Approximation algorithms for car-sharing problems

Kelin Luo
Department of Mathematics and Computer Science, Eindhoven University of Technology, Eindhoven, the Netherlands
k.luo@tue.nl

Frits C. R. Spieksma
Department of Mathematics and Computer Science, Eindhoven University of Technology, Eindhoven, the Netherlands
k.luo@tue.nl

Abstract
We consider several variants of a car-sharing problem. Given are a number of requests each consisting of a pick-up location and a drop-off location, a number of cars, and nonnegative, symmetric travel times that satisfy the triangle inequality. Each request needs to be served by a car, which means that a car must first visit the pick-up location of the request, and then visit the drop-off location of the request. Each car can serve two requests. One problem is to serve all requests with the minimum total travel time (called $CS_{\text{sum}}$), and the other problem is to serve all requests with the minimum total latency (called $CS_{\text{lat}}$). We also study the special case where the pick-up and drop-off location of a request coincide. We propose two basic algorithms, called the match and assign algorithm and the transportation algorithm. We show that the best of the resulting two solutions is a $2$-approximation for $CS_{\text{sum}}$ (and a $7/5$-approximation for its special case), and a $5/3$-approximation for $CS_{\text{lat}}$ (and a $3/2$-approximation for its special case); these ratios are better than the ratios of the individual algorithms. Finally, we indicate how our algorithms can be applied to more general settings where each car can serve more than two requests, or where cars have distinct speeds.

2012 ACM Subject Classification Theory of computation → Approximation algorithms analysis

Keywords and phrases car-sharing, approximation Algorithms, matching

Digital Object Identifier 10.4230/LIPIcs...

Funding Kelin Luo: This project has received funding from the European Union’s Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement number 754462.
Frits C. R. Spieksma: This project has received funding from the NWO Gravitation Project NETWORKS, Grant Number 024.002.003.

1 Introduction
We investigate the following car-sharing problem: there are $n$ cars (or servers), with car $k$ stationed at location $d_k$ ($1 \leq k \leq n$), and there are $2n$ requests, each request $i$ consisting of a pick-up location $s_i$ and a drop-off location $t_i$ ($1 \leq i \leq 2n$). Between every pair of locations $x$ and $y$, a number $w(x, y)$ is given; these numbers can be interpreted as the distance between locations $x$ and $y$ or the time needed to travel between $x$ and $y$. These distances, or times, are non-negative, symmetric, and satisfy the triangle inequality. Each request needs to be served by a car, which means that a car must first visit the pick-up location, and then visit the drop-off location. The car sharing problem is to assign all requests to the cars such that each car serves exactly two requests while minimizing total travel time, and/or while

1 corresponding author

minimizing total waiting time (called total latency), incurred by customers that submitted the requests. Let us elaborate on these two objectives.

- **Minimize total travel time.** The total travel time is the travel time each car drives to serve its requests, summed over the cars. From the company or drivers’ perspective, this is important since this objective reflects minimizing costs while serving all requests. From a societal point of view, this objective also helps to reduce emissions. We use $CS_{sum}$ to refer to the car-sharing problem with the objective to minimize total travel time. The special case of $CS_{sum}$ where the pick-up and drop-off location is identical for each request, is denoted by $CS_{sum,s}=t$.

- **Minimize total latency.** The total latency represents the sum of the travel times needed for each individual customer to arrive at her/his drop-off location, summed over the customers. From the customers’ perspective, this is important because it helps customers to reach their destinations as soon as possible. We use $CS_{lat}$ to refer to the car-sharing problem with the objective to minimize total latency. The special case of $CS_{lat}$ where the pick-up and drop-off location is identical for each request, is denoted by $CS_{lat,s}=t$.

**Motivation**

Consider a working day morning. A large number of requests, each consisting of a pick-up and a drop-off location has been submitted by the customers. The car-sharing company has to assign these requests to available cars. In many practical situations, each request is allowed to occupy at most two seats in a car (see Uber [1]). Thus, in a regular car where at most four seats are available, at most two requests can be combined. Since the company knows the location of the available cars, an instance of our problem arises.

Another application can be found in the area of collective transport for specific groups of people. For instance, the company Transvision [2] organizes collective transport by collecting all requests in a particular region of the Netherlands, combines them, and offers them to regular transport companies. In their setting customers must make their request the evening before the day of the actual transport; the number of requests for a day often exceeds 5,000. However, in this application, it is true that a car may pick up more than two requests (we come back to this issue in Section 5).

The problems $CS_{sum,s}=t$ and $CS_{lat,s}=t$ are natural special cases of $CS_{sum}$ and $CS_{lat}$ respectively, and can be used to model situations where items have to be delivered to clients (whose location is known and fixed). For instance, one can imagine a retailer sending out trucks to satisfy clients’ demand where each truck is used to satisfy two clients.

**Related Work.**

In the literature, car-sharing systems are increasingly studied. Agatz et al. [3] consider the problem of matching cars and customers in real-time with the aim to minimize the total number of vehicle-miles driven. Stiglic et al. [8] show that a small increase in the flexibility of either the cars or the customers can significantly increase the performance of a dynamic car-sharing system. Furthermore, Wang et al. [9] introduce the notion of stability for a car sharing system and they present methods to establish stable or nearly stable solutions. More recently, Ashlagi et al. [4] study the problem of matching requests to cars that arrive online and need to be assigned after at most a prespecified number of time periods. Every pair of requests that is matched yields a profit and the goal of the car sharing system is to match requests with maximum total profit.
Bei and Zhang [5] introduce $CS_{\text{sum}}$, and give a 2.5-approximation algorithm for it. In fact, both $CS_{\text{sum}}$ and $CS_{\text{lat}}$ are a special case of the so-called two-to-one assignment problem (2-1-AP) investigated by Goossens et al. [7]. Given a set $G$ of $n$ green elements, and a set $R$ of $2n$ red elements, we call a triple a set of three elements that consist of a single green element and two red elements; further, there is a cost-coefficient for each triple. The 2-1-AP problem is to find a collection of triples covering each element exactly once, while the sum of the corresponding cost-coefficients is minimized. In the context of our car sharing problem, the green elements represent the cars, and the red elements represent the requests. For the special case of 2-1-AP where the cost of each triple $(i,j,k)$ is defined as the sum of the three corresponding distances, i.e., $cost(i,j,k) = d_{ij} + d_{jk} + d_{ki}$, that satisfy the triangle inequality, Goossens et al. [7] give an algorithm with approximation ratio $4/3$. The definition of the cost-coefficients in $CS_{\text{sum}}$, as well as in $CS_{\text{lat}}$ differs from this expression; we refer to Section 2 for a precise definition.

