Approximation Algorithms for Capacitated Location Routing*

Tobias Harks Felix G. König
Jannik Matuschke

Technische Universität Berlin, Institut für Mathematik
Straße des 17. Juni 136, 10623 Berlin, Germany
{harks,fkoenig,matuschke}@math.tu-berlin.de.

Preprint 2010/10

May 5, 2010
Revised: February 7, 2012

Abstract

An *approximation algorithm* for an optimization problem runs in polynomial time for all instances and is guaranteed to deliver solutions with bounded optimality gap. We derive such algorithms for different variants of *capacitated location routing*, an important generalization of vehicle routing where the cost of opening the depots from which vehicles operate is taken into account. Our results originate from combining algorithms and lower bounds for different relaxations of the original problem, and besides location routing we also obtain approximation algorithms for *multi-depot capacitated vehicle routing* by this framework. Moreover, we extend our results to further generalizations of both problems, including a *prize-collecting* variant, a *group* version, and a variant where *cross-docking* is allowed. We finally present a computational study of our approximation algorithm for capacitated location routing on benchmark instances and large-scale randomly generated instances. Our study reveals that the quality of the computed solutions is much closer to optimality than the provable approximation factor.

1 Introduction

The broad realm of vehicle routing addresses the omnipresent logistic challenge of minimizing the cost of operating vehicles performing pickups and/or deliveries of goods for clients from a given set of depots. In many logistics applications, however, the cost of opening these depots constitutes a second major cost driver. Integrating this aspect of location decisions into the model leads to an additional and distinct optimization challenge. The two families corresponding to the routing and location subproblems, namely *vehicle routing* and *facility location*, have been studied extensively from practical as well as theoretical points of view. The integrated problem of jointly making location and routing decisions is known as *location routing* and has received significant attention in the operations research community as well.

*This work was supported by the European Regional Development Fund (ERDF) and is part of a joint project with 4flow AG, Berlin, Germany.

A basic variant of location routing is the *capacitated location routing problem* (CLR) defined as follows. We are given an undirected connected graph $G = (V, E)$, where the node set $V = \mathscr{C} \dot{\cup} \mathscr{F}$ of the graph is partitioned into a set of *clients* $\mathscr{C}$ and a set of *facilities* $\mathscr{F}$. We will use the term *facilities* interchangeably with the term *depots*. There are cost functions $c : E \to \mathbb{R}^+$ on the edge set, and $\phi : \mathscr{F} \to \mathbb{R}^+$ associated with the depots modeling *opening costs*. Every potential depot maintains an unbounded fleet of vehicles, each with a *uniform capacity* $u > 0$. Each client $v \in \mathscr{C}$ has a demand $d_v > 0$. A feasible solution to CLR is given by a tuple $(F, \mathscr{T})$, where $F \subseteq \mathscr{F}$ is a set of open depots and $\mathscr{T}$ is a set of tours $\{T_1, \ldots, T_k\}$ such that (1) every tour starts at an open depot and returns to the same depot at the end, (2) the demand of every client is served by the tours by which it is visited, and (3) the total demand served by a tour does not exceed u. The total cost of a solution is defined by $\sum_{T \in \mathscr{T}} c(T) + \sum_{w \in F} \phi(w)$, where $c(T) = \sum_{e \in T} c_e$ denotes the routing cost of tour T. Note that we may assume without loss of generality that G is complete and the edge costs c satisfy the triangle inequality: If this is not the case, we replace G by its metric closure, i.e., introducing an edge between each pair of vertices with the cost of a shortest path between those vertices in the original graph. Furthermore, note that this model also implicitly covers depot-dependent fixed costs per tour, i.e., each vehicle sent out from depot v incurs a cost of $a_v \in \mathbb{R}^+$. This can be easily modeled by adding $\frac{1}{2}a_v$ to the cost of all edges incident to v, as each tour originating at v contains exactly two of these edges.

In the version of the problem described above, a client's demand may be split up and served by multiple facilities, which is not always desired or even possible in practice. This motivates the following terminology. A solution to CLR fulfills the *single-assignment property* (cf. Nagy and Salhi (2007); Laporte et al. (1988)), if the demand of each client is served by exactly one facility. A solution fulfills the *single-tour property*, if each client's demand is served by exactly one tour. Clearly, this latter property can only be fulfilled if $d_v \leq u$ for all $v \in \mathscr{C}$.

The special case of CLR where location decisions have already been made (i.e., $\phi \equiv 0$) is the *multi-depot capacitated vehicle routing problem* (MDCVR). Note that in the uncapacitated case ($u = \infty$), CLR and MDCVR are equivalent: By triangle inequality, every optimal solution to either problem can be transformed such that each depot is visited by at most one tour (without increasing cost). Hence, facility opening cost can be modeled by adding $\frac{1}{2}\phi(v)$ to $c(e)$ for all edges e incident to a facility $v \in \mathscr{F}$.

Not surprisingly, CLR contains *NP*-hard combinatorial optimization problems as a special case. When there is only one facility and infinite vehicle capacity, for instance, the problem becomes the travelling salesman problem. Or, when demands are uniform and match the vehicle capacity, CLR becomes the metric uncapacitated facility location (UFL) problem, as an optimal routing corresponds to finding shortest paths from each client to an open facility.

Because of this intrinsic hardness, an *exact* solution method for most location routing problems including CLR is very likely to perform poorly on some problem instances. Speaking more formally, its *worst-case running time* is likely to grow *exponentially* with problem size (Haimovich et al., 1988). In fact, even for simple variants of vehicle routing problems, only relatively small instances are solved to optimality, see the book by Toth and Vigo (2002) and references therein. On the other hand, problem sizes encountered in real-life problems have grown tremendously over the past years (and are expected to grow further), thus fast heuristics are becoming increasingly important for solving location and vehicle routing problems (Cordeau et al., 2002; Desrochers et al., 1988). While *(meta-)heuristics* used today deliver feasible solutions to larger instances in reasonable time, there is usually *no guaranteed bound* regarding solution quality. Merely for some restricted special cases, there are heuristics for which such bounds are known, see Haimovich et al. (1988).

To address this apparent dilemma regarding worst-case running time and guaranteed solution quality, we use *approximation algorithms* in this paper, a solution methodology in the

intersection of mathematics, computer science, and operations research. An approximation algorithm for an *NP*-hard combinatorial optimization problem is a heuristic enjoying two desirable properties: Its worst-case running time is bounded by a polynomial in problem size, and there are provable a priori bounds (constant numbers in the best case) on the worst-case quality of the solution generated:

Definition 1.1. An algorithm *ALG* for a minimization problem P is a ρ*-approximation algorithm* if it runs in time polynomial in the input size, and for every instance I of P, we have

$$\mathrm{ALG}(I) \leq \rho \cdot \mathrm{OPT}(I),$$

where $\mathrm{ALG}(I)$ and $\mathrm{OPT}(I)$ denote the objective values of the solution returned by *ALG* and of an optimal solution for I, respectively.

While this worst-case guarantee gives theoretical evidence for the reasonability of the algorithm, the quality of solutions may be much closer to optimality in practice than the approximation factor indicates. A standard reference containing approximation algorithms for a multitude of hard optimization problems is the book of Hochbaum (1997). Another recent and very good reference, containing a detailed introduction to the various techniques used in the design of approximation algorithms, is the book by Williamson and Shmoys (2011).

Within this framework, we devise a constant factor approximation algorithm for CLR with arbitrary demands. For MDCVR with arbitrary demands, we obtain an improved approximation factor, which is, to the best of our knowledge, the best constant factor approximation algorithm for this problem to date. Moreover, we consider three practically relevant extensions of the above model. Suppose a company has the option not to serve all clients' demands itself, but to outsource any number of transports to clients at given customer-dependent prices. This extended model is known as the *prize-collecting* capacitated location routing problem (PC-CLR). In the second extension, we consider *group* capacitated location routing (G-CLR) where the set of clients is partitioned into groups $\mathscr{C}_1, \ldots, \mathscr{C}_k$, with $\mathscr{C} = \bigcup_{i=1}^{k} \mathscr{C}_i$. In a feasible solution, only one client from each group needs to be served. Applications include intermodal transport networks, where goods can be transferred from one logistics network to the next at one of several hub locations. In the third extension, *cross-docking* is allowed: We allow consolidation tours which do not visit a facility, but contain one node where they meet with other tours. From there, spare capacity on the latter tours is utilized jointly to forward all demand of the consolidation tour to facilities. Being of profound practical interest (see e.g. Vahdani and Zandieh (2010); Wen et al. (2009)), cross-docking operations may significantly improve capacity utilization and hence reduce total cost. We extend our constant factor approximations to all three of these variants, where for the group version, our approximation guarantee depends on the cardinality of the largest group.

1.1 Previous Results

Location routing (as the integration of vehicle routing and facility location) has occupied a central place in the operations research literature over the past decades. Since hundreds of papers have been published in this broad area, we will give pointers to text books and survey articles when referring to the main streams in location routing. However, we give a concise overview of works regarding approximation algorithms on the subject.

Location Routing. Perhaps one of the earliest models of location routing appears in the paper by Webb (1968). Laporte (1988) gives a comprehensive overview of the literature prior to the

late 80s. More recent survey articles summarizing heuristic algorithms and mathematical programming formulations for many variants of location routing can be found in Mina et al. (1998) and Nagy and Salhi (2007). Very recently, there have been several works on integer programming formulations for CLR with capacitated facilities using strengthened cut inequalities, see Belenguer et al. (2011) and Contardo et al. (2010).

There are only a few works that are concerned with approximation theory for location routing problems. For unbounded vehicle capacity, a $(2-\frac{1}{|V|-1})$-approximation algorithm is given by Goemans and Williamson (1995). Glicksman and Penn (2008) generalize this result to the case of (uncapacitated) group location routing, where one is given a system of groups of clients, and only one client from each group needs to be served. Among other results, they derive a $(2-\frac{1}{|V|-1})L$-approximation algorithm, where L denotes the cardinality of the largest group. Finally, Chen and Chen (2009) provide a 24-approximation for location routing with soft facility capacities (i.e., facilities can be installed multiple times, each copy capable to serve a limited amount of demand, while vehicle loads are still unbounded).

Vehicle Routing. When facilities can be opened at no cost, location routing becomes the multi-depot vehicle routing problem, for which countless exact and heuristic solution methods have been proposed. For an overview of the rich literature in this field, we refer the reader to the books edited by Toth and Vigo (2002), Golden and Assad (1988), and the surveys of Baldacci et al. (2010), Cordeau et al. (2002, 2007), Haimovich et al. (1988), Laporte (2009), and Laporte and Semet (2001). For vehicle routing problems with additional side constraints (such as time-windows, heterogeneous fleets, fleets of limited size) see also Bräysy and Gendreau (2005), Baldacci et al. (2010), and Desrochers et al. (1988).

There is a large body of work regarding the classical capacitated vehicle routing problem (with a single depot) including the seminal PTAS of Haimovich and Rinnoy Kan (1985) for geometric distances. Li and Simchi-Levi (1990) consider the multi-depot capacitated vehicle routing problem (MDCVR) and present, among other results, a $(2+2\rho_{\mathrm{TSP}})$-approximation algorithm for arbitrary, unsplittable demands, where ρ_{TSP} denotes the factor of an approximation algorithm for the travelling salesman problem. This is the best previously known approximation algorithm for this version of the problem, with $\rho_{\mathrm{TSP}} = 3/2$ using the algorithm by Christofides (1976). There is also a PTAS for the case of Euclidean distances and uniform demands, albeit with running time exponential in vehicle capacity as well as the number of depots (Cardon et al., 2008).

Charikar et al. (2001b) studied the related k-delivery TSP in which a single vehicle with capacity k needs to transport n (unit-sized) items located at arbitrary locations to given demand points. For this problem they derive a 5-approximation.

Facility Location. Approximation algorithms for metric uncapacitated facility location (UFL) constitute a central topic in combinatorial optimization. As a reference, we point the reader to the 1.52-approximation of Mahdian et al. (2006). Using ideas of Chudak and Shmoys (2003), a recent publication by Byrka and Aardal (2010) improves this factor to 1.5, also introducing a bifactor approximation that provides separate approximation ratios for connection and opening costs with respect to an initially solved LP relaxation.

Ravi and Sinha (2006) study the related *capacitated cable facility location problem*. As in CLR, one is given a complete undirected graph with metric costs on the edges. A set of clients needs to be served from facilities with associated opening costs. Facilities need to be opened and clients need to be connected to open facilities by Steiner trees, where an edge e of a tree is associated with a number of cables bought for the corresponding connection, each at price

$c(e)$. Each cable has uniform capacity u, and each connection needs to comprise enough cables to provide capacity no less than the number of clients depending on it. The authors propose a $(\rho_{\text{UFL}} + \rho_{\text{ST}})$-approximation algorithm, where ρ_{UFL} and ρ_{ST} denote the approximation factors of algorithms for UFL and Steiner tree, respectively, which are used as subroutines. Their algorithm computes a feasible solution by merging a UFL and a Steiner tree solution. The merging procedure first routes the entire demand along the Steiner tree and then iteratively relieves overloaded subtrees of excessive demand by rerouting it to a closest open facility in the UFL solution.

The approximation algorithms in this paper use a similar technique of merging two solutions (UFL and a minimum spanning tree) by iteratively rerouting demand from overloaded subtrees of the spanning tree to a closest open facility of the UFL solution. Since our model is tour-based, however, we cannot argue on individual link capacities, or use corresponding flow arguments. The merging procedure in Ravi and Sinha (2006) crucially relies on the flexibility to install sufficient cable capacity on individual edges, and to fractionally route flow under these capacity constraints. In contrast, we have to decide about buying complete tours from open facilities, requiring a different rerouting procedure.

Extended Models. In the *prize-collecting* (PC) version of the above problems, a feasible solution does not have to serve all clients. Instead, an individual penalty may be paid for each unserved client. Thereby, PC can precisely model outsourcing decisions and is hence of profound practical interest. For the PC version of UFL, Jain et al. (2003) claim to obtain a 2-approximation, improving the 3-approximation by Charikar et al. (2001a), but omitting a complete proof. We are not aware of any previous approximation results for PC vehicle routing or PC location routing.

