Bifactor Approximation for Location Routing with Vehicle and Facility Capacities

Oscar F. Carrasco Heine[a,*], Antonia Demleitner[b], Jannik Matuschke[a]

[a]*Research Centre for Operations Management, KU Leuven*
[b]*valantic Supply Chain Excellence GmbH*

Abstract

Location Routing is a fundamental planning problem in logistics, in which strategic location decisions on the placement of facilities (depots, distribution centers, warehouses etc.) are taken based on accurate estimates of operational routing costs. We present an approximation algorithm, i.e., an algorithm with proven worst-case guarantees both in terms of running time and solution quality, for the general capacitated version of this problem, in which both vehicles and facilities are capacitated. Before, such approximation algorithms were only known for the special case where facilities are uncapacitated or where their capacities can be extended arbitrarily at linear cost. Previously established lower bounds that are known to approximate the optimal solution value well in the uncapacitated case can be off by an arbitrary factor in the general case. We show that this issue can be overcome by a bifactor approximation algorithm that may slightly exceed facility capacities by an adjustable, arbitrarily small fraction of the vehicle capacity while approximating the optimal cost by a constant factor. In addition to these proven worst-case guarantees, we also assess the practical performance of our algorithm in a comprehensive computational study, showing that the approach allows efficient computation of near-optimal solutions for instance sizes beyond the reach of current state-of-the-art heuristics.
Keywords: Combinatorial optimization, approximation algorithms, location routing

1. Introduction

Location decisions play an important role in transport logistics. The placement of distribution centers, warehouses, and depots shapes the topology of logistics networks and has a major influence on overall logistics costs. When making strategic decisions on the location of such facilities, both the costs for opening and maintaining the facilities as well as the anticipated transportation costs for serving clients from these facilities have to be taken in consideration. In the classic Facility Location problem, one of the most widely

*Corresponding author
Email addresses: `felipe.carrasco@kuleuven.be` (Oscar F. Carrasco Heine), `antonia.demleitner@sce.valantic.com` (Antonia Demleitner), `jannik.matuschke@kuleuven.be` (Jannik Matuschke)

studied models in location analysis, these latter costs are assumed to be linear, with every client receiving a dedicated connection to a facility. While this simplification may be appropriate in some situations (in particular outside logistics), in other application contexts, such as the design of local distribution networks, the transportation cost strongly depends on the actual tours that vehicles take to serve the clients.

It has long been observed that basing location decisions on oversimplified cost models can lead to inferior solutions with costs significantly exceeding those of an optimal solution (Maranzana, 1964; Webb, 1968; Salhi & Rand, 1989; Salhi & Nagy, 1999). This motivates the study of *Location Routing*, an integrated approach that combines Facility Location with a Vehicle Routing problem: Given a finite set of possible facility locations, the task is to determine a subset of these facilities that is to be opened and to plan tours originating from the open facilities in order to supply clients and satisfy their demand. These tours have to respect capacity constraints for vehicles (total demand served by a single tour) and facilities (total demand served by a facility). All of this should be done at minimum total cost, which is the sum of opening costs for the facilities and the routing cost, i.e., the total length of all tours.

By combining Facility Location and Vehicle Routing, each of which is an *NP*-hard problem in its own right, Location Routing constitutes a computationally challenging problem. Numerous heuristic and exponential-time exact approaches have been proposed in literature; see the recent surveys by Prodhon & Prins (2014) and Drexl & Schneider (2015). In this paper, we study *approximation algorithms* for Location Routing, that is, algorithms that come with a proven worst-case guarantee both on their running time and the quality of the produced solution, measured in terms of deviation from the cost of an optimal solution. So far, algorithms with polynomial running time and constant-factor approximation guarantee have only been known for variants of the problem in which facilities are uncapacitated (Ravi & Sinha, 2006; Harks et al., 2013) or for soft-capacitated variants, where facility capacities can be extended arbitrarily at linear cost (Chen & Chen, 2009a,b). The case with hard facility capacities is considerably more challenging, as known lower bounds on the optimal solution value that work well for uncapacitated facilities fail to work in the capacitated case.

Our main contribution is a polynomial-time *bifactor* approximation algorithm that overcomes this issue by computing solutions in which the capacity of any facility can be exceeded by at most a small, adjustable factor while the cost is within a constant factor of the optimal solution. While these theoretical implications constitute the main appeal of our algorithm, we also study its practical performance in computational experiments. We show that the quality of computed solutions is close to those computed by current state-of-the-art heuristics, while its polynomial running time translates into a high degree of scalability, allowing it to tackle instances of sizes beyond the reach of those methods.

Structure of this paper. The remainder of this paper is structured as follows. In the rest of Section 1, we give a formal problem definition, followed by a discussion of related work and our contributions. In

Section 2, we discuss two lower bounds on the optimal solution value. In Section 3 we present and analyze our approximation algorithm, establishing our main result. Section 4 presents heuristic variations of this algorithm, which will be of particular importance for the experiments implemented in Section 5. Finally, Section 6 contains our conclusions.

1.1. Formal Problem Definition

The *Capacitated Location Routing (CLR)* is defined as follows: As input, we are given a set F of possible facility locations, as well as a set C of clients who need to be served. Each facility $w \in F$ has an opening cost $f(w) \geq 0$ and a limited capacity $u(w) \in \mathbb{Z}_+$. Every client $v \in C$ has an individual demand $d(v) \in \mathbb{Z}_+$ that needs to be satisfied. They are served by an unlimited fleet of identical vehicles, each having capacity $\bar{u} \in \mathbb{Z}_+$. There further is a metric[1] distance function $c : V \times V \to \mathbb{R}_+$, where $V := F \cup C$, so that $c(v, w)$ denotes the cost for traveling from v to w.

A *tour* T consists of a facility w_T, a sequence of clients $v_T^1, \dots, v_T^k$, and *service values* $x_T(v)$ for $v \in V$ with $x_T(v) = 0$ for all $v \in C \setminus \{v_T^1, \dots, v_T^k\}$. The *cost* of tour T is $c(T) := c(w_T, v_T^1) + \sum_{i=1}^{k-1} c(v_T^i, v_T^{i+1}) + c(v_T^k, w_T)$, i.e., the total distance covered by a vehicle starting at w_T, visiting the clients in order of the sequence, and returning to w_T.

The goal is to decide on a set of facilities $F' \subseteq F$ to open and a set of tours $\mathcal{T}$ such that

(C1) each tour originates from an open facility: $w_T \in F'$ for every $T \in \mathcal{T}$,

(C2) the demand of every client must be served entirely by the tours visiting it:
$$\textstyle\sum_{T \in \mathcal{T}} x_T(v) = d(v) \text{ for all } v \in C,$$

(C3) the total demand served by any tour is at most the vehicle capacity:
$$\textstyle\sum_{v \in C} x_T(v) \leq \bar{u} \text{ for all } T \in \mathcal{T},$$

(C4) the total demand served from any facility is at most its capacity:
$$\textstyle\sum_{T \in \mathcal{T} : w_T = w} \sum_{v \in C} x_T(v) \leq u(w) \text{ for all } w \in F,$$

minimizing the total cost $\sum_{w \in F'} f(w) + \sum_{T \in \mathcal{T}} c(T)$.

CLR is a combination of two fundamental optimization problems: Capacitated Multi-Depot Vehicle Routing and Capacitated Facility Location, both of which can be seen as special cases of CLR.

Capacitated Multi-Depot Vehicle Routing (CMDVR). The special case where $f \equiv 0$ is known as Capacitated Multi-Depot Vehicle Routing. Note that in this case, it can be assumed that all facilities are opened without loss of generality and thus only the routing aspect is relevant.

[1] In particular, for every $u, v, w \in V$, it holds that $c(u, v) \leq c(u, w) + c(w, v)$.

Capacitated Facility Location (CFL). In the Capacitated Facility Location problem, we are given the same input as in CLR, with the omission of a vehicle capacity. A feasible solution consists of a set of facilities $F' \subseteq F$ to be opened and service values $x(v, w)$ for each client $v \in C$ and $w \in F'$ such that $\sum_{w \in F'} x(v, w) = d(v)$ for each $v \in C$ and $\sum_{v \in C} x(v, w) \leq u(w)$ for all $w \in F$. The cost of such a solution is $\sum_{w \in F'} f(w) + \sum_{v \in C} c(v, w)x(v, w)$. This problem is equivalent to an instance of CLR in which $\bar{u} = 1$ and c is scaled down by a factor of $\frac{1}{2}$. Indeed, by a standard argument using the integrality of the bipartite matching polytope, it is easy to check that for this special case of CLR, there always exists an optimal solution in which every tour corresponds to a round trip visiting only a single client, serving a single unit of demand. Conversely, because client demands and facility capacities are integer, there exists an optimal solution to the CFL instance in which all $x(v, w)$ are integer. Interpreting $x(v, w)$ as the number of dedicated round-trips from facility w to client v establishes a one-to-one correspondence among optimal solutions of these special types for the CFL and CLR instance, respectively, of same cost.

In many application contexts, there are additional desirable properties for location routing solutions that we did not include as hard constraints in our definition of CLR. Two noteworthy examples are the single-sourcing and single-tour properties.

Single-sourcing/single-tour property. We say that a solution $(F', \mathcal{T})$ fulfills the *single-sourcing property* if for every client $v \in C$ there is a facility w_v such that $w_T = w_v$ for all $T \in \mathcal{T}$ with $x_T(v) > 0$. Furthermore, $(F', \mathcal{T})$ fulfills the *single-tour property*, if for every client $v \in C$ there is a unique tour $T \in \mathcal{T}$ with $x_T(v) = d(v)$.

1.2. Previous Results on Approximating Location Routing and Related Problems

We give a brief review of literature on Facility Location, Vehicle Routing, and Location Routing, with a focus on approximation algorithms. An *α-approximation algorithm* for an optimization problem is an algorithm that, given an instance of the problem, computes in polynomial time in the size of the input a solution whose cost is at most α times the cost of an optimal solution to that instance; in this context, $\alpha \geq 1$ is called the *approximation factor*.

The study of approximation algorithms is motivated by the fact that many fundamental optimization problems are NP-hard, leaving little hope for solution methods that are both efficient and exact. Besides their potential to serve as practically usable heuristics, with the additional benefit of an a priori guarantee on the quality of the produced solutions in the worst case, the study of approximation algorithms also yields insights in the inherent complexities of the corresponding optimization problems and the quality of lower bounds on the optimal solution value. For an introduction to the topic see the textbook by Williamson & Shmoys (2011).

Facility Location. Facility Location problems have been a major focus in approximation algorithms over the last decades. The approximability of the Uncapacitated Facility Location problem (UFL), in which there is no upper bound on the demand served by any individual facility, is very well understood in particular. A large variety of techniques have been shown to be successful in obtaining constant-factor approximations for this setting, including greedy algorithms (Jain et al., 2003), LP-based methods (Byrka & Aardal, 2010), and local search (Korupolu et al., 2000), culminating in the currently best known approximation ratio of 1.488 (Li, 2013).

The capacitated version turns out to be much more challenging. For many years, approximation guarantees could only be proven for a local search with an add/swap/remove neighborhood (Korupolu et al., 2000), yielding approximation ratios of 3 for the case of uniform capacities (Aggarwal et al., 2013) and 5 for arbitrary capacities (Bansal et al., 2012). The difficulty in approximating CFL can be in part be attributed to the fact that the natural LP relaxation for the problem turns out to yield only very weak lower bounds on the optimal solution value in the capacitated setting, with an unbounded gap between optimal integer and fractional solutions. Abrams et al. (2002) showed that the integrality gap can be bounded by a constant when allowing a constant blow-up in facility capacities, providing a bifactor approximation of the problem. Only recently, An et al. (2017) derived an LP-based constant-factor approximation (without exceeding facility capacities) for CFL, using a considerably more involved LP relaxation. Moreover, even in the case of uniform capacities, it is NP-hard to decide if a given instance of CFL admits a feasible solution fulfilling the single-sourcing property, a consequence of a straightforward reduction from the Bin Packing problem (Levi et al., 2012) that also carries over to CLR.