Our results

We formulate and analyze three polynomial-time approximation algorithms for our car-sharing problems: a match and assign algorithm MA, a transportation algorithm TA, and a combined algorithm CA which runs MA and TA, and then outputs the best of the two solutions. These algorithms are extended versions of algorithms in [7] using as additional input a parameter $\alpha \in \{1, 2\}$ and a vector $v \in \{u, \mu\}$; for their precise description, we refer to Section 3. All of them run in time $O(n^3)$. Here, we establish their worst-case ratio's (see Williamson and Shmoys [10] for appropriate terminology).

| Table 1 Overview of our results |
|---------------------------------|
| Problem                  | MA(1, $u$) | MA(2, $\mu$) | TA(1) | TA(2) | CA(1, $u$) | CA(2, $\mu$) |
| $CS_{\text{sum}}$          | $2^*$ (Lemma 5) | 3            | $3^*$ (Lemma 6) | 4 | $2^*$ (Theorem 7) | 3            |
| $CS_{\text{sum},s=t}$      | $3^*/2^*$ | 3/2          | $3^*$ | 4 | $7/5^*$ (Theorem 8) | 10/7         |
| $CS_{\text{lat}}$           | 4          | $2^*$        | 3     | $2^*$ | 3          | 5/3 (Theorem 10) |
| $CS_{\text{lat},s=t}$       | 2          | $2^*$        | 2     | $2^*$ | 8/5        | $3/2^*$ (Theorem 11) |

For all four problems, we show how the above mentioned algorithms behave with respect to their worst-case ratios. Notice that for $CS_{\text{sum},s=t}$, $CS_{\text{lat}}$ and $CS_{\text{lat},s=t}$, the worst-case ratio of the combined algorithm CA is strictly better than each of the two worst-case ratios of the algorithms CA consists of. An overview of all our results is shown in Table 1; $u$ and $\mu$ are defined in Section 2; an "*" means that the corresponding worst-case ratio is tight; results not attributed to a specific Lemma or Theorem are proven in Appendix C.

The two problems $CS_{\text{sum}}$ and $CS_{\text{lat}}$ only differ in their objectives, i.e., anything that is a feasible solution in $CS_{\text{sum}}$ is also a feasible solution in $CS_{\text{lat}}$ and vice versa. A similar statement can be made for $CS_{\text{sum},s=t}$ and $CS_{\text{lat},s=t}$. Hence, we can use a two-criteria approach to judge a solution; more concretely, we call an algorithm a $(\rho, \varphi)$-approximation algorithm if the cost of its solution is at most $\rho$ times the cost of an optimal solution to $CS_{\text{sum}}$, and at most $\varphi$ times the cost of an optimal solution of $CS_{\text{lat}}$. Therefore, CA(1, $u$) is a $(2, 3)$-approximation (resp. $(\frac{7}{5}, \frac{8}{5})$-approximation) algorithm and CA(2, $\mu$) is a $(4, \frac{7}{5})$-approximation (resp. $(\frac{24}{17}, \frac{17}{12})$-approximation) algorithm. Notice that, from a worst-case perspective, none of these two algorithms dominates the other.
2 Preliminaries

Notation. We consider a setting with \( n \) cars, denoted by \( D = \{1, 2, \ldots, n\} \), with car \( k \) at location \( d_k \) (\( k \in D \)). There are \( 2n \) requests \( R = \{1, 2, \ldots, 2n\} \), where request \( i \) is specified by the pick-up location \( s_i \) and the drop-off location \( t_i \) (\( i \in R \)). The distance (or cost, or travel time) between location \( x_1 \) and \( x_2 \) is denoted by \( w(x_1, x_2) \). We assume here that distances \( w(x_1, x_2) \) are non-negative, symmetric, and satisfy the triangle inequality. Furthermore, we extend the notation of distance between two locations to the distance of a path: \( w(x_1, x_2, \ldots, x_k) = \sum_{i=1}^{k-1} w(x_i, x_{i+1}) \). We want to find an allocation \( M = \{(k, R_k) : k \in D, R_k \subseteq R\} \), where \( |R_k| = 2 \) and \( R_1, R_2, \ldots, R_n \) are pairwise disjoint; the set \( R_k \) contains the two requests that are assigned to car \( k \). For each car \( k \in D \) and \( (k, R_k) \in M \) where \( R_k \) contains request \( i \) and request \( j \), i.e., \( R_k = \{i, j\} \), we denote the travel time of serving the requests in \( R_k \) by:

\[
\text{cost}(k, \{i, j\}) = \min\{w(d_k, s_i, t_i, t_j), w(d_k, s_i, t_j, t_i), w(d_k, s_j, t_i, s_j, t_j), w(d_k, s_j, t_i, t_j), w(d_k, s_i, t_i, s_j, t_j), w(d_k, s_i, t_i, t_j)\}. \tag{1}
\]

Notice that the six terms in (1) correspond to all distinct ways of visiting the locations \( s_i, s_j, t_i \) and \( t_j \) where \( s_i \) is visited before \( t_i \) and \( s_j \) is visited before \( t_j \). We view \( \text{cost}(k, \{i, j\}) \) as consisting of two parts: one term expressing the travel time between \( d_k \) and the first pick-up location \( s_i \) or \( s_j \); and another term capturing the travel time from the first pick-up location to the last drop-off location. For convenience, we give a definition of the latter quantity.

\[
u_{ij} \equiv \min\{w(s_i, s_j, t_i, t_j), w(s_i, s_j, t_j, t_i), w(s_i, t_i, s_j, t_j)\}
\quad \text{for each } i, j \in R \times R, i \neq j. \tag{2}
\]

Notice that the \( u_{ij} \)'s are not necessarily symmetric. For \( \text{CS}_{\text{sum},w} \), \( u_{ij} \equiv w(s_i, s_j) \). Using (1) and (2), the travel time needed to serve requests in \( R_k = \{i, j\} \) (\( k \in D \)) is then given by:

\[
\text{cost}(k, \{i, j\}) = \min\{w(d_k, s_i) + u_{ij}, w(d_k, s_j) + u_{ji}\}. \tag{3}
\]

We denote the travel time of an allocation \( M \) by:

\[
\text{cost}(M) = \sum_{(k, R_k) \in M} \text{cost}(k, R_k). \tag{4}
\]

In \( \text{CS}_{\text{sum}} \) and \( \text{CS}_{\text{sum},w} \), the goal is to find an allocation \( M \) that minimizes \( \text{cost}(M) \).

Let us now consider \( \text{CS}_{\text{lat}} \). Here, we focus on the waiting time as perceived by an individual customer, from the moment the car leaves its location until the moment the customer reaches its drop-off location. More formally, we denote the latency of serving requests in \( R_k = \{i, j\} \) by:

\[
\text{wait}(k, \{i, j\}) \equiv \min\{w(d_k, s_i, s_j, t_i), w(d_k, s_i, s_j, t_j), w(d_k, s_i, t_i, t_j), w(d_k, s_i, t_i, s_j), w(d_k, s_j, s_i, t_i), w(d_k, s_j, s_i, t_j)\}. \tag{5}
\]

Again, the six terms in (5) correspond to all distinct ways of visiting the locations \( s_i, s_j, t_i \) and \( t_j \) where \( s_i \) is visited before \( t_i \) and \( s_j \) is visited before \( t_j \). We view \( \text{wait}(k, R_k) \) as consisting of two parts: one term expressing the waiting time between \( d_k \) and the first pick-up location \( s_i \) or \( s_j \); and another term capturing the waiting time from the first pick-up
location to the last drop-off location. For convenience, we give a definition of the latter quantity.

\[
\mu_{ij} \equiv \min \{w(s_i, s_j, t_j), w(s_i, s_j, t_i), w(s_i, t_i) + w(s_i, s_j, t_i, t_j) + w(s_i, s_j, t_i) + w(s_i, s_j, t_j, t_i)\} \text{ for each } i, j \in R \times R, i \neq j. \tag{6}
\]

Notice that the \(\mu_{ij}\)'s are not necessarily symmetric. For \(CS_{lat,s=t}\), \(\mu_{ij} \equiv w(s_i, s_j)\). Using (5) and (6), the latency needed to serve requests in \(R_k = \{i, j\} (k \in D)\) is then given by:

\[
\text{wait}(k, \{i, j\}) = \min \{2w(d_k, s_i) + \mu_{ij}, 2w(d_k, s_j) + \mu_{ji}\}. \tag{7}
\]

We denote the latency of an allocation \(M\) by:

\[
\text{wait}(M) = \sum_{(k, R_k) \in M} \text{wait}(k, R_k). \tag{8}
\]

Thus, in \(CS_{lat}\) and \(CS_{lat,s=t}\), the goal is to find an allocation \(M\) that minimizes \(\text{wait}(M)\).

