



An Asymptotically Optimal Approximation Algorithm for the Travelling Car Renter Problem

Lehilton L. C. Pedrosa 

Institute of Computing, University of Campinas, SP, Brazil
lehilton@ic.unicamp.br

Greis Y. O. Quesquén 

Institute of Computing, University of Campinas, SP, Brazil
greis.quesquen@students.ic.unicamp.br

Rafael C. S. Schouery 

Institute of Computing, University of Campinas, SP, Brazil
rafael@ic.unicamp.br

Abstract

In the classical Travelling Salesman Problem (TSP), one wants to find a route that visits a set of n cities, such that the total travelled distance is minimum. An often considered generalization is the Travelling Car Renter Problem (CaRS), in which the route is travelled by renting a set of cars and the cost to travel between two given cities depends on the car that is used. The car renter may choose to swap vehicles at any city, but must pay a fee to return the car to its pickup location. This problem appears in logistics and urban transportation when the vehicles can be provided by multiple companies, such as in the tourism sector. In this paper, we consider the case in which the return fee is some fixed number $g \geq 0$, which we call the Uniform CaRS (UCaRS). We show that, already for this version, there is no $o(\log n)$ -approximation algorithm unless $P = NP$. The main contribution is an $O(\log n)$ -approximation algorithm for the problem, which is based on the randomized rounding of an exponentially large LP-relaxation.

2012 ACM Subject Classification Theory of computation \rightarrow Approximation algorithms analysis; Theory of computation \rightarrow Routing and network design problems

Keywords and phrases Approximation Algorithm, Travelling Car Renter Problem, LP-rounding, Separation Problem

Digital Object Identifier 10.4230/OASICS.ATMOS.2019.14

Funding Supported by grant #2015/11937-9, São Paulo Research Foundation (FAPESP), and grants #425340/2016-3, #313026/2017-3, #308689/2017-8, #425806/2018-9, #422829/2018-8, National Council for Scientific and Technological Development (CNPq).

1 Introduction

Transportation is a key aspect of modern society, as it connects people, businesses and services. Related problems are spread in many areas, such as logistics [18], supply-chain [34] and, in particular, urban transportation [12, 27, 2]. To the latter, the transportation decisions are determinant to, e.g., the diffusion of the population with the growth of cities and urban areas, the movement of people who commute from or to school and work, and easy access of tourists and visitors to events or leisure activities. Vehicles comprise the most used means in urban transportation, which can be divided into two main traffic modes: public transportation and private cars. The first is usually considered to be a cheap form of transport and helps cities to reduce traffic congestion and the overall level of pollution. The second mode is more flexible and convenient but has a relatively higher price and consumption [23].



© Lehilton L. C. Pedrosa, Greis Y. O. Quesquén, and Rafael C. S. Schouery;
licensed under Creative Commons License CC-BY

19th Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2019).

Editors: Valentina Cacchiani and Alberto Marchetti-Spaccamela; Article No. 14; pp. 14:1–14:15

OpenAccess Series in Informatics



Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

A special trend in urban transportation, especially for private cars, is the offer of individual transport services, such as sharing or rental. Car sharing allows customers to reserve, access and use vehicles remotely [7], and vehicles are rented on an hourly basis by members looking for a high level of convenience and a low level of commitment [1]. On the other hand, car rental may be less straightforward but allows rents on a day, week, month or even on a yearly basis. Independent of the form, in most cases, the vehicle rental problems are discussed from a company's point of view [31], whose goal is to maximize the total profit or the number of customers served. Some works, however, consider problems from a client's perspective [17, 15, 5, 26], who has a set of available rental services and wants to choose which services to hire to minimize its total cost.

A problem that aims at minimizing costs associated with the service's user appears in the tourism industry [5]. Goldberg et al. [17] considered the Travelling Car Renter Problem (CaRS), which can be described as follows. A tourist wants to visit a set of cities and, to this, there are many rental services which offer a set of available cars. The cost to rent each car depends on the route taken by this car from the pickup point to its return location. Moreover, for each time the tourist rents a car, there is an additional fee, which corresponds to the cost to return the car to its pickup location. The cost of both the route and the return fee may depend on the selected car. The objective is to find a closed route that visits all the cities and comes back to the original place, such that the cost of all rented cars is minimum. The classical Travelling Salesman Problem (TSP) is a particular case of this problem. Indeed, TSP corresponds to instances of CaRS in which there is only one available car, and the return fee is zero. Since TSP is NP-hard, CaRS is NP-hard as well.

In this paper, we consider the version of CaRS in which the return fee is fixed regardless of the return point, and is the same for every car, which we call the *Uniform CaRS* (UCaRS). Formally, an instance of UCaRS is composed of set of cities $V = \{1, 2, \dots, n\}$, a set of cars $C = \{1, 2, \dots, r\}$, a return fee $g \geq 0$ and, for each car $i \in C$, an associated cost function $d_i : V \times V \rightarrow \mathbb{R}_{\geq 0}$. A solution is a sequence of walks P_1, P_2, \dots, P_s on V , associated with indexes $e_1, e_2, \dots, e_s \in C$, respectively, such that $P = (P_1 P_2 \dots P_s)$ is a closed route, and each city of V appears in P at least once. Denote by $E(H)$ the set of edges of a graph H . The objective of the problem is to find a solution which minimizes the sum

$$\sum_{i=0}^s \left(\sum_{(u,v) \in E(P_i)} d_{e_i}(u,v) + g \right),$$

which is the cost of edges in each walk P_i corresponding to car e_i plus the return fees.

1.1 Our contribution and summary of techniques

In this paper, we study UCaRS from the perspective of approximation algorithms. An algorithm for a minimization problem is an $\alpha(n)$ -approximation algorithm if it runs in polynomial time and, for every instance I of size n of the problem, it returns a solution with value at most $\alpha(n) \cdot \text{OPT}(I)$, where $\text{OPT}(I)$ the value of an optimal solution for I . An $\alpha(n)$ -approximation algorithm is said to be asymptotically optimal if any approximation algorithm has factor $\Omega(\alpha(n))$ unless $P = NP$.

We assume that for each car $i \in C$, the function d_i is metric, i.e., each function d_i satisfies the following assumptions:

1. (*symmetry*) for every $u, v \in V$, $d_i(u, v) = d_i(v, u)$, and
2. (*triangle inequality*) for every $u, v, w \in V$, $d_i(u, v) \leq d_i(u, w) + d_i(w, v)$.