Vehicle Routing. Due to their wide range of applications, the Vehicle Routing problems have attracted considerable attention in operations research; see the textbooks by Toth & Vigo (2003) and Golden et al. (2008) for an overview. The fundamental *Traveling Salesperson problem* (TSP), corresponding to the case of a single facility (called *depot* in most of vehicle routing literature) with uncapacitated vehicles, serves as a benchmark both in the theoretical study and the practical design of routing algorithms. In terms of approximation algorithms, the threshold set by the simple and elegant $3/2$-approximation by Christofides (1976) for TSP has only recently been broken by a considerably more intricate design (Karlin et al., 2021). For the case of multiple facilities and capacitated vehicles, Li & Simchi-Levi (1990) devised a tour-partitioning technique that can be used to turn any ρ-approximation for TSP into a $(2+\rho)$-approximation. To the best of our knowledge, no approximation results are known for Multi-Depot Vehicle Routing with capacitated depots.

Location Routing. Ravi & Sinha (2006) paved the way to approximating Location Routing problems with capacitated vehicles by studying the closely related *Capacitated-Cable Facility Location* (CCFL) problem. This problem corresponds to CLR without facility capacities, except that clients are not served by tours but are connected to open facilities via trees of capacitated cables. Ravi & Sinha (2006) showed that two lower bounds on the optimal cost in a given CCFL instance can be derived by computing solutions to appropriately defined instances of UFL and the Steiner Tree problem, respectively. Using this insight, they derived a $(\rho_{\mathrm{UFL}} + \rho_{\mathrm{ST}})$-approximation for CCFL, where ρ_{UFL} and ρ_{ST} denote approximation factors of algorithms for UFL and Steiner Tree, respectively. Based on their framework, Harks et al. (2013) showed how to obtain a 4.32-approximation for Location Routing with capacitated vehicles but uncapacitated facilities (ULR). Our algorithm, discussed below, combines this framework with an LP-rounding scheme to balance the load at individual facilities to obtain a bifactor approximation for CLR.

Ravi & Sinha's (2006) article concludes with the question whether it is possible to obtain constant-factor approximation results for variants of CCFL with additional constraints, mentioning facility capacities as a notable example of practical importance. Chen & Chen (2009a,b) give a partial answer to this question by studying two *soft-capacitated* variants of CCFL and Location Routing, respectively: The *Soft-capacitated Facility Location and Cable Installation* (SC-FLCI) problem corresponds to CCFL, in which arbitrarily many copies of each facility can be opened, each copy costing $f(w)$ and being able to serve a demand of $u(w)$. The *Access Network Design* (AND) problem, motivated by a problem in telecommunication network design, corresponds to location routing in which both vehicles and facilities are uncapacitated, but in which an additional cost $a(w)$ is incurred for each unit of demand served by each facility $w \in F$, representing a linear cost for installing sufficient capacity at each facility. Note that in both problems, no hard upper bound on the total amount of demand served by any facility exists—hence the name "soft-capacitated". Chen & Chen provide a 19.84-approximation for SC-FLCI (2009a) and a 12-approximation for AND (2009b), both based on a primal-dual approach.

1.3. Our Contribution

In this paper, we study approximation algorithms for CLR with uniform vehicle capacities and arbitrary facility capacities. We start by pointing out that, contrary to the case without facility capacities, the natural adaptation of the combined tree/facility-location lower bound by Ravi & Sinha (2006) to the case with facility capacities can be arbitrarily far off from the value of an optimal solution.

Motivated by this observation and the fact that computing feasible solutions fulfilling the single-sourcing property is *NP*-hard, we turn our attention to bifactor approximation: Our main result is an algorithm that given a Location Routing instance and a parameter $\varepsilon \in (0, 1]$, computes in polynomial time a solution in which every facility w receives a load of at most $u(w) + \varepsilon\bar{u}$, where $u(w)$ is the capacity of the facility and $\bar{u}$ is the uniform vehicle capacity. The cost of this solution is bounded by $4 + {}^{2\alpha}/_{\varepsilon}$ times that of an optimal solution to the original instance, where α is the approximation guarantee of an algorithm used to compute the CFL lower bound.[2] For the special case of Capacitated Multi-Depot Vehicle Routing, we can further

[2]The current best known approximation factors for CFL are $\alpha = 5$ for the general case (Bansal et al., 2012) and $\alpha = 3$ for the case of uniform facility capacities (Aggarwal et al., 2013), respectively.

assume $\alpha = 1$, obtaining a factor of $4 + 2/\varepsilon$. The solutions produced by our algorithms furthermore fulfill the single-sourcing and the stronger single-tour property (i.e., every client is served entirely by a single visit of a vehicle), as long as all client demands are bounded by $\varepsilon\bar{u}$.

Our algorithm uses an adaptation of the framework by Ravi & Sinha (2006) to cluster the clients into groups. It then assigns these groups to open facilities via a rounding scheme for a linear program that balances the load at individual facilities. The approximation guarantee is proven using the aforementioned tree/facility-location lower bounds. As an interesting consequence, solutions with cost close to these lower bounds exist and can be computed when relaxing facility capacities by an arbitrarily small factor.

We remark that, in many application contexts (such as regional warehouses in e-commerce or wholesaling), typical facility capacities are at least an order of magnitude larger than vehicle capacities, and so the violation of facility capacities caused by our algorithm is relatively small even when ε is set to 1. It is also important to point out that, differently from the soft-capacitated variants of the problem discussed earlier, the total demand served by any facility $w \in F$ is strictly limited by the a priori upper bound $u(w) + \varepsilon\bar{u}$. This makes our results applicable to settings in which small extensions of facility capacities are admissible but larger extensions (which are assumed to be possible in the soft-capacitated setting) are impossible, e.g., due to physical limitations.

Complementing our theoretical results, we devise several heuristic modifications to the algorithm that improve its practical performance. In particular, by replacing the aforementioned load-balancing linear program by an integer program of moderate size, we show how our framework can be used to obtain solutions that strictly respect the original facility capacities while still achieving a comparably low cost.

Finally, we analyze the empirical performance of our algorithm with and without heuristic improvements in an extensive computational study on a set of widely used benchmark instances from literature as well as additional randomly generated instances. The algorithm exhibits consistent performance across all instance sizes, indicating a high degree of scalability. While it does not outdo previous exact and heuristic approaches without polynomial run time or worst-case approximation guarantees in terms of average solution cost, it stays reasonably close to the best known solutions computed by such methods (7.05% on average), needing only a fraction of the computation time used by those methods. It can thus tackle instances of size beyond the reach of existing algorithms (the largest instances in our experiments contain 10000 clients, whereas the largest instances from literature contain 600 clients). For smaller instances, it can moreover serve as a construction heuristic that very quickly generates solutions of adequate quality, which can be further improved, e.g., in a local search or branch-and-bound approach. The underlying algorithmic idea can also serve as a conceptual basis for stronger heuristics, as we discuss in the conclusion of this paper.

2. Tree and Facility-Location Lower Bounds

In the following we discuss two combinatorial lower bounds introduced by Ravi & Sinha (2006) for the capacitated cable facility-location problem in the slightly adapted form used by Harks et al. (2013) for Location Routing with capacitated vehicles and uncapacitated facilities. Both bounds are straightforward to adapt to the case with facility capacities. Throughout the rest of the paper, we will let OPT denote the cost of an optimal solution for a CLR instance (the instance in question will always be clear from the context).

2.1. Spanning Tree Lower Bound

The first bound is based on a minimum spanning tree—it ignores both vehicle and facility capacities and can be applied to CLR without adaptation; see Lemma 2 by Harks et al. (2013) for a proof.

Lemma 1. *For a given CLR instance, consider the graph* $G = (V \cup \{r\}, E)$ *with* $E := \{\{r,w\} : w \in F\} \cup \{\{v,w\} : v \in C, w \in F\} \cup \{\{v,v'\} : v, v' \in C, v \neq w\}$. *Define costs* $c'(r,w) = 0$, $c'(v,w) = c(v,w) + \frac{1}{2} f(w)$ *for all* $v \in C$, $w \in F$, *and* $c'(v,v') = c(v,v')$ *for all other* $\{v,v'\} \in E$. *Let* T' *be a minimum spanning tree in* G *with respect to weights* c' *and let* $L' := \sum_{e \in T'} c'(e)$. *Then,* $L' \leq \mathrm{OPT}$.

2.2. Capacitated Facility Location Lower Bound

The second lower bound is based on facility location. For ULR the bound uses an instance of Uncapacitated Facility Location. It is straightforward to see that by using a corresponding instance of Capacitated Facility Location instead, one obtains a lower bound for CLR.

Lemma 2. *For a given CLR instance, consider the following CFL instance: clients, facilities, demands, opening costs, and facility capacities remain the same as in the CLR instance, but the distances are set to* $\tilde{c} := {}^{2c}/_{\bar{u}}$. *Let* $(\tilde{F}, \tilde{x})$ *be an optimal solution to this CFL instance and let* $\tilde{L} := \sum_{w \in \tilde{F}} \big(f(w) + \sum_{v \in C} \tilde{c}(v,w) \tilde{x}(v,w) \big)$. *Then* $\tilde{L} \leq \mathrm{OPT}$.

Proof. Let $(F', \mathcal{T})$ be an optimal solution to the CLR instance. For $v \in C$ and $w \in F'$ define $x'(v,w) = \sum_{T \in \mathcal{T} : w_T = w} x_T(v)$. Note that (F', x') is a feasible solution to the CFL instance. Moreover,

$$\begin{aligned}
\sum_{v \in C} \sum_{w \in F'} \tilde{c}(v,w) x'(v,w) &= \sum_{v \in C} \sum_{w \in F'} \sum_{T \in \mathcal{T} : w_T = w} \frac{2}{\bar{u}} c(v,w) x_T(v) \\
&= \sum_{T \in \mathcal{T}} \sum_{v \in C} 2c(v, w_T) \frac{x_T(v)}{\bar{u}} \\
&\leq \sum_{T \in \mathcal{T}} c(T) \sum_{v \in C} \frac{x_T(v)}{\bar{u}} \leq \sum_{T \in \mathcal{T}} c(T),
\end{aligned}$$

where the penultimate inequality follows from the fact that any tour T containing a client v can be decomposed into a w_T-v-path and a v-w_T-path, each of which has length at most $c(v, w_T)$ by triangle inequality. Hence, the total cost of the CFL solution (F', x') is at most the cost of the CLR solution $(F', \mathcal{T})$. □

2.3. Approximation Gap for Lower Bounds in the Capacitated Setting

In the setting where facilities are uncapacitated, the approximation result of Harks et al. (2013) implies that, for any instance, the maximum of the two lower bounds is within a constant factor of the value of an optimal solution. The following lemma reveals that this no longer holds in the capacitated setting.

Lemma 3. *For any $n \in \mathbb{N}$, there exists an instance of CLR with n clients and two facilities such that* $\mathrm{OPT} \geq (n-1)\max\{L', \tilde{L}\}$, *where L' and $\tilde{L}$ are the values of the lower bounds described in Lemmas 1 and 2, respectively.*

Proof. Consider a CLR instance with $n = |C|$ clients with unit demands $d \equiv 1$. There are two possible facility locations: $F = \{w_1, w_2\}$ with $f(w_1) = f(w_2) = 0$ and $u(w_1) = u(w_2) = \bar{u} = n - 1$. All clients are located at the same position as w_1, which has a distance of 1 to w_2. That is, $d(v, w_1) = 0$ and $d(v, w_2) = d(w_1, w_2) = 1$ for all $v \in C$.

Observe that for this instance the value of the MST lower bound as described in Lemma 1 is 0 because facility opening cost are 0 and every client is located at distance 0 to a facility. Observe further that the value of the CFL lower bound as described in Lemma 2 is $2/\bar{u} = 2/(n-1)$, as all but one client can be served by facility w_1 at cost 0. Note, however, that any feasible solution to the CLR instance needs to contain a tour connecting at least one client to facility w_2 and that the cost of such a tour is at least 2. Thus the cost of an optimal solution for the constructed instance is at least $n - 1$ times the larger of the two lower bounds. □

In the next section, however, we will show that a solution within a constant factor of the two lower bounds can still be obtained when slightly relaxing the facility capacities. The following corollary is an implication of the analysis presented in Section 3.