A natural variant of the latency objective is one where the latency is counted with respect to the pick-up location as opposed to the drop-off location in \(CS_{lat}\). Clearly, then the drop-off location becomes irrelevant, and in fact our approximation results for \(CS_{lat,s=t}\) become valid for this variant.

Remark. Bei and Zhang [5] prove that the special car-sharing problem, i.e., \(CS_{sum,s=t}\) is NP-hard. Their proof can also be used to prove that \(CS_{lat,s=t}\) is NP-hard. In fact, we point out that the arguments presented in Goossens et al. [7] allow to establish the APX-hardness of these two problems.

Paper Outline In Section 3, we present two algorithms, i.e., the match and assign algorithm and the transportation algorithm. In Section 4, we prove the approximation ratios. Section 5 concludes the paper.

3 Algorithms

We give three polynomial-time approximation algorithms for our car-sharing problems. In Section 3.1 we describe the match and assign algorithm \(MA(\alpha, v)\), and in Section 3.2 we describe the transportation algorithm \(TA(\alpha)\). As described before, the third algorithm, \(CA(\alpha, v)\), simply runs \(MA(\alpha, v)\) and \(TA(\alpha)\), and then outputs the best of the two solutions. Let \(\alpha \in \{1, 2\}\) be the coefficient that weights the travel time between the car location and the first pick-up location, and let \(v \in \{u, \mu\}\) (where \(u\) is defined in (2), and \(\mu\) is defined in (6)).

3.1 The match and assign algorithm

The match-and-assign algorithm \(MA(\alpha, v)\) goes through two steps: in the first step, the algorithm pairs the requests based on their combined serving cost, and in the second step, the algorithm assigns the request-pairs to the cars.

The key characteristics of algorithm \(MA(\alpha, v)\) are found in lines 4 and 9 where the costs of the first and second step are defined. A resulting quantity is \(v_1(M_1) + v_2(M_2)\); we now prove two lemma’s concerning this quantity, which will be of use in Section 4.

Lemma 1. For each \(\alpha \in \{1, 2\}\) and \(v \in \{u, \mu\}\), we have:

\[
v_1(M_1) + v_2(M_2) = \sum_{(k, \{i, j\}) \in M_2} \min \{\alpha w(d_k, s_i) + v_{ij}, \alpha w(d_k, s_j) + v_{ji}\}.
\]
Approximation algorithms for car-sharing problems

Algorithm 1 Match-and-assign algorithm (MA(α, v))

1: Input: non-negative weighted graph \( G = (V, E, w) \), requests \( R = \{i = (s_i, t_i) : 1 \leq i \leq m, s_i, t_i \in V \} \), cars \( D = \{k : 1 \leq k \leq n, d_k \in V \} \), \( \alpha \in \{1, 2\} \) and \( v \in \{u, \mu\} \).
2: Output: An allocation \( MA = \{(k, \{i, j\}) : k \in D, i, j \in R\} \).
3: For \( i, j \in R \) do
4: \( v_1(i, j) = \frac{v_{ij} + v_{ji}}{2} \)
5: end for
6: Let \( G_1 \equiv (R, v_1) \) be the complete weighted graph where an edge between vertex \( i \in R \) and vertex \( j \in R \) has weight \( v_1(i, j) \).
7: Find a minimum weight perfect matching \( M_1 \) in \( G_1 \equiv (R, v_1) \) with weight \( v_1(M_1) \).
8: For \( k \in D \) and \( \{i, j\} \in M_1 \) with \( v_{ij} \geq v_{ji} \) do
9: \( v_2(k, \{i, j\}) = \min\{\alpha w(d_k, s_i) + \frac{v_{ij} - v_{ji}}{2}, \alpha w(d_k, s_j) - \frac{v_{ij} - v_{ji}}{2}\} \)
10: end for
11: Let \( G_2 \equiv (D \cup M_1, v_2) \) be the complete bipartite graph with left vertex-set \( D \), right vertex-set \( M_1 \) and edges with weight \( v_2(k, \{i, j\}) \) for \( k \in D \), and \( \{i, j\} \in M_1 \).
12: Find a minimum weight perfect matching \( M_2 \) in \( G_2 \equiv (D \cup M_1, v_2) \) with weight \( v_2(M_2) \).
13: Output allocation \( MA = M_2 \).

Proof. Without loss of generality, for any \( \{i, j\} \in M_1 \), suppose \( v_{ij} - v_{ji} \geq 0 \) (the other case is symmetric).

\[
v_1(M_1) + v_2(M_2) = \sum_{\{i,j\} \in M_1} \frac{v_{ij} + v_{ji}}{2} + \sum_{\{k,\{i,j\}\} \in M_2} \min\{\alpha w(d_k, s_i) + \frac{v_{ij} - v_{ji}}{2}, \alpha w(d_k, s_j) - \frac{v_{ij} - v_{ji}}{2}\}
= \sum_{\{k,\{i,j\}\} \in M_2} \min\{\alpha w(d_k, s_i) + v_{ij}, \alpha w(d_k, s_j) + v_{ji}\}.
\]

The first equality follows from lines 4 and 9 in Algorithm 1.

Lemma 2. For \( \alpha \in \{1, 2\} \), \( v \in \{u, \mu\} \), and for each allocation \( M \), we have:

\[
v_1(M_1) + v_2(M_2) \leq \frac{1}{2} \sum_{\{k,\{i,j\}\} \in M} (\alpha w(d_k, s_i) + w(d_k, s_j)) + v_{ij} + v_{ji}.
\]

Proof. For an allocation \( M \), let \( M_R = \{R_k : (k, R_k) \in M\} \). Observe that

\[
v_1(M_1) \leq \sum_{\{i,j\} \in M_R} \frac{v_{ij} + v_{ji}}{2}, \tag{9}
\]

since \( M_1 \) is a minimum weight perfect matching in \( G_1 \equiv (R, v_1) \).

We claim that

\[
v_2(M_2) \leq \frac{1}{2} \sum_{\{k,\{i,j\}\} \in M} \alpha w(d_k, s_i) + w(d_k, s_j). \tag{10}
\]

When summing (9) and (10), the lemma follows.

Hence, it remains to prove (10). Consider an allocation \( M \), and consider the matching \( M_1 \) found in the first step of MA. Based on \( M \) and \( M_1 \), we construct the graph \( G' = (R \cup D, M_1 \cup \{(i, j, k) : (k, \{i, j\}) \in M\}) \). Note that every vertex in graph \( G' \) has degree 2. Thus, we can partition \( G' \) into a set of disjoint cycles called \( C \); each cycle \( c \in C \) can
be written as \( c = (i_1, j_1, k_1, i_2, j_2, k_2, \ldots, i_h, j_h, i_1) \), where \( \{i_s, j_s\} \in M_1, (k_h, \{j_h, i_1\}) \in M \) for \( 1 \leq s < h \) and \( (k_h, \{j_h, i_1\}) \in M \). Consider now, for each cycle \( c \in C \), the following two matchings called \( M^c_\ell \) and \( M^c_r \):

\[ M^c_\ell = \{ \{i_1, j_1\}, \{i_2, j_2\}, \ldots, \{i_h, j_h\}, k_h \} \]

\[ M^c_r = \{ \{k_1, i_2\}, \{k_2, i_3\}, \ldots, \{k_h, i_1\} \} \]