Intuitively, the symmetry means that going from a city u to a city v using a car i costs the same as going from city v to city u using the same car. Also, the triangle inequality means that, to go from u to v , taking a direct route is always one of the most economical choices. If no restrictions on the cost functions are made, then it is very unlikely that the problem would have any approximation, since this version of UCARS generalizes the non-metric TSP, which cannot be approximated by any computable function unless $P = NP$ [32].

For TSP, one requires that a solution P is a Hamiltonian cycle, i.e., it is a closed walk which visits every city of V exactly once. For most applications, this assumption is without loss of generality, since the triangle inequality implies that any closed walk corresponds to a cycle spanning the same cities of no larger cost. For UCARS, however, we observe that, if a solution were required to visit each city exactly once, then the problem would not admit any approximation algorithm, even under the assumption that every cost function d_i is metric. Indeed, we show that if such a variant has an $\alpha(n)$ -approximation algorithm for any computable function $\alpha(n)$, then one can solve the Hamiltonian Cycle Problem (HamC) in polynomial time. Since HamC is NP-hard, this would imply $P = NP$.

As for hardness results, we show that UCARS is as hard to approximate as the Set Cover Problem (SC), which implies that UCARS has no approximation algorithm with factor $o(\log n)$, unless $P = NP$ [20, 9]. Then, we note that there is a natural correspondence between instances of UCARS and instances of Group TSP (G-TSP), which is a generalization of TSP whose instance is comprised of a family of groups of cities, and whose goal is to find a route that visits at least one city of each group. Namely, we show that UCARS can be reduced to G-TSP preserving the approximation factor, implying an approximation factor of $O(\log^2 n \log r)$ for UCARS by results from Garg et al. [13] and Fakcharoenphol et al. [10].

Our main contribution is a better approximation algorithm, which achieves a factor of $O(\log n)$. This algorithm is asymptotically optimal because of the proved lower bound. The reduction to G-TSP gives only a polylogarithmic factor, thus our algorithm takes a different approach. We observe that a solution of UCARS can be seen as a set of walks which cover the set of cities in V . Recall that in SC, given a family of subsets of V , each with a given weight, one wants to select some subsets such that the whole set V is covered, and the total weight of selected subsets is minimized. This suggests that an instance of UCARS could be reduced to an instance of SC, where the subsets correspond to possible sets of walks whose weights are the cost of visiting the corresponding cities plus the return fee.

This reduction does not work as is, however, for two main reasons. First, there are exponentially many distinct walks, and thus it does not run in polynomial time. Second, a solution for the SC instance does not take into account connectivity, and thus the union of the walks attained from the SC solution may not form a closed walk. Yet, we use this reduction to obtain a linear programming (LP) formulation which gives a lower bound on the optimal value. While this formulation is still exponential, we show how to obtain a good solution to its relaxation in polynomial time. Then, in a first phase, we obtain a set of walks that visit every city, by using a randomized algorithm which rounds the LP solution, and whose expected cost is at an $O(\log n)$ factor of the optimal value. The set of walks might be disconnected, thus, in a second phase, we show how to combine the walks into a single closed walk of not much larger cost.

The most involved and technical part of our algorithm is finding a feasible solution for the LP formulation with bounded value. To this, we note that the number of variables is exponential, but there are only polynomially many constraints. Then, an optimal solution can be obtained if the corresponding dual separation problem can be solved in polynomial time. This separation problem, for its turn, is NP-hard, and even hard to approximate by any constant. Thus, we show that a restriction of the original LP can still be used to obtain a lower bound, and that, for this restriction, we can compute an approximate solution.

1.2 Related works

CaRS was introduced by Silva et al. [30, 17], who proposed a memetic algorithm and a hybridization between GRASP and VND heuristics. Following, several other heuristic methods have been proposed for the problem, such as ant-colony optimization [29], transgenetic algorithms [31, 16], and evolutionary algorithms [11, 5, 8, 26, 6]. Exact algorithms based on mathematical formulations have also been proposed, which include mixed integer programming formulations [24, 5, 26, 25], and models based on the Quadratic Assignment Problem (QAP) or on network-flow formulations [14]. To the best of our knowledge, no approximation algorithms for CaRS or one of its variants have been discussed before this work.

It is folklore that the so-called Double-MST is a 2-approximation for the metric TSP. This algorithm calculates a Minimum Spanning Tree (MST) of the input graph G , then duplicates all its edges to complete a Eulerian graph, i.e., a graph for which there is a route visiting each edge exactly once. By making shortcuts, one obtains a Hamiltonian cycle whose cost no more than twice the cost of the MST.

SC is a classical NP-hard problem, and it is NP-hard to approximate SC by a factor $(1 - \varepsilon) \log n$, where n is the size of the ground set, and $\varepsilon > 0$ is an arbitrary constant [20, 9]. For this problem, a simple greedy algorithm is an $O(\log n)$ approximation [4]. It is well known that the same factor can also be obtained by a probabilistic algorithm which rounds its LP relaxation by independently selecting each set according to the corresponding variable [33].

G-TSP is a generalization of both TSP and SC, and thus the SC hardness holds for G-TSP as well. In fact, even if the cost function corresponds to the Euclidean distance, it is hard to approximate G-TSP by any constant [28]. A closely related problem is the Group Steiner Problem (G-ST). In this problem, an instance is composed of an edge-weighted graph on n vertices and m groups of vertices. The objective is to find a minimum weight tree that contains at least one vertex from every group. For this problem, it is unlikely that a $O(\log^{2-\varepsilon} m)$ -approximation algorithm exist, for any $\varepsilon > 0$ [21]. The best-known algorithm has a factor of $O(\log^2 n \log m)$ [13, 3].

1.3 Paper organization

The remainder of the paper is organized as follows. In Section 2, we discuss the inapproximability of UCaRS and give the reduction from UCaRS to G-TSP. In Section 3, we present a covering integer programming formulation which gives a lower bound for the problem. Then, we show how to use the LP relaxation to derive a solution for the problem whose cost is at most at a factor $O(\log n)$ of the optimal value. This is done by a randomized LP rounding algorithm which takes as the input a pre-computed LP solution. In Section 4, we show how to obtain such a solution. First, we describe the separation problem corresponding to the LP formulation and observe that it is NP-hard. Then, we consider a restriction of the formulation for which the corresponding separation problem can be approximated and show that the value of this restriction is not too far from the optimal value. In Section 5, we discuss the used techniques and possible extensions to similar problems.

2 Inapproximability

As in the case of TSP, if the cost functions of an instance of UCaRS are arbitrary, then the problem cannot be approximated by any function computable in polynomial time unless $P = NP$. Next, we observe that, for the version of the problem which requires a solution to be a Hamiltonian cycle, this hardness holds even if we assume that each cost function d_i is symmetric and satisfies the triangle inequality.