Corollary 1. *For any $\varepsilon \in (0, 1]$ and any instance of CLR, there exists a solution in which constraints (C1)-(C3) are fulfilled, the load at any facility $w \in F$ is no more than $u(w) + \varepsilon\bar{u}$, and the total cost is no more than $4L' + \frac{2}{\varepsilon}\tilde{L}$, where L' and $\tilde{L}$ are the values of the lower bounds described in Lemmas 1 and 2, respectively.*

3. Approximation Algorithm

In the following, we present a bifactor approximation algorithm for CLR. The algorithm and the accompanying analysis prove the following result:

Theorem 1. *There is an algorithm that, given $\varepsilon \in (0, 1]$, computes in polynomial time a solution to a given instance of CLR such that constraints (C1)-(C3) are fulfilled, the load at any facility $w \in F$ is no more than $u(w) + \varepsilon\bar{u}$, and the total cost is no more than $(4 + \frac{2\alpha}{\varepsilon})\,\mathrm{OPT}$, where $\alpha \geq 1$ is the approximation factor of an algorithm for CFL.*

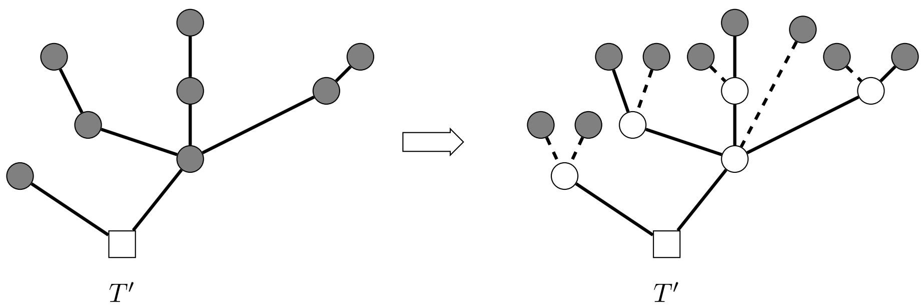

Figure 1: Preprocessing of the tree. If a client v does not occur as a leaf, or if $d(v) > \bar{\bar{u}}$, it is replaced by a dummy node whose demand is equal to 0. For each such client, $\ell := \lceil d(v)/\bar{\bar{u}} \rceil$ nodes are added at distance 0 from the original, each with a demand of $d(v)/\ell$. In the figure, hollow circles represent dummy nodes, filled circles represent clients with positive demand, and the square represents a facility. In the modified tree, each of these new nodes is connected to the dummy node corresponding to v (the corresponding edges are depicted as dashed lines).

Throughout this section let $\varepsilon \in (0, 1]$ be the parameter given in the input of the algorithm and define $\bar{\bar{u}} := \varepsilon \bar{u}$.

Overview. The algorithm consists of three steps. In the first step, a minimum spanning tree for the instance described in Lemma 1 is partitioned into clusters such that each cluster contains clients with a total demand of at most $\bar{\bar{u}}$ and every cluster with demand less than $\bar{\bar{u}}/2$ contains a facility. The clustering technique is based on a procedure for relieving overloaded subtrees by Alpert et al. (2003) and Ravi & Sinha (2006). In the second step, the clusters are assigned to open facilities via a rounding procedure for an assignment LP. Using the fact that most clusters have large aggregated demand, it is shown that solutions to the CFL instance described in Lemma 2 induce feasible solutions of bounded cost for this LP. The rounding procedure might allocate one additional cluster per facility, thus resulting in an additional demand of at most $\bar{\bar{u}}$ at any open facility. Finally, each cluster is converted into a tour by adding an edge to its assigned facility and using the classic doubling-and-shortcutting technique for TSP.

3.1. Step 1: Clustering

The first step of the algorithm consists of a clustering procedure that partitions the set of clients into appropriate clusters and associated trees connecting the nodes within each cluster.

Preprocessing of the tree. The clustering procedure starts with a minimum spanning tree T' for the instance (G, c') described in Lemma 1. Note that we can assume T' to contain the edges $\{r, w\}$ for all $w \in F$ without loss of generality (as those edges have cost 0). We further modify the tree such that clients only occur at leafs and all clients have demand at most $\bar{\bar{u}}$. This can be achieved by splitting each client v in $\lceil d(v)/\bar{\bar{u}} \rceil$ nodes, distributing the demand uniformly among them, and by introducing an additional *dummy node* at the location of each client that takes the role of the original, possibly internal, node of the client in the

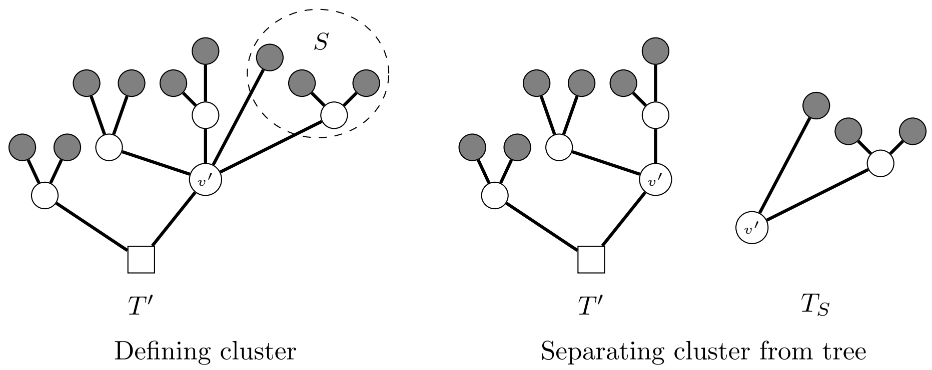

Figure 2: A step of the clustering procedure. As in Fig. 1, hollow circles represent dummy nodes (with no demand) and filled circles represent clients (with positive demand). The procedure finds a node v' with $d_{T'}(v') > \bar{\bar{u}}$ but $d_{T'}(v) \leq \bar{\bar{u}}$ for all $v \in K_{T'}(v')$. It greedily constructs a set S such that $d(S) \geq \bar{\bar{u}}/2$. The set S is added to $\mathcal{S}$, and the corresponding tree T_S is removed from T'.

tree. The nodes resulting from splitting the client, which have positive demand, are attached to this dummy node as leafs. See Fig. 1 for an illustration. Dummy nodes will be removed when constructing the tours at the end of the algorithm, so they do not occur in the final solution. Note that neither of these operations increases the cost of the tree and that clients with demand at most $\bar{\bar{u}}$ remain represented by a single client.

Notation. For $v, w \in V(T')$ we say that w is a *descendant* of v in T' if v is on the unique r-w-path in T'. We say that w is a *child* of v in T', if v directly precedes w on the unique r-w-path in T'. We use the notation $T'[v]$ to denote the subtree of T' containing v and all its descendants. We denote the set of children of v in T' by $K_{T'}(v)$. We further let $d_{T'}(v) := \sum_{v' \in V(T'[v]) \cap C} d(v')$ denote the total demand in the subtree of v.

Clustering. The clustering procedure creates a family $\mathcal{S}$ (initially empty) of client sets, together with corresponding trees T_S for each $S \in \mathcal{S}$. To this end, the procedure repeatedly identifies a node $v' \in V(T') \setminus \{r\}$ such that $d_{T'}(v') > \bar{\bar{u}}$ but $d_{T'}(v) \leq \bar{\bar{u}}$ for all $v \in K_{T'}(v')$. Then a subset $L \subseteq K_{T'}(v')$ of the children of v' is greedily selected so that $\bar{\bar{u}}/2 \leq \sum_{v \in L} d_{T'}(v) \leq \bar{\bar{u}}$. More specifically, starting with $L = \emptyset$, children of v' are sequentially added in non-increasing order of $d_{T'}(v)$ until adding the next child would violate the upper bound. Note that L is non-empty, because we assume $d(v) \leq \bar{\bar{u}}$ for all $v \in C$, and because the first child that is not added has at most the demand of the last child that was added, the total demand in L must be at least $\bar{\bar{u}}/2$. The client set $S := \bigcup_{v \in L} V(T'[v]) \cap C$ is added as a new element of $\mathcal{S}$, associated with the corresponding tree $T_S := \bigcup_{v \in L} T'[v] \cup \{v, v'\}$ consisting of the subtrees induced by the nodes in L and the edges connecting these subtrees to v'. The edges of T_S and its nodes except for v' are removed from T' (see Fig. 2). Once $d_{T'}(v') \leq \bar{\bar{u}}$ for all nodes $v' \in V(T')$ in the remaining tree, the procedure identifies a set of facilities F_1 containing all facilities w for which $V(T'[w]) \cap C \neq \emptyset$. For each $w \in F_1$, the set $V(T'[w]) \cap C$ is added as an additional cluster to $\mathcal{S}$, together with the corresponding tree $T'[w]$. The algorithm returns the

Algorithm 1: Clustering

Let T' be a minimum spanning tree in the graph G' with weights c'.

Initialize $\mathcal{S} := \emptyset$.

while there is $v \in V(T') \setminus \{r\}$ with $d_{T'}(v) > \bar{\bar{u}}$ **do**

  Let $v' \in V(T') \setminus \{r\}$ such that $d_{T'}(v') > \bar{\bar{u}}$ but $d_{T'}(v) \leq \bar{\bar{u}}$ for all $v \in K_{T'}(v')$.

  Let $L \subseteq K_{T'}(v')$ be such that $\bar{\bar{u}}/2 \leq \sum_{v \in L} d_{T'}(v) \leq \bar{\bar{u}}$.

  Add the set $S := \bigcup_{v \in L} V(T'[v]) \cap C$ to $\mathcal{S}$ and let $T_S := \bigcup_{v \in L} T'[v] \cup \{v, v'\}$.

  Remove all edges of T_S and all nodes of $V(T_S) \setminus \{v'\}$ from T'.

Let $F_1 := \{w \in F : V(T'[w]) \cap C \neq \emptyset\}$.

for $w \in F_1$ **do**

  Add the set $S := V(T'[w]) \cap C$ to $\mathcal{S}$ and let $T_S := T[w]$.

return $(\mathcal{S}, \{T_S : S \in \mathcal{S}\}, F_1)$

clustering $\mathcal{S}$, the corresponding trees T_S for $S \in \mathcal{S}$ and the set of facilities F_1. A pseudo-code listing of the clustering procedure is given in Algorithm 1 and an illustration is given in Fig. 2.

Note that while it is possible for the node sets $V(T_S)$ and $V(T_{S'})$ for $S, S' \in \mathcal{S}$ with $S \neq S'$ to overlap at a dummy node or a facility, the client sets S and S' are disjoint (recall that clients only occur at leafs of the tree T') and the same is true for the edge sets of the trees T_S and $T_{S'}$. Hence $\mathcal{S}$ indeed comprises a partition of the clients and the trees T_S partition the original tree T' (with the exception of the edges $\{r, w\}$ for $w \in F$ which do not occur in any tree). These observations and additional properties of the clustering following immediately from its construction in the algorithm are summarized below.

Lemma 4. *Algorithm 1 computes in polynomial time a partition $\mathcal{S}$ of the clients together with a tree T_S for each $S \in \mathcal{S}$ such that:*

- *$T_S \subseteq T'$ and $E(T_S) \cap E(T_{S'}) = \emptyset$ for all $S, S' \in \mathcal{S}$ with $S \neq S'$,*
- *$S \subseteq V(T_S)$ for all $S \in \mathcal{S}$,*
- *$\sum_{v \in S} d(v) \leq \bar{\bar{u}}$ for all $S \in \mathcal{S}$.*

Moreover, defining $\mathcal{S}' := \{S \in \mathcal{S} : \sum_{v \in S \cap C} d(v) < \bar{\bar{u}}/2\}$, there is a unique facility $w_S \in V(T_S)$ for every $S \in \mathcal{S}'$, and $w_S \neq w_{S'}$ for $S \neq S'$. Defining $F_1 := \{w \in F : V(T'[w]) \cap C \neq \emptyset\}$, every facility $w \in F_1$ is incident to at least one edge in $T' \setminus \{r, w\}$.