Obviously, both \( M_\ell = \bigcup_{c \in C} M^c_\ell \), and \( M_r = \bigcup_{c \in C} M^c_r \) are a perfect matching in \( G_2 = (D \cup M_1, v_2) \). Given the definition of \( v_2(k, \{i, j\}) \) (see line 9 of Algorithm MA), we derive for each pair of requests \( \{i, j\} \) and two cars \( a, b : v_2(a, \{i, j\}) + v_2(b, \{i, j\}) \leq \alpha w(d_a, i) + v_2 \left( \left\{ \frac{\alpha}{2} \right\} w(d_b, j) - \frac{1}{2} \right) = \alpha (w(d_a, i) + w(d_b, j)) \). Similarly, it follows that: \( v_2(a, \{i, j\}) + v_2(b, \{i, j\}) \leq \alpha (w(d_a, i) + w(d_b, j)) \). Thus, for each \( c \in C \):

\[
\sum_{(k, \{i, j\}) \in M^c_\ell} v_2(k, \{i, j\}) + \sum_{(k, \{i, j\}) \in M^c_r} v_2(k, \{i, j\}) \leq \sum_{(k, \{i, j\}) \in M} \alpha (w(d_k, s_i) + w(d_k, s_j)). \tag{11}
\]

Note that \( M_2 \) is a minimum weight perfect matching in \( G_2 = (D \cup M_1, v_2) \), and both \( M_\ell \) and \( M_r \) are a perfect matching in \( G_2 = (D \cup M_1, v_2) \). Thus:

\[
v_2(M_2) \leq \sum_{c \in C} \min \{v_2(M^c_\ell), v_2(M^c_r)\}
\]

\[
\leq \frac{1}{2} \sum_{c \in C} \left( \sum_{(k, \{i, j\}) \in M^c_\ell} v_2(k, \{i, j\}) + \sum_{(k, \{i, j\}) \in M^c_r} v_2(k, \{i, j\}) \right)
\]

\[
\leq \frac{1}{2} \sum_{(k, \{i, j\}) \in M} \alpha (w(d_k, s_i) + w(d_k, s_j)).
\]

The last inequality follows from (11), and hence (10) is proven.

### 3.2 The transportation algorithm

In this section, we present the transportation algorithm. The idea of the algorithm is to replace each car \( k \in D \) by 2 virtual cars called \( \gamma(k) \) and \( \delta(k) \), resulting in two car sets \( \Gamma = \{\gamma(1), \ldots, \gamma(n)\} \) and \( \Delta = \{\delta(1), \ldots, \delta(n)\} \). Next we assign the requests to the \( 2n \) cars using a particular definition of the costs; a solution is found by letting car \( k \in D \) serve the requests assigned to car \( \gamma(k) \) and \( \delta(k) \).

**Algorithm 2** Transportation algorithm (TA(\( \alpha \))

1: **Input:** non-negative weighted graph \( G = (V, E, w) \), requests \( R = \{i = (s_i, t_i) : 1 \leq i \leq m, s_i, t_i \in V\} \), cars \( D = \{k : 1 \leq k \leq n, d_k \in V\} \), two virtual car sets \( \Gamma = \{\gamma(1), \ldots, \gamma(n)\} \), and \( \Delta = \{\delta(1), \ldots, \delta(n)\} \), and \( \alpha \in \{1, 2\} \).

2: **Output:** An allocation \( \text{TA} = \{(k, \{i, j\}) : k \in D, i, j \in R\} \).

3: For \( k \in D, i \in R \) do

4: \( v_3(\gamma(k), i) = \alpha w(d_k, s_i, t_i) + w(t_i, d_k) \)

5: \( v_3(\delta(k), i) = w(d_k, s_i, t_i) \)

6: end for

7: Let \( G_3 \equiv (\Gamma \cup \Delta \cup R, v_3) \) be the complete bipartite graph with \emph{left} vertex-set \( \Gamma \cup \Delta \), \emph{right} vertex-set \( R \), and edges with weight \( v_3(x, i) \) for \( x \in \Gamma \cup \Delta \) and \( i \in R \).

8: Find a minimum weight perfect matching \( M_3 \) in \( G_3 \equiv (\Gamma \cup \Delta \cup R, v_3) \) with weight \( v_3(M_3) \).

9: Output allocation \( \text{TA} = M_4 \equiv \{(k, \{i, j\}) : (\gamma(k), i), (\delta(k), j) \in M_3, k \in D\} \).
XX:8 Approximation algorithms for car-sharing problems

The crucial points of algorithm TA(α) are found in lines 4 and 5 where the costs of assigning a request are defined; a resulting quantity is \( v_3(M_3) \). We now prove two lemmas concerning this quantity \( v_3(M_3) \), which will be of use in Section 4.

- **Lemma 3.** For each \( \alpha \in \{1, 2\} \), we have:
  \[
  v_3(M_3) = \sum_{(k, (i,j)) \in M_4} \min\{\alpha w(d_k, s_i, t_i) + w(t_i, d_k, s_j, t_j), \alpha w(d_k, s_j, t_j) + w(t_j, d_k, s_i, t_i)\}.
  \]

  This lemma follows directly from the definition of the costs in lines 4 and 5 of TA.

- **Lemma 4.** For \( \alpha \in \{1, 2\} \) and for each allocation \( M \), we have:
  \[
  v_3(M_3) \leq \sum_{(k, (i,j)) \in M} \min\{\alpha w(d_k, s_i, t_i) + w(t_i, d_k, s_j, t_j), \alpha w(d_k, s_j, t_j) + w(t_j, d_k, s_i, t_i)\}.
  \]

  This lemma is obvious since \( M_3 \) is a minimum weight perfect matching.

  **Remark.** Both MA(α, v) and TA(α) runs in time \( O(n^3) \) since a minimum matching \( M \) in a weighted graph of \( n \) vertices can be found in time \( O(n^3) \) [6].

4 Approximation results

In this section, we analyze the combined algorithm CA(α, v), i.e., the best of the two algorithms MA(α, v) and TA(α), for \( CS_{sum} \), \( CS_{sum,s=t} \), \( CS_{lat} \) and \( CS_{lat,s=t} \). We denote the allocation by the match and assign algorithm MA(α, v) (resp. the transportation algorithm TA(α)) by MA (resp. TA). We denote an optimal allocation of a specific problem by \( M^* = \{(k, R_k^i) : k \in D\} \) with \( R_k^i = \{i, j\} \) \((i, j) \in R\). Let \( M_3 = \{(k, R_k^i) : (k, R_k^i) \in M^*\} \) denote the pairs of requests in \( M^* \). With a slight abuse of notation, we use CA(I) to denote the allocation found by CA for instance \( I \).

4.1 Approximation results for \( CS_{sum} \)

We first establish the worst-case ratios of MA(1, u) and TA(1), and next prove that CA(1, u) is a 2-approximation algorithm.

- **Lemma 5.** MA(1, u) is a 2-approximation algorithm for \( CS_{sum} \).