In the *group* variant, the set of clients is partitioned into disjoint subsets, or groups of clients, and only one client from every group has to be served. Group facility location is closely related to unweighted set cover, as we shall see in Section 4. For the group case of uncapacitated vehicle and location routing, the only previous result we know of is the algorithm by Glicksman and Penn (2008) mentioned above.

Finally, in capacitated location routing and multi-depot capacitated vehicle routing, *cross-docking* may be allowed in certain application scenarios. Here, some clients are served by consolidation tours which do not connect directly to a facility, but meet with other tours having spare capacity. These latter tours jointly forward all demand required by the consolidation tour to their respective facilities. Cross-docking plays a significant role in numerous logistics applications, and some heuristic approaches have recently been proposed for vehicle routing with cross-docking (Vahdani and Zandieh, 2010; Wen et al., 2009). This model also exhibits strong similarity to a practically relevant problem called mixed truck delivery which is studied in Liu et al. (2003). Here, delivery tours are sought as well, and clients may be served by tours either from facilities or from hubs, which are in turn served by facilities. The authors develop a heuristic solution approach and present computational results suggesting that routing cost can be reduced on average by around 10% for random instances when allowing cross-docking. Our model corresponds to the case where each client node may also function as a hub.

1.2 Our contribution and structure of the paper

In Section 2 we develop a framework for combining approximation algorithms for facility location with spanning or Steiner tree algorithms in order to obtain approximation algorithms for capacitated location routing and multi-depot capacitated vehicle routing problems. We apply our technique to devise a constant factor approximation algorithm for CLR with arbitrary

demands. We are not aware of any previous results regarding constant factor approximations for CLR. For MDCVR, we obtain an improved approximation guarantee which is, to the best of our knowledge, the best approximation factor to date. In Sections 3, 4 and 5 we study the prize-collecting, group, and cross-docking variants. We extend our approximation algorithm to all three variants. While we derive constant factor approximations for the prize-collecting and cross-docking versions, the approximation guarantee for the group version depends on the cardinality of the largest group. In fact, we show that this version of the problem does not allow for a constant factor approximation by providing a lower bound on the achievable approximation factor depending on the number of groups. In Section 6 we present a computational study of our algorithm for CLR, where we compare solution quality and running time with those of other algorithms for CLR from the literature on benchmark instances. It turns out that in practice, the algorithm's performance greatly exceeds its theoretically proven approximation guarantee. On the benchmark test set, the quality of our solutions is on average within a factor of 1.1–1.2 of best known solutions. While the increase in cost over other algorithm is mild, our algorithm's running time is several magnitudes faster, taking only negligible time on benchmark instances. To further demonstrate this computational efficiency we test our algorithm on a set of large-scale randomly generated instances (1000–10000 customers, 100–1000 facilities per instance). We are not aware of any previous work considering CLR instances of comparable size. We conclude the paper in Section 7 with a brief summary and a discussion of open problems.

2 Approximation Algorithm for Capacitated Location Routing

In this section, we present our main approximation result. After deriving two lower bounds, we present our algorithm for CLR followed by its analysis. Finally, we describe a specialization for multi-depot capacitated vehicle routing yielding an improved approximation guarantee.

Before we start, we introduce some additional notation. As described in the introduction, a feasible solution to CLR consists of a set of open facilities F and a set of tours (or, in mathematical terms, closed walks) $\mathscr{T}$ such that (1) each tour visits an open facility, (2) the demand of each client is served by the tours by which it is visited, and (3) the demand transported by a tour does not exceed the vehicle capacity u. The second and third condition can be expressed by the existence of *demand assignments*, i.e., non-negative values x_{vT} for each $v \in \mathscr{C}$ and $T \in \mathscr{T}$ fulfilling (2) $\sum_{T \in \mathscr{T}: v \in V(T)} x_{vT} = d_v$ for all $v \in \mathscr{C}$ and (3) $\sum_{v \in \mathscr{C}} x_{vT} \leq u$ for all $T \in \mathscr{T}$. Note that these demand assignments can be computed efficiently from the tuple $(F, \mathscr{T})$ and it is thus not important whether they are part of the formal solution output. However, we will use them in proofs throughout the paper.

Using this notation, we once again give a formal definition of the basic version of the capacitated location routing problem, which is the subject of this section.

Problem 2.1 (Capacitated Location Routing)**.**

Input: a graph $G = (\mathscr{C} \cup \mathscr{F}, E)$, metric edge costs $c : E \to \mathbb{R}^+$, opening costs $\phi : \mathscr{F} \to \mathbb{R}^+$, demands $d : \mathscr{C} \to \mathbb{R}^+$, vehicle capacity $u \in \mathbb{R}^+$

Task: Find a set of facilities $F \subseteq \mathscr{F}$ and a set of closed walks $\mathscr{T}$ with a demand assignment $x : \mathscr{C} \times \mathscr{T} \to \mathbb{R}^+$ such that

(1) $V(T) \cap F \neq \emptyset$ for all $T \in \mathscr{T}$,

(2) $\sum_{T \in \mathscr{T}: v \in V(T)} x_{vT} = d_v$ for all $v \in \mathscr{C}$,

(3) $\sum_{v\in\mathscr{C}} x_{vT} \le u$ for all $T \in \mathscr{T}$,

minimizing the cost $\sum_{w\in F} \phi(w) + \sum_{T\in\mathscr{T}} \sum_{e\in T} c_e$.

2.1 Two Lower Bounds

We provide two lower bounds on the optimal solution, which will be used to derive a constant approximation factor for our algorithm.

Lemma 2.2. *Given an instance of CLR, consider an uncapacitated facility location (UFL) instance defined as follows. The sets of clients and facilities remain the same as in CLR, but we set the costs of edges to $\tilde{c} := \frac{2}{u}c$. Then, the cost of an optimal solution to UFL (w.r.t. $\tilde{c}$) is at most the cost of an optimal solution to CLR (w.r.t. c).*

Proof. Consider a feasible solution $(F, \mathscr{T})$ of CLR with demand assignments x_{vT}. Construct a solution U of the UFL instance by opening all facilities that were opened by CLR and connecting each client $v \in \mathscr{C}$ to a closest open facility $w(v) \in F$. The connection cost of U is $\tilde{c}(U) = \sum_{v\in\mathscr{C}} \tilde{c}_{vw(v)} d_v$. We will show that $\tilde{c}(U) \le \sum_{T\in\mathscr{T}} c(T)$, which proves the lemma.

Consider a flow f constructed from the CLR solution as follows. For every client $v \in \mathscr{C}$ and each tour T serving v, partition T into two paths from the facility to the client and send x_{vT} units of flow along each path. Note that the amount of flow carried by edge $e \in E$ is at most u times the number of tours containing e and, thus, e is contained in at least $\left\lceil \frac{f_e}{u} \right\rceil$ tours. Denoting the cost of f w.r.t. $\tilde{c}$ by $\tilde{c}(f)$ we deduce

$$\frac{1}{2}\tilde{c}(f) = \frac{1}{u}\sum_{e\in E} c_e f_e \le \sum_{e\in E} c_e \left\lceil \frac{f_e}{u} \right\rceil \le \sum_{e\in E} \sum_{T\in\mathscr{T}: e\in T} c_e = \sum_{T\in\mathscr{T}} c(T).$$

Note that the construction of the flow f induces a path decomposition. Let $\mathscr{P}_v$ be the set of all paths from a facility to client $v \in \mathscr{C}$ used in the construction of f and let f_P be the flow value assigned to that path. Note that $\sum_{P\in\mathscr{P}_v} f_P = \sum_{T\in\mathscr{T}} 2x_{vT} = 2d_v$, because every tour contributes two paths for every client it serves. Furthermore, $\tilde{c}(P) \ge \tilde{c}_{vw(v)}$, i.e., the length of any of the facility-client-paths is at least the distance to a closest facility. Thus, we obtain

$$\tilde{c}(f) = \sum_{v\in\mathscr{C}} \sum_{P\in\mathscr{P}_v} \tilde{c}(P) f_P \ge \sum_{v\in\mathscr{C}} \tilde{c}_{vw(v)} \sum_{P\in\mathscr{P}_v} f_P = \sum_{v\in\mathscr{C}} \tilde{c}_{vw(v)} 2d_v = 2\tilde{c}(U)$$

showing that $\tilde{c}(U) \le \sum_{T\in\mathscr{T}} c(T)$. □

Lemma 2.3. *Given an instance of CLR, consider the graph $G' = (V \cup \{r\}, E \cup E')$, where $E' = \{\{r,w\} : w \in \mathscr{F}\}$ and define costs $c'_{rw} = 0$, $c'_{vw} = c_{vw} + \frac{1}{2}\phi(w)$ for all $v \in \mathscr{C}$, $w \in \mathscr{F}$, and $c'_{vw} = c_{vw}$ for all other $\{v,w\} \in E$. Then the cost of a minimum spanning tree in G' with respect to the costs c' is a lower bound on the cost of an optimal solution of CLR (w.r.t. c).*

Proof. Consider a feasible solution $(F, \mathscr{T})$ to CLR. We will construct a spanning tree in G' that has at most the same cost. For every open facility $w \in F$, let $T_1, \dots, T_j$ be an arbitrary ordering of the tours based at w with $T_i = (w, v_1^i, \dots, v_{l_i}^i, w)$ where l_i is the number of clients in T_i. For $i = 1, \dots, j-1$, replace the last edge $\{v_{l_i}^i, w\}$ of T_i and the first edge $\{w, v_1^{i+1}\}$ of T_{i+1} by the direct edge $\{v_{l_i}^i, v_1^{i+1}\}$. Also remove the final edge $\{v_{l_j}^j, w\}$ of T_j. As a result, we get a walk P_w from w to $v_{l_j}^j$ along all clients that are served by w. Note that $c'(P_w) = \sum_{e\in P_w} c'(e) \le \sum_{i=1}^{j} c(T_i) + \frac{1}{2}\phi(w)$ by triangle inequality and the fact that P_w contains only one edge incident to w.

Now let S be the union of all P_w for $w \in F$ and all edges in E'. As S spans all facilities and contains a walk from any client to a facility, it contains a spanning tree of G' with cost at most $c'(S) \leq \sum_{T \in \mathscr{T}} c(T) + \sum_{w \in F} \phi(w)$. □

2.2 Algorithm

We construct an approximate solution to CLR from an approximate solution to the UFL instance described in Lemma 2.2 and a minimum spanning tree on the graph G' as described in Lemma 2.3. Essentially, the idea is to decompose the spanning tree into subtrees with demands between $u/2$ and u, which can then be turned into tours by doubling edges. These tours are serviced by facilities opened by either the spanning tree or the UFL solution. The cost of the resulting solution is bounded by the sum of twice the cost of the spanning tree, twice the connection cost of the UFL solution, and once the opening cost of the UFL solution. Using the bifactor approximation algorithm of Byrka and Aardal (2010) for UFL, we obtain a total approximation factor of 4.38 for CLR.

We now describe the algorithm in more detail. After solving the UFL instance approximately and computing a minimum spanning tree, we open all facilities that are opened in the UFL solution and also all facilities w that are incident to an edge other than $\{w,r\}$ in the spanning tree S. Any client with demand $d_v \geq u$ is assigned to a closest open facility and served by $\lceil \frac{d_v}{u} \rceil$ tours comprising only the assigned facility and the client. We proceed to describe how to construct tours for the remaining demands by merging the given spanning tree on G' with a UFL solution to obtain a feasible solution to CLR (this will later be referred to as the "merge phase"). For a better understanding, direct the spanning tree towards the root r and denote the subtree rooted at node v by S_v with D_v being the sum of the demands of all clients in S_v.

If z is a facility and the total demand in S_z is at most u, we turn this subtree into a tour based at z by doubling edges and short-cutting by triangle inequality. If the total demand in S_z exceeds u, we will relieve this subtree by rerouting excessive demand to other open facilities, charging the costs to the UFL solution, until the remaining demand is at most u. This last step resembles a technique introduced by Ravi and Sinha (2006).

We now describe our rerouting procedure in detail. Let v be a node in S_z such that $D_v > u$ but $D_w \leq u$ for all children w of v. Let I be the set containing all subtrees S_w with w being a child of v and the set $\{v\}$ itself. We want to make sure that less than u units of demand have to be routed to the parent of v in the tree and the rest of the demand is connected with additional edges paid for by the UFL solution. To this end, we greedily partition I into groups $I_0, \ldots, I_k$ such that the sum of demands of all subtrees in each group I_j is at most u but at least $u/2$ (unless $j = 0$). We keep the connection of all trees in I_0 to the node v, but we extract the trees of all other groups from the spanning tree (including the edges connecting them with v). For each $j = 1, \ldots, k$, the subtrees in group I_j together with the edges to v form one single tree which can be turned into a tour by doubling edges and short-cutting. Among all clients on this tour we choose one with the cheapest connection cost to an open facility and insert this facility into the tour, paying at most twice the cost of the corresponding edge by triangle inequality. Observe that this edge carries at least $u/2$ units of demand. We repeat this procedure until the total demand in the subtree S_z is at most u. Then we turn the remainder of S_z into a tour, again by doubling edges.

2.3 Analysis

We analyze the algorithm presented in the previous section to show that it is a 4.38-approximation for CLR. We start by estimating the cost of the solution produced in the merge phase against the cost of the spanning tree and the facility location solution.

Algorithm 1: Algorithm for CLR.