3.2. Step 2: Assignment

The second step of the algorithm constructs an assignment of the clusters constructed in the first step to a set of open facilities.

The assignment LP. To this end, the algorithm first computes an approximate solution to the CFL instance described in Lemma 2, using an α-approximation algorithm for CFL. Let us denote the set of facilities opened in this solution by F_2 and the corresponding assignment by $\tilde{x}$. Recall further the clustering $\mathcal{S}$ and the set of facilities F_1 returned by the first step of the algorithm, and define $F' := F_1 \cup F_2$. For $S \in \mathcal{S}$ and $w \in F$, we define $c(S,w) := \min_{v \in V(T_S)} c(v,w)$ and $d(S) := \sum_{v \in S} d(v)$. Consider the following assignment LP that assigns the demand of each cluster produced by the first step of the algorithm to a facility in F':

$$\begin{array}{llcll} \min & \displaystyle\sum_{S\in\mathcal{S}}\sum_{w\in F'} \frac{c(S,w)}{d(S)} x(S,w) & & & \\ \text{s.t.} & \displaystyle\sum_{S\in\mathcal{S}} x(S,w) & \le & u(w) & \forall w \in F' \\ & \displaystyle\sum_{w\in F'} x(S,w) & = & d(S) & \forall S \in \mathcal{S} \\ & x(S,w) & \ge & 0 & \forall S \in \mathcal{S}, w \in F' \end{array} \tag{1}$$

In the proof of the following lemma, we show that the CFL solution induces a feasible solution to (1) of bounded cost.

Lemma 5. *LP (1) has a feasible solution with objective function value at most* $\frac{1}{\varepsilon}\sum_{v\in C}\sum_{w\in F'} \tilde{c}(v,w)\tilde{x}(v,w) + \sum_{S\in\mathcal{S}'} c(T_S)$.

Proof. To prove the lemma, we first construct an instance of a network flow problem in an undirected graph $\bar{G} = (\bar{V}, \bar{E})$ on the node set $\bar{V} := V(T') \setminus \{r\}$ (thus, V' contains all clients and facilities, as well as potential dummy nodes introduced in the preprocessing step). The edge set $\bar{E} = \bar{E}_1 \cup \bar{E}_2$ consists of the edges $\bar{E}_1 := \{\{v,w\} : v \in C, w \in F, \tilde{x}(v,w) > 0\}$ representing the support of the CFL assignment and $\bar{E}_2 := \bigcup_{S\in\mathcal{S}'} T_S$, i.e., the union of all trees T_S corresponding to clusters S with low demand $d(S) < \bar{u}/2$.

For $e = \{v,w\} \in \bar{E}_1$ we set the capacity $u(e) := \tilde{x}(v,w)$ and for $e \in \bar{E}_2$ we set the capacity $u(e) := d(S)$, where $S \in \mathcal{S}'$ is the unique cluster with $e \in T_S$. (Note that because $\bar{G}$ is undirected, flow can use each edge in either direction.) For $w \in F'$ let S_w denote the unique cluster $S \in \mathcal{S}'$ with $w_S = w$, if it exists, and let $S_w = \emptyset$ otherwise. We will consider each facility $w \in F'$ as a source with supply $s(w) := u(w) - d(S_w)$. We will consider any client $v \in \bar{C} := C \setminus \bigcup_{S\in\mathcal{S}'} S$ that is not in a cluster of low demand as a sink with demand $r(v) := d(v)$.

The following claim establishes the existence of a flow in $\bar{G}$ that saturates all these demands while not exceeding any edge capacity or any supply at a facility. To make this formal, let $\mathcal{P}_{vw}$ for any $v, w \in \bar{V}$ denote the set of v-w-paths in the graph $\bar{G}$ and let $\mathcal{P} := \bigcup_{v\in\bar{C}, w\in F'} \mathcal{P}_{wv}$.

Claim 1. *There exists a flow* $y \in \mathbb{R}_+^{\mathcal{P}}$ *such that* $\sum_{P\in\mathcal{P}: e\in P} y(P) \le u(e)$ *for all* $e \in E$, $\sum_{v\in\bar{C}}\sum_{P\in\mathcal{P}_{vw}} y(P) \le s(w)$ *for all* $w \in F'$, *and* $\sum_{w\in F'}\sum_{P\in\mathcal{P}_{vw}} y(P) = r(v)$ *for all* $v \in \bar{C}$.

We postpone the proof of the claim and first show how it implies the lemma. Note that by choice of the capacities u, the cost of the flow y in terms of c is bounded by

$$\sum_{e \in \bar{E}} c(e) \sum_{P \in \mathcal{P}: e \in P} y(P) \leq \sum_{v \in C} \sum_{w \in F'} c(v,w)\tilde{x}(v,w) + \sum_{S \in \mathcal{S}'} \sum_{\{v,w\} \in T_S} c(v,w) d(S). \tag{2}$$

Let $\bar{y}(v,w) := \sum_{P \in \mathcal{P}_{vw}} y(P)$ be the amount of flow that is sent from $w \in F'$ to $v \in \bar{C}$ in the flow y. Consider the vector x defined by $x(S, w_S) := d(S)$ for all $S \in \mathcal{S}'$ and $x(S,w) := \sum_{v \in S} \bar{y}(v,w)$ for all $S \in \mathcal{S} \setminus \mathcal{S}'$ and $w \in F'$, with $x(S,w) = 0$ for all other pairs $S \in \mathcal{S}'$ and $w \neq w_S$. We show that this vector is a feasible solution to (1). The first set of constraints is fulfilled because $\sum_{S \in \mathcal{S}} x(S,w) = d(S_w) + \sum_{v \in \bar{C}} \bar{y}(v,w) \leq d(S_w) + s(w) = u(w)$ for all $w \in F'$. The second set of constraints is fulfilled because $\sum_{w \in F'} x(S,w) = x(S,w_s)$ for all $S \in \mathcal{S}'$ and $\sum_{w \in F'} x(S,w) = \sum_{w \in F'} \sum_{w \in F'} \bar{y}(v,w) = \sum_{v \in S} r(v) = d(S)$ for all $S \in \mathcal{S} \setminus \mathcal{S}'$. Moreover, the cost of this feasible solution can be bounded as follows:

$$\begin{aligned}
\sum_{S \in \mathcal{S}} \sum_{w \in F'} \frac{c(S,w)}{d(S)} x(S,w) &\leq \sum_{S \in \mathcal{S} \setminus \mathcal{S}'} \sum_{w \in F'} \sum_{v \in S} \frac{c(v,w)}{d(S)} \bar{y}(v,w) \\
&\leq \frac{2}{\bar{\bar{u}}} \sum_{S \in \mathcal{S} \setminus \mathcal{S}'} \sum_{w \in F'} \sum_{v \in S} c(v,w) \sum_{P \in \mathcal{P}_{vw}} y(P) \\
&\leq \frac{2}{\bar{\bar{u}}} \sum_{w \in F'} \sum_{v \in C} \sum_{P \in \mathcal{P}_{vw}} \sum_{e \in P} c(e) y(P) \\
&= \frac{2}{\bar{\bar{u}}} \sum_{e \in \bar{E}} \sum_{P \in \mathcal{P}: e \in P} c(e) y(P),
\end{aligned} \tag{3}$$

where the first inequality follows from $c(S, w_S) = 0$ for all $S \in \mathcal{S}'$, the second follows from the fact that $d(S) \geq \bar{\bar{u}}/2$ for all $S \in \mathcal{S} \setminus \mathcal{S}'$, and the third follows from the fact that c is a metric and hence $c(v,w) \leq \sum_{e \in P} c(e)$ for any $P \in \mathcal{P}_{vw}$. Combining (2) and (3) yields

$$\sum_{S \in \mathcal{S}} \sum_{w \in F'} \frac{c(S,w)}{d(S)} x(S,w) \leq \frac{1}{\varepsilon} \sum_{v \in C} \sum_{w \in F'} \tilde{c}(v,w)\tilde{x}(v,w) + \sum_{S \in \mathcal{S}'} \sum_{\{v,w\} \in T_S} c(v,w)$$

using the definition of $\tilde{c}$ and the fact that ${}^{d(S)}/_{\bar{\bar{u}}} \leq {}^{1}/_{2}$ for $S \in \mathcal{S}'$. We thus established that Claim 1 implies Lemma 5. The proof of the former is given below. □

Proof of Claim 1. By the max-flow/min-cut theorem, a flow satisfying the demands r while respecting supply limits s and capacities u exists if and only if

$$\sum_{v \in A \cap \bar{C}} r(v) - \sum_{w \in A \cap F'} s(w) \leq \sum_{e \in \delta(A)} u(e)$$

for any node set $A \subseteq V'$, where $\delta(A)$ is the cut induced by A, i.e., the set of edges $e \in \bar{E}$ with exactly one endpoint in A and the other in $V' \setminus A$. It thus suffices to show that the above inequality holds for any $A \subseteq V'$.

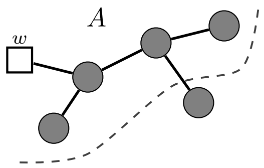

Figure 3: The cut induced by set A in the proof of Claim 1. If $w \in A \cap F_1$ but $S_w \setminus A \neq \emptyset$, then at least one edge of T_{S_w} crosses the cut.

Let $A \subseteq V'$ and note that

$$\begin{aligned}\sum_{v \in A \cap \bar{C}} r(v) - \sum_{w \in A \cap F'} s(w) &= \sum_{v \in A \cap C} d(v) - \sum_{S \in \mathcal{S}'} \sum_{v \in A \cap S} d(v) - \sum_{w \in A \cap F'} (u(w) - d(S_w)) \\ &= \sum_{v \in A \cap C} d(v) - \sum_{w \in A \cap F'} u(w) + \sum_{w \in A \cap F_1} d(S_w \setminus A) \\ &\leq \sum_{v \in A \cap C} \sum_{w \in F' \setminus A} \tilde{x}(v, w) + \sum_{w \in A \cap F_1} d(S_w \setminus A),\end{aligned}$$

where the last inequality follows from the fact that $\tilde{x}$ is a feasible assignment for the CFL instance with open facilities $F_2 \subseteq F'$. Finally, note that $\sum_{v \in A \cap C} \sum_{w \in F' \setminus A} \tilde{x}(v, w) \leq \sum_{e \in \delta(A) \cap \bar{E}_1} u(e)$ and that $\delta(A) \cap \bar{E}_2$ contains at least one edge $e \in T_{S_w}$ for every $w \in A \cap F_1$ for which $S_w \setminus A \neq \emptyset$ (see Fig. 3 for an illustration). Because any such edge $e \in T_{S_w} \subseteq \bar{E}_2$ has capacity $u(e) = d(S_w)$, we obtain $\sum_{w \in A \cap F_1} d(S_w \setminus A) \leq \sum_{e \in \delta(A) \cap \bar{E}_2} u(e)$, proving the claim. ☐

Rounding the assignment. The algorithm computes an optimal extreme point solution x to LP (1). The following rounding procedure transforms x into a new solution x' with $x'(S, w) \in \{0, d(S)\}$ for all $S \in \mathcal{S}$ and $w \in F'$. Starting with $x' := x$, the rounding procedure maintains a helper graph $G_{x'} = (\mathcal{S} \cup F', E_{x'})$ with edge set $E_{x'} = \{\{S, w\} : S \in \mathcal{S}, w \in F', 0 < x'(S, w) < d(S)\}$, which is updated along with the solution. As long as $E_{x'} \neq \emptyset$, it iteratively applies the following modification to x' (which intuitively is a flow augmentation along a path in the helper graph): Let $w, w' \in F'$ be two facilities such that both w and w' are incident to exactly one edge in G_x and such that there is a w-w'-path P in G_x (it is argued that in the proof of Lemma 6 that these always exist while the graph is non-empty). Let $w_0 := w, S_1, w_1, \ldots, w_{k-1}, S_k, w_k := w'$ denote the nodes along path P in order of traversal from w to w' and let $I := \{(S_i, w_{i-1}) : i \in \{1, \ldots, k\}\}$ and $I' := \{(S_i, w_i) : i \in \{1, \ldots, k\}\}$. Without loss of generality (by swapping the roles of w and w' if necessary), we can assume that $\sum_{(S,w) \in I} c(S, w) \leq \sum_{(S,w) \in I'} c(S, w)$. Let $\Delta := \min\{d(S) - x(S, w) : (S, w) \in I\} \cup \{x(S, w) : (S, w) \in I'\}$ and update x by increasing $x(S, w)$ by Δ for all $(S, w) \in I$ and decreasing $x(S, w)$ by Δ for all $(S, w) \in I'$.