► **Theorem 1.** *Let $\alpha(n)$ be a function computable in polynomial time with $\alpha(n) \geq 1$. If $P \neq NP$, then there is no $\alpha(n)$ -approximation algorithm for the problem of, given an instance of UCARS, finding a solution P which is a Hamiltonian cycle with minimum cost.*

Thus, from now on, we allow routes which visit the same city more than once. Even if every cost function is metric and the solution is not required to be a Hamiltonian cycle, next lemma states that UCARS is SC-hard, i.e., there is a reduction from the unweighted version of SC to UCARS. Recall that an instance of unweighted SC is composed of subsets S_1, S_2, \dots, S_m of a ground set E . The objective is to find a set of indices e_1, e_2, \dots, e_s such that $S_{e_1} \cup S_{e_2} \cup \dots \cup S_{e_s} = E$ and s is minimum.

► **Theorem 2.** *If there is an α -approximation for UCARS, then there is an α -approximation for unweighted SC.*

Using the hardness result for SC [9], this immediately implies the following.

► **Corollary 3.** *If $P \neq NP$, there is no $(1 - \varepsilon) \log n$ -approximation for UCARS, for any $\varepsilon > 0$.*

On the other hand, one can reduce an instance of UCARS to an instance of G-TSP, such that the optimal value of the first is at a constant factor of the optimal value of the second, and vice-versa. An instance of G-TSP is formed by a graph G , a cost function on the edges d' , and sets $S_1, S_2, \dots, S_m \subseteq V(G)$. A solution is a closed walk P such that $S_i \cap V(P) \neq \emptyset$ for $1 \leq i \leq m$. The objective is to find a solution which minimizes the sum $\sum_{(u,v) \in E(P)} d'(u, v)$.

► **Lemma 4.** *If G-TSP admits an α -approximation, then UCARS admits an α -approximation.*

Since an $\alpha(n)$ -approximation for G-ST implies a $2\alpha(n)$ -approximation for G-TSP, using the $O(\log^2 n \log m)$ -approximation for G-ST by Garg et al. [13, 10], we obtain the following.

► **Theorem 5.** *There exists an $O(\log^2 n \log r)$ -approximation for UCARS.*

3 An LP rounding algorithm

In this section, we describe the LP rounding algorithm with factor $O(\log n)$. We start with an integer programming formulation which gives a lower bound for the optimal value.

3.1 A covering formulation

The main idea for the algorithm comes from the observation that a solution consists of a set of walks that visit the whole set of cities. Each walk corresponds to a pair of a car i and a subset of cities S , which we call a *component*. The cost of each component corresponds to the total weight of its edges plus the return fee. For the sake of simplicity, instead of considering every possible walk for each subset of cities, we consider each subset only once for each car. Thus, instead of looking for a set of segments that forms a closed walk, we are only interested in finding a set of connected components that cover the set of cities. The cost of each representative connected component corresponds to the weight of a minimum spanning tree of S according to d_i plus the return fee g . Note that any minimum walk which visits S has cost at most twice that of the minimum spanning tree. This is sufficient for our purposes since we are only interested in an asymptotic factor $O(\log n)$.

In the following, we denote by \mathcal{S} the set of all pairs (i, S) with $S \subseteq V$ and $i \in C$. For each $(i, S) \in \mathcal{S}$, we denote by $\text{MST}_i(S)$ the cost of a minimum spanning tree of S with respect to d_i . The cost of (i, S) is defined as $\text{cost}(i, S) = \text{MST}_i(S) + g$. In the following integer

14:6 Optimal Approximation Algorithm for the Travelling Car Renter Problem

linear program, for each $(i, S) \in \mathcal{S}$, there is a binary variable $x_{(i,S)}$ which indicates whether the component (i, S) belongs to the solution.

$$\begin{aligned} & \text{minimize} && \sum_{(i,S) \in \mathcal{S}} x_{(i,S)} \text{cost}(i, S) \\ & \text{subject to} && \sum_{(i,S) \in \mathcal{S}: v \in S} x_{(i,S)} \geq 1, \quad \forall v \in V, \\ & && x_{(i,S)} \in \{0, 1\}, \quad \forall (i, S) \in \mathcal{S}. \end{aligned} \tag{IP}$$

In the following, given an integer linear program or a linear program (Q) , denote by $\text{OPT}(Q)$ the optimal value of (Q) . Also, we denote by OPT the value of an optimal solution for the instance of UCaRS. We observe that a solution for the UCaRS instance induces a feasible solution of (IP), thus $\text{OPT}(\text{IP})$ is a lower bound for the optimal value.

► **Lemma 6.** *Consider an instance of UCaRS and the corresponding formulation (IP). Let OPT be the value of an optimal solution for this instance. Then, $\text{OPT}(\text{IP}) \leq \text{OPT}$.*

Let (P) be the linear relaxation of (IP). Notice that (P) has an exponential number of variables and therefore we do not know how to solve this problem directly. However, we can obtain an approximate solution with only a polynomial number of non-zero variables, according to the following lemma, which is a central result for the algorithm. The proof is given in Section 4.

► **Lemma 7.** *There is an algorithm that, in polynomial time, finds a feasible solution for (P) whose value is at most $c \cdot \text{OPT}(\text{IP})$, for some constant c .*

3.2 Rounding the fractional solution

Next, we assume that we are given a solution x of (P), and the objective is to find a solution for the instance of UCaRS by rounding the value of variables $x_{(i,S)}$ for each $(i, S) \in \mathcal{S}$. The rounding algorithm can be broken into three phases. In the first phase, we find a set of components which cover V . In the second phase, we complement the set of components to obtain a connected cover. In the third phase, we turn the set of components into a route that visits each city at least once.

Covering phase

Note that for each $(i, S) \in \mathcal{S}$, we can assume that $0 \leq x_{(i,S)} \leq 1$. Thus, the value of $x_{(i,S)}$ can be interpreted as a probability that component (i, S) is used in an integral solution. We start with an empty set of components \mathcal{C} and execute the following iteration: include each component (i, S) in \mathcal{C} independently with probability $x_{(i,S)}$. The expected cost of the included components in this iteration is at most the value of x , but this does not guarantee that every city is covered by some component of \mathcal{C} . If we repeat this process a sufficient number of times, then the probability of a city remaining uncovered tends to zero. After $\log n$ iterations, the probability that a city is uncovered is small, then for each uncovered city v , we include a component corresponding to a singleton $\{v\}$ and an arbitrary car, say, the first one. The steps for the covering phase are summarized in Figure 1.

The following lemma states that indeed the obtained set covers all the cities. The proof follows directly from the algorithm.