Input: An instance of CLR.
Output: A feasible solution to CLR.
UFL phase:
Create an UFL instance with edge costs $\tilde{c} = \frac{2}{u}c$ as described in Lemma 2.2.
Apply the bifactor approximation algorithm of Byrka and Aardal with $\gamma = 2.38$ on this instance and let F_1 be the set of facilities opened in the resulting UFL solution.
Tree phase:
Construct the graph G' with edge costs c' as described in Lemma 2.3 and compute a minimum spanning tree S.
Let F_2 be the set of facilities that are incident to an edge in $S \cap E$.
Large demand phase:
Open all facilities in $F_1 \cup F_2$.
forall the $v \in \mathscr{C}$ *with* $d_v \geq u$ **do**
  Construct $\lceil \frac{d_v}{u} \rceil$ copies of a tour from v to a closest open facility.
  Add the tours to $\mathscr{T}$ and remove the corresponding demand d_v.
end
Merge phase:
forall the $z \in F_2$ **do**
  while $D_z > u$ **do**
    Let $v \in V(S_z)$ such that $D_v > u$ but $D_w \leq u$ for all children w of v.
    Let $I = \{V(S_w) : w \text{ is a child of } v\} \cup \{\{v\}\}$.
    Find a partition $I = I_0 \dot{\cup} \ldots \dot{\cup} I_k$, such that $\sum_{v \in I_j} d_v \leq u$ for all $j \in \{0, \ldots, k\}$ and $\sum_{v \in I_j} d_v > \frac{u}{2}$ for all $j \in \{1, \ldots, k\}$.
    forall the $j \in \{1, \ldots, k\}$ **do**
      Find a pair (w, z') such that w is a vertex of a tree in I_j, $z' \in F_1 \cup F_2$ and c_{wz} is minimal.
      Construct a tour visiting all vertices of trees in I_j and z' by doubling wz and the edges of all trees in I_j and short-cutting.
      Add the tour to $\mathscr{T}$ and remove the corresponding subtrees in I_j from S.
    end
  end
  Construct a tour from S_z by doubling all edges and short-cutting.
  Add the tour to $\mathscr{T}$.
end
Clean-up phase:
Remove all facilities from $F_1 \cup F_2$ that are not on any of the tours in $\mathscr{T}$.
return $(F_1 \cup F_2, \mathscr{T})$;

Lemma 2.4. *The solution to CLR constructed by Algorithm 1 in the large demand and merge phases from the spanning tree S and the UFL solution U has cost at most* $2c'(S) + 2\tilde{c}(U) + \phi(U)$.

Proof. Every tour constructed in the large demand phase for a client $v \in \mathscr{C}$ has cost at most $2\lceil \frac{d_v}{u} \rceil c_{vw(v)}$, where $w(v)$ is a closest open facility in U. This is bounded by twice the connection cost for v in the UFL solution as $2\lceil \frac{d_v}{u} \rceil c_{vw(v)} \leq 2 \cdot 2\frac{d_v}{u} c_{vw(v)} \leq 2\tilde{c}_{vw(v)} d_v$, because $\frac{d_v}{u} \geq 1$.

Consider a tour T constructed during an iteration of the inner "for" loop of the merge phase in Algorithm 1. The cost of the tour is at most twice the cost of the edges of the corresponding subtree plus $2c_{wz'}$. Observe that, by the choice of w and z', the edge $\{w, z'\}$ is at most as expensive

as any other edge used in U to connect any of the clients x on the tour to its facility $y(x)$. As the sum of the demands on the tour is at least $\frac{u}{2}$, we obtain

$$\sum_{x \in V(T)} \tilde{c}_{xy(x)} d_x \geq \tilde{c}_{wz'} \sum_{x \in V(T)} d_x \geq \tilde{c}_{wz'} \frac{u}{2} = c_{wz'}.$$

Thus, the total cost of all tours constructed in the inner loop amounts to at most twice the connection cost of U plus twice the costs of the corresponding subtrees. The tours constructed in the outer loop and the opening costs of all facilities in F_2 are bounded by twice the costs of the remaining subtrees S_z (w.r.t. c'), and the opening costs of all facilities in F_1 are $\phi(U)$. As all subtrees are pairwise disjoint, summing everything up yields the desired result. □

Consequently, if S is a minimum spanning tree and U is a ρ-approximation to a minimum cost solution to the UFL instance, the merge phase of Algorithm 1 returns an $(2+2\rho)$-approximation to CLR. Note, however, that in this analysis $\phi(U)$ is counted twice while the actual solution only pays it once. We can improve the approximation factor by using a bifactor approximation algorithm for UFL of Byrka and Aardal (2010). Given a parameter $\gamma > 1.678$, this algorithm returns a solution whose opening cost exceeds the opening cost of an initially computed optimal fractional LP solution U_{LP} by at most a factor of γ, and whose connection cost exceeds the connection cost of the fractional solution by at most $1+2e^{-\gamma}$. In this way, we obtain a solution U with $2\tilde{c}(U)+\phi(U) \leq 2(1+2e^{-\gamma})\tilde{c}(U_{\mathrm{LP}})+\gamma\phi(U_{\mathrm{LP}})$, which is bounded by $\gamma(\tilde{c}(U_{\mathrm{LP}})+\phi(U_{\mathrm{LP}}))$ for all $\gamma \geq 2.38$. Choosing $\gamma = 2.38$, Lemma 2.4 yields our main result.

Theorem 2.5. *Algorithm 1 is a* 4.38*-approximation algorithm for CLR (Problem 2.1). The solution it produces fulfills the single-assignment property. If $d_v \leq u$ for all $v \in \mathscr{C}$, it furthermore fulfills the single-tour property.*

On the other hand, the approximation ratio of our algorithm improves naturally for classes of instances that allow a better UFL approximation. One example are graphs with Euclidean edge cost. Here, a PTAS for UFL (Arora et al., 1998) can be applied to obtain a $(4+\varepsilon)$-approximation for CLR.

2.4 Special Case: Multi-Depot Capacitated Vehicle Routing

The special case of CLR, where opening facilities does not incur cost ($\phi \equiv 0$) is the *multi-depot capacitated vehicle routing problem* (MDCVR) as considered in Li and Simchi-Levi (1990); Cardon et al. (2008). By a slight modification of Algorithm 1, we obtain an improved approximation ratio for this problem: Instead of solving the UFL instance approximately in the UFL phase, we solve it exactly by opening all facilities and assigning clients to facilities along shortest client-facility paths. We thus can replace the factors incurred by the bifactor UFL-algorithm by 1 and obtain the following result.

Theorem 2.6. *When solving the UFL instance by shortest path computation, Algorithm 1 is a* 4*-approximation algorithm for MDCVR. The solution it produces fulfills the single-assignment property. If $d_v \leq u$ for all $v \in \mathscr{C}$, it furthermore fulfills the single-tour property.*

Note that this improves the previously best known approximation guarantee of 5 for MDCVR in Li and Simchi-Levi (1990) yielding the single-assignment property.

3 Prize-Collecting Location Routing

We now apply our algorithmic framework for CLR and MDCVR to the *prize-collecting* (PC) variant of these problems. In a prize-collecting setting, we can decide for each client whether to serve it by our solution, or to pay a penalty for not serving it. Note that prize-collecting can naturally be viewed as a way of incorporating outsourcing decisions into an optimization model: In this case, a customer's penalty corresponds to the cost of having it served by an outside service provider. As outsourcing is an important option in many logistics applications, the prize-collecting variants of CLR and MDCVR are highly relevant in practice. Moreover, it is not hard to see that PC-CLR and PC-MDCVR are generalization of CLR and MDCVR, respectively: By setting penalties high enough, we can force any optimal solution to serve all clients.

Formally, an instance of PC-CLR comprises an instance of CLR together with a penalty function $p : \mathscr{C} \to \mathbb{R}^+$, and a solution is now a three-tuple $(F, \mathscr{T}, C)$, where $F \subseteq \mathscr{F}$ is a set of open facilities as before, $C \subseteq \mathscr{C}$ is the set of clients served, and $\mathscr{T}$ is a set of tours as before, except that we require only the demands of clients in C to be served by $\mathscr{T}$. The cost of a solution to PC-CLR is $\sum_{T\in\mathscr{T}} c(T) + \sum_{w\in F} \phi(w) + \sum_{v\in\mathscr{C}\setminus C} p(v)$. As before, PC-MDCVR is the special case of PC-CLR where $\phi \equiv 0$.

Problem 3.1 (Prize-Collecting Capacitated Location Routing)**.**

Input: a graph $G = (\mathscr{C} \cup \mathscr{F}, E)$, metric edge costs $c : E \to \mathbb{R}^+$, opening costs $\phi : \mathscr{F} \to \mathbb{R}^+$, demands $d : \mathscr{C} \to \mathbb{R}^+$, vehicle capacity $u \in \mathbb{R}^+$, penalties $p : \mathscr{C} \to \mathbb{R}^+$

Task: Find a set of facilities $F \subseteq \mathscr{F}$, a set of clients $C \subseteq \mathscr{C}$, and a set of closed walks $\mathscr{T}$ with a demand assignment $x : C \times \mathscr{T} \to \mathbb{R}^+$ such that

(1) $V(T) \cap F \neq \emptyset$ for all $T \in \mathscr{T}$,

(2) $\sum_{T\in\mathscr{T}: v\in V(T)} x_{vT} = d_v$ for all $v \in C$,

(3) $\sum_{v\in C} x_{vT} \leq u$ for all $T \in \mathscr{T}$,

minimizing the cost $\sum_{w\in F} \phi(w) + \sum_{T\in\mathscr{T}} \sum_{e\in T} c_e + \sum_{v\in\mathscr{C}\setminus C} p(v)$.

3.1 Algorithm

The key challenge in solving the prize-collecting variant by our algorithm lies in the choice of C: On the one hand, both our solution to UFL (Lemma 2.2) and our spanning tree (Lemma 2.3) need to serve the same set of clients in order for our rerouting procedure to work. On the other hand, we need to ensure that the sum of the costs of these partial solutions remains a lower bound for the original problem. We accomplish this by utilizing an approximation algorithm for PC-UFL, and an LP-based approximation algorithm for the prize-collecting Steiner tree to determine two respective sets of customers served. We then compute a solution to PC-CLR serving exactly those customers served by both the tree and the facility location solution.

A formal description of the algorithm is given in Algorithm 2. We will prove that it is a $(\rho_{\text{PC-ST}} + 2\rho_{\text{PC-UFL}})$-approximation algorithm for PC-CLR, where $\rho_{\text{PC-ST}}$ and $\rho_{\text{PC-UFL}}$ denote the approximation factors of the approximation algorithms used for prize-collecting Steiner tree (w.r.t. the undirected cut relaxation) and PC-UFL, respectively. Currently, the best known ap-

proximation algorithm for PC-UFL achieves an approximation ratio of $\rho_{\text{PC-UFL}} = 2$ (Jain et al., 2003), while for prize-collecting Steiner tree the algorithm of Goemans and Williamson (1995) achieves an approximation factor of $2 - \frac{1}{|V|}$, meeting the integrality gap of the LP relaxation. Using these algorithms results in an approximation factor of 6 for our algorithm.

Algorithm 2: Algorithm for PC-CLR.

Input: An instance of PC-CLR.
Output: A feasible solution to PC-CLR.
UFL phase:
Create a UFL instance as in the UFL phase of Algorithm 1. Add p to obtain an instance of PC-UFL.
Run an approximation algorithm for PC-UFL. Let U be the returned UFL solution and C_1 denote the set of served clients and F_1 be the set of opened facilities in U.
Steiner tree phase:
Construct the graph G' as in Lemma 2.3.
Run an approximation algorithm for prize-collecting Steiner tree on the instance given by G', the terminal set $\mathscr{C} \cup \{r\}$ and penalties p. Let S be the resulting tree and C_2 denote set of connected customers and F_2 be the set of connected facilities in the Steiner tree.
Merge phase:
Set $C := C_1 \cap C_2$.
Run the large demand and merge phases of Algorithm 1 using U and S, serving only clients in C. Let $\mathscr{T}$ be the resulting set of tours.
return $(F_1 \cup F_2, \mathscr{T}, C)$.

First note that an equivalent of Lemma 2.2 still holds in a prize-collecting setting: In its proof, we constructed a feasible solutions to a scaled instance of UFL from any feasible solution to CLR without increasing cost. It is easy to see that this construction adapts naturally when transferring the set of clients served from an optimal PC-CLR solution to a feasible solutions to PC-UFL: The penalties for customers not served are exactly the same in both solutions.

To obtain the second, tree based lower bound, we consider a prize-collecting Steiner tree instance defined as follows. We add a root node r to the network and connect it to all facilities, i.e., we consider the graph $G' = (V \cup \{r\}, E \cup E')$ with $E' = \{\{r,w\} : w \in \mathscr{F}\}$ as constructed in Lemma 2.3. We then extend the cost function c to E' by defining cost $c_{rw} = \frac{1}{2}\phi_w$ for each $w \in \mathscr{F}$ and define new penalties by setting $p' := \frac{1}{2}p$. We let $R = \mathscr{C} \cup \{r\}$ be the set of terminals. We will use an approximation algorithm on this prize-collecting Steiner tree instance that is based on the following undirected cut relaxation.

$$
\begin{aligned}
& \min \quad \sum_{e \in E \cup E'} c(e)y(e) + \sum_{N \subseteq \mathscr{C}} \Big(\sum_{v \in N} p'(v)\Big) z(N) \\
(\text{PC-ST}_{\text{LP}}) \qquad & \text{s.t.} \sum_{e \in \delta_{G'}(S)} y(e) + \sum_{N \subseteq \mathscr{C}:\, S \cap \mathscr{C} \subseteq N} z(N) \geq 1 \qquad \forall S \subseteq V,\ S \cap \mathscr{C} \neq \emptyset \\
& \qquad y \geq 0
\end{aligned}
$$

Here, $\delta_{G'}(S)$ denotes the cut in G' induced by the vertex set S, i.e., the set of all edges of G' that have one endpoint in S and one endpoint outside of S. The intuition for the LP formulation is the following: Given a feasible solution to prize-collecting Steiner tree, define $z(N) = 1$ for the set N of clients that are not connected to the Steiner tree, and $z(N) = 0$ for all other sets of clients. Moreover, set $y(e) = 1$ if edge e is in the Steiner tree, $y(e) = 0$ otherwise. The inequalities follow from the fact that any cut that separates a served terminal from the root has

to be crossed by at least one edge of the tree.

Lemma 3.2. $\text{OPT}(\textit{PC-ST}_{LP}) \leq \frac{1}{2}\textit{OPT}(\textit{PC-CLR})$

Proof. Let $(F, \mathscr{T}, C)$ be an optimal solution to PC-CLR. Construct a solution $(\tilde{z}, \tilde{y})$ to PC-ST_{LP} by setting $\tilde{z}(N^* := \mathscr{C} \setminus C) = 1$ and $\tilde{z}(N) = 0$ for all other $N \subseteq \mathscr{C}$, and $\tilde{y}(\{r,w\}) = 1$ for all $w \in F$, $\tilde{y}(\{r,w\}) = 0$ for all $w \in \mathscr{F} \setminus F$, and $\tilde{y}(e) = \frac{1}{2}|\{T \in \mathscr{T} : e \in E(T)\}|$. It is easy to observe that the constructed solution $(\tilde{y}, \tilde{z})$ has cost $\frac{1}{2}\sum_{w \in F} \phi(w) + \frac{1}{2}\sum_{v \in \mathscr{C} \setminus C} p(v) + \frac{1}{2}\sum_{T \in \mathscr{T}} c(T)$.