The following lemma shows that that this procedure terminates after a linear number of iterations and results in a solution x' whose cost is at most that of x and which allocates at most a demand of $u + \bar{\bar{u}}$ to

Algorithm 2: Assignment

Let x be an optimal extreme point solution to (1). Initialize $x' := x$.

while there is $S \in \mathcal{S}$ and $w \in F'$ with $0 < x'(S,w) < d(S)$ **do**

  Let $E_{x'} := \{\{S,w\} : S \in \mathcal{S}, w \in F', 0 < x'(S,w) < d(S)\}$.

  Let $w, w' \in F'$ such that both w and w' have degree 1 and there exists a w-w'-path $P = (w_0, S_1, w_1, \ldots, S_k, w_k)$ in the graph $G_{x'} = (\mathcal{S} \cup F', E_{x'})$. (note: $w_0 = w$, $w_k = w'$)

  Let $I := \{(S_i, w_{i-1}) : i \in \{1,\ldots,k\}\}$ and $I' := \{(S_i, w_i) : i \in \{1,\ldots,k\}\}$.

  if $\sum_{(S,w)\in I} c(S,w) \leq \sum_{(S,w)\in I'} c(S,w)$ **then**

    Let $\Delta := \min\{d(S) - x'(S,w) : (S,w) \in I\} \cup \{x'(S,w) : (S,w) \in I'\}$.

    Let $x'(S,w) := x'(S,w) + \Delta$ for all $(S,w) \in I$.

    Let $x'(S,w) := x'(S,w) - \Delta$ for all $(S,w) \in I'$.

  else

    Let $\Delta := \min\{d(S) - x'(S,w) : (S,w) \in I'\} \cup \{x'(S,w) : (S,w) \in I\}$.

    Let $x'(S,w) := x'(S,w) + \Delta$ for all $(S,w) \in I'$.

    Let $x'(S,w) := x'(S,w) - \Delta$ for all $(S,w) \in I$.

return x'

each facility.[3]

Lemma 6. *The assignment procedure (Algorithm 2) computes in polynomial time (after at most $|\mathcal{S}|+|F'|-1$ iterations) a vector x' with $x'(S,w) \in \{0, d(S)\}$ for all $S \in \mathcal{S}$ and all $w \in F'$ such that*

- $\sum_{S\in\mathcal{S}} \sum_{w\in F'} \frac{c(S,w)}{d(S)} x'(S,w) \leq \sum_{S\in\mathcal{S}} \sum_{w\in F'} \frac{c(S,w)}{d(S)} x(S,w)$,
- $\sum_{w\in F'} x'(S,w) = d(S)$ *for all* $S \in \mathcal{S}$*, and*
- $\sum_{S\in\mathcal{S}} x'(S,w) \leq \bar{\bar{u}} + \sum_{S\in\mathcal{S}} x(S,w)$ *for all* $w \in F'$*.*

Proof. Note that any modification applied to x' in the algorithm does not increase the cost and keeps $\sum_{w\in F'} x'(S,w)$ invariant for all $S \in \mathcal{S}$ (because the selected path always starts and ends at a facility). Hence if the algorithm terminates, the first and second condition are automatically met.

Initially, because x is an extreme point solution to a transportation problem, the graph $G_{x'}$ is a forest before the start of the first iteration. During the course of the algorithm, once a variable $x'(S,w)$ is set to

[3] We remark that procedures yielding the same guarantees as those in Lemma 6 are implied by numerous rounding procedures for more general linear programs, such as the classic 2-approximation for the generalized assignment problem (Shmoys & Tardos, 1993) or more recent approximation results for single-source unsplittable flow (Morell & Skutella, 2020). However, the procedure outlined here is specifically tailored to the special case resulting from LP (1), yielding a simpler and more efficient algorithm.

either 0 or $d(S)$ its value is never changed again (because the corresponding edge cannot occur in the path P anymore). Hence, in any iteration, the set of edges in $E_{x'}$ is a subset of the edges from the previous iteration, and $G_{x'}$ is a forest throughout the run of the algorithm.

We further observe that no cluster node $S \in \mathcal{S}$ can have degree 1 in $G_{x'}$, as $0 < x'(S, w) < d(S)$ for some w implies that there must be at least one other w' with $0 < x'(S, w) < d(S)$. Hence, any non-singleton connected component of $G_{x'}$ is a tree whose leafs are elements of F'. Hence, as long as $E_{x'}$ is non-empty, there are indeed $w, w' \in F'$ with degree 1 that are connected by a w-w'-path in $G_{x'}$.

Furthermore, note that $\sum_{S \in \mathcal{S}} x'(S, w)$ for some $w \in F'$ can only change while w is a leaf in $G_{x'}$ (because the changes in the variables x' cancel out for w when it occurs in the interior of the path P). Consider the first iteration when $w \in F'$ becomes a leaf (if any such iteration exists) and note that $\sum_{S \in \mathcal{S}} x'(S, w) = \sum_{S \in \mathcal{S}} x(S, w) \leq u(w)$ at the begin of this iteration. Let $S' \in \mathcal{S}$ be the unique cluster such that $\{S', w\} \in E_{x'}$. Because for any $S \neq S'$, the variable $x'(S, w)$ will not be changed anymore and $x'(S', w)$ is never increased beyond $d(S') \leq \bar{\bar{u}}$, we obtain $\sum_{S \in \mathcal{S}} x'(S, w) \leq \sum_{S \in \mathcal{S}} x(S, w) + \bar{\bar{u}}$ throughout the algorithm.

Finally, by choice of Δ, there is at least one variable $x'(S, w)$ in each iteration whose value is set to either 0 or $d(S)$. Hence the number of edges in $E_{x'}$ decreases by at least one in each iteration. Because $E_{x'}$ is a forest, it contains at most $|\mathcal{S}| + |F'| - 1$ edges initially and the algorithm terminates after $|\mathcal{S}| + |F'| - 1$ iterations. Note further that $E_{x'} = \emptyset$ implies $x'(S, w) \in \{0, d(S)\}$ for all $S \in \mathcal{S}$ and $w \in F'$. □

3.3. Step 3: Constructing tours

In the final step of the algorithm, the trees covering each cluster are connected to the facilities as indicated by the rounded assignment x'. For each cluster, the resulting tree is turned into a tour via the classic edge-doubling-and-shortcutting procedure. For every $S \in \mathcal{S}$, let $w_S \in F'$ be the unique facility with $x'(S, w_S) = d(S)$ and let $v \in V(T_S)$ be such that $c(v, w_S) = c(S, w_S)$. Define $\bar{T}_S := T_S \cup \{v, w_S\}$. Note that $\bar{T}_S$ is a connected graph spanning (a superset of) the nodes in $S \cup \{w_S\}$ and hence a tour visiting the clients in S and facility w_S of length at most $2\big(c(T_S) + c(S, w_S)\big)$ can be computed using the classic double tree algorithm: (1) double the edges in $\bar{T}_S$, (2) compute a Eulerian tour on the doubled tree, (3) shortcut the tour by skipping any nodes not in $(S \cap C) \cup \{w_S\}$ and any repeated visits of a node. The algorithm returns the resulting tours together with the set of facilities F' to be opened.

Analysis. In the constructed solution, the total demand on all tours based at facility $w \in F'$ is

$$\sum_{\substack{S \in \mathcal{S} \\ w_S = w}} d(S) = \sum_{S \in \mathcal{S}} x'(S, w) \leq \bar{\bar{u}} + \sum_{S \in \mathcal{S}} x(S, w) \leq u(w) + \varepsilon \bar{u}$$

by Lemma 6. The cost of the tours constructed by the algorithm is bounded by $\sum_{S\in\mathcal{S}} 2(c(T_S) + c(S, w_s))$. Moreover,

$$\sum_{S\in\mathcal{S}} c(S, w_S) = \sum_{w\in F'} \sum_{S\in\mathcal{S}} \frac{c(S,w)}{d(S)} x'(S,w) \le \frac{1}{\varepsilon} \sum_{v\in C} \sum_{w\in F'} \tilde{c}(v,w)\tilde{x}(v,w) + \sum_{S\in\mathcal{S}'} c(T_S)$$

by Lemmas 5 and 6. We conclude that the total cost of the constructed tours is bounded by $4\sum_{S\in\mathcal{S}} c(T_S) + \frac{2}{\varepsilon}\sum_{v\in C}\sum_{w\in F'} \tilde{c}(v,w)\tilde{x}(v,w)$. Finally, note that $4\sum_{S\in\mathcal{S}} c(T_S) + \sum_{w\in F_1} f(w) \le 4c'(T')$, because, by Lemma 4, the trees T_S for $S \in \mathcal{S}$ are edge-disjoint and every facility $w \in F_1$ is incident to at least one edge $e \in T'$ other than $\{r, w\}$. Combining these observations, we can bound the total cost of the produced solution by

$$\begin{aligned}
\sum_{w\in F'} f(w) + \sum_{T\in\mathcal{T}} c(T) &\le \sum_{w\in F_1} f(w) + \sum_{w\in F_2} f(w) + 4\sum_{S\in\mathcal{S}} c(T_S) + \frac{2}{\varepsilon}\sum_{v\in C}\sum_{w\in F'} \tilde{c}(v,w)\tilde{x}(v,w) \\
&\le 4c'(T') + \sum_{w\in F_2} f(w) + \frac{2}{\varepsilon}\sum_{v\in C}\sum_{w\in F'} \tilde{c}(v,w)\tilde{x}(v,w) \\
&\le \left(4 + \frac{2\alpha}{\varepsilon}\right) \mathrm{OPT},
\end{aligned}$$

where α is the approximation factor for the CFL solution computed in step 2. As each of the three steps can be executed in polynomial time, this concludes the analysis of the algorithm and the proof of Theorem 1.

Finally, we remark that clients $v \in C$ whose original demand is less than or equal to $\varepsilon\bar{u}$ are not split during the preprocessing (recall that the algorithm is executed on a modified instance in which client demands are at most $\bar{\bar{u}}$), and are thus assigned to a single tour in the algorithm.

4. Algorithm Variants and Heuristic Improvements

In this section, we discuss some heuristic modifications and variants of the algorithm that, while not yielding better worst-case guarantees, help to improve the practical performance of the algorithm. We will assess the effect of these modifications in Section 5.

Accounting for Facilities Opened in Step 1. The algorithm presented in the preceding section computes two sets of facilities, F_1 in Step 1 and F_2 in Step 2, and the final solution opens all facilities in $F_1 \cup F_2$. As the set of facilities in F_1 is already known at the beginning of Step 2, the cost of those facilities can be reduced to 0 in the CFL instance that is solved in that step. This encourages the algorithm to make use of the facilities opened in Step 1 instead of opening additional facilities, heuristically reducing the opening cost of the produced solution.