► **Lemma 8.** *Let \mathcal{C} be the set of components returned by the covering phase of the algorithm. Then, for every $v \in V$, there is a component $(i, S) \in \mathcal{C}$ such that $v \in S$.*

1. Compute an (P) solution x using Lemma 7.
2. Let $\mathcal{C} \leftarrow \emptyset$ and repeat $\lceil \log n \rceil$ times:
 - For each (i, S) include (i, S) in \mathcal{C} with probability $x_{(i,S)}$.
3. For each $v \in V$:
 - If $v \notin S$ for all $(i, S) \in \mathcal{C}$, then add $(1, \{v\})$ to \mathcal{C} .
4. Return \mathcal{C} .

■ **Figure 1** First phase of the rounding algorithm: covering.

Lemma 10 estimates the expected cost of \mathcal{C} . First, using standard analysis, we bound the probability that a city is not covered by one component included in the random step.

► **Lemma 9.** *Let \mathcal{C}' be the set of components included in any iteration of step 2 of the covering phase of the algorithm. Also, let V' be the set of cities v for which there exists $(i, S) \in \mathcal{C}'$ with $v \in S$. Then, $\Pr(v \notin V') \leq 1/n$ for every $v \in V$.*

► **Lemma 10.** *Let \mathcal{C} be the set of components returned by the covering phase. Then*

$$\mathbb{E} \left[\sum_{(i,S) \in \mathcal{C}} \text{cost}(i, S) \right] \leq O(\log n) \cdot \text{OPT}.$$

Proof. Consider set \mathcal{C}_ℓ of components (i, S) included in the iteration ℓ of step 2. Since each component (i, S) is included with probability $x_{(i,S)}$, the expected cost of these components is

$$\begin{aligned} \mathbb{E} \left[\sum_{(i,S) \in \mathcal{C}_\ell} \text{cost}(i, S) \right] &= \sum_{(i,S) \in \mathcal{C}} \Pr((i, S) \in \mathcal{C}_\ell) \text{cost}(i, S) \\ &= \sum_{(i,S) \in \mathcal{C}} x_{(i,S)} \text{cost}(i, S) \leq c \cdot \text{OPT}(\text{IP}), \end{aligned}$$

where the inequality comes from the fact that, from Lemma 7, the objective value for x is at most $c \cdot \text{OPT}(\text{IP})$, for some constant c . Thus, the expected cost of the set of components \mathcal{C}' drawn in step 2 can be bounded as

$$\mathbb{E} \left[\sum_{(i,S) \in \mathcal{C}'} \text{cost}(i, S) \right] \leq \sum_{\ell=1}^{\lceil \log n \rceil} \mathbb{E} \left[\sum_{(i,S) \in \mathcal{C}_\ell} \text{cost}(i, S) \right] \leq \lceil \log n \rceil \cdot c \cdot \text{OPT}(\text{IP}).$$

Let V' be the set of vertices covered by some component of \mathcal{C}' . For each $v \in V \setminus V'$, a new component of the form $(1, \{v\})$ is added in step 3 of the first phase of the algorithm. Using Lemma 9, this happens with probability at most $1/n$. Let \mathcal{C}_+ be the set of such components and observe that $\mathcal{C}_+ = \mathcal{C} \setminus \mathcal{C}'$. Also, note that for any component $(i, S) \in \mathcal{C}_+$, a minimum spanning tree of S contains no edges, and thus $\text{cost}(i, S) = g$. It follows that

$$\mathbb{E} \left[\sum_{(i,S) \in \mathcal{C}_+} \text{cost}(i, S) \right] = \sum_{v \in V} \Pr(v \notin V') \text{cost}(1, \{v\}) \leq \sum_{v \in V} \frac{1}{n} g = g.$$

Observe that any feasible solution of (IP) has at least one component, thus $g \leq \text{OPT}(\text{IP})$. From Lemma 6, we know that $\text{OPT}(\text{IP}) \leq \text{OPT}$. Combining everything, the costs of all components in \mathcal{C} add up to $O(\log n) \cdot \text{OPT}$. ◀

1. Let $\mathcal{D} \leftarrow \emptyset$.
2. While $\mathcal{C} \cup \mathcal{D}$ induces a disconnected graph H :
 - Find vertices v and v' in distinct components of H which minimize $d_{\min}(v, v')$.
 - Add $(j, \{v, v'\})$ to \mathcal{D} , where j is such that $d_j(v, v') = d_{\min}(v, v')$.
3. Return \mathcal{D} .

■ **Figure 2** Second phase of the rounding algorithm: connection.

Connection phase

After the first phase, the set of components \mathcal{C} covers all cities V , but they are possibly disconnected. So we want to connect \mathcal{C} by selecting additional components, \mathcal{D} , each of which connect two components of \mathcal{C} . Since each such an additional component corresponds to an edge which connects two cities, v, v' , we may simply select the car i with the smallest cost $d_i(v, v')$. Thus, we consider an edge-weight function d_{\min} such that for every pair of cities $v, v' \in V$, we define $d_{\min}(v, v') = \min\{d_i(v, v') : i \in \mathcal{C}\}$.

We execute the following. Start with an empty set \mathcal{D} . Then, while the graph H induced by the minimum spanning trees of $\mathcal{C} \cup \mathcal{D}$ is disconnected, find vertices v and v' such that v and v' are in distinct connected components of H , and $d_{\min}(v, v')$ is minimum. Include component $(j, \{v, v'\})$ in \mathcal{D} , where j is such that $d_j(v, v') = d_{\min}(v, v')$. After the last iteration, the minimum spanning trees of $\mathcal{C} \cup \mathcal{D}$ induce a connected graph H which contains all vertices. The steps for the connection phase are summarized in Figure 2.

To bound the cost of components in \mathcal{D} , we need the following auxiliary lemma.

► **Lemma 11.** *Let T be a minimum spanning tree of V with respect to d_{\min} . Then*

$$\sum_{(v, v') \in E(T)} d_{\min}(v, v') \leq OPT.$$

Proof. Consider an optimal solution $P = P_1, P_2, \dots, P_s$ of UCARS. For each $(v, v') \in E(P)$, let $\phi(v, v') \in \mathcal{C}$ be the car associated with edge (v, v') . Let T be a minimum spanning tree of P . Since P spans every vertex of V , T is also a spanning tree of V . We get

$$\sum_{(v, v') \in E(T)} d_{\min}(v, v') \leq \sum_{(v, v') \in E(T)} d_{\phi(v, v')}(v, v') \leq OPT,$$

where the last inequality holds because T is a minimum spanning tree of P . ◀

The edge cost of components of \mathcal{D} is bounded by the next lemma.