It remains to show that $(\tilde{z}, \tilde{y})$ is feasible for PC-ST_{LP}. So let S denote an arbitrary subset of V with $S \cap \mathscr{C} \neq \emptyset$. If S contains an open facility w, then $\{r,w\} \in \delta_{G'}(S)$, and by definition of $\tilde{y}$, the constraint for S is fulfilled. Else, if $S \cap C = \emptyset$, then S contains only unserved clients and the set $\{N \subseteq \mathscr{C} : S \cap \mathscr{C} \subseteq N\}$ contains N^*. Hence, by definition of $\tilde{z}$, the constraint for S is satisfied as well. Finally, if S does not contain an open facility and $S \cap C \neq \emptyset$, then there is a a client $v \in C \cap S$ connected to an open facility outside of S by a tour. At least two edges of this tour lie in the cut $\delta_{G'}(S)$, hence the constraint for S is again satisfied by definition of $\tilde{y}$. □

Theorem 3.3. *Using the algorithm of Goemans and Williamson (1995) in its Steiner tree phase, Algorithm 2 is a* $(2 + 2\rho_{\textit{PC-UFL}})$*-approximation algorithm for PC-CLR (Problem 3.1). The solution it produces fulfills the single-assignment property. If* $d_v \leq u$ *for all* $v \in \mathscr{C}$*, it furthermore fulfills the single-tour property.*

Proof. Since the algorithm uses the large demand and merge phases of Algorithm 1, the claims of the theorem regarding single-assignment and single-tour properties follow directly from Theorem 2.5.

Moreover, by Lemma 2.4, the cost of the solution returned in the merge phase is bounded by

$$\begin{aligned} 2c(S) + 2\tilde{c}(U) + \phi(U) + \sum_{v \in \mathscr{C} \setminus C} p(v) &\leq 2c(S) + \sum_{v \in \mathscr{C} \setminus C_1} 2p'(v) + 2\tilde{c}(U) + \phi(U) + \sum_{v \in \mathscr{C} \setminus C_2} p(v) \\ &\leq 2 \cdot 2\,\text{OPT}(\text{PC-ST}_{\text{LP}}) + 2\rho_{\text{PC-UFL}} \cdot \text{OPT}(\text{PC-UFL}) \\ &\leq (2 + 2\rho_{\text{PC-UFL}})\,\text{OPT}, \end{aligned}$$

where the second to last inequality stems from the fact that the algorithm by Goemans and Williamson (1995) is a 2-approximation, and the last from Lemma 3.2. □

Similar to Section 2.4, we can replace the algorithm for PC-UFL by shortest path computations for the case of PC-MDCVR, which solve this subproblem to optimality: A client is connected to a facility if and only if the shortest path distance to its closest facility is no greater than its penalty. This yields an improved approximation ratio for PC-MDCVR.

Theorem 3.4. *For the case* $\phi \equiv 0$*, PC-UFL can be solved exactly by shortest path computations. Thereby, Algorithm 2 becomes a* 4*-approximation algorithm for PC-MDCVR. The solution it produces fulfills the single-assignment property. If* $d_v \leq u$ *for all* $v \in \mathscr{C}$*, it furthermore fulfills the single-tour property.*

4 Group Location Routing

We now consider a group version of location routing (G-CLR) where the set of clients is partitioned into groups $\mathscr{C}_1, \ldots, \mathscr{C}_k$, with $\mathscr{C} = \bigcup_{i=1}^{k} \mathscr{C}_i$ and only one client from each group needs to be served. The uncapacitated version of this problem was studied by Glicksman and Penn (2008), who give a $(2 - \frac{1}{|V|-1})L$-approximation algorithm with L being the cardinality of the largest

group. Their idea is to solve an LP relaxation of the problem and use the resulting fractional solution to decide which client is to be served from each group. We extend this approach to the capacitated case which is significantly more complex: In the absence of vehicle capacities, facility opening costs can be transferred to edges of the graph, i.e. location routing is equivalent to multi-depot vehicle routing in this case. In contrast to Glicksman and Penn (2008), our LP relaxation has to explicitly incorporate the facility location aspect of the problem.

The dependence of our approximation factor on the parameter L gives rise to the question whether there is a constant factor approximation algorithm for G-CLR that is independent of any parameters in the input. At the end of this section, we answer this question in the negative by showing that there is no $o(\log(k))$-approxmation algorithm for G-CLR.

Problem 4.1 (Group Capacitated Location Routing)**.**

Input: a graph $G = (\mathscr{C} \cup \mathscr{F}, E)$, metric edge costs $c : E \to \mathbb{R}^+$, opening costs $\phi : \mathscr{F} \to \mathbb{R}^+$, demands $d : \mathscr{C} \to \mathbb{R}^+$, vehicle capacity $u \in \mathbb{R}^+$, a partition $\mathscr{C}_1, \dots, \mathscr{C}_k$ of $\mathscr{C}$

Task: Find a set of facilities $F \subseteq \mathscr{F}$, a set of clients $C \subseteq \mathscr{C}$ and a set of closed walks $\mathscr{T}$ with a demand assignment $x : C \times \mathscr{T} \to \mathbb{R}^+$ such that

(1) $V(T) \cap F \neq \emptyset$ for all $T \in \mathscr{T}$,

(2) $\sum_{T \in \mathscr{T} : v \in V(T)} x_{vT} = d_v$ for all $v \in C$,

(3) $\sum_{v \in \mathscr{C}} x_{vT} \leq u$ for all $T \in \mathscr{T}$,

(4) $C \cap \mathscr{C}_i \neq \emptyset$ for all $i \in \{1, \dots, k\}$,

minimizing the cost $\sum_{w \in F} \phi(w) + \sum_{T \in \mathscr{T}} \sum_{e \in T} c_e$.

4.1 LP relaxation

In order to obtain an approximation for G-CLR, we describe how to transform a solution of G-CLR into a multi-commodity flow variable assignment on the arcs and vertices of a directed graph. We then prove a set of valid inequalities fulfilled by all assignments obtained from feasible G-CLR solutions. The LP relaxation resulting from these inequalities can be used to decide on a set of representatives, one for each client group. Replacing each group by its representative, we obtain an instance of (non-group) CLR which can be approximated by an adaption of Algorithm 1 with the spanning tree replaced by a Steiner tree. We will show that the resulting solution to G-CLR is a $4.38L$-approximation.

While the problem remains based on an undirected graph, it is more convenient to consider its directed equivalent in our LP relaxation: We replace each undirected edge e by two oppositely directed arcs a_e^+ and a_e^- with costs $c(a_e^+) = c(a_e^-) = c(e)$ and denote the set of all such arcs by A. We start constructing a multi-commodity flow on the edges in A from a given (undirected) solution of G-CLR by fixing an arbitrary orientation for every tour. Let $y(a)$ be the number of tours using arc $a \in A$. Let $T_{v \leftarrow w}(a)$ and $T_{v \to w}(a)$ be the index sets of all tours that serve client $v \in \mathscr{C}$ from facility $w \in \mathscr{F}$ with an occurrence of arc $a \in A$ on the path from w to v or, respectively, from v to w. Accordingly, define variables $x_{v \leftarrow w}(a) = \sum_{i \in T_{v \leftarrow w}(a)} x_{vi}$ and $x_{v \to w}(a) = \sum_{i \in T_{v \to w}(a)} x_{vi}$ for all arcs, where the x_{vi} are the demand assignments introduced at the beginning of Section 2. Finally, for each facility $w \in \mathscr{F}$, let $z(w) = 1$ if w is open and $z(w) = 0$ otherwise.

The values $x_{v\leftarrow w}(a)$ and $x_{v\rightarrow w}(a)$ can be interpreted as multi-commodity flow with two commodities $v \leftarrow w$ and $v \rightarrow w$ for each pair $v \in \mathscr{C}$ and $w \in \mathscr{F}$, respectively. The first commodity corresponds to goods transported from facility w to client v, the second commodity $v \rightarrow w$ emulates the empty truck capacity on the tour returning from v to w. We define the flow balance of node $v \in V$ with respect to commodity $h \in \{v \leftarrow w, v \rightarrow w \ : \ v \in \mathscr{C}, w \in \mathscr{F}\}$ as $b_h(v) := \sum_{a\in\delta^+(v)} x_h(a) - \sum_{a\in\delta^-(v)} x_h(a)$.

First observe that the total amount of flow on any arc can at most be the capacity u times the number of tours using the arc, i.e.,

$$\sum_{v\in\mathscr{C}} \sum_{w\in\mathscr{F}} (x_{v\leftarrow w}(a) + x_{v\rightarrow w}(a)) \leq u y(a) \quad \forall a \in A. \tag{1}$$

Furthermore, we obtain

$$\sum_{v\in\mathscr{C}_i} \sum_{w\in\mathscr{F}} \tfrac{1}{d_v}(x_{v\leftarrow w}(a) + x_{v\rightarrow w}(a)) \leq y(a) \quad \forall a \in A, i \in \{1,\dots,k\} \tag{2}$$

by observing that the left hand side of the equation is at most 1 per tour that is using the arc: Only one client v in a group is served, only d_v units are transported to this client in total, and in any tour, each arc occurs either before or after v but never both.

By construction of x, flow conservation holds for each commodity at all nodes that neither correspond to its facility nor to its client. Furthermore, at clients $v \in \mathscr{C}$, the value of any commodity $v \rightarrow w$ for some $w \in \mathscr{F}$ leaving the client equals the value of $v \leftarrow w$ entering it:

$$b_{v\rightarrow w}(p) = 0 = b_{v\leftarrow w}(p) \qquad \forall v \in \mathscr{C}, w \in \mathscr{F}, p \in V \setminus \{v,w\} \tag{3}$$

$$b_{v\leftarrow w}(v) = -b_{v\rightarrow w}(v) = b_{v\rightarrow w}(w) = -b_{v\leftarrow w}(w) \qquad \forall v \in \mathscr{C}, w \in \mathscr{F} \tag{4}$$

Moreover, as one client from every group needs to be served, the variables fulfill

$$\sum_{v\in\mathscr{C}_i} \sum_{w\in\mathscr{F}} \tfrac{1}{d_v} b_{w\rightarrow v}(v) = 1 \qquad \forall i \in \{1,\dots,k\}. \tag{5}$$

Finally, at most d_v units of flow are sent from an open facility to client v and thus

$$\sum_{v\in\mathscr{C}_i} \tfrac{1}{d_v} b_{v\leftarrow w}(v) \leq z_w \quad \forall w \in \mathscr{F}, i \in \{1,\dots,k\}. \tag{6}$$

We conclude that the value of an optimal solution to the group location routing problem is at least the value of an optimal solution of the following LP.

$$(\text{G-CLR}_{\text{LP}}) \qquad \begin{aligned} \min \quad & \sum_{a\in A} c(a)y(a) + \sum_{w\in\mathscr{F}} \phi_w z_w \\ \text{s.t.} \quad & x,y,z \text{ fulfill } (1)-(6) \\ & x,y,z \geq 0 \end{aligned}$$

Let (x^*, y^*, z^*) be an optimal solution to G-CLR$_{\text{LP}}$. For $i \in \{1,\dots,k\}$, let $r_i \in \mathscr{C}_i$ be a client with $\sum_{w\in\mathscr{F}} \frac{b^*_{v\leftarrow w}}{d_v}$ maximum over all $v \in \mathscr{C}_i$. We now define the set of group representatives as $R := \{r_1,\dots,r_k\}$. The following inequality will be useful for deriving lower bounds on OPT(G-CLR$_{\text{LP}}$).

Lemma 4.2. *Let $L := \max\{|\mathscr{C}_i| : i \in \{1,\dots,k\}\}$. Then $L \cdot \sum_{w\in\mathscr{F}} b^*_{r_i\leftarrow w} \geq d_{r_i}$ for all $i \in \{1,\dots,k\}$.*

Proof. By (5), in each group C_i, there has to be at least one client $v \in C_i$ with $\sum_{w\in\mathscr{F}} \frac{b^*_{v\leftarrow w}}{d_v} \geq \frac{1}{L}$, and thus this inequality holds for r_i in particular. □

Now denote the instance of (non-group) CLR defined by the set of representatives R by CLR(R). Consider the following LP relaxation for the uncapacitated facility location problem arising from CLR(R) as described in Lemma 2.2. We will use it to derive a lower bound on the value of an optimal solution to G-CLR$_{\text{LP}}$.

$$
\begin{aligned}
(\text{UFL}_{\text{LP}}(R)) \qquad & \min \quad \sum_{v\in R}\sum_{w\in\mathscr{F}} \tilde{c}_{vw}x_{vw} + \sum_{w\in\mathscr{F}} \phi_w z_w & \\
& \text{s.t.} \quad \sum_{w\in\mathscr{F}} x_{vw} \geq d_v & \forall v \in R \\
& \quad \tfrac{1}{d_v} x_{vw} \leq z_w & \forall v \in R, w \in \mathscr{F} \\
& \quad x, z \geq 0 &
\end{aligned}
$$

Lemma 4.3. $\text{OPT}(UFL_{LP}(R)) \leq L \cdot \text{OPT}(G\text{-}CLR_{LP})$.