Reducing CFL Instance Sizes. Step 2 of the algorithm presented in Section 3 requires approximately solving a CFL instance of the same size as the original CLR instance in order to determine the set F_2. While polynomial-time constant factor approximation algorithms for CFL exist, these local-search based algorithms still require significant computational effort and dominate the running time of our algorithm as instance sizes increase. As an alternative approach, one can solve the smaller CFL instance obtained from merging the clusters computed in Step 1 of the algorithm, i.e., similarly to the assignment LP (1), we define $\tilde{c}(S, w) := \min_{v \in V(T_S)} \tilde{c}(v, w)$ and $d(S) := \sum_{v \in S} d(v)$ for $S \in \mathcal{S}$ and $w \in F$. Not only does this modification reduce the number of clients in the CFL instance roughly by a factor of $\bar{\bar{u}}$, the resulting instance also intuitively anticipates the use of the computed set F_2 in the construction of the final solution, where facilities are indeed connected to clusters rather then individual clients. It is thus to be expected that this modification will in fact reduce not only computation time but also solution cost.[4]

Integer Program for Combined CFL and Assignment. Instead of approximately solving a CFL instance to determine a set of facilities to be opened and then applying the rounding procedure to the solution of (1) in order to assign clusters to open facilities, these two steps can be directly addressed in a single step using the following integer program:

$$\begin{array}{llcll}
\min & \displaystyle\sum_{S \in \mathcal{S}} \sum_{w \in F} 2c(S, w) y(S, w) & + & \sum_{w \in F} f(w) z(w) & \\
\text{s.t.} & \displaystyle\sum_{S \in \mathcal{S}} d(S) y(S, w) & \leq & u(w) z(w) & \forall w \in F \\
& \displaystyle\sum_{w \in F} y(S, w) & = & 1 & \forall S \in \mathcal{S} \\
& y(S, w) & \in & \{0, 1\} & \forall S \in \mathcal{S}, w \in F \\
& z(w) & \in & \{0, 1\} & \forall w \in F
\end{array} \tag{4}$$

Here, $c(S, w)$ is defined in the same way as in Section 3.2. The factor 2 in the objective function coefficients for the variables y accounts for the fact that a vehicle not only needs to visit a cluster, but will also travel back to the corresponding facility afterwards, resulting in a better estimation of the routing costs.

A solution to IP (4) not only yields a set of facilities to be opened (those $w \in F$ with $z(w) = 1$) but also an assignment of client clusters to open facilities, which can be used in Step 3 of the algorithm. Even more, such an assignment also leads to a feasible CLR solution, as capacity constraints are satisfied due to the first constrained of (4).

As with the preceding modification, this approach benefits from the fact that the number of clusters is significantly smaller than the number of clients in the original input instance. Thus solving the IP is tractable at least for moderately sized CLR instance. We remark, however, that contrary to LP (1), which is always feasible when the original CLR instance is feasible, there is no such guarantee for IP (4). To address this issue, we suggest the following hierarchical approach: Whenever IP (4) turns out to be infeasible, find

[4] We remark that the resulting CFL instance does not necessarily fulfill the triangle inequality, however, most CFL algorithms can still be applied as heuristics in this case, as this property is only used in their analysis.

the smallest $\gamma > 1$ such that the IP (4) with the right hand-side of the first constraint replaced by $\gamma u(w)z(w)$ is feasible and proceed with an optimal solution to this modified IP.

Post-optimization of Tours via LKH. While typical CLR instances may consist of a large number of clients, individual tours usually only serve a comparably small number of clients due to the limited vehicle capacity (Menezes et al., 2016). For such small sets (even up to several hundreds) of clients, an optimal or almost-optimal tour can be found efficiently using LKH, the implementation of the Lin-Kernighan heuristic by Helsgaun (2000), in Step 3 of the algorithm.

5. Computational Study

In this section, we assess the practical performance of our algorithm in a computational study on a great variety of instances. We investigate the quality of produced solutions and the scalability of the approach.

5.1. Implementation Details

In our experiments, we tested four variants of our algorithm, differing in the implementation of Step 2 and the use of LKH post-optimization:

- $A_{\mathrm{LS}}^{\mathrm{DTS}}$: the algorithm described in Section 3, but using the improvements described in Section 4, i.e., the CFL instance in Step 2 is constructed with modified facility costs and clustered clients; 3-approximation local search procedure by Aggarwal et al. (2013) is used to solve the CFL instance in Step 2
- $A_{\mathrm{IP}}^{\mathrm{DTS}}$: the algorithm described in Section 3, but using the assignment IP (4) to replace Step 2, as described in Section 4
- $A_{\mathrm{LS}}^{\mathrm{LKH}}$: same as $A_{\mathrm{LS}}^{\mathrm{DTS}}$, but the tours are optimized using LKH as described in Section 4
- $A_{\mathrm{IP}}^{\mathrm{LKH}}$: same as $A_{\mathrm{IP}}^{\mathrm{DTS}}$, but the tours are optimized using LKH as described in Section 4

Parameter ε was fixed to 1. Preliminary experiments showed that, not surprisingly, smaller values of ε reduce capacity violations but increase solution cost quite drastically (for $\varepsilon = \frac{1}{2}$ the average total cost went up by a factor of 1.5 on the tested instances). As the IP-based algorithms $A_{\mathrm{IP}}^{\mathrm{DTS}}$ and $A_{\mathrm{IP}}^{\mathrm{LKH}}$ also produce feasible solutions in almost all cases, complete experiments with smaller values of ε were omitted.

The algorithm was implemented in Python 3.7.3, using Gurobi 8.1.1 to solve subproblems (1) and (4), respectively. To avoid excessive execution times on larger instances, we use Gurobi's *TimeLimit* option, which returns the best found solution within the specified time interval. We also preemptively terminate the optimization process for IP (4) if the incumbent solution remains unchanged for a specified period.

Similarly, we interrupt the local search procedure for CFL if the whole procedure exceeds a specified time limit, or if the current solution cannot be improved within a certain time limit. The exact values of the time limits were set according to instance size and are specified in Section 5.2. Note that while preemptive termination can slightly deteriorate the quality of the solutions produced by our algorithm, we still compare them against valid lower bounds, which are computed separately, without imposing time limits.

Extensive use of Python's *networkx* library was made, although avoiding methods that rely on randomness or arbitrariness in order to obtain reproducible results. We also used the LKH implementation available in the *elkai* library. The experiments were run on an AMD Ryzen 5 PRO 2400G processor at 3.6 GHz, with 16 GB RAM.

5.2. Test Instances

We tested the algorithms both on benchmark instances previously used in literature as well as on newly generated random instances.

Benchmark instances from literature. Many instance sets for CLR have been published in literature over the previous decades. Among them, instance sets provided by Barreto et al. (2007), Prins et al. (2006), and Tuzun & Burke (1999) have emerged as a standard benchmark for CLR algorithms. The first one consists of 13 instances with both facility and vehicle capacity. The number of clients ranges from 21 to 150, and the amount of possible facility locations ranges from 5 to 14. Prins et al.'s collection contains 30 instances in which the amount of clients ranges from 20 to 200, and the number of potential facility locations ranges from 5 to 10. Again, both vehicles and facilities have limited capacity. Finally, Tuzun & Burke's set is the largest, with 36 instances. Vehicles have limited capacity, while the capacity of each facility is equal to the total demand of all clients, making them uncapacitated in practice. The number of clients of these instances lies between 100 and 200, while the number of possible locations lies between 10 and 20. We compare our algorithms against the best known solution (BKS) values for these instances, as reported by Schneider & Drexl (2017). We will denote this set of 79 instances by **PBT**.

In addition, we included the collection created by Schneider & Löffler (2019), which contains slightly larger instances. In the 202 instances included in this set, the number of facilities ranges from 5 to 30, and the number of clients lies between 100 and 600, following a similar structure as the instances by Prins et al. (2006). This set not only allows us to test our method on bigger instances than the previous standard, but Schneider & Löffler also provide best known solutions which we can compare our results with. We will denote this instance collection by **S&L**.

Additional randomly generated instances. To assess the scalability of our approach, we created additional randomly generated instances with the number of clients ranging between 50 and 10000. In generating these instances, we slightly adapted the procedures specified by Tuzun & Burke (1999) and by Schneider & Löffler (2019). The main components of this process were the following:

- The number of clients of an instance is specified by a value from the following range: 50, 100, 150, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 2500, 5000, or 10000. For a given instance, we will refer to this value as the *instance size* n. The number of potential facility locations in any instance of size n is simply given by $n/20$, except when the number of clients is either 50 or 150, in which case the number of possible facility locations equals 5 and 10, respectively.
- A number of conglomerates is specified from the set $\{0, 3, 5\}$. This number is used to randomly generate the distance metric described below.
- The vehicle capacity of an instance is specified from the set $\{70, 150, 300\}$.
- The uniform facility capacity is specified from the set $\{400, 600, 1200\}$.
- For each instance, one of three possible ranges for the facility costs is chosen: $[2, 4]$, $[200, 400]$, or $[20000, 40000]$. The cost of each facility is drawn uniformly at random from the chosen range. This leads to instances in which the opening costs are either considerably lower than the routing costs, comparable to them, or considerably higher.
- To determine the (Euclidean) distances, clients and facilities are randomly assigned a point in $[0, 1000]^2$ as follows: If the number of conglomerates specified in the input is 0, all clients and facilities are placed uniformly and independently at random in this region. If the number of conglomerates is 3 or 5, the region $[0, 1000]^2$ is divided in a 3×3-grid of 9 cells, of which 3 or 5, respectively, are chosen as *conglomeration areas*. Then, 80% of the clients and 80% of the facilities are randomly distributed across these conglomeration areas, and each one of them is assigned uniformly at random a coordinate in the corresponding cell. The other 20% of clients and facilities are then equally distributed across the remaining cells, and each one of them is assigned uniformly at random a coordinate in the corresponding cell as well.
- The demand of each client is drawn independently and uniformly from the interval $[10, 20]$.

Each instance created is named n-k-abc, where n stands for the number of clients, k for the number of client conglomerates, and $a, b, c \in \{\mathrm{s}, \mathrm{m}, \mathrm{l}\}$, referring to small, medium, or large vehicle capacity, facility costs, or facility capacity, respectively. According to this scheme, we considered all possible combinations for any given value of $n \leq 1000$ and create $3^4 = 81$ instances for each such value. For the larger instances (size 2500 to 10000), we used Taguchi's orthogonal array design (1960) to cover possible combinations of the parameters in a fractional factorial design with 9 combinations that we tested for $n \in \{2500, 5000, 10000\}$; see Table 1 for the detailed design.

# of conglomerates	facility cost	vehicle capacity	facility capacity
	s	s	s
0	m	m	m
	l	l	l
	s	m	l
3	m	l	s
	l	s	m
	s	l	m
5	m	s	l
	l	m	s

Table 1: Experimental design for XL instances.

In total, we thus obtained 999 instances. We partition this collection into four sets according to instance size: **S** for instances with $50 \leq n \leq 200$, **M** for instances with $300 \leq n \leq 600$, **L** for instances with $700 \leq n \leq 1000$, and **XL** for instances with $2500 \leq n \leq 10000$.

Lower bounds for additional instances. Since no previous solutions are available for the newly generated instances, we use the lower bounds described in Lemmas 1 and 2 to measure the quality of the produced solutions. During the clustering step of the algorithm, the corresponding MST needs to be computed, providing us with the lower bound specified in Lemma 1. Additionally, we compute solutions to the CFL instance described in Lemma 2 for every instance, resulting in the second lower bound. For instances of size S to L, we compute exact solutions to these CFL instances using the standard integer programming formulation for CFL. In the case of the XL instances, due to memory constraints, we used a sparsification approach to reduce the size of the integer programming formulation[5]. For each instance, the greater of these two lower bounds (CFL or MST) is chosen as a reference. The gap is then computed as $\Delta_{\text{LB}} = {}^{(\text{ALG}-\text{LB})}/_{\text{LB}}$, where ALG and LB denote the cost of the solution produced by the algorithm and the lower bound, respectively.

Time limits by instance size. As mentioned in Section 5.1, time limits were set to avoid excessive computation times for large instances. The following time limits were set depending on the instance size:

- For improving the incumbent solution in either the local search or while solving the integer program in Step 2, time limits of 60, 90, and 120 minutes were set for instances in sets M, L, and XL, respectively.

[5]This procedure consists in dividing the area in smaller cells, and letting clients only be directly served by facilities in neighboring cells. Each cell additionally contains an artificial hub that can re-distribute the supply of facilities that are located further away. On instances for which the exact solution to CFL could be computed, this sparsification led to a mild deterioration of at most 2.4% in the computed lower bounds.

- For the entire solution process of either the local search or solving the integer program in Step 2, time limits of 180, 270, and 360 minutes were set for instances in sets M, L, and XL, respectively.
- No time limits were set for the instances belonging to sets PBT, S&L, and S.