  **Proof.** We assume wlog that, for each \( (k, \{i, j\}) \in M^* \), \( \text{cost}(k, \{i, j\}) = w(d_k, s_i) + w(d_k, s_j) + u_{ij} \). We have:

  \[
  \text{cost}(MA(1, u)) = \sum_{(k, \{i,j\}) \in M} \min\{w(d_k, s_i) + u_{ij}, w(d_k, s_j) + u_{ji}\} \quad (\text{by } (3) \text{ and } (4))
  \]

  \[
  = v_1(M_1) + v_2(M_2) \quad (\text{by Lemma } [1])
  \]

  \[
  \leq \frac{1}{2} \sum_{(k, (i,j)) \in M^*} (w(d_k, s_i) + w(d_k, s_j) + w(t_i, d_k, s_j) + u_{ij}) \quad (\text{by Lemma } [2])
  \]

  \[
  \leq \frac{1}{2} \sum_{(k, (i,j)) \in M^*} (2w(d_k, s_i) + w(s_i, s_j) + 3w_{ij})
  \]

  \[
  \leq \frac{1}{2} \sum_{(k, (i,j)) \in M^*} (2w(d_k, s_i) + 4w_{ij}) \quad (\text{since } w(s_i, s_j) \leq w_{ij})
  \]

  \[
  \leq \frac{1}{2} \sum_{(k, (i,j)) \in M^*} 4\text{cost}(k, \{i, j\}) \quad (\text{by the assumption } \text{cost}(k, \{i, j\}) = w(d_k, s_i) + w_{ij})
  \]

  \[
  = 2 \text{ cost}(M^*). \quad (12)
  \]
The second inequality follows from the triangle inequality, and since \( u_{ij} \leq 2u_{ij} \) for each request pair \( \{i, j\} \in R^2 \); the corresponding proof can be found in Appendix A.1.\[\phantom{\text{Proof.}}\]

Notice that the statement in Lemma 5 is actually tight by the instance depicted in Figure 1.

\[\textbf{Lemma 6.} \ TA(1) \text{ is a 3-approximation algorithm for } CS_{sum}.\]

\[\textbf{Proof.} \text{ We assume wlog that, for each } (k, \{i, j\}) \in M^*, \text{ cost}(k, \{i, j\}) = w(d_k, s_i) + u_{ij}. \text{ We have:}\]

\[
\begin{align*}
\text{cost}(\text{TA}(1)) &= \sum_{(k, \{i, j\}) \in TA} \min\{w(d_k, s_i) + u_{ij}, w(d_k, s_j) + u_{ij}\} \quad \text{(by (3))} \\
&\leq v_3(M_4) \quad \text{(by Lemma 3)} \\
&\leq \sum_{(k, \{i, j\}) \in M^*} w(d_k, s_i, t_i, d_k, s_j, t_j) \quad \text{(by Lemma 4)} \\
&\leq \sum_{(k, \{i, j\}) \in M^*} 3\text{cost}(k, \{i, j\}) \quad \text{(by cost}(k, \{i, j\}) = w(d_k, s_i) + u_{ij}) \\
&= 3\ \text{cost}(M^*)
\end{align*}
\]

(13)

The third inequality holds since \( 3\text{cost}(k, \{i, j\}) \geq w(d_k, s_i, t_i, d_k, s_j, t_j) \) for each \( (k, \{i, j\}) \in R. \) The proof can be found in Appendix A.2.\[\phantom{\text{Proof.}}\]

Notice that the statement in Lemma 6 is actually tight by the instance depicted in Figure 2 restricted to the subgraph induced by the nodes \((s_2, s_6), (k_3, k_4), \) and \((s_7, s_8).\)

Recall that the combined algorithm CA(1, u) runs MA(1, u) and TA(1), and then outputs the best of the two solutions. We now state the main result of this section.

\[\textbf{Theorem 7.} \ CA(1, u) \text{ is a 2-approximation algorithm for } CS_{sum}. \text{ Moreover, there exists an instance } I \text{ for which cost}(CA(I)) = 2\ \text{cost}(M^*(I)).\]

\[\textbf{Proof.} \text{ It is obvious that, as cost}(CA(1, u)) = \min\{\text{cost}(MA(1, u)), \text{cost}(TA(1))\}, \text{ Lemma’s 5 and 6 imply that } CA(1, u) \text{ is a 2-approximation algorithm for } CS_{sum}. \text{ We now provide an instance for which this ratio is achieved.}\]

Consider the instance \( I \) depicted in Figure 1. This instance has \( n = 2 \) with \( D = \{1, 2\} \) and \( R = \{1, 2, 3, 4\}. \) Locations corresponding to distinct vertices in Figure I are at distance 1. Observe that an optimal solution is \( M^*(I) = \{(k_1, \{1, 3\}), (k_2, \{2, 4\})\} \) with cost\((M^*(I)) = 2. \)

Note that \( M_R = \{(1, 3), \{2, 4\}\}. \) Let us now analyse the performance of MA(1, u) and TA(1) on instance \( I.\)

Based on the \( u_{ij} \) values as defined in \( (3), \) MA(1, u) can find, in the first step, matching \( M_1 = \{(1, 2), \{3, 4\} \text{ with } v_1(M_1) = 3. \) Then, no matter how the second step matches the pairs to cars (since two cars stay at the same location), the total cost of MA(1, u) will be 4. TA(1) can assign request 1 to car \( \gamma(1), \) and request 2 to car \( \delta(1), \) and similarly, request 3 to car \( \gamma(2), \) and request 4 to car \( \delta(2). \) Note that \( v_3(\{(k_1, 1), (k_1, 2), (k_2, 3), (k_2, 4)\}) = v_3(\{(k_1, 1), (k_1, 3), (k_2, 2), (k_2, 4)\}) = 6. \) Thus, the total cost of TA(1) is 4.

To summarize, the instance in Figure 1 is a worst-case instance for the combined algorithm CA(1, u).\[\phantom{\text{Proof.}}\]
Approximation algorithms for car-sharing problems

4.2 Approximation results for CS_{sum,s=t}

Let us now consider the special case of CS_{sum} where the pick-up and drop-off location is identical to each request, CS_{sum,s=t}. From (12) and (13), we have cost(MA(1,u)) = v_1(M_1) + v_2(M_2) and cost(TA(1)) ≤ v_3(M_3). Note that u_{ij} = u_{ji} = w(s_i, s_j) in CS_{sum,s=t}.

Now, we show that for CS_{sum,s=t} the algorithm CA(1,u) is a 7/5-approximation algorithm, a ratio which is strictly better than the ratio of MA(1,u) and TA(1) for this problem.

**Theorem 8.** CA(1,u) is a 7/5-approximation algorithm for CS_{sum,s=t}. Moreover, there exists an instance I for which cost(CA(I)) = 7/5 cost(M^*(I)).

**Proof.** We assume wlog that, for each \((k, \{i, j\}) \in M^*, cost(k, \{i, j\}) = w(d_k, s_i) + u_{ij}\). We have:

\[
5 \text{cost}(CA(1,u)) \leq 4 \text{cost}(MA(1,u)) + \text{cost}(TA(1)) \\
\leq 4(v_1(M_1) + v_2(M_2)) + v_3(M_3) \quad \text{(using (12) and (13))} \\
\leq \sum_{(k, \{i, j\}) \in M^*} (4w(d_k, s_i) + 3w(d_k, s_j) + 4w(s_i, s_j)) \quad \text{(by Lemma's 2 and 4)} \\
\leq \sum_{(k, \{i, j\}) \in M^*} (7w(d_k, s_i) + 7w(s_i, s_j)) \quad \text{(by the triangle inequality)} \\
= \sum_{(k, \{i, j\}) \in M^*} 7 \text{cost}(k, \{i, j\}) \quad \text{(by cost(k, \{i, j\}) = w(d_k, s_i) + u_{ij})} \\
= 7 \text{cost}(M^*) \quad \text{(14)}
\]

We now provide an instance for which this ratio is achieved. Consider the instance I depicted in Figure 1. If two points are not connected by an edge, their distance equals 5. Observe that an optimal solution is \(\{(k_1, \{1, 3\}), (k_2, \{2, 4\}), (k_3, \{5, 6\}), (k_4, \{7, 8\})\}\) with cost(M^*(I))=10. Note that \(M^*_R = \{\{1, 3\}, \{2, 4\}, \{5, 6\}, \{7, 8\}\}\). Let us now analyse the performance of MA(1,u) and TA(1) on instance I.