Proof. Consider the solution $(\tilde{x},\tilde{z})$ to UFL$_{\text{LP}}(R)$ obtained by setting $\tilde{z}_w = L \cdot z^*_w$ and $\tilde{x}_{vw} = L \cdot b^*_{v\leftarrow w}(w)$ for all $v \in R, w \in \mathscr{F}$. Observe that by Lemma 4.2, we have for each representative r_i

$$\sum_{w\in\mathscr{F}} \tilde{x}_{r_i w} = L \cdot \sum_{w\in\mathscr{F}} b^*_{r_i\leftarrow w}(r_i) \geq d_{r_i}.$$

Together with (6), this immediately implies that $(\tilde{x},\tilde{z})$ is a feasible solution to UFL$_{\text{LP}}$. The flow of each commodity $v \leftarrow w$ ($v \rightarrow w$) can be decomposed into flow on v-w-paths (w-v-paths), each of which has at length at least c_{vw} by triangle inequality. Combining this with (1), we obtain

$$
\begin{aligned}
\sum_{a\in A} c(a)y^*(a) &\geq \sum_{a\in A} \tfrac{c_a}{u} \sum_{v\in\mathscr{C}}\sum_{w\in\mathscr{F}} (x^*_{v\leftarrow w}(a) + x^*_{v\rightarrow w}(a)) \geq \sum_{v\in\mathscr{C}}\sum_{w\in\mathscr{F}} \tfrac{2}{u} c_{vw} b^*_{v\leftarrow w}(v) \\
&\geq \tfrac{1}{L} \cdot \sum_{v\in R}\sum_{w\in\mathscr{F}} \tfrac{2}{u} c_{vw}\tilde{x}_{vw} = \tfrac{1}{L} \cdot \sum_{v\in R}\sum_{w\in\mathscr{F}} \tilde{c}_{vw}\tilde{x}_{vw}.
\end{aligned}
$$

Furthermore, $L \cdot \sum_{w\in\mathscr{F}} \phi_w z^*_w = \sum_{w\in\mathscr{F}} \phi_w \tilde{z}_w$ by construction, which implies $\text{OPT}(\text{UFL}_{\text{LP}}) \leq L \cdot \text{OPT}(\text{G-CLR}_{\text{LP}})$. □

A second lower bound can be obtained from the LP relaxation of a Steiner tree instance defined similar to that in Section 3. Again, we consider the graph $G' = (V \cup \{r\}, E \cup E')$ with $E' = \{\{r,w\} : w \in \mathscr{F}\}$ as constructed in Lemma 2.3. We then extend the cost function c to E' by defining cost $c_{rw} = \frac{1}{2}\phi_w$ for each $w \in \mathscr{F}$. We now consider the undirected cut relaxation of the Steiner tree instance on G' with terminals $R \cup \{r\}$.

$$
\begin{aligned}
(\text{ST}_{\text{LP}}(R)) \qquad & \min \quad \sum_{e\in E\cup E'} c(e)y(e) & \\
& \text{s.t.} \quad \sum_{e\in\delta_{G'}(S)} y(e) \geq 1 & \forall S \subseteq V,\ S \cap R \neq \emptyset \\
& \quad y \geq 0 &
\end{aligned}
$$

Lemma 4.4. $\text{OPT}(ST_{LP}(R)) \leq \frac{1}{2} L \cdot \text{OPT}(G\text{-}CLR_{LP})$

Proof. Consider the solution $\tilde{y}$ to ST$_{\text{LP}}(R)$ obtained by setting $\tilde{y}(\{v,w\}) = \frac{1}{2}L \cdot (y^*(vw) + y^*(wv))$ for all $v,w \in V$ and $\tilde{y}(\{r,w\}) = L \cdot z^*_w$ for all $w \in \mathscr{F}$. Let $S \subseteq V$ with $r_i \in S$ for some $i \in \{1,\dots k\}$. By flow conservation and (6) we obtain

$$\sum_{a\in\delta^+(S)} x^*_{r_i\rightarrow w}(a) + \sum_{w\in S} d_{r_i} z_w \geq b^*_{r_i\rightarrow w}(r_i) \ \text{ and } \sum_{a\in\delta^-(S)} x^*_{r_i\leftarrow w}(a) + \sum_{w\in S} d_{r_i} z^*_w \geq b^*_{r_i\rightarrow w}(r_i),$$

where $\delta^+(S) = \{vw \in A : v \in S, w \in V \setminus S\}$ and $\delta^-(S) = \{vw \in A : v \in V \setminus S, w \in S\}$. By construction of $\tilde{y}$ and inequality (2) we obtain

$$\begin{aligned}\sum_{e \in \delta_{G'}(S)} \tilde{y}(e) &= \frac{1}{2} L \left(\sum_{a \in \delta^+(S)} y^*(a) + \sum_{a \in \delta^-(S)} y^*(a) \right) + \sum_{w \in S} z^*_w \\ &\geq \frac{L}{2d_{r_i}} \left(\sum_{a \in \delta^+(S)} x^*_{r_i \to w}(a) + \sum_{a \in \delta^-(S)} x^*_{r_i \leftarrow w}(a) + 2 \cdot \sum_{w \in S} d_{r_i} z^*_w \right) \\ &\geq \tfrac{L}{d_{r_i}} \cdot b^*_{r_i \to w}(r_i).\end{aligned}$$

The last expression is at least 1 by Lemma 4.2. Thus, $\tilde{y}$ is a feasible solution to $\mathrm{ST}_{\mathrm{LP}}(R)$ with $\sum_{e \in E \cup E'} c(e)\tilde{y}(e) = \frac{1}{2} L (\sum_{a \in A} c(a) y^*(a) + \sum_{w \in \mathscr{F}} \phi_w) = \frac{1}{2} L \cdot \mathrm{OPT}(\text{G-CLR}_{\mathrm{LP}})$. □

Remark 4.5. The LP relaxation presented in this section also yields an alternative proof of the minimum spanning tree lower bound in Lemma 2.3 for the non-group case, using the bidirected cut formulation of the spanning tree polytope. However, the direct and combinatorial proof of Lemma 2.3 given in Section 2.1 appears to be more intuitive and elegant.

4.2 Algorithm

Lemma 4.3 and Lemma 4.4 immediately lead to a 4.38L-approximation algorithm for G-CLR: Compute an optimal solution to G-CLR$_{\mathrm{LP}}$, obtain a set of representatives R from this solution and compute an approximation to the resulting instance CLR(R) with Algorithm 1, using an LP-based Steiner tree 2-approximation algorithm instead of a minimum spanning tree computation.

Algorithm 3: Algorithm for GCLR.

Input: An instance of G-CLR.
Output: A feasible solution to G-CLR.
Compute an optimal solution (x^*, y^*, z^*) to G-CLR$_{\mathrm{LP}}$.
forall the $i \in \{1, \ldots, k\}$ **do**
  Let $r_i \in \mathscr{C}_i$ be a client with $\sum_{w \in \mathscr{F}} \frac{b^*_{v \leftarrow w}}{d_v}$ maximum over all $v \in \mathscr{C}_i$.
  $R = R \cup \{r_i\}$
end
Construct the graph G' with extended edge costs $c_{rw} = \frac{1}{2}\phi_w$ for $w \in \mathscr{F}$.
Apply the algorithm of Goemans and Williamson to obtain a Steiner tree S with terminal set $R \cup \{r\}$ in G'.
Apply Algorithm 1 on the instance CLR(R) with the minimum spanning tree computed in the tree phase replaced by the Steiner tree S. Return the computed solution.

Theorem 4.6. *Algorithm 3 is a* 4.38L*-approximation for G-CLR (Problem 4.1). There is a* 4L*-approximation for G-MDCVR.*

Proof. The cost of the Steiner tree computed by the algorithm of Goemans and Williamson (1995) is a at most $2 \cdot \mathrm{OPT}(\mathrm{ST}_{\mathrm{LP}}(R))$. The UFL solution U computed in Algorithm 1 approximates the opening cost of an optimal solution to $\mathrm{UFL}_{\mathrm{LP}}(R)$ by γ, and its connection cost by $(1+2e^{-\gamma})$, because the LP relaxation is equivalent to the one used in the algorithm of Byrka and Aardal (2010). Thus, Lemmas 4.3 and 4.4 yield $2c(S) + 2\tilde{c}(U) + \phi(U) \leq 2 \cdot \mathrm{OPT}(\mathrm{ST}_{\mathrm{LP}}(R)) + \gamma \cdot \mathrm{OPT}(\mathrm{UFL}_{\mathrm{LP}}(R)) \leq 4.38L \cdot \mathrm{OPT}(\mathrm{GCLR})$. □

4.3 Lower bound on the approximability

Observing that the approximation guarantee of Algorithm 3 depends on the cardinality of the largest group, it is natural to ask whether the group version of CLR is indeed considerably harder than the standard version or whether there is a constant factor approximation whose performance is independent of any instance parameters. We answer this question negatively by showing that there is no approximation algorithm for G-CLR with a factor better than $\mathcal{O}(\log(k))$.

In fact, the inapproximability result already holds for the special case of G-CLR with unit demands and unit capacity, which corresponds to the group version of metric uncapacitated facility location (G-UFL), as well as for the uncapacitated case considered in Glicksman and Penn (2008). It is derived by a straightforward reduction from unweighted set cover.

Proposition 4.7. *There exists a constant $\alpha > 0$ such that there is no $\alpha \log(k)$-approximation for G-UFL, unless $P = NP$.*

Proof. We reduce the unweighted set cover problem, for which the same $\log(n)$-approximability-threshold has been proven by Feige (1998), to G-UFL. An instance of unweighted set cover consists of a ground set H and a set system $\mathscr{S} \subseteq 2^H$ together with costs c_S for every $S \in \mathscr{S}$. The task is to choose a subset $\mathscr{S}'$ of $\mathscr{S}$ such that every element of the ground set is covered, i.e., $\bigcup_{S \in \mathscr{S}'} S = H$, while minimizing the total cost $\sum_{S \in \mathscr{S}'} c_S$.

We create a G-UFL instance by introducing a facility w_S for each $S \in \mathscr{S}$ and setting $\phi(w_S) := c_S$. For every $h \in H$ and every $S \in \mathscr{S}$ with $h \in S$ we introduce a client v_{hS}. We also introduce a client group $\mathscr{C}_h$ for each element $h \in H$ of the ground set and let it contain all clients v_{hS}. Finally, we set $c_{w_S, v_{hS'}} = 0$, whenever $S = S'$, and to ∞ otherwise.

Note that any feasible solution to this G-UFL instance with finite costs corresponds to a feasible solution to set cover with the same costs, by selecting the sets corresponding to open facilities. As for every client group there is an open facility with connection cost 0 to one of its members, every set is covered. Likewise, every feasible solution to set cover induces a solution to G-UFL by opening the facilities corresponding to the chosen sets. Since every element of the ground set is covered, for every client group there is a member that has connection cost 0 to an open facility. Thus, any γ-approximation for G-UFL (or the group version of UFL) immediately implies a γ-approximation for set cover. Choosing the same α as used in Feige (1998) for set cover, we conclude that there is no $\alpha \log(k)$-approximation for G-UFL (unless $P = NP$), since this would imply a $\alpha \log(|H|)$-approximation for set cover (note that $|H|$ is the number of groups in the constructed G–UFL instance). □

Corollary 4.8. *There exists a constant $\alpha > 0$ such that there is no $\alpha \log(k)$-approximation for G-CLR (even if $u = \infty$), unless $P = NP$.*

Proof. As G-UFL is a special case of G-CLR, the inapproximability also holds for the latter. The reduction also works if $u = \infty$, as connection costs are either 0 or ∞ and so serving all clients at a facility on one tour instead of serving them separately does not change the costs. □

5 Location Routing with Cross-Docking

A major trade-off in classic vehicle routing applications is good capacity utilization versus low cost of each tour conducted. Especially in applications where clients with small demands are located far away from facilities, significant cost savings can be realized by allowing *consolidation tours*. In such a tour, a vehicle is positioned at a client node to collect goods from other vehicles passing through. Then, it starts on its own tour to distribute the goods collected. Essentially, the demand of a tour of clients is consolidated at one node and forwarded to facilities via

other tours from there. The necessitated process of loading goods from one vehicle to another at a client node is commonly referred to as *cross-docking*. The example in Figure 1 shows that cross-docking may indeed lead to cost savings.

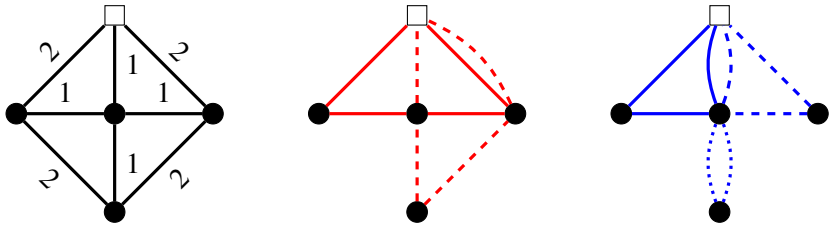

Figure 1: A CLR instance with $u = 5$. The numbers on the edges indicate the edge costs. The demand at the central client is 1, the demand at the other clients is 3. The optimal routing scheme in Solution 1 without cross-docking has total cost 12. The routing scheme in Solution 2 uses cross-docking to consolidate the tours at the central vertex. Its total cost is 10.

Formally, a *solution with cross-docking* to CLR is again a tuple $(F, \mathscr{T})$, where $F \subseteq \mathscr{F}$ is a set of open facilities and $\mathscr{T}$ is a set of tours, but $\mathscr{T} = \mathscr{T}_\mathrm{F} \dot\cup \mathscr{T}_\mathrm{H}$ is now partitioned into a set of *facility tours* $\mathscr{T}_\mathrm{F}$ and a set of *consolidation tours* $\mathscr{T}_\mathrm{H}$. We now require that (1a) every facility tour visits an open facility, (1b) in every consolidation tour $T \in \mathscr{T}_\mathrm{H}$, exactly one client $h \in V(T)$ is designated as the *hub* consolidating the hub-demand $d_T^h := \sum_{v \in \mathscr{C}} x_{vT}$ of all clients served by the tour, (2) the demand of every client (including the additional demand occurring if the client is the hub of one or more consolidation tours) is served by the tours by which it is visited, and (3) the demand served by a tour does not exceed u. More precisely, there are non-negative values x_{vT} such that $\sum_{T \in \mathscr{T}_\mathrm{F} : v \in V(T)} x_{vT} = d_v + \sum_{T \in \mathscr{T}_\mathrm{H} : v = h_T} d_T^h$ for all $v \in \mathscr{C}$, and $\sum_{v \in \mathscr{C}} x_{vT} \leq u$ for all $T \in \mathscr{T}$. We say that a solution fulfills the *single-vehicle-to-client* property if each client's demand arrives on a single vehicle, which can be important in practice. Note that, in contrast to the single-tour property, single-vehicle-to-client deliveries still can be split up on earlier segments of the transportation route or even originate from distinct facilities.