► **Lemma 12.** *Let \mathcal{D} be the set of components returned by the connection phase of the algorithm. Then*

$$|\mathcal{D}| \leq |\mathcal{C}| \quad \text{and} \quad \sum_{(i, S) \in \mathcal{D}} MST_i(S) \leq OPT.$$

Routing phase

Given the set of components \mathcal{C} and \mathcal{D} , we construct a closed walk which visits all cities by considering each component at a time. Let $r \in V$ be an arbitrary vertex and start a trivial closed walk P composed only of r . Now, while there are components of $\mathcal{C} \cup \mathcal{D}$ which have not been considered yet, find one such a component (i, S) such that $S \cap V(P) \neq \emptyset$ and let

1. Let $r \in V$ and make $P \leftarrow (r)$.
2. Make $\mathcal{N} \leftarrow \mathcal{C} \cup \mathcal{D}$.
3. While $\mathcal{N} \neq \emptyset$:
 - Find $(i, S) \in \mathcal{N}$ such that $S \cap V(P) \neq \emptyset$, and let $v \in S \cap V(P)$.
 - Construct a cycle P' of $S \setminus V(P) \cup \{v\}$.
 - For each edge $(u, u') \in E(P')$, make $\phi(u, u') \leftarrow i$.
 - Insert P' into P .
 - Make $\mathcal{N} \leftarrow \mathcal{N} \setminus \{(i, S)\}$.
4. Return P, ϕ .

■ **Figure 3** Third phase of the rounding algorithm: routing.

$v \in S \cap V(P)$. Then, from a minimum spanning tree T' of S with respect to d_i , construct a cycle P' of $S \setminus V(P) \cup \{v\}$. Observe that, by doubling the edges of T' and then taking shortcuts, one can build such a cycle with cost at most $2 \text{MST}_i(S)$. The edges of P' are labelled with the index of car i , and the walk P is extended by including P' .

Note that P visits all vertices and can be decomposed into a sequence of maximal walks with the same label. Therefore, it induces a feasible solution for the instance of UCARS. The steps for the routing phase are summarized in Figure 3.

The cost of the returned solution corresponds to travelled edges and to the return fees. The edge cost is bounded as follows.

► **Lemma 13.** *Let P, ϕ be the solution returned by the routing phase of the algorithm. Then,*

$$\sum_{(v,v') \in E(P)} d_{\phi(v,v')}(v, v') \leq 2 \cdot \sum_{(i,S) \in \mathcal{C} \cup \mathcal{D}} \text{MST}_i(S).$$

Proof. Recall that to each component $(i, S) \in \mathcal{C} \cup \mathcal{D}$, one constructs a cycle P' by doubling the edges of minimum spanning tree of S and taking shortcuts, then the cycle P' has cost at most $2 \text{MST}_i(S)$. Now observe that $E(P)$ is the union of the edges of cycles P' and each such a cycle P' corresponds to a distinct component $(i, S) \in \mathcal{C} \cup \mathcal{D}$. ◀

Finally, we can show the main result.

► **Theorem 14.** *There exists a randomized $O(\log n)$ -approximation algorithm for UCARS.*

Proof. By Lemma 7, the solution x of (IP) is computed in polynomial time. Since each of the three phases of the LP rounding algorithm also runs in polynomial time, the whole algorithm is polynomial.

Using Lemma 13, the expected edge cost of edges is

$$\begin{aligned} \mathbb{E} \left[\sum_{(v,v') \in E(P)} d_{\phi(v,v')}(v, v') \right] &\leq 2 \cdot \mathbb{E} \left[\sum_{(i,S) \in \mathcal{C}} \text{MST}_i(S) \right] + \mathbb{E} \left[\sum_{(i,S) \in \mathcal{D}} \text{MST}_i(S) \right] \\ &\leq O(\log n) \cdot \text{OPT} + \text{OPT} = O(\log n) \cdot \text{OPT}, \end{aligned}$$

where the last inequality follows from Lemmas 10 and 12.

Now it remains to bound the expected cost of the return fees. There is a return fee of cost g for each vertex in the walk P where there is a switching car, thus it is sufficient to count the number of vertices in P whose adjacent edges have different labels. In each iteration of step 3 of the routing phase of the algorithm, the labels of every edge in the inserted subwalk P' are the same, thus the number of swaps in P increases by at most 2. Since this step is executed for $|\mathcal{C}| + |\mathcal{D}|$ iterations, the total expected return fee is at most

14:10 Optimal Approximation Algorithm for the Travelling Car Renter Problem

$$\mathbb{E}[2 \cdot (|\mathcal{C}| + |\mathcal{D}|) \cdot g] \leq \mathbb{E}[4 \cdot |\mathcal{C}| \cdot g] \leq \mathbb{E}\left[4 \cdot \sum_{(i,S) \in \mathcal{C}} \text{cost}(i, S)\right] \leq O(\log n) \cdot \text{OPT},$$

where the first inequality comes from Lemma 12, the second inequality is due to $g \leq \text{cost}(i, S)$, and the last inequality comes from Lemma 10.

Adding up the expected costs of edges and the return fees, we conclude that the expected cost of the returned solution cost is at most $O(\log n) \cdot \text{OPT}$, and the theorem follows. ◀

4 Approximating the LP-relaxation

The objective of this section is to find a feasible solution of (P) and prove Lemma 7. The main ingredient will be solving a relaxed version of the separation problem and using the Ellipsoid method. For the sake of completeness, we begin by briefly reviewing this method and identifying the corresponding separation problem.

4.1 The separation problem

Remember that (P) has an exponential number of variables. A strategy to tackle this difficulty is solving the separation problem corresponding to the dual formulation. The idea is that, if the number of variables in the dual is bounded by a polynomial, then one can use the Ellipsoid method. Informally, this method consists of iteratively picking a candidate solution y and querying an *oracle* for the separation problem. Given vector y , the separation problem is the task of either deciding that y is feasible, or finding a violated constraint. If y is not feasible, another solution y is picked, until a feasible optimal solution is found

► **Theorem 15** ([19]). *Let (Q) be a non-empty linear program. Then an optimal solution can be found querying an oracle for its separation problem only a polynomial number of times.*

The dual formulation of (P) is given next.

$$\begin{aligned} & \text{maximize} && \sum_{v \in V} y_v \\ & \text{subject to} && \sum_{v \in S} y_v - \text{MST}_i(S) \leq g \quad \forall (i, S) \in \mathcal{S} \\ & && y_v \geq 0 \quad \forall v \in V. \end{aligned} \tag{D}$$

For a set $A \subseteq V$, we define $y(A) = \sum_{v \in A} y_v$. Note that the separation problem consists of finding $(i, S) \in \mathcal{S}$ for which the constraint $y(S) - \text{MST}_i(S) \leq g$ is violated, or showing that there is no such constraint. Since the number of cars is polynomial, we can solve the problem separately for each $i \in C$. For some $i \in C$, it is enough to find $S \subseteq V$ such that $y(S) - \text{MST}_i(S)$ is maximum and verify that this value is greater than g . Unfortunately, this problem corresponds to the Net Worth Maximization Problem (NWMP), which has no constant-factor approximation algorithm unless $P = NP$ [22].