Based on the \(u_{ij}\) values as defined in (3), MA(1,u) can find, in the first step, matching \(M_1 = \{\{1, 2\}, \{3, 4\}, \{5, 6\}, \{7, 8\}\}\). Then, no matter how the second step matches the pairs to cars (since two cars stay at the same location), the total cost of MA(1,u) will be 14.

TA(1) can assign request 1 to car \(\gamma(1)\), and request 3 to car \(\delta(1)\), and similarly, request 2 to car \(\gamma(2)\), and request 4 to car \(\delta(2)\); request 5 to car \(\gamma(3)\), and request 7 to car \(\delta(3)\); request 4 to car \(\gamma(4)\), and request 6 to car \(\delta(4)\). Note that \(v_3(\{(k_3, 5), (k_3, 7), (k_4, 6), (k_4, 8)\}) = v_3(\{(k_3, 5), (k_3, 6), (k_4, 7), (k_4, 8)\}) = 6\). Thus the total cost of TA(1) is 14.

\[\text{Figure 1} \quad \text{A worst-case instance for the combined algorithm CA(1,u) of CS_{sum}.}\]
To summarize, the instance in Figure 2 is a worst-case instance for the combined algorithm CA(1, u).

4.3 Approximation results for CSlat

For CSlat, we analyze algorithm CA(2, µ), which outputs the best of the two solutions, MA(2, µ) and TA(2).

Lemma 9. For each \((k, \{i, j\}) \in D \times R^2\), \(2w(d_k, s_i) + 2w(d_k, s_j) + \mu_{ij} + \mu_{ji} + \min\{2w(d_k, s_i, t_i) + w(t_i, d_k, s_j, t_j), 2w(d_k, s_j, t_j) + w(t_j, d_k, s_i, t_i)\} \leq \min\{8w(d_k, s_i) + 5\mu_{ij}, 8w(d_k, s_j) + 5\mu_{ji}\}.

The proof can be found in Appendix A.3.

Theorem 10. CA(2, µ) is a 5/3-approximation algorithm for CSlat.

Proof. 

\[
3\text{wait}(CA(2, \mu)) \leq 2\text{wait}(MA(2, \mu)) + \text{wait}(TA(2)) \\
\leq 2(v_1(M_1) + v_2(M_2) + v_3(M_3)) \quad \text{(by Lemma 1 and Lemma 2)} \\
\leq \sum_{\langle k, \{i, j\}\rangle \in M^*} \min\{8w(d_k, s_i) + 5\mu_{ij}, 8w(d_k, s_j) + 5\mu_{ji}\} \\
\leq \sum_{\langle k, \{i, j\}\rangle \in M^*} 5\min\{2w(d_k, s_i) + \mu_{ij}, 2w(d_k, s_j) + \mu_{ji}\} \\
= 5 \text{wait}(M^*). 
\]

The third inequality follows from Lemma 2, Lemma 3, and Lemma 9, i.e., \(2(v_1(M_1) + v_2(M_2) + v_3(M_3) \leq \sum_{\langle k, \{i, j\}\rangle \in M^*}(2w(d_k, s_i) + 2w(d_k, s_j) + \mu_{ij} + \mu_{ji} + \min\{2w(d_k, s_i, t_i) + w(t_i, d_k, s_j, t_j), 2w(d_k, s_j, t_j) + w(t_j, d_k, s_i, t_i)\}) \leq \sum_{\langle k, \{i, j\}\rangle \in M^*} \min\{8w(d_k, s_i) + 5\mu_{ij}, 8w(d_k, s_j) + 5\mu_{ji}\}. \)

4.4 Approximation results for CSlat,s,t

Let us now consider the special case of CSlat where the pick-up and drop-off location is identical to each request, CSlat,s,t. Note that \(\mu_{ij} = \mu_{ji} = w(s_i, s_j)\). Now, we show that for CSlat,s,t algorithm CA(2, µ) is a \(3/2\)-approximation algorithm, a ratio which is strictly better than the ratio’s of MA(2, µ) and TA(2) for this problem.

Theorem 11. CA(2, µ) is a 3/2-approximation algorithm for CSlat,s,t. Moreover, there exists an instance I for which \(\text{wait}(CA(I)) = 3/2 \text{wait}(M^*(I))\).
Approximation algorithms for car-sharing problems

We have:

$$\text{wait}(M^*) \leq \text{wait}(\text{TA}(2))$$

Proof. We assume wlog that, for each $(k, \{i, j\}) \in M^*$, $\text{wait}(k, \{i, j\}) = 2w(d_{k}, s_{i}) + \mu_{ij}$.

We have:

$$2\text{wait}(CA(2, \mu)) \leq \text{wait}(MA(2, \mu)) + \text{wait}(\text{TA}(2))$$

$$\leq \sum_{(k, \{i, j\}) \in MA} \min\{2w(d_{k}, s_{i}) + \mu_{ij}, 2w(s_{k}, s_{j}) + \mu_{ji}\}$$

$$+ \sum_{(k, \{i, j\}) \in TA} \min\{2w(d_{k}, s_{i}) + \mu_{ij}, 2w(s_{k}, s_{j}) + \mu_{ji}\} \quad \text{(by (7) and (8))}$$

$$\leq v_1(M_1) + v_2(M_2) + v_3(M_3) \quad \text{(by Lemma 1 and Lemma 3)}$$

$$\leq \sum_{(k, \{i, j\}) \in M^*} (w(s_{i}, s_{j}) + w(d_{k}, s_{i}) + w(d_{k}, s_{j}) + 3w(d_{k}, s_{i}) + w(d_{k}, s_{j}))$$

$$\leq \sum_{(k, \{i, j\}) \in M^*} (3w(s_{i}, s_{j}) + 6w(d_{k}, s_{i})) \quad \text{(by the triangle inequality)}$$

$$= \sum_{(k, \{i, j\}) \in M^*} 3\text{wait}(k, \{i, j\}) \quad \text{(by \text{wait}(k, \{i, j\}) = 2w(d_{k}, s_{i}) + \mu_{ij})}$$

$$= 3\text{wait}(M^*).$$

The fourth inequality holds by Lemma 2 and 4.

We now provide an instance for which this ratio is achieved. Consider the instance $I$ depicted in Figure 3. If two points are not connected by an edge, their distance equals 5. Observe that an optimal solution is $\{(k_1, \{1, 3\}), (k_2, \{2, 4\}), (k_3, \{5, 6\}), (k_4, \{7, 8\})\}$ with $\text{wait}(M^*(I))=8$. Note that $M^*_R = \{(1, 3), (2, 4), (5, 6), (7, 8)\}$. Let us now analyse the performance of $MA(2, \mu)$ and $\text{TA}(2)$ on instance $I$.

Based on the $u_{ij}$ values as defined in (3), $MA(2, \mu)$ can find, in the first step, matching $M_1 = \{(1, 2), (3, 4), (5, 6), (7, 8)\}$. Then, no matter how the second step matches the pairs to cars, the total cost of $MA(2, \mu)$ will be 12.