Problem 5.1 (Capacitated Location Routing with Cross-Docking)**.**

Input: a graph $G = (\mathscr{C} \cup \mathscr{F}, E)$, metric edge costs $c : E \to \mathbb{R}^+$, opening costs $\phi : \mathscr{F} \to \mathbb{R}^+$, demands $d : \mathscr{C} \to \mathbb{R}^+$, vehicle capacity $u \in \mathbb{R}^+$

Task: Find a set of facilities $F \subseteq \mathscr{F}$ and a set of closed walks $\mathscr{T}$ partitioned into facility tours $\mathscr{T}_\mathrm{F}$ and consolidation tours $\mathscr{T}_\mathrm{H}$, with a demand assignment $x : \mathscr{C} \times \mathscr{T} \to \mathbb{R}^+$ such that

(1a) $V(T) \cap F \neq \emptyset$ for all $T \in \mathscr{T}_\mathrm{F}$,

(1b) each consolidation tour $T \in \mathscr{T}_\mathrm{H}$ has a dedicated hub $h_T \in V(T)$,

(2) $\sum_{T \in \mathscr{T}_\mathrm{F} : v \in V(T)} x_{vT} = d_v + \sum_{T \in \mathscr{T}_\mathrm{H} : v = h_T} \sum_{v \in \mathscr{C}} x_{vT}$ for all $v \in \mathscr{C}$,

(3) $\sum_{v \in \mathscr{C}} x_{vT} \leq u$ for all $T \in \mathscr{T}$,

minimizing the cost $\sum_{w \in F} \phi(w) + \sum_{T \in \mathscr{T}} \sum_{e \in T} c_e$.

5.1 Algorithm

We now describe how to adapt Algorithm 1 in order to allow for cross-docking. As before, we start by computing a solution to a UFL instance as defined in Lemma 2.2 and a minimum spanning tree for the modified graph G' as defined in Lemma 2.3. Then, we modify our rerouting procedure as stated formally in Algorithm 4.

Algorithm 4: Algorithm for CLR with cross-docking.

Input: An instance of CLR.
Output: A feasible solution to CLR with cross-docking.
UFL phase:
Create an UFL instance with edge costs $\tilde{c} = \frac{2}{u}c$ as described in Lemma 2.2.
Apply the 1.5-approximation algorithm of Byrka and Aardal on this instance and let F_1 be the set of facilities opened in the resulting UFL solution.
Run tree and large demand phase of Algorithm 1.
Merge phase:
forall the $z \in F_2$ **do**
  while $D_z > u$ **do**
    Let $v \in V(S_z)$ such that $D_v > u$ but $D_w \leq u$ for all children w of v.
    Let $I = \{V(S_w) : w \text{ is a child of } v\} \cup \{\{v\}\}$.
    For every $R \in I$ find a pair (v_R, z_R) such that $v_R \in V(R)$ and $z_R \in F_1 \cup F_2$ and $c_{v_R z_R}$ is minimal.
    Order the sets in I non-decreasingly by $c_{v_R z_R}$ and include the first $\left\lfloor \frac{D_v}{u} \right\rfloor$ sets in I_t.
    Let $I_\mathrm{s} := I \setminus I_\mathrm{t}$.
    forall the $R \in I_t$ **do**
      Construct a tour visiting z_R, v and all vertices in R by adding $v_R z_R$ to the tree and then doubling edges and short-cutting.
      Add the tour to $\mathscr{T}_\mathrm{F}$ and remove the subtrees corresponding to the elements of R from S.
    end
    forall the $R \in I_s$ **do**
      Construct a tour visiting v and all vertices in R by doubling edges and short-cutting.
      Add the tour to $\mathscr{T}_\mathrm{H}$ with hub v and remove the subtrees corresponding to the elements of R from S.
    end
  end
  Construct a tour from S_z by doubling all edges and short-cutting.
  Add the tour to $\mathscr{T}_\mathrm{F}$.
end
Run clean-up phase of Algorithm 1.

As in Section 2.2, we consider a node v with $D_v > u$ but $D_w \leq u$ for all children w of v and let I be the set containing all subtrees S_w, with w being a child of v, and $\{v\}$ itself. For each of these sets $R \in I$, we determine a node v_R with cheapest connection cost $c_{v_R z_R}$ to an open facility z_R. We order the sets $R \in I$ non-decreasingly by $c_{v_R z_R}$ and define the set of *sink trees* I_t as the first $\left\lfloor \frac{D_v}{u} \right\rfloor$ elements of I. The remaining elements $I \setminus I_\mathrm{t}$ comprise the set of *source trees* I_s. Each sink tree $R \in I_t$ is turned into a facility tour by doubling edges and inserting the open facility closest to v_R, paying at most twice the tree edges plus the connection cost of v_R to its facility. Each source tree is turned into a consolidation tour with hub v by doubling the edges and short-cutting.

Note that by this construction, each facility tour visits v. Hence, any spare capacity on a facility tour can be filled by hub demands ensuing at v from consolidation tours. Furthermore, the sum of all demands that cannot be served by the facility tours constructed is strictly less than u.

5.2 Analysis

We first point out that our lower bounds from Section 2.1 remain valid when allowing cross-docking. These results can easily be obtained by slight modification of the corresponding proofs of Lemma 2.2 and Lemma 2.3, respectively.

Lemma 5.2. *Consider a UFL instance as defined in Lemma 2.2. The cost of its optimal solution (w.r.t. $\tilde{c}$) is at most the cost of an optimal solution to CLR with cross-docking (w.r.t. c).*

Proof. Consider an optimal solution $(F, \mathscr{T})$ of CLR with cross-docking and demand assignments x_{vT}. As in the proof of Lemma 2.2, we can construct a flow f from the CLR solution as follows. For every client $v \in \mathscr{C}$ and each tour $T \in \mathscr{T}$ serving v, partition T into two paths from the facility or hub of the tour to the client and send x_{vT} units of flow along either path. Since flow is sent along two paths for every client/tour pair and all hub demand is forwarded along further tours to facilities, the net flow transported to any client $v \in \mathscr{C}$ equals $2d_v$. We can thus apply flow decomposition on f to obtain a set of client-facility paths $\mathscr{P}_\mathrm{P}$ and cycles $\mathscr{P}_\mathrm{C}$, respectively, with corresponding flow values f_P for every $P \in \mathscr{P}_\mathrm{P} \cup \mathscr{P}_\mathrm{C}$. On this flow decomposition, we can apply the same arguments as in the proof of Lemma 2.2. □

Lemma 5.3. *The cost of a minimum spanning tree in the graph G' w.r.t. costs c' as defined in Lemma 2.3 is a lower bound on the cost of an optimal solution to CLR with cross-docking (w.r.t. c).*

Proof. Let $(F, \mathscr{T})$ be a feasible solution to CLR with cross-docking. Note that $S = \bigcup_{T \in \mathscr{T}} T \cup E'$ spans all vertices of the graph G' since for every client there is a path from a facility to this client along edges used in the tours. We can modify S such that it spans the graph G' but every facility opened in the CLR solution is incident to at most one edge in E by applying the technique from the proof of Lemma 2.3 on the set of facility tours. This set contains a spanning tree of cost at most the cost of the CLR solution. □

It turns out that guaranteeing demand u on each of the tours constructed in the rerouting procedure yields an improved approximation guarantee for the merge phase of our algorithm.

Lemma 5.4. *The merge phase of Algorithm 4 constructs a solution to CLR with cross-docking with cost at most $2c'(S) + \tilde{c}(U) + \phi(U)$ from the spanning tree S and the UFL solution U.*

Proof. Observe that each facility tour constructed in the inner loop serves a total demand of u. Thus, the central inequality in the proof Lemma 2.4 changes to $\sum_{x \in V(T)} \tilde{c}_{xy(x)} d_x \geq 2c_{wz'}$. Accordingly, the connection cost of the UFL solution is paid only once. □

Intuitively, the improved bound in Lemma 5.4 arises from the tight capacity utilization of vehicles that are paid for by the UFL solution. We immediately obtain a better approximation guarantee for Algorithm 4 when using the 1.5-approximation of Byrka and Aardal (2010) for constructing the UFL solution U.

Theorem 5.5. *Algorithm 4 is a 3.5-approximation algorithm for CLR with cross-docking (Problem 5.1). If $d_v \leq u$ for all $v \in \mathscr{C}$, the obtained solution satisfies the single-vehicle-to-client property.*

Again, in the case of MDCVR with $\phi \equiv 0$, we can apply shortest path computations to solve the UFL instance exactly.

Theorem 5.6. *When solving the UFL instance by shortest path computation, Algorithm 4 is a* 3*-approximation algorithm for MDCVR with cross-docking. If $d_v \leq u$ for all $v \in \mathscr{C}$, the obtained solution satisfies the single-vehicle-to-client property.*

We remark that, while Algorithm 1 produces a solution without cross-docking, its approximation factor still holds for the case where cross-docking is allowed as all lower bounds used in Theorem 2.5 remain valid. Thus, we obtain the following bounds on the improvements realizable by cross-docking in CLR and MDCVR.

Corollary 5.7.

1. *Algorithm 1 is a* 4.38*-approximation for CLR with cross-docking and a* 4*-approximation for MDCVR with cross-docking. The produced solution fulfills the single-assignment property. If $d_v \leq u$ for all $v \in \mathscr{C}$, it fulfills the single-tour property.*

2. *The value of an optimal solution for CLR without cross-docking is at most* 4.38 *times the value of a solution with cross-docking. The value of an optimal solution for MDCVR without cross-docking is at most* 4 *times the value of a solution with cross-docking.*

We close this section by observing that the validity of the lower bounds extend to the cases of prize-collecting as well as group location routing with cross-docking. We can thus combine the merge phase of Algorithm 4 with the modifications introduced in Algorithm 2 for PC-CLR and Algorithm 3 for G-CLR, respectively.

Theorem 5.8. *There is a $(2+\rho_{PC\text{-}UFL})$-approximation algorithm for PC-CLR with cross-docking. There is a* 3*-approximation algorithm for PC-MDCVR with cross-docking.*

Theorem 5.9. *There is a* 3.5*L-approximation algorithm for G-CLR with cross-docking. There is a* 3*L-approximation algorithm for G-MDCVR with cross-docking.*

6 Computational Study

In Section 2, we have proven that our polynomial time algorithm for CLR is guaranteed to compute solutions which are at most 4.38 times as expensive as the optimum. In this section, we shall see that the algorithm's performance in practice exceeds this theoretical worst-case estimate by far. We would like to emphasize that we do not expect our algorithm to compete with (meta-)heuristic approaches without an approximation guarantee. Rather, the question addressed in this computational study is how much solution quality on typical instances needs to be sacrificed in exchange for polynomial running time and a worst case performance guarantee across all instances.

For our experiments, we implemented Algorithm 1 with the following minor modifications: First, instead of using the bifactor approximation algorithm of Byrka and Aardal in the UFL phase, we implemented the greedy approximation algorithm of Jain et al. (2003). While the latter has a slightly worse approximation guarantee of 1.861, it is purely combinatorial, avoiding randomization and linear programming, and far easier to implement. In this context, note that although the instances in our study are equipped with Euclidian distances, we do not apply the PTAS of Arora et al. (1998) as it is not tailored for practical use in regards of running time. Moreover, before applying Prim's algorithm (see e.g. Cormen et al. (2001)) in the tree phase,

we set the opening costs of all facilities opened in the UFL phase to zero; doing so turns out to yield slightly improved results, while it does not interfere with our theoretical analysis of the algorithm. Finally, once the algorithm has computed all tours, we added an option to improve each single tour by solving the corresponding travelling salesman problem (TSP) using LKH, an implementation of the Lin-Kernighan heuristic described in Helsgaun (2000).

Fact 6.1. *Our implementation of Algorithm 1 has an approximation guarantee of* 5.722.

Fact 6.2. *The running time of our implementation of Algorithm 1 is* $\mathcal{O}(n^2m)$*, where n and m denote the number of clients and facilities, respectively.*

Fact 6.1 results directly from Lemma 2.4 and the approximation factor of the greedy algorithm used in the UFL phase. The running time of the implementation is dominated by that of the UFL phase, cf. Jain et al. (2003). Moreover, experiments in Helsgaun (2000) indicate that the practical running time of LKH is quite low (close to quadratic). Our study supports this observation, as the additional running time when employing the option for a-posteriori tour optimization by LKH turns out to be small, immeasurable on moderately sized instances.

We report results for two different sets of instances: The first, referred to as the benchmark set, comprises 45 instances appearing frequently in the location routing literature, see the references appearing below. Here, we compare our results with those obtained by recent (meta-)heuristic algorithms as well as best known solutions (bks) from the literature. While the benchmark instances are moderate in size (20–200 clients, 5–20 facilities), our second test set consists of 27 randomly generated instances which are considerably larger (up to 10000 clients and 1000 facilities). Our implementation was done in C++ using GCC 4.5 under SUSE Linux 11.3, and all computations were conducted on an Intel Core2 Duo E8400 processor at 3GHz with 4GB RAM.

6.1 Benchmark instances

Key properties of the benchmark instances used are listed in Table 1. The first 36 instances were introduced in Tuzun and Burke (1999), the last nine in Barreto et al. (2007); we will refer to them as sets *TB* and *B*, respectively. While set *TB* is adopted as-is, our set *B* contains only those instances introduced in Barreto et al. (2007) which do not have a capacity limit on facilities, as only those mirror the location routing problem addressed here.

The best known solution values reported for *TB* were obtained in Prins et al. (2007). For *B*, some proven optima were already reported in Barreto et al. (2007), while the remaining instances were solved to proven optimality in Baldacci et al. (2009), as reported in Contardo et al. (2010).

Table 2 contains gaps to bks and cpu times for our implementation of Algorithm 1, with and without a-posteriori optimization of tours using LKH, compared to those of four other algorithms for CLR: a greedy randomized adaptive search procedure (GRASP) proposed in Prins et al. (2006); a Lagrangean relaxation granular tabu search (LRGTS) developed in Prins et al. (2007); a two-phase tabu search (TS) studied in Tuzun and Burke (1999); and finally an exact branch-and-cut-and-price approach (BCP) proposed in Baldacci et al. (2009). Results for algorithms GRASP and LRGTS are stated in Prins et al. (2007) for all 45 benchmark instances, while results for TS and BPS are only available in the corresponding works for the instances in *TB* and *B*, respectively.