4.2 Cost-restricted components

To be able to use the Ellipsoid method, we will define a slightly different linear program, but whose separation problem is easier. If one considers only sets S for which $\text{MST}_i(S)$ is small, i.e., less than g , then maximizing $y(S)$ is a good approximation for the optimization version of the separation problem. Since, a priori, $\text{MST}_i(S)$ is greater than g , we will consider a relaxation of (D) which contains only constraints where $\text{MST}_i(S) \leq g$.

Let $\mathcal{S}_{\leq g}$ be the subset of components $(i, S) \in \mathcal{S}$ such that $\text{MST}_i(S) \leq g$, and define $(\mathbf{P}_{\leq g})$ to be the restriction of (\mathbf{P}) which contains only variables corresponding to $(i, S) \in \mathcal{S}_{\leq g}$. We note that the value of this restriction is at a constant factor of the value of (\mathbf{IP}) .

► **Lemma 16.** $\text{OPT}(\mathbf{P}_{\leq g}) \leq 4 \cdot \text{OPT}(\mathbf{IP})$.

Proof. We build a feasible solution for $(\mathbf{P}_{\leq g})$ from a feasible solution for (\mathbf{IP}) . For each $U \subseteq \mathcal{S}$, define $\text{cost}(U) = \sum_{(i,S) \in U} \text{cost}(i, S)$.

Note that a solution x of \mathbf{IP} corresponds to a set $U^* \subseteq \mathcal{S}$ such that $\text{cost}(U^*) = \text{OPT}(\mathbf{IP})$. We build an integral solution of $(\mathbf{P}_{\leq g})$ corresponding to a set $U \subseteq \mathcal{S}_{\leq g}$. If $g = 0$, we add component $(1, \{v\})$ to U for $v \in V$, thus $\text{cost}(U) = 0$, and we are done. Thus, assume $g > 0$.

Consider a component $(i, S) \in U^*$. We can partition S into parts S' such that $\text{MST}_i(S') \leq g$. First, let T be a minimum spanning tree of S with respect to d_i . Then, obtain a closed walk P by doubling the edges of T and finding an Eulerian walk whose edge weight is at most $2 \cdot \text{MST}_i(S)$. Finally, break P into disjoint maximal subwalks whose edges add up to at most g . To do this, greedily find a maximal prefix P' of P whose edges weights do not add up more than g , and remove $V(P')$ from P . Observe that the total weight of the subwalks is at most $2 \cdot \text{MST}_i(S)$, and that the number of subwalks is at most $\lceil (2 \cdot \text{MST}_i(S))/g \rceil$. Therefore, the component (i, S) can be replaced by a set of components whose total cost is $2 \cdot \text{MST}_i(S) + \lceil (2 \cdot \text{MST}_i(S))/g \rceil g \leq 4 \cdot \text{MST}_i(S) + g \leq 4 \cdot \text{cost}(i, S)$.

Repeating this procedure for each component of U^* , we obtain a set U of components (i, S) such that $\text{MST}_i(S) \leq g$ and $\text{cost}(U) \leq 4 \cdot \text{cost}(U^*)$. Since every vertex is in some component, U induces a feasible solution for $(\mathbf{P}_{\leq g})$, and the lemma holds. ◀

4.3 Constructing an approximate solution

Let $(\mathbf{D}_{\leq g})$ be the dual formulation of $(\mathbf{P}_{\leq g})$. Note that $(\mathbf{D}_{\leq g})$ is a relaxation of (\mathbf{D}) which contains only constraints corresponding to $(i, S) \in \mathcal{S}_{\leq g}$. Instead of solving $(\mathbf{D}_{\leq g})$, we will consider a different linear program in which we replace constraints $\sum_{v \in S} y_v - \text{MST}_i(S) \leq g$ by constraints $\sum_{v \in S} y_v \leq g$. The resulting linear program is described below.

$$\begin{aligned} & \text{maximize} && \sum_{v \in V} y_v \\ & \text{subject to} && \sum_{v \in S} y_v \leq g, \quad \forall (i, S) \in \mathcal{S}_{\leq g}, \\ & && y_v \geq 0, \quad \forall v \in V. \end{aligned} \tag{\mathbf{D}'_{\leq g}}$$

Now, the separation problem of $(\mathbf{D}'_{\leq g})$ corresponds to, given a vector y and an index i , find a set $S \subseteq V$ such that $y(S)$ is maximum and $\text{MST}_i(S) \leq g$. This problem corresponds to the Budget Steiner Tree Problem (BSTP), which is, again, an NP-Hard maximization problem. However, unlike NWMP, it admits a constant-factor approximation algorithm with factor $5 + \varepsilon$, for every $\varepsilon > 0$ [22]. Thus, instead of solving $(\mathbf{D}'_{\leq g})$ directly, we will obtain a solution to a relaxation.

To do this, we execute the Ellipsoid method, but stop if we cannot decide whether the candidate solution is infeasible for $(\mathbf{D}'_{\leq g})$. Thus, it may return an infeasible solution whose value might be larger than $\text{OPT}(\mathbf{D}'_{\leq g})$. More precisely, we execute the following algorithm. First, we initialize $\mathcal{A} = \emptyset$ and start the Ellipsoid method. Suppose that we are given a candidate solution y . For each car i , we execute the $(5 + \varepsilon)$ -approximation algorithm for the BSTP whose input is y and the distance function is d_i , and find a set $A_i \subseteq V$ with $\text{MST}_i(A) \leq g$. Let j be the car for which $y(A_j)$ is maximum. There are two cases to consider:

14:12 Optimal Approximation Algorithm for the Travelling Car Renter Problem

- a) If $y(A_j) > g$, then we add the constraint $y(A_j) \leq g$, which is a constraint of $(D'_{\leq g})$, because $(j, A_j) \in \mathcal{S}_{\leq g}$. We make $\mathcal{A} = \mathcal{A} \cup \{(j, A_j)\}$ and the execution of Ellipsoid method resumes with the set of constraints corresponding to \mathcal{A} .
- b) If $y(A_j) \leq g$, then we stop and return $(y, \mathcal{A}, (j, A_j))$.

