.1	6.9
u159	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	0.8	2.8	**0.0**	**0.0**	**0.0**	4.3	**0.0**	**0.0**	**0.0**	3.9
ulysses16	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
ulysses22	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**

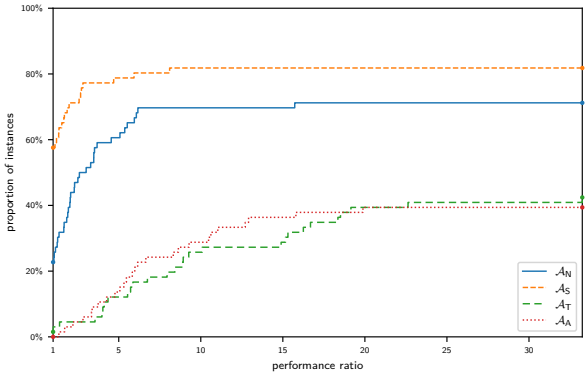
–: no primal bound found within the time limit. Bold: smallest gap per instance and number of trips.



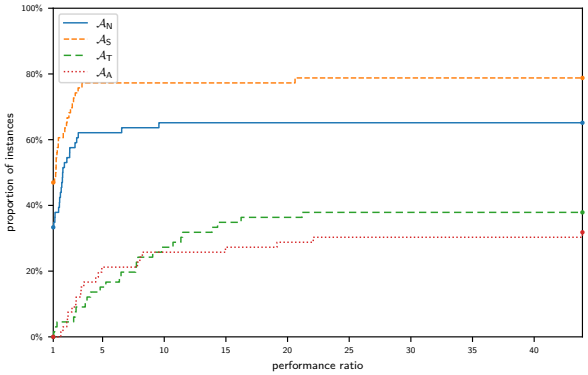
(a) I_1, 10 trips



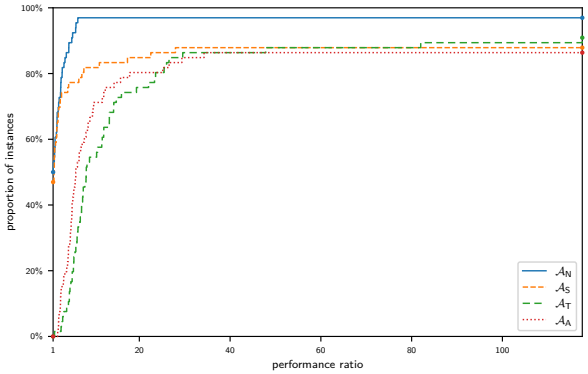
(b) I_1, 20 trips



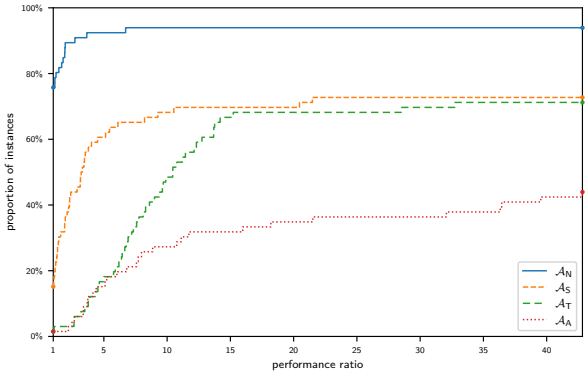
(c) I_1, 30 trips



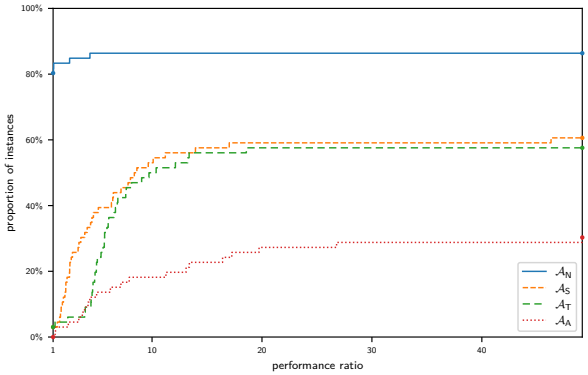
(d) I_1, 40 trips



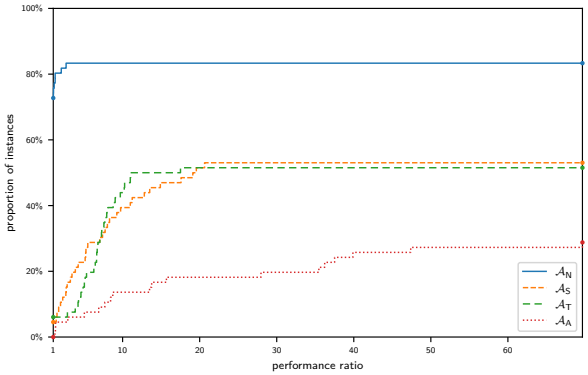
(e) I_2, 10 trips



(f) I_2, 20 trips



(g) I_2, 30 trips



(h) I_2, 40 trips

Figure A.1: Performance profiles with respect to the solving times.