$\text{TA}(2)$ can assign request 3 to car $\gamma(1)$ and request 1 to car $\delta(1)$, and similarly, request 4 to car $\gamma(2)$ and request 2 to car $\delta(2)$; request 5 to car $\gamma(3)$ and request 7 to car $\delta(3)$; request 4 to car $\gamma(4)$ and request 6 to car $\delta(4)$. Note that $v_3(\{(k_3, 5), (k_3, 7), (k_4, 6), (k_4, 8)\}) = v_1(\{(k_3, 5), (k_3, 6), (k_4, 7), (k_4, 8)\}) = 8$. Thus the total cost of $\text{TA}(2)$ is 12.

To summarize, the instance in Figure 3 is a worst-case instance for the combined algorithm $CA(2, \mu)$. \hfill □
5 Conclusions

We have analyzed two algorithms for four different versions of a car sharing problem. One algorithm, called match and assign, first matches the requests into pairs, and then assigns the pairs to the cars. Another algorithm, called transportation assigns two requests to each car. These two algorithms emphasize different ingredients of the total cost and the total latency. Accordingly, we have proved that (for most problem variants) the worst-case ratio of the algorithm defined by the best of the two corresponding solutions is strictly better than the worst-case ratios of the individual algorithms.

We point out that the algorithms can be generalized to handle a variety of situations. We now list these situations, and shortly comment on the corresponding worst-case behavior.

Generalized car-sharing problem: \(|R| = a \cdot n\). In this situation, \(a\cdot n\) requests are given, and each car can serve \(a\) requests, \(a \geq 2\). We can extend algorithm TA for the resulting problem by replacing a single car by a cars, and use an appropriately defined cost between a request in \(R\) and a car. We obtain that the extended TA algorithm is a \((2a - 1)\)-approximation for \(CS_{sum}\) and an \(a\)-approximation for \(CS_{lat}\), generalizing our results for \(a = 2\). We refer to Appendix B for the proofs.

Related car-sharing problem: different speed. In this situation, we allow that cars have different speeds. Indeed, let car \(k\) have speed \(p_k\) for each \(k \in D\). We denote the travel time of serving requests in \(R_k\) by \(\text{cost}(k, R_k)/p_k\) and the travel time of an allocation \(M\) by \(\sum_{(k, R_k) \in M} \text{cost}(k, R_k)/p_k\). Analogously, we can adapt the latency of serving requests in \(R_k\) and the latency of an allocation \(M\). Although it is unclear how to generalize algorithm MA, we can still use algorithm TA(\(\alpha\)) in this situation. By defining \(v_3(k, i)\) in terms of the cost above, we get the worst-case ratios of TA(\(\alpha\)) as shown in Table 1.

Car redundancy or deficiency: \(2|D| > |R|\) or \(2|D| < |R|\). We shortly sketch how to modify TA(\(\alpha\)) for this situation. For the problem with \(2|D| > |R|\), by adding a number of dummy requests \(R_d\) with \(|R_d| = 2n - |R|\), where the distance between any two requests in \(R_d\) is 0, the distance between a request in \(R_d\) and \(R\) is a large constant, and the distance between a request in \(R_d\) and a car in \(D\) is 0, an instance of our problem arises. For the problem with \(2|D| < |R|\), by adding a number of dummy cars \(D_d\) with \(|D_d| = \lceil|R|/2 \rceil - n\), where the distance between a car in \(D_d\) and a request in \(R\) is 0, an instance of our problem arises. We claim that minor modifications of the proofs for TA imply that TA(1) is a 3-approximation algorithm for \(CS_{sum}\), and TA(2) is a 2-approximation algorithm for \(CS_{lat}\). On the other hand, it is not clear how to generalize MA(\(\alpha, v\)) for this situation.

References

1 How uberpool works. https://www.uber.com/nl/en/ride/uberpool/ Accessed: 2020-01-20.
2 Transvision. https://www.transvision.nl/ Accessed: 2020-01-23.
3 Niels Agatz, Ann Campbell, Moritz Fleischmann, and Martin Savelsbergh. Time slot management in attended home delivery. Transportation Science, 45(3):435–449, 2011.
4 Itai Ashlagi, Maximilien Burq, Chinmoy Dutta, Patrick Jaillet, Amin Saberi, and Chris Sholley. Edge weighted online windowed matching. In Proceedings of the 2019 ACM Conference on Economics and Computation, pages 729–742, 2019.
5 Xiaohui Bei and Shengyu Zhang. Algorithms for trip-vehicle assignment in ride-sharing. In Thirty-Second AAAI Conference on Artificial Intelligence, 2018.
6 Harold N Gabow. Data structures for weighted matching and nearest common ancestors with linking. In Proceedings of the first annual ACM-SIAM symposium on Discrete algorithms, pages 434–443, 1990.
Approximation algorithms for car-sharing problems

7 Dries Goossens, Sergey Polyakovskiy, Frits CR Spieksma, and Gerhard J Woeginger. Between a rock and a hard place: the two-to-one assignment problem. *Mathematical methods of operations research*, 76(2):223–237, 2012.

8 Mitja Stiglic, Niels Agatz, Martin Savelsbergh, and Mirko Gradisar. Making dynamic ride-sharing work: The impact of driver and rider flexibility. *Transportation Research Part E: Logistics and Transportation Review*, 91:190–207, 2016.

9 Xing Wang, Niels Agatz, and Alan Erera. Stable matching for dynamic ride-sharing systems. *Transportation Science*, 52(4):850–867, 2018.

10 David P Williamson and David B Shmoys. *The design of approximation algorithms*. Cambridge university press, 2011.
A Omitted Proofs

This section contains proofs omitted in the main body.

A.1 Proof of the claim \( u_{ji} \leq 2u_{ij} \) in Lemma 5

If \( u_{ji} \leq u_{ij} \), the claim is proved; otherwise, we prove the claim by distinguishing three cases based on \( u_{ij} \). If \( u_{ij} = w(s_i, s_j, t_i, t_j) \), then, according to (2), \( u_{ij} \leq w(s_j, s_i) + w(s_i, t_j) + w(t_i, t_j) \), and we have \( u_{ji} \leq 2u_{ij} \); if \( u_{ij} = w(s_i, s_j, t_j, t_i) \), then, according to (3), \( u_{ij} \leq w(s_j, s_i) + w(s_i, t_j) + w(t_j, t_i) \), and we have \( u_{ij} \leq 2u_{ij} \); otherwise, \( u_{ij} = w(s_i, t_i, s_j, t_j) \), and according to (2), \( u_{ij} \leq w(s_j, t_i) + w(s_j, s_i) + w(s_i, t_i) \), so we have \( u_{ij} \leq 2u_{ij} \).

A.2 Proof of the claim \( w(d_k, s_i, t_i, d_k, s_j, t_j) \leq 3cost(k, \{i, j\}) \) in Lemma 6

According to (3), \( 3cost(k, \{i, j\}) = 3w(d_k, s_i) + 3u_{ij} \). The claim \( 3cost(k, \{i, j\}) \geq w(d_k, s_i, t_i, d_k, s_j, t_j) \) is proved by distinguishing three cases based on \( u_{ij} \):

If \( u_{ij} = w(s_i, s_j, t_i, t_j) \), the inequality \( 3cost(k, \{i, j\}) \geq w(d_k, s_i) + w(s_i, s_j) + w(s_j, t_i) + w(t_j, t_i) + w(d_k, s_j) + w(s_j, t_j) + w(t_i, s_i) + w(t_i, t_j) \) follows by applying the triangle inequality.

If \( u_{ij} = w(s_i, s_j, t_i, t_j) \), the inequality \( 3cost(k, \{i, j\}) \geq w(d_k, s_i) + w(s_i, s_j) + w(s_j, t_i) + w(t_j, t_i) + w(d_k, s_j) + w(s_j, t_j) + w(t_i, s_i) + w(t_i, t_j) \) follows by applying the triangle inequality.