The previous algorithm returns a set \mathcal{A} of polynomial size since we stopped before the end of the Ellipsoid algorithm. Let (D'_A) be the relaxation of $(D'_{\leq g})$ where only constraints corresponding to components in \mathcal{A} are added. Similarly, let (D_A) be the relaxation of $(D_{\leq g})$ where only the constraints corresponding to components in \mathcal{A} are added.

Next lemmas relate the values of (D_A) and (D'_A) and of D'_A and $D'_{\leq g}$.

► **Lemma 17.** $OPT(D_A) \leq 2 \cdot OPT(D'_A)$.

Proof. Let y be an optimal solution for (D_A) and $(i, S) \in \mathcal{A}$. Since y is feasible for (D_A) , we have $y(S) \leq g + MST_i(S)$, and, since $(i, S) \in \mathcal{S}_{\leq g}$, we have $MST_i(S) \leq g$. Then,

$$\frac{y(S)}{2} \leq \frac{MST(S) + g}{2} \leq \frac{g + g}{2} = g.$$

Since the choice of (i, S) is arbitrary, this inequality holds for every $(i, S) \in \mathcal{A}$, thus $\frac{y}{2}$ is feasible for (D'_A) . Therefore the value of the solution $\frac{y}{2}$ is not larger than the value of an optimal solution and we conclude that $OPT(D_A) = 2 \cdot \frac{y}{2}(V) \leq 2 \cdot OPT(D'_A)$. ◀

► **Lemma 18.** $OPT(D'_A) \leq (5 + \varepsilon) \cdot OPT(D'_{\leq g})$.

4.4 Proof of Lemma 7

► **Lemma 7.** *There is an algorithm that, in polynomial time, finds a feasible solution for (P) whose value is at most $c \cdot OPT(IP)$, for some constant c .*

Proof. Note that dual formulation of (D_A) corresponds to a linear program (P_A) which is similar to (P), but contains only variables corresponding to components in \mathcal{A} .

We execute the following algorithm. First, execute the modified Ellipsoid method described in Subsection 4.3 and obtain a set \mathcal{A} . Next, construct the linear program (P_A) . Then, solve (P_A) with any polynomial-time algorithm (e.g., Ellipsoid method), and obtain a solution x , indexed in \mathcal{A} . Finally, return the extension of x in which, for each $(i, S) \notin \mathcal{A}$, we represent $x_{(i,S)} = 0$ implicitly.

Notice that the above algorithm runs in polynomial time since \mathcal{A} has polynomial size and therefore (P_A) has a polynomial size too. Also, note that x is a feasible solution for (P) since (P_A) is a restriction of (P).

It remains to bound the value of an optimal solution for (P_A) . Using lemmas 17 and 18, we have $OPT(P_A) = OPT(D_A) \leq 2 \cdot OPT(D'_A) \leq 2 \cdot (5 + \varepsilon) \cdot OPT(D'_{\leq g})$. Now observe that $(D'_{\leq g})$ is a restriction of $(D_{\leq g})$, thus $OPT(P_A) \leq 2 \cdot (5 + \varepsilon) \cdot OPT(D_{\leq g}) = 2 \cdot (5 + \varepsilon) \cdot OPT(P_{\leq g})$. Since, by Lemma 16, we have $OPT(P_{\leq g}) \leq 4 \cdot OPT(IP)$, the lemma follows. ◀

5 Concluding remarks

While our $O(\log n)$ -approximation is probabilistic, it can be derandomized by techniques of conditional probabilities. Also, this paper focus on the asymptotic analysis, so changes to the algorithm may improve the hidden constant which multiplies $\log n$. Notice that our algorithm achieves the same asymptotic guarantee for the variant of the problem in which, instead of a closed walk, asks for a spanning tree composed of multiples subtrees.

We believe that the techniques of our algorithm can generalize to other variants of CaRS or similar problems. For example, one might consider the version in which the return fee depends on the car, that is, there is a return fee g_i for each car $i \in C$. A possible direction is considering the more general version of CaRS, in which, for each car i , the return fee is a given distance function f_i which depends on the return and pickup locations.

References

- 1 Fleura Bardhi and Giana M. Eckhardt. Access-Based Consumption: The Case of Car Sharing. *Journal of Consumer Research*, 39(4):881–898, December 2012. doi:10.1086/666376.
- 2 John Black. *Urban Transport Planning*. Routledge, May 2018. doi:10.4324/9781351068604.
- 3 Chandra Chekuri, Guy Even, and Guy Kortsarz. A greedy approximation algorithm for the group Steiner problem. *Discrete Applied Mathematics*, 154(1):15–34, January 2006. doi:10.1016/j.dam.2005.07.010.
- 4 V. Chvatal. A Greedy Heuristic for the Set-Covering Problem. *Mathematics of Operations Research*, 4(3):233–235, 1979. doi:10.1287/moor.4.3.233.
- 5 André Renato Villela da Silva and Luiz Satoru Ochi. An efficient hybrid algorithm for the Traveling Car Renter Problem. *Expert Systems with Applications*, 64:132–140, December 2016. doi:10.1016/j.eswa.2016.07.038.
- 6 André Renato Villela da Silva and Luiz Satoru Ochi. Um Algoritmo Evolutivo para o Problema do Caixeiro Alugador. In *Proceeding Series of the Brazilian Society of Applied and Computational Mathematics*, volume 5, 2017. doi:10.5540/03.2017.005.01.0460.
- 7 Kenan Degirmenci and Michael H. Breitner. Carsharing: A literature review and a perspective for information systems research. In Lars Beckmann, editor, *Multikonferenz Wirtschaftsinformatik (MKWI)*, Paderborn, Germany, February 2014. URL: <https://eprints.qut.edu.au/105684/>.
- 8 Sávio S. Dias, Luiz Satoru Ochi, Victor M. C. Machado, Luidi Gelabert Simonetti, and André Renato Villela da Silva. Uma Heurística Baseada em ILS Para o Problema do Caixeiro Alugador. In *Anais do XLVIII SBPO Simpósio Brasileiro de Pesquisa Operacional*, pages 1625–1636, 2016.
- 9 Irit Dinur and David Steurer. Analytical Approach to Parallel Repetition. In *Proceedings of the Forty-sixth Annual ACM Symposium on Theory of Computing*, STOC '14, pages 624–633, New York, NY, USA, 2014. ACM. doi:10.1145/2591796.2591884.
- 10 Jittat Fakcharoenphol, Satish Rao, Satish Rao, and Kunal Talwar. A Tight Bound on Approximating Arbitrary Metrics by Tree Metrics. In *Proceedings of the Thirty-fifth Annual ACM Symposium on Theory of Computing*, STOC '03, pages 448–455, New York, NY, USA, 2003. ACM. doi:10.1145/780542.780608.
- 11 Denis Felipe, Elizabeth Ferreira Gouvêa Goldberg, and Marco César Goldberg. Scientific algorithms for the Car Renter Salesman Problem. In *2014 IEEE Congress on Evolutionary Computation (CEC)*, pages 873–879. IEEE, July 2014. doi:10.1109/cec.2014.6900556.
- 12 David Foot. *Operational Urban Models*. Routledge, October 2017. doi:10.4324/97813515105307.
- 13 Naveen Garg, Goran Konjevod, and R. Ravi. A Polylogarithmic Approximation Algorithm for the Group Steiner Tree Problem. *Journal of Algorithms*, 37(1):66–84, October 2000. doi:10.1006/jagm.2000.1096.
- 14 Marco César Goldberg, Elizabeth Ferreira Gouvêa Goldberg, Henrique P. L. Luna, Matheus da S. Menezes, and Lucas Corrales. Integer programming models and linearizations for the traveling car renter problem. *Optimization Letters*, 12(4):743–761, April 2017. doi:10.1007/s11590-017-1138-5.
- 15 Marco César Goldberg, Elizabeth Ferreira Gouvêa Goldberg, Matheus da S. Menezes, and Henrique P. L. Luna. Quota traveling car renter problem: Model and evolutionary algorithm. *Information Sciences*, 367-368:232–245, November 2016. doi:10.1016/j.ins.2016.05.027.