5.3. Results

In this section, we present and analyze the results of our computational experiments. We focus on the results obtained by $A_{\mathrm{LS}}^{\mathrm{DTS}}$ and $A_{\mathrm{IP}}^{\mathrm{LKH}}$. While the former is the closest representation of the bifactor approximation described in Section 3, the latter results in the best solution quality. We discuss the differences between all four variants at the end.

Solution quality on benchmark instances. For the benchmark instances from literature, we compare the solutions obtained by our algorithms to the best known solution for the respective instance. We report the gap $\Delta_{\mathrm{BKS}} = {}^{(\mathrm{ALG}-\mathrm{BKS})}/_{\mathrm{BKS}}$, where ALG and BKS denote the cost of the solution produced by the algorithm and the best known solution, respectively. For algorithm $A_{\mathrm{LS}}^{\mathrm{DTS}}$, we observed average gaps of 18.64% (22.49% when restricted to those cases where the solution of the algorithm does not violate any facility capacities) on the instance set PBT and 7.74% (13.97% when restricted to feasible solutions) on the instance set S&L. Algorithm $A_{\mathrm{IP}}^{\mathrm{LKH}}$ computed feasible solutions for all instances of both sets, with average gaps of 11.69% for PBT and 5.23% for S&L. See Fig. 4 for a detailed depiction of the results. We observe that the algorithm thus computes solutions much closer to best known (in many cases proven optimal) solutions than guaranteed by our worst-case analysis in Section 3. For 14 of the S&L instances, $A_{\mathrm{IP}}^{\mathrm{LKH}}$ even computes new best known solutions.

Solution quality on newly generated instances. Using the lower bounds described in Section 5.2, we were able to measure the performance of our algorithm on our newly created instances. We observe that the distribution of the lower bound gap is not significantly affected by instance size, with global average gaps of 63.88% (81.80% when restricted to feasible solutions) for $A_{\mathrm{LS}}^{\mathrm{DTS}}$ and 56.44% (55.61%) for $A_{\mathrm{IP}}^{\mathrm{LKH}}$ (see Fig. 5). These gaps are all considerably lower than the worst-case guarantee of the algorithm. Additionally, $A_{\mathrm{IP}}^{\mathrm{LKH}}$ results in several feasible near-optimal solutions, with gaps as low as 2.15%.

Among the infeasible solutions to the instances in set XL, two outliers were left out of Fig. 5: One solution with a gap of 388.25% computed by $A_{\mathrm{LS}}^{\mathrm{DTS}}$ and one with a gap of 416.38% computed by $A_{\mathrm{IP}}^{\mathrm{LKH}}$. In both cases, the respective procedures in Step 2 of the algorithm (LS or integer program, respectively) were interrupted while the incumbent solutions were still frequently being improved. Nevertheless, both values are still below the guarantee provided by the theoretical analysis.

As we will discuss below, most instances in the set XL resulted in a premature termination of the algorithm due to the time limit. However, with the exception of the two outliers mentioned above, this had

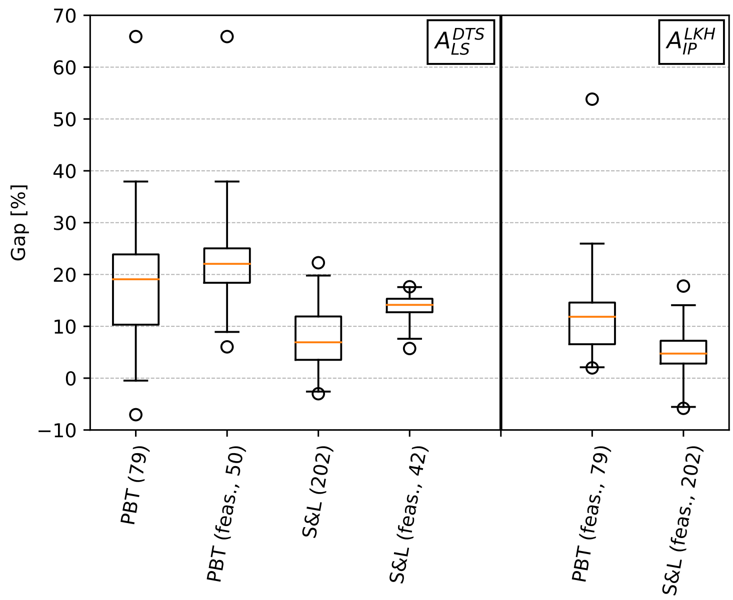

Figure 4: Gap to best known solution for instances in the benchmark set. For algorithm $A_{\mathrm{LS}}^{\mathrm{DTS}}$, the results for all instances in the respective set and the results for only those instances in the set where the algorithm produced a feasible solution (i.e., no violation of capacity limits; indicated by 'feas.') is shown. Numbers in parentheses indicate the respective number of instances. Algorithm $A_{\mathrm{IP}}^{\mathrm{LKH}}$ produced feasible solutions for all instances in the benchmark set. Each box represents the range from the 25th to the 75th percentile. The orange line marks the median. The whiskers represent the remaining 25 percentiles in each in each direction, with the exception for the lowest and highest value, which are marked by circles.

no significant influence on the lower bound gaps, which exhibit practically the same distribution for XL as for S, M, and L. This robustness against early interruption of the optimization process in Step 2 can be seen as another indication for the scalability of our approach.

Running time. The average running time for instances in PBT was 0.42 seconds for $A_{\mathrm{LS}}^{\mathrm{DTS}}$ and 0.34 seconds for $A_{\mathrm{IP}}^{\mathrm{LKH}}$. The average running time for instances in S&L was 7.71 seconds for $A_{\mathrm{LS}}^{\mathrm{DTS}}$ and 15.12 seconds for $A_{\mathrm{IP}}^{\mathrm{LKH}}$. The average running time for the instances in S, M, and L was 31.88 seconds for $A_{\mathrm{LS}}^{\mathrm{DTS}}$ and 254.52 seconds for $A_{\mathrm{IP}}^{\mathrm{LKH}}$, while, on average, the instances in XL required 294.09 minutes to be solved with $A_{\mathrm{LS}}^{\mathrm{DTS}}$ and 132.61 minutes to be solved with $A_{\mathrm{IP}}^{\mathrm{LKH}}$.

Each instance in PBT was solved in less than 1.30 seconds by $A_{\mathrm{LS}}^{\mathrm{DTS}}$ and less than 1.50 seconds by $A_{\mathrm{IP}}^{\mathrm{LKH}}$. Each instance in S&L could be solved in less than 52.20 seconds by $A_{\mathrm{LS}}^{\mathrm{DTS}}$ and less than 859.90 seconds by $A_{\mathrm{IP}}^{\mathrm{LKH}}$, with 90% of them actually being solved in less than 20.64 seconds by $A_{\mathrm{LS}}^{\mathrm{DTS}}$ and less than 19.68 seconds by $A_{\mathrm{IP}}^{\mathrm{LKH}}$. Furthermore, $A_{\mathrm{LS}}^{\mathrm{DTS}}$ solved all of the instances in S, M, and L within the set time limits, while $A_{\mathrm{IP}}^{\mathrm{LKH}}$ solved 96.30% of those instances within the time limit. Among the instances in XL, $A_{\mathrm{LS}}^{\mathrm{DTS}}$ solved 33.33% of the instances within the time limit, while $A_{\mathrm{IP}}^{\mathrm{LKH}}$ solved 59.26%.

While the measured times cannot be directly compared with previously reported run times for other

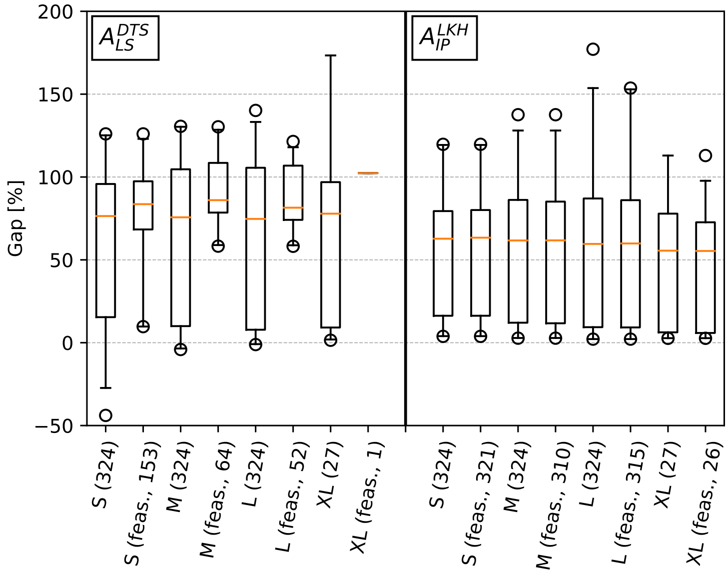

Figure 5: Gap to lower bound for newly generated instances. For each algorithm, the results for all instances in the respective set and the results for only those instances in the set where the algorithm produced a feasible solution (indicated by 'feas.') is shown. Numbers in parentheses indicate the respective number of instances. One outlier for the instance set XL was omitted for each of the two algorithms. See Fig. 4 for an explanation of the box-and-whisker diagrams.

heuristics in literature, we observe that all run times previously reported for the benchmark instances are several orders of magnitude larger (e.g., Schneider & Löffler (2019) report running times in the order of 1 *min* for their heuristic on the instance set PBT, and an average of around 75 *min* for the instance set S&L). Moreover, the instances in sets L and XL (with up to 10000 clients and 500 facilities) are at least an order of magnitude larger than any instances considered in the CLR literature before. Thus, the polynomial run time of our algorithm translates to a high degree of scalability.

We remark that we did not optimize our implementation for speed. The main computational bottleneck of our method is Step 2, i.e., approximating the CFL instance in case of $A_{\mathrm{LS}}^{\mathrm{DTS}}$ or solving IP (4) in case of $A_{\mathrm{IP}}^{\mathrm{LKH}}$. Replacing the local search with another solution approach to CFL, or speeding up the solution process for IP (4) by adding valid inequalities or employing a decomposition approach, respectively, could significantly speed up the solution process and put even larger instance sizes within reach.

Facility loads. In our experiments, 918 out of 1280 (71.72%) solutions computed by $A_{\mathrm{LS}}^{\mathrm{DTS}}$ exceed the capacity at least one facility. In these solutions, on average, an overloaded facility serves an excessive demand of 13.24% of its capacity on average and 73.50% at maximum (on some of the instances where vehicle capacities are close to facility capacities). Over all solutions computed by $A_{\mathrm{LS}}^{\mathrm{DTS}}$, 27.27% of the open facilities had to serve an excess of demand.

For $A_{\mathrm{IP}}^{\mathrm{LKH}}$ on the other hand, facility capacities were only exceeded in 27 out 1280 (2.11%) instances, with an average excess of 3.79% of capacity among all overloaded facilities and a worst-case of 15.25%. On average, only 0.22% of the open facilities had to serve an excess of demand when applying this variant of the algorithm. We conclude that the use of the IP for assigning clusters to facilities results in feasible solutions in the vast majority of cases and that in the cases where it does result in feasible solutions, a small extension of the capacity of individual facilities suffices.

Comparing LS and IP. To measure the direct effect of the assignment method used in Step 2, we compare the variant using local search with that using the IP using the same routing heuristic, respectively. Across all instances, the average increase in costs when using $A_{\mathrm{IP}}^{\mathrm{DTS}}$ with respect to the costs obtained when using $A_{\mathrm{LS}}^{\mathrm{DTS}}$ is 2.58%. Similarly, the average increase in costs when using $A_{\mathrm{IP}}^{\mathrm{LKH}}$ with respect to the costs obtained when using $A_{\mathrm{LS}}^{\mathrm{LKH}}$ is 2.76%. However, recall that most solutions produced by A_{LS}^{X} are infeasible for $X \in \{DTS, LKH\}$ (feasibility is not affected by the routing heuristic). If we restrict our analysis to instances in which both A_{LS}^{X} and A_{IP}^{X} result in feasible solutions, the effect is reversed: across all instances, the average decrease in costs when using $A_{\mathrm{IP}}^{\mathrm{DTS}}$ with respect to the costs obtained when using $A_{\mathrm{LS}}^{\mathrm{DTS}}$ is 0.19%. The same average decrease in costs is obtained when using $A_{\mathrm{IP}}^{\mathrm{LKH}}$ instead of $A_{\mathrm{LS}}^{\mathrm{LKH}}$.