Please note that our algorithms, GRASP and LRGTS, TS, and BCP were tested on different machines, so the cpu times stated should not be compared directly. Since all tests were performed on modern desktop computers, however, we do believe that a comparison of the magnitudes of running times remains feasible.

name	#facilities	#clients	∅ demand	vehicle capacity	bks value
111112	10	100	15.17	150	1468.40
111122	20	100	15.00	150	1449.20
111212	10	100	14.39	150	1396.46
111222	20	100	15.19	150	1432.29
112112	10	100	15.28	150	1167.53
112122	20	100	14.32	150	1102.70
112212	10	100	15.06	150	793.97
112222	20	100	14.73	150	728.30
113112	10	100	14.81	150	1238.49
113122	20	100	15.10	150	1246.34
113212	10	100	14.73	150	902.38
113222	20	100	14.78	150	1021.31
121112	10	200	14.95	150	2281.78
121122	20	200	15.15	150	2185.55
121212	10	200	14.81	150	2234.78
121222	20	200	14.94	150	2259.52
122112	10	200	15.24	150	2101.90
122122	20	200	14.47	150	1709.56
122212	10	200	14.69	150	1467.54
122222	20	200	15.21	150	1084.78
123112	10	200	15.13	150	1973.28
123122	20	200	14.66	150	1957.23
123212	10	200	15.09	150	1771.06
123222	20	200	15.29	150	1393.62
131112	10	150	14.79	150	1866.75
131122	20	150	14.93	150	1841.86
131212	10	150	15.02	150	1981.37
131222	20	150	14.71	150	1809.25
132112	10	150	14.95	150	1448.27
132122	20	150	14.75	150	1444.25
132212	10	150	14.91	150	1206.73
132222	20	150	15.15	150	931.94
133112	10	150	14.95	150	1699.92
133122	20	150	14.93	150	1401.82
133212	10	150	15.18	150	1199.51
133222	20	150	14.91	150	1152.86
Chr69-100x10	10	100	14.58	200	842.90*
Chr69-50x5	5	50	15.54	160	565.60*
Chr69-75x10	10	75	18.19	160	861.60*
Gas67-22x5	5	22	463.14	4500	585.11*
Gas67-29x5	5	29	439.66	4500	512.10*
Gas67-32x5	5	32	917.81	8000	562.20*
Gas67-32x5-2	5	32	917.81	11000	504.30*
Gas67-36x5	5	36	25.00	250	460.40*
Min92-27x5	5	27	311.48	2500	3062.00*

Table 1: Properties of benchmark instances and cost of a best known solution (bks, * denotes proven optimality). The bks values for the first 36 instances are from Prins et al. (2007), those for the last nine from a series of papers by Baldacci et al. (2009), Barreto et al. (2007), and Tuzun and Burke (1999).

instance	approx		approx+tsp		GRASP		LRGTS		TS/BCP	
	gap	cpu	gap	cpu	gap	cpu	gap	cpu	gap	cpu
111112	0.207	0.00	0.079	0.00	0.039	32.40	0.015	3.30	0.060	6.01
111122	0.235	0.00	0.117	0.00	0.054	40.70	0.016	6.50	0.057	5.71
111212	0.133	0.00	0.043	0.00	0.019	27.60	0.011	4.20	0.034	3.36
111222	0.342	0.00	0.246	0.00	0.035	36.20	0.008	7.40	0.055	5.52
112112	0.164	0.00	0.076	0.00	0.028	27.70	0.017	6.90	0.054	5.45
112122	0.133	0.00	0.095	0.01	0.019	34.30	0.012	6.80	0.027	2.66
112212	0.086	0.00	0.041	0.00	0.025	22.50	0.024	5.20	0.039	3.92
112222	0.119	0.00	0.070	0.00	0.027	37.30	0.020	5.90	0.017	1.68
113112	0.183	0.00	0.090	0.00	0.028	21.50	0.024	4.30	0.063	6.34
113122	0.201	0.00	0.131	0.00	0.021	36.00	0.008	6.30	0.023	2.26
113212	0.140	0.00	0.082	0.00	0.011	20.30	0.012	4.00	0.020	2.04
113222	0.166	0.00	0.126	0.00	0.004	38.40	0.007	4.90	0.023	2.05
131112	0.253	0.01	0.142	0.01	0.075	113.00	0.042	12.50	0.072	7.19
131122	0.230	0.01	0.110	0.01	0.026	161.40	0.018	18.50	0.028	2.77
131212	0.153	0.00	0.067	0.01	0.027	100.00	0.015	11.10	0.021	2.06
131222	0.206	0.01	0.102	0.01	0.026	132.40	0.006	15.80	0.025	2.53
132112	0.163	0.01	0.081	0.01	0.041	117.70	0.000	22.00	0.074	7.43
132122	0.301	0.01	0.230	0.02	0.009	166.10	0.034	28.00	0.024	2.39
132212	0.101	0.01	0.050	0.00	0.028	106.70	0.004	14.60	0.020	2.04
132222	0.170	0.00	0.123	0.01	0.010	142.40	0.005	13.70	0.018	1.75
133112	0.155	0.01	0.098	0.00	0.022	92.80	0.017	17.90	0.037	3.68
133122	0.127	0.01	0.075	0.01	0.017	128.40	0.016	18.50	0.062	6.17
133212	0.128	0.00	0.068	0.01	0.020	88.50	0.014	14.50	0.054	5.43
133222	0.081	0.00	0.029	0.01	0.068	134.90	0.008	14.30	0.026	2.55
121112	0.217	0.01	0.145	0.01	0.055	308.00	0.016	32.60	0.053	4.28
121122	0.139	0.01	0.050	0.02	0.047	410.00	0.010	39.60	0.012	1.20
121212	0.191	0.01	0.105	0.02	0.017	311.40	0.012	32.80	0.024	2.39
121222	0.225	0.02	0.122	0.02	0.042	418.90	0.004	40.20	0.047	4.26
122112	0.145	0.02	0.088	0.02	0.017	338.00	0.009	47.20	0.027	2.70
122122	0.179	0.02	0.125	0.02	0.057	370.00	0.017	59.30	0.045	4.53
122212	0.107	0.01	0.050	0.01	0.020	242.70	0.014	36.70	0.056	5.60
122222	0.119	0.01	0.049	0.00	0.010	308.50	0.005	38.70	0.026	2.60
123112	0.170	0.01	0.081	0.01	0.036	282.80	0.005	41.60	0.042	4.20
123122	0.126	0.01	0.050	0.02	0.068	399.20	0.015	51.80	0.023	2.31
123212	0.183	0.02	0.146	0.02	0.010	199.00	0.009	34.00	0.060	6.00
123222	0.182	0.01	0.134	0.01	0.011	296.30	0.005	43.20	0.015	1.52
Chr69-100x10*	0.283	0.00	0.108	0.00	0.022	25.50	0.000	28.20	0.000	13074.7
Chr69-50x5*	0.220	0.00	0.079	0.00	0.059	2.30	0.037	2.40	0.000	112.9
Chr69-75x10*	0.177	0.00	0.104	0.00	0.000	9.80	0.002	10.10	0.000	3413.5
Gas67-22x5*	0.244	0.00	0.021	0.00	0.000	0.20	0.004	0.20	0.000	6.0
Gas67-29x5*	0.279	0.00	0.165	0.00	0.006	0.40	0.000	0.40	0.000	178.2
Gas67-32x5*	0.245	0.00	0.179	0.00	0.017	0.60	0.040	0.60	0.000	63.4
Gas67-32x5-2*	0.205	0.00	0.123	0.00	0.000	0.50	0.001	0.50	0.000	117.9
Gas67-36x5*	0.448	0.00	0.094	0.00	0.000	0.80	0.035	0.70	0.000	2.9
Min92-27x5*	0.181	0.00	0.115	0.00	0.000	0.40	0.001	0.30	0.000	47.0
∅	**0.188**	0.01	**0.100**	0.01	**0.026**	128.54	**0.013**	17.96	**0.038** **0.000**	3.74 1890.69

Table 2: Gaps to best known solution (bks) and cpu times for various algorithms on benchmark instances (* signifies proven optimality of bks). Results for algorithm TS are only available for the first 36 instances, those for BCP only for the last nine, hence they share a column. The last row contains average values, with those for TS (first 36 instances) and BCP (last nine) one above the other.

On average, our approximation algorithm delivers solutions with cost about 19% above the bks value. This figure improves to 10% when LKH is used to optimize tours a-posteriori. Moreover, the running time of our algorithm is negligible on these instances, regardless of whether LKH is used or not. In comparison, the (meta-)heuristic algorithms GRASP, LRGTS, and TS compute solutions with objective 1–4% above that of bks on average, while their running times vary strongly from 1–7 seconds (TS) to up to 7 minutes (GRASP). The exact approach (BCP) is able to find optimal solutions for all instances in B, while its running time is naturally very high (up to several hours).

Since gaps to bks for GRASP and LRGTS are no greater for the instances in TB than for those in B, where optimality has been proven, it seems reasonable to assume that the gap between bks and an optimum solution is generally small. In this case, our algorithm vastly outperforms its theoretical approximation guarantee of 5.722. When employing a simple post-optimization step using LKH, it yields solutions within a factor of 1.25 of bks on all instances, within 1.1 on average. Moreover, its polynomial running time is reflected in very small cpu times on these benchmark instances. When compared to (meta-)heuristic algorithms, solution quality suffers only by a single-digit percentage on average, while cpu times are improved by several magnitudes. Moreover, recall that this improvement in running time comes in addition to the advantage of having a guarantee on solution quality across all possible instances, including malicious examples where (meta-)heuristics might perform very poorly. In light of its extremely fast running time, our algorithm can also be used to compute feasible start solutions for other search heuristics.

6.2 Larger, randomly generated instances

The extremely fast running time of our algorithm on benchmark instances, which are all of moderate size, suggests that our algorithm is suitable for larger instances as well. To the best of our knowledge, no instances of CLR which are significantly larger than those in the benchmark set have been solved in the literature; hence, we generated a random test set from three input parameters: size, facility opening cost, and vehicle capacity.

Instances were generated on three base networks of different sizes: M (1000 clients, 100 facilities), L (5000, 500), and XL (10000, 1000). Facility opening costs were drawn uniformly at random from three different ranges: $[0;100]$, $[100;200]$, and $[200;500]$. Vehicle capacities were set to either 9, 100, or 1000, while client demands were drawn uniformly at random from $[0;10]$ in all cases. Finally, x- and y-coordinates for clients and facilities were drawn uniformly at random from $[0;100]$, and Euclidean distances $d(i,j) := \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}$ are used in all instances. Our approach of generating the random instances is similar to the approach of Tuzun and Burke (1999), except that we did not use clustering. The experimental design, using the same base network with different parameters, allows us to compare the effects of these parameters on solution structure, performance of the algorithm, and quality of the lower bounds derived from MST and UFL subproblems, respectively.

All possible combinations of the three input parameters yield 27 different instances, which we name by their size, indexed with their choice of facility opening cost and vehicle capacity. E.g., $M_{2,2}$ is an instance with 1000 clients, 100 facilities, facility opening costs in $[100;200]$, and vehicle capacity 100.

Key properties of the solutions computed by our algorithm, again with and without LKH, together with cpu times are depicted in Table 3. The column 'lower bound' denotes the better of the two lower bounds arising from the UFL and MST instances as described in Lemmas 2.2 and 2.3. While the minimum spanning tree computed within the algorithm is optimal and can thus be directly used as lower bound, deriving a reasonable UFL lower bound requires more

name	lower bound	#open fac.	fac. cost	#tours	approx			approx+tsp		
					cost	gap	cpu	cost	gap	cpu
$M_{1,1}$	8800.8^{O}	33	779.4	757	13563.4	0.541	0.47	13478.9	0.532	0.55
$M_{1,2}$	41249.5^{O}	13	82	58	4111.6	0.961	0.95	3499.05	0.669	1.00
$M_{1,3}$	2096.3^{T}	8	31.8	10	3343.6	0.595	1.59	2478.9	0.183	1.65
$M_{2,1}$	12166.6^{O}	18	2157.9	756	18086.2	0.487	0.53	17997	0.479	0.62
$M_{2,2}$	2288.8^{O}	5	528.2	55	5098.9	1.228	0.92	4468.74	0.952	0.98
$M_{2,3}$	2151.6^{T}	1	205.2	6	3520.2	0.636	1.62	2620.84	0.218	1.67
$M_{3,1}$	15432.2^{O}	10	2370.3	756	23008.8	0.491	0.68	22926.8	0.486	0.76
$M_{3,2}$	2938.7^{O}	3	869.1	55	6012	1.046	1.27	5345.92	0.819	1.32
$M_{3,3}$	2203.4^{T}	1	414.3	6	3656.8	0.66	0.83	2779.25	0.261	0.88
$L_{1,1}$	17502^{D}	128	2426.4	3695	32473	0.855	23.39	32325.9	0.847	35.90
$L_{1,2}$	4607^{T}	47	337.3	272	9433.5	1.048	50.84	8106.1	0.76	59.29
$L_{1,3}$	4607^{T}	16	42.1	31	7344.7	0.594	120.12	5463.71	0.186	121.95
$L_{2,1}$	29519.6^{D}	50	6000.5	3694	50380.6	0.707	28.89	50229.7	0.702	41.35
$L_{2,2}$	5946.5^{D}	10	1163.8	271	13435.5	1.259	65.79	12059.5	1.028	73.81
$L_{2,3}$	4659.4^{T}	2	306	29	8477	0.819	162.99	6624.78	0.422	165.08
$L_{3,1}$	38728.3^{D}	31	7293.4	3694	64058.9	0.654	38.3	63905.1	0.65	50.94
$L_{3,2}$	7515.9^{D}	6	1473	271	15694.9	1.088	89.8	14372.4	0.912	97.81
$L_{3,3}$	4709.4^{T}	1	409.3	29	8835.3	0.876	210.71	6966.54	0.479	213.44
$XL_{1,1}$	25449.7^{D}	229	3394.8	7480	48879.5	0.921	136.48	48677.1	0.913	214.23
$XL_{1,2}$	6494.6^{T}	78	405.1	554	13752.3	1.117	314.81	11872	0.828	369.47
$XL_{1,3}$	6494.6^{T}	33	52.7	69	10400	0.601	741.08	7754.66	0.194	749.91
$XL_{2,1}$	46601.8^{D}	82	9264.9	7473	77796.3	0.669	165.57	77580.6	0.665	243.40
$XL_{2,2}$	9253.7^{D}	17	1752.7	547	20133.7	1.176	383.13	18159.1	0.962	434.58
$XL_{2,3}$	6550.3^{T}	4	507.8	57	12018.2	0.835	879.48	9296.1	0.419	886.83
$XL_{3,1}$	60461.3^{D}	48	10593.1	7473	101676	0.682	228.14	101454	0.678	307.12
$XL_{3,2}$	11838.1^{D}	11	2255.2	547	23304.5	0.969	518.15	21341.5	0.803	570.46
$XL_{3,3}$	6600.5^{T}	2	610.2	57	13091.1	0.983	1314.86	10389.9	0.574	1322.69
$\varnothing$	14328.43	32.85	2063.94	1433.41	22651.35	0.833	203.01	21562	0.616	221.03

Table 3: Best known lower bounds (T: MST, O: optimal UFL solution, D: dual UFL solution), solution properties, costs, gaps and cpu times of the approximation algorithm with and without tsp post-optimization for random instances.

care, as the UFL solutions used in the algorithm are only approximations: For smaller instances (size M), we computed the optimal solution value of the corresponding UFL instances using the mixed integer programming solver CPLEX 12.1 (IBM Corp., 2009). For the instances of size L and XL, where using a MIP solver was not possible, we derived a lower bound by constructing a dual solution from the client bids occuring in the UFL greedy algorithm by Jain et al. (2003).