- 16 Marco César Goldberg, Elizabeth Ferreira Gouvêa Goldberg, Paulo Henrique Asconavieta da Silva, Matheus da S. Menezes, and Henrique P. L. Luna. A transgenetic algorithm applied to the Traveling Car Renter Problem. *Expert Systems with Applications*, 40(16):6298–6310, November 2013. doi:10.1016/j.eswa.2013.05.072.
- 17 Marco César Goldberg, Paulo Henrique Asconavieta da Silva, and Elizabeth Ferreira Gouvêa Goldberg. Memetic algorithm for the Traveling Car Renter Problem: An experimental investigation. *Memetic Computing*, 4(2):89–108, September 2011. doi:10.1007/s12293-011-0070-y.
- 18 M. Grazia Speranza. Trends in transportation and logistics. *European Journal of Operational Research*, 264(3):830–836, February 2018. doi:10.1016/j.ejor.2016.08.032.
- 19 Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric Algorithms and Combinatorial Optimization*, volume 2 of *Algorithms and Combinatorics*. Springer, 1988. doi:10.1007/978-3-642-97881-4.
- 20 Sudipto Guha and Samir Khuller. Greedy Strikes Back: Improved Facility Location Algorithms. *Journal of Algorithms*, 31(1):228–248, April 1999. doi:10.1006/jagm.1998.0993.
- 21 Eran Halperin and Robert Krauthgamer. Polylogarithmic Inapproximability. In *Proceedings of the Thirty-fifth Annual ACM Symposium on Theory of Computing*, STOC '03, pages 585–594, New York, NY, USA, 2003. ACM. doi:10.1145/780542.780628.
- 22 David S. Johnson, Maria Minkoff, and Steven Phillips. The Prize Collecting Steiner Tree Problem: Theory and Practice. In *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '00, pages 760–769, Philadelphia, PA, USA, 2000. Society for Industrial and Applied Mathematics. URL: <http://dl.acm.org/citation.cfm?id=338219.338637>.
- 23 Xin Luan, Lin Cheng, Yang Zhou, and Fang Tang. Strategies of Car-Sharing Promotion in Real Market. In *2018 3rd IEEE International Conference on Intelligent Transportation Engineering (ICITE)*, pages 159–163. IEEE, September 2018. doi:10.1109/icite.2018.8492652.
- 24 Matheus da S. Menezes. *O problema do Caixeiro alugador com coleta de prêmios: Um estudo algorítmico*. PhD thesis, Universidade Federal do Rio Grande do Norte, 2014.
- 25 Brenner Humberto Ojeda Rios. *Hibridização de meta-heurísticas com métodos baseados em programação linear para o problema do caixeiro alugador*. Master's thesis, Universidade Federal do Rio Grande do Norte, 2017.
- 26 Brenner Humberto Ojeda Rios, Elizabeth Ferreira Gouvêa Goldberg, and Greis Yvet Oropeza Quesquen. A hybrid metaheuristic using a corrected formulation for the Traveling Car Renter Salesman Problem. In *2017 IEEE Congress on Evolutionary Computation (CEC)*, pages 2308–2314. IEEE, June 2017. doi:10.1109/cec.2017.7969584.
- 27 Philipp Rode, Graham Floater, Nikolas Thomopoulos, James Docherty, Peter Schwinger, Anjali Mahendra, and Wanli Fang. Accessibility in Cities: Transport and Urban Form. In *Disrupting Mobility*, pages 239–273. Springer International Publishing, 2017. doi:10.1007/978-3-319-51602-8_15.
- 28 Shmuel Safra and Oded Schwartz. On the complexity of approximating tsp with neighborhoods and related problems. *Computational Complexity*, 14(4):281–307, March 2006. doi:10.1007/s00037-005-0200-3.
- 29 Paulo Henrique Asconavieta da Silva. *O problema do caixeiro viajante alugador: Um estudo algorítmico*. PhD thesis, Universidade Federal do Rio Grande do Norte, 2011.
- 30 Paulo Henrique Asconavieta da Silva, Marco César Goldberg, and Elizabeth Ferreira Gouvêa Goldberg. The car renter salesman problem: An algorithmic study. In *Anais do XLII SBPO Simpósio Brasileiro de Pesquisa Operacional*, 2010.
- 31 Paulo Henrique Asconavieta da Silva, Marco César Goldberg, and Elizabeth Ferreira Gouvêa Goldberg. Evolutionary algorithm for the Car Renter Salesman. In *2011 IEEE Congress of Evolutionary Computation (CEC)*, pages 593–600. IEEE, June 2011. doi:10.1109/cec.2011.5949673.
- 32 Vijay V. Vazirani. *Approximation Algorithms*. Springer, Berlin, 2001. doi:10.1007/978-3-662-04565-7.

- 33 David P. Williamson and David B. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, Cambridge, 2011. doi:10.1017/CB09780511921735.
- 34 Henk Zijm and Matthias Klumpp. Logistics and Supply Chain Management: Developments and Trends. In *Logistics and Supply Chain Innovation*, pages 1–20. Springer International Publishing, August 2015. doi:10.1007/978-3-319-22288-2_1.