We conclude that, on average, costs are not significantly affected by the choice of assignment heuristic in Step 2. As we saw earlier, running times tend to increase when using A_{IP}^{X} instead of A_{LS}^{X}. However, A_{IP}^{X} computes feasible solutions in most cases, making it a viable candidate for use in practice. This difference can also be observed in Fig. 6: Almost all solutions produced by $A_{\mathrm{IP}}^{\mathrm{DTS}}$ lie left to the 0-excess line, or at least close to it.

Effect of post-optimization. Across all instances and methods, the average routing cost improvement obtained after applying LKH is 8.20%. As mentioned earlier, computation times for applying this post-optimization step are negligible.

Lower bounds. In 70.9% of all instances, CFL resulted in a better lower bound than MST. Moreover, CFL lower bounds are on average 459.83% larger than MST lower bounds. In general, one would expect CFL to result in a better bound when facility costs are considerably higher than routing costs, in which case the costs of the CFL solution should provide a good estimation of the actual CLR costs. If routing costs are considerably higher than facility costs instead and when vehicle capacity utilization is low, MST is expected to provide a better estimates. Thus, a possible explanation for the better performance of the CFL lower bound in our study is that facility costs are significantly higher than routing costs for most instances. In fact, this is also a plausible explanation for the clustering that can be observed in Fig. 6: The lower cluster (below a threshold of approximately 25% on the vertical axis) coincides with the instances for which facility costs are classified as "large". For larger facility costs, the lower bound provided by CFL provides an increasingly better approximation to the optimal cost of the corresponding CLR instance.

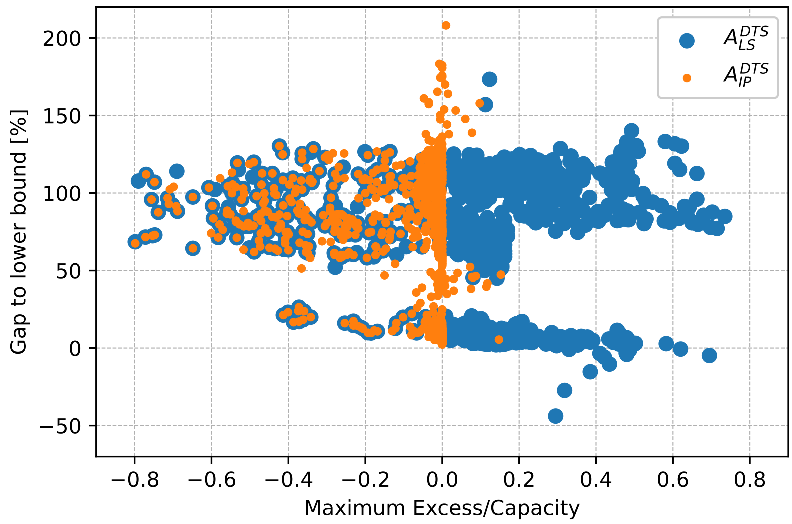

Figure 6: Maximum excessive load vs. gap to lower bound. Each point corresponds to a solution: All instances from all sets are separately solved with $A_{\mathrm{LS}}^{\mathrm{DTS}}$ and $A_{\mathrm{IP}}^{\mathrm{DTS}}$. The vertical coordinate denotes the gap to the lower bound for the solution. The horizontal coordinate denotes the largest excessive demand of a facility relative to its capacity in the respective solution. The same two outliers excluded from Fig. 5 were excluded here as well.

Summary of main findings. The computational results for $A_{\mathrm{LS}}^{\mathrm{DTS}}$ indicate that the algorithm largely outperforms the worst-case guarantee proven in Section 3. Moreover, using the heuristic improvements discussed in Section 4, variant $A_{\mathrm{IP}}^{\mathrm{LKH}}$ obtains solutions that do not exceed any facility capacities for the vast majority of instances. On the benchmark instances the produced solutions are within approximately 7.05% of the best known solutions on average. The experiments on newly generated instances provide evidence that these results can be extended to larger instance sizes as the performance relative to lower bounds remains the same from small instance sizes (comparable to the benchmark instances) to very large instance sizes. We conclude that the algorithmic approach can serve as a basis for a fast heuristic that allows computation of high-quality solutions even on very large instance sizes.

6. Conclusions and Outlook

In this paper, we proposed a bifactor approximation for CLR, with worst-case guarantees for both capacity utilization and costs. The algorithm makes use of a minimum spanning tree in the input graph and a CFL solution with respect to modified connection costs, which are both lower bounds for optimal CLR solutions.

An interesting open question is the existence of a constant-factor approximation for CLR that strictly adheres to the facility capacities. The question for such an algorithm was already raised by Ravi & Sinha (2006). Chen & Chen (2009a,b) gave a positive answer for soft-capacitated variants of the problem. Our theoretical results can be seen as a natural next step towards an approximation algorithm strictly respecting hard capacities. As demonstrated by the example given in the proof of Lemma 3, however, such an algorithm would require the use of new and stronger lower bounds on the optimal solution value, which appear to be a natural starting point for future research. Furthermore, our computational experiments indicate that our approach can also serve as the basis of an efficient heuristic for use in practice. While the IP variant of the algorithm obtains feasible solutions in most cases, future research will hopefully reveal additional heuristic methods that allow to achieve feasibility for the remaining cases in which small capacity violations at some facilities still occur, thus providing implementable solutions for those application contexts in which small capacity extensions are not realizable. Further heuristic improvements of the algorithm might be possible, for example by applying a stronger heuristic for the single-depot vehicle routing problem to each open facility and the clients assigned to it, reducing the routing cost of the solution.

Acknowledgements

We thank three anonymous reviewers for their suggestions that helped improving the presentation of the paper. This work was supported with internal funds of KU Leuven. Part of this work was done while the second and third authors were affiliated with Technische Universität München.

References

Abrams, Z., Meyerson, A., Munagala, K., & Plotkin, S. (2002). On the integrality gap of capacitated facility location. *Carnegie Mellon University, CMU-CS-02-199*, .

Aggarwal, A., Louis, A., Bansal, M., Garg, N., Gupta, N., Gupta, S., & Jain, S. (2013). A 3-approximation algorithm for the facility location problem with uniform capacities. *Mathematical Programming*, *141*, 527–547.

Alpert, C. J., Kahng, A. B., Liu, B., Mandoiu, I. I., & Zelikovsky, A. Z. (2003). Minimum buffered routing with bounded capacitive load for slew rate and reliability control. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, *22*, 241–253.

An, H.-C., Singh, M., & Svensson, O. (2017). LP-based algorithms for capacitated facility location. *SIAM Journal on Computing*, *46*, 272–306.

Bansal, M., Garg, N., & Gupta, N. (2012). A 5-approximation for capacitated facility location. In *European Symposium on Algorithms* (pp. 133–144). Springer.

Barreto, S., Ferreira, C., Paixao, J., & Sousa Santos, B. (2007). Using clustering analysis in a capacitated location-routing problem. *European Journal of Operational Research*, *179*, 968–977.

Byrka, J., & Aardal, K. (2010). An optimal bifactor approximation algorithm for the metric uncapacitated facility location problem. *SIAM Journal on Computing*, *39*, 2212–2231.

Chen, X., & Chen, B. (2009a). Approximation algorithms for soft-capacitated facility location in capacitated network design. *Algorithmica*, *53*, 263–297.

Chen, X., & Chen, B. (2009b). Cost-effective designs of fault-tolerant access networks in communication systems. *Networks: An International Journal*, *53*, 382–391.

Christofides, N. (1976). *Worst-case analysis of a new heuristic for the travelling salesman problem*. Technical Report Carnegie-Mellon Univ Pittsburgh Pa Management Sciences Research Group.

Drexl, M., & Schneider, M. (2015). A survey of variants and extensions of the location-routing problem. *European Journal of Operational Research*, *241*, 283–308.

Golden, B. L., Raghavan, S., & Wasil, E. A. (2008). *The vehicle routing problem: latest advances and new challenges* volume 43. Springer Science & Business Media.

Harks, T., König, F. G., & Matuschke, J. (2013). Approximation algorithms for capacitated location routing. *Transportation Science*, *47*, 3–22.

Helsgaun, K. (2000). An effective implementation of the Lin–Kernighan traveling salesman heuristic. *European Journal of Operational Research*, *126*, 106–130.

Jain, K., Mahdian, M., Markakis, E., Saberi, A., & Vazirani, V. V. (2003). Greedy facility location algorithms analyzed using dual fitting with factor-revealing LP. *Journal of the ACM (JACM)*, *50*, 795–824.

Karlin, A. R., Klein, N., & Gharan, S. O. (2021). A (slightly) improved approximation algorithm for metric tsp. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing* (pp. 32–45).

Korupolu, M. R., Plaxton, C. G., & Rajaraman, R. (2000). Analysis of a local search heuristic for facility location problems. *Journal of Algorithms*, *37*, 146–188.

Levi, R., Shmoys, D. B., & Swamy, C. (2012). LP-based approximation algorithms for capacitated facility location. *Mathematical Programming*, *131*, 365–379.

Li, C.-L., & Simchi-Levi, D. (1990). Worst-case analysis of heuristics for multidepot capacitated vehicle routing problems. *ORSA Journal on Computing*, *2*, 64–73.

Li, S. (2013). A 1.488 approximation algorithm for the uncapacitated facility location problem. *Information and Computation*, *222*, 45–58.

Maranzana, F. (1964). On the location of supply points to minimize transport costs. *Journal of the Operational Research Society*, *15*, 261–270.

Menezes, M. B., Ruiz-Hernández, D., & Verter, V. (2016). A rough-cut approach for evaluating location-routing decisions via approximation algorithms. *Transportation Research Part B: Methodological*, *87*, 89–106.

Morell, S., & Skutella, M. (2020). Single source unsplittable flows with arc-wise lower and upper bounds. In *International Conference on Integer Programming and Combinatorial Optimization* (pp. 294–306). Springer.

Prins, C., Prodhon, C., & Wolfler Calvo, R. (2006). Solving the capacitated location-routing problem by a GRASP complemented by a learning process and a path relinking. *4OR*, *4*, 221–238.

Prodhon, C., & Prins, C. (2014). A survey of recent research on location-routing problems. *European Journal of Operational Research*, *238*, 1–17.

Ravi, R., & Sinha, A. (2006). Approximation algorithms for problems combining facility location and network design. *Operations Research*, *54*, 73–81.

Salhi, S., & Nagy, G. (1999). Consistency and robustness in location-routing. *Studies in Locational Analysis*, (pp. 3–19).

Salhi, S., & Rand, G. K. (1989). The effect of ignoring routes when locating depots. *European Journal of Operational Research*, *39*, 150–156.

Schneider, M., & Drexl, M. (2017). A survey of the standard location-routing problem. *Annals of Operations Research*, *259*, 389–414.

Schneider, M., & Löffler, M. (2019). Large composite neighborhoods for the capacitated location-routing problem. *Transportation Science*, *53*, 301–318.

Shmoys, D. B., & Tardos, É. (1993). An approximation algorithm for the generalized assignment problem. *Mathematical Programming*, *62*, 461–474.

Taguchi, G. (1960). Table of orthogonal arrays and linear graphs. *Rep. Stat. Appl. Res., JUSE*, *6*, 1–52.

Toth, P., & Vigo, D. (2003). The granular tabu search and its application to the vehicle-routing problem. *INFORMS Journal on Computing*, *15*, 333–346.

Tuzun, D., & Burke, L. I. (1999). A two-phase tabu search approach to the location routing problem. *European Journal of Operational Research*, *116*, 87–99.

Webb, M. (1968). Cost functions in the location of depots for multiple-delivery journeys. *Journal of the Operational Research Society*, *19*, 311–320.

Williamson, D. P., & Shmoys, D. B. (2011). *The Design of Approximation Algorithms*. Cambridge university press.