Cpu time for the largest instances is at most about twenty minutes. On average, using LKH to optimize tours a-posteriori reduces total cost by about 5%, while increasing cpu time by roughly 10%. Naturally, the effect of using LKH on both solution quality and cpu time is more significant when vehicle capacity is large (i.e., tours are long). Regarding the lower bounds, we observe that the MST yields stronger bounds for larger vehicle capacities, while the UFL bound is stronger when vehicle capacities are small. On average, our algorithm shows a gap of 61.6% to the corresponding lower bounds when LKH post-optimization is enabled.

While we do not expect the lower bounds to be very close to the optimum solution values, we do not have any other primal solutions to compare with our results on instances of similar size. However, we encourage the authors of other algorithms for CLR to perform experiments on our random test set, which are available for download at `http://www.coga.tu-berlin.de/clrlib`, and compare their results to ours.

7 Summary & Outlook

Approximation algorithms combine efficient running times with provable a priori guarantees on solution quality. We applied this concept to several versions of capacitated location routing problems, which extend classical vehicle routing problems by depot location decisions. Variants of our algorithms also yield improved approximation guarantees for multi-depot capacitated vehicle routing.

We constructed a 4.38-approximation algorithm for capacitated location routing with arbitrary client demands, the first constant-factor approximation known for this problem. For the case of multi-depot capacitated vehicle routing, our algorithm improves the best known approximation ratio from 5 to 4. We then extended our algorithms to practically relevant generalizations of these problems, namely a prize-collecting version with penalties for non-served clients, and a group version, where one client from each group needs to be chosen. In all three cases, a variant where cross-docking is allowed leads to better approximation factors. All algorithms in our framework are based on computing an uncapacitated solution via a minimum spanning tree or Steiner tree, and rerouting excess demand according to the solution of a scaled facility location problem (or along shortest paths for multi-depot capacitated vehicle routing).

Finally, we demonstrated in a computational study that our algorithm for CLR is also of practical relevance. Our computational experiments revealed that the actual solution quality achieved by our algorithm is much closer to optimality than suggested by the theoretical bounds. On a benchmark set of instances from the literature, our algorithm for CLR computes solutions with cost within a factor of 1.1–1.2 of best known solutions on average. Moreover, we demonstrated that our algorithm is extremely fast, running in negligible time on benchmark instances. Thus, it might be a valuable tool for solving large-scale problems.

A further investigation of the algorithms in this paper would be of practical as well as theoretical interest: Given its fast running time, using its solution as a starting point in local search frameworks might lead to improved results of those heuristics without a significant increase in running time. Further experiments could be conducted on extended location routing models, e.g., with capacities on facilities, or heterogeneous vehicle fleets. Although the theoretical approximation guarantee might be lost in these cases, the algorithm could be adapted to still be an

efficient heuristic for those problems.

On the theoretical side, it might be possible to sharpen our analysis and prove stronger theoretical approximation guarantees. Moreover, our paper does not address the issue of lower bounds on the possible approximation factor for basic capacitated location routing. It is easy to see that it cannot be approximated better than by a factor of 1.5 (unless $P = NP$), which is the best known lower bound for approximating a single-depot vehicle routing problem with uniform vehicle capacities (Golden and Wong, 1981). However, an analysis that takes into account both the location and routing aspects of the problem might lead to stronger inapproximability results.

Moreover, our algorithms strongly rely on the technique of tree-to-tour-conversion, thereby incurring an additional factor of 2 in their approximation ratios. It would be interesting to find out if a more tour-specific approach, e.g., the tour partitioning techniques widely used for vehicle routing problems (Li and Simchi-Levi, 1990), could lead to better approximation factors.

The cross-docking model considered in this work assumes that the cost of actual cross-docking operations is negligible, and the operations can be performed at arbitrary client nodes. Deriving approximation algorithms for the case where cross-docking operations incur cost and are restricted to certain cross-docking facilities is an open problem. In Section 5, we point out that the possible improvement due to cross-docking is bounded by a factor of at most 4.38. We suspect the actual bound to be much smaller and leave its determination as a further open question for future research.

Acknowledgements: We would like to thank two anonymous referees, the guest editor, and the editor in chief for motivating us to conduct the computational study in Section 6.

References

Arora, S., P. Raghavan, S. Rao. 1998. Approximation schemes for Euclidean k-medians and related problems. *STOC '98*. Association for Computing Machinery, New York, NY, 106–113.

Baldacci, R., A. Mingozzi, R. Wolfer-Calvo. 2009. The capacitated location routing problem. Presented at ROUTE 2009, Rolighed, Denmark.

Baldacci, R., P. Toth, D. Vigo. 2010. Exact algorithms for routing problems under vehicle capacity constraints. *Annals of Operations Research* **175**(1) 213–245.

Barreto, S., C. Ferreira, J. Paixão, B. S. Santos. 2007. Using clustering analysis in a capacitated location-routing problem. *European Journal of Operational Research* **179**(3) 968 – 977.

Belenguer, J.-M., E. Benavent, C. Prins, C. Prodhon, R. W. Calvo. 2011. A branch-and-cut method for the capacitated location-routing problem. *Computers & OR* **38**(6) 931–941.

Bräysy, O., M. Gendreau. 2005. Vehicle routing problem with time windows, part I: Route construction and local search algorithms. *Transportation Science* **39**(1) 104–118.

Byrka, J., K. Aardal. 2010. An optimal bifactor approximation algorithm for the metric uncapacitated facility location problem. *SIAM Journal on Computing* **39**(6) 2212–2231.

Cardon, S., E. Dommers, C. Eksin, R. Sitters, L. Stougie. 2008. A PTAS for the multiple depot vehicle routing problem. SPOR Report 2008-03, Technische Universiteit Eindhoven, Eindhoven.

Charikar, M., S. Khuller, D. M. Mount, G. Narasimhan. 2001a. Algorithms for facility location problems with outliers. *SODA '01*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 642–651.

Charikar, M., S. Khuller, B. Raghavachari. 2001b. Algorithms for capacitated vehicle routing. *SIAM Journal on Computing* **31**(3) 665–682.

Chen, X., B. Chen. 2009. Cost-effective designs of fault-tolerant access networks in communication systems. *Networks* **53**(4) 382–391.

Christofides, N. 1976. Worst-case analysis of a new heuristic for the travelling salesman problem. Report 388, Graduate School of Industrial Administration, Carnegie Mellon University, Pittsburgh, PA.

Chudak, F. A., D. B. Shmoys. 2003. Improved approximation algorithms for the uncapacitated facility location problem. *SIAM Journal on Computing* **33**(1) 1–25.

Contardo, C., J.-F. Cordeau, B. Gendron. 2010. A branch-and-cut algorithm for the capactiated location problem. Unpublished manuscript.

Cordeau, J. F., M. Gendreau, G. Laporte, J. Y. Potvin, F. Semet. 2002. A guide to vehicle routing heuristics. *Journal of the Operational Research Society* **53** 512–522.

Cordeau, J. F., G. Laporte, M. W. P. Savelsbergh, D. Vigo. 2007. Vehicle routing. C. Barnhart, G. Laporte, eds., *Transportation*, vol. 14 of *Handbooks in Operations Research and Management Science*, Elsevier, Amsterdam. 367–428.

Cormen, T. H., C. E. Leiserson, R. L. Rivest, C. Stein. 2001. *Introduction to Algorithms*. The MIT Press, New York.

Desrochers, M., J. K. Lenstra, M. W. P. Savelsbergh, F. Soumis. 1988. Vehicle routing with time windows: Optimization and approximation. B. L. Golden, A. A. Assad, eds., *Vehicle Routing: Methods and Studies*, North-Holland, Amsterdam. 65–84.

Feige, U. 1998. A threshold of ln n for approximating set cover. *Journal of the ACM (JACM)* **45**(4) 634–652.

Glicksman, H., M. Penn. 2008. Approximation algorithms for group prize-collecting and location-routing problems. *Discrete Applied Mathematics* **156**(17) 3238–3247.

Goemans, M. X., D. P. Williamson. 1995. A general approximation technique for constrained forest problems. *SIAM Journal on Computing* **24**(2) 296–317.

Golden, B., R. Wong. 1981. Capacitated arc routing problems. *Networks* **11**(3) 305–315.

Golden, B. L., A. A. Assad, eds. 1988. *Vehicle Routing: Methods and Studies*. North-Holland, Amsterdam.

Haimovich, M., A. H. G. Rinnoy Kan. 1985. Bounds and heuristics for capacitated routing problems. *Mathematics of Operations Research* **10**(4) 527–542.

Haimovich, M., A. H. G. Rinnoy Kan, L. Stougie. 1988. Analysis of heuristics for vehicle routing problems. B. L. Golden, A. A. Assad, eds., *Vehicle Routing: Methods and Studies*, North-Holland, Amsterdam. 47–61.

Helsgaun, K. 2000. An effective implementation of the Lin-Kernighan traveling salesman heuristic. *European Journal of Operations Research* **126**(1) 106–130.

Hochbaum, D. S. 1997. *Approximation algorithms for NP-hard problems*. PWS Publishing, Boston, MA.

IBM Corp. 2009. *IBM ILOG CPLEX Optimizer 12.1.0*. `http://www.ibm.com/software/integration/optimization/cplex-optimizer/`.

Jain, K., M. Mahdian, E. Markakis, A. Saberi, V. V. Vazirani. 2003. Greedy facility location algorithms analyzed using dual fitting with factor-revealing LP. *Journal of the ACM* **50**(6) 795–824.

Laporte, G. 1988. Location-routing problems. B. L. Golden, A. A. Assad, eds., *Vehicle Routing: Methods and Studies*, North-Holland, Amsterdam. 163–198.

Laporte, G. 2009. Fifty years of vehicle routing. *Transportation Science* **43**(4) 408–416.

Laporte, G., Y. Nobert, S. Taillefer. 1988. Solving a family of multi-depot vehicle routing and location-routing problems. *Transportation Science* **22**(3) 161–172.

Laporte, G., F. Semet. 2001. Classical heuristics for the capacitated VRP. P. Toth, D. Vigo, eds., *The vehicle routing problem*, Society for Industrial and Applied Mathematics, Philadelphia, PA. 109–128.

Li, C., D. Simchi-Levi. 1990. Worst-case analysis of heuristics for multidepot capacitated vehicle routing problems. *ORSA Journal on Computing* **2**(1) 64–73.

Liu, J., C.-L. Li, C.-Y. Chan. 2003. Mixed truck delivery systems with both hub-and-spoke and direct shipment. *Transportation Research Part E: Logistics and Transportation Review* **39**(4) 325–339.

Mahdian, M., Y. Ye, J. Zhang. 2006. Approximation algorithms for metric facility location problems. *SIAM Journal on Computing* **36**(2) 411–432.

Mina, H., V. Jayaraman, R. Srivastava. 1998. Combined location-routing problems: A synthesis and future research directions. *European Journal of Operational Research* **108**(1) 1–15.

Nagy, G., S. Salhi. 2007. Location-routing: Issues, models and methods. *European Journal of Operational Research* **177**(2) 649–672.

Prins, C., C. Prodhon, R. W. Calvo. 2006. Solving the capacitated location-routing problem by a grasp complemented by a learning process and a path relinking. *4OR* **4**(3) 221–238.

Prins, C., C. Prodhon, A. B. Ruiz, P. Soriano, R. W. Calvo. 2007. Solving the capacitated location-routing problem by a cooperative lagrangean relaxation-granular tabu search heuristic. *Transportation Science* **41**(4) 470–483.

Ravi, R., A. Sinha. 2006. Approximation algorithms for problems combining facility location and network design. *Operations Research* **54**(1) 73–81.

Toth, P., D. Vigo. 2002. *The Vehicle Routing Problem*. Monographs on Discrete Mathematics and Applications. Society for Industrial and Applied Mathematics, Philadelphia, PA.

Tuzun, D., L. I. Burke. 1999. A two-phase tabu search approach to the location routing problem. *European Journal of Operational Research* **116**(1) 87–99.

Vahdani, B., M. Zandieh. 2010. Scheduling trucks in cross-docking systems: Robust meta-heuristics. *Computers & Industrial Engineering* **58**(1) 12–24.

Webb, M. H. J. 1968. Cost functions in the location of depots for multiple-delivery journeys. *Operations Research Quarterly* **19** 311–320.

Wen, M., J. Larsen, J. Clausen, J.-F. Cordeau, G. Laporte. 2009. Vehicle routing with cross-docking. *Journal of the Operational Research Society* **60**(11) 1708–1718.

Williamson, D., D. Shmoys. 2011. *The Design of Approximation Algorithms*. Cambridge University Press.