If \( u_{ij} = w(s_i, s_j, t_i, t_j) \), the inequality \( 3cost(k, \{i, j\}) \geq w(d_k, s_i) + w(s_i, s_j) + w(s_j, t_i) + w(t_j, t_i) + w(d_k, s_j) + w(s_j, t_j) + w(t_i, s_i) + w(t_i, t_j) \) follows by applying the triangle inequality.

All three cases entail the inequality \( w(d_k, s_i, t_i, d_k, s_j, t_j) \leq 3cost(k, \{i, j\}) \).

A.3 Proof of Lemma 9

We first prove \( 2w(d_k, s_i) + 2w(d_k, s_j) + \mu_{ij} + \mu_{ji} + \min\{2w(d_k, s_i) + w(t_i, d_k, s_j, t_j), 2w(d_k, s_j, t_j) + w(t_j, d_k, s_i, t_i)\} \leq 8w(d_k, s_i) + 5\mu_{ij} \). We distinguish three cases based on \( \mu_{ij} \):

Case 1 \( \mu_{ij} = \mu_{ij} = w(s_i, t_i) + w(s_i, t_j) \). We have \( 2w(d_k, s_i) + 2w(d_k, s_j) + w(t_i, s_i) + w(t_j, s_j) \leq 8w(d_k, s_i) + 5w(t_j, s_j) + 5w(t_i, s_i) \), since \( w(s_i, t_i) + w(s_i, t_j) + w(t_i, s_i) + w(t_j, s_j) \leq 8w(d_k, s_i) + 5w(t_j, s_j) + 5w(t_i, s_i) \), and \( u_{ij} = 2w(s_i, t_i) + 2w(s_i, t_j) + w(t_i, s_i) + w(t_j, s_j) \leq 4w(s_i, t_i) + 4w(s_i, t_j) + w(t_i, s_i) + w(t_j, s_j) \), since \( u_{ij} = 2w(s_i, t_i) + 2w(s_i, t_j) + w(t_i, s_i) + w(t_j, s_j) \leq 8w(d_k, s_i) + 5\mu_{ij} \).

Case 2 \( \mu_{ij} = \mu_{ij} = w(s_i, t_i) + 2w(s_i, s_j) + 2w(s_i, s_j) + w(t_i, s_i) + w(t_i, s_j) \). We have \( 2w(d_k, s_i) + 2w(d_k, s_j) + w(t_i, s_i) + w(t_i, s_j) \leq 8w(d_k, s_i) + 5w(t_j, s_j) + 5w(t_i, s_i) \), since \( w(s_i, t_i) + w(s_i, t_i) + w(t_i, s_i) + w(t_i, s_j) \leq 8w(d_k, s_i) + 5w(t_j, s_j) + 5w(t_i, s_i) \), and \( u_{ij} = 2w(s_i, t_i) + 2w(s_i, t_j) + w(t_i, s_i) + w(t_i, s_j) \leq 4w(s_i, t_i) + 4w(s_i, t_j) + w(t_i, s_i) + w(t_i, s_j) \), since \( u_{ij} = 2w(s_i, t_i) + 2w(s_i, t_j) + w(t_i, s_i) + w(t_i, s_j) \leq 8w(d_k, s_i) + 5\mu_{ij} \).

Case 3 \( \mu_{ij} = \mu_{ij} = w(s_i, t_i) + 2w(s_i, s_j) + 2w(s_i, s_j) + w(t_i, s_i) + w(t_i, s_j) \). We have \( 2w(d_k, s_i) + 2w(d_k, s_j) + w(t_i, s_i) + w(t_i, s_j) \leq 8w(d_k, s_i) + 5w(t_j, s_j) + 5w(t_i, s_i) \), since \( w(s_i, t_i) + w(s_i, t_i) + w(t_i, s_i) + w(t_i, s_j) \leq 8w(d_k, s_i) + 5w(t_j, s_j) + 5w(t_i, s_i) \), and \( u_{ij} = 2w(s_i, t_i) + 2w(s_i, t_j) + w(t_i, s_i) + w(t_i, s_j) \leq 4w(s_i, t_i) + 4w(s_i, t_j) + w(t_i, s_i) + w(t_i, s_j) \), since \( u_{ij} = 2w(s_i, t_i) + 2w(s_i, t_j) + w(t_i, s_i) + w(t_i, s_j) \leq 8w(d_k, s_i) + 5\mu_{ij} \).

Analogously, we have \( 2w(d_k, s_i) + 2w(s_i, s_j) + 2w(d_k, s_j) + w(t_i, s_i) + w(t_i, s_j) \leq 8w(d_k, s_i) + 5\mu_{ij} \).
B Transportation algorithm for the generalized car-sharing problem

Here, we present the transportation algorithm for the generalized car-sharing problem, in which each car serves a \((a > 2)\) requests. In the transportation algorithm, we replace each car \(k \in D\) by a virtual cars \(\gamma_1(k), \ldots, \gamma_a(k)\), resulting in car sets \(\Gamma_1 = \{\gamma_1(1), \ldots, \gamma_1(n)\}, \ldots, \Gamma_a = \{\gamma_a(1), \ldots, \gamma_a(n)\}\). Next we assign requests to the \(a \cdot n\) cars using a particular definition of the costs.

By extending the definition of the travel time in \([1]\), let \(\text{cost}(k, R_k)\) denote the travel time needed to serve requests in \(R_k\) and \(\text{cost}(M) = \sum_{k=1}^n \text{cost}(k, R_k)\) denote the travel time of allocation \(M = \{(k, R_k) : k \in D, R_k \in R\}\), in which \(\text{cost}(k, R_k) = \text{cost}(k, \{i_1, i_2, \ldots, i_a\})\) is the minimum travel time of visiting all locations \(\{d_k, s_{i_1}, t_{i_1}, \ldots, s_{i_a}, t_{i_a}\}\) where \(s_{i_a}\) is visited before \(t_{i_a}\) for all \(i_a \in R_k\). Similarly, by extending the definition of waiting time in \([5]\), let \(\text{wait}(k, R_k)\) denote the latency of serving requests in \(R_k\) and \(\text{wait}(M) = \sum_{k=1}^n \text{wait}(k, R_k)\) denote the latency of allocation \(M = \{(k, R_k) : k \in D, R_k \in R\}\) where \(|R_k| = a\).

B.1 \(\text{TA}(a)_{\text{sum}}\) for \(\text{CS}_{\text{sum}}\)

\begin{algorithm}
\textbf{Algorithm 3} Transportation algorithm for \(\text{CS}_{\text{sum}}\) (\(\text{TA}(a)_{\text{sum}}\))
\begin{enumerate}
\item \textbf{Input:} non-negative weighted graph \(G = (V, E, w)\), requests \(R = \{i = (s_i, t_i) : 1 \leq i \leq m, s_i, t_i \in V\}\), cars \(D = \{k : 1 \leq k \leq n, d_k \in V\}\), virtual car sets \(\Gamma_1, \ldots, \Gamma_a\).
\item \textbf{Output:} An allocation \(\text{TA} = \{(k, R_k) : k \in D, R_k \in R\}\).
\item For \(k = 1, 2, \ldots, n - 1\) do
\item For \(1 \leq j \leq a, i \in R\) do
\item \(v_3(\gamma_j(k), i) = w(d_k, s_i, t_i) + w(t_i, d_k)\)
\item end for
\item end for
\item For \(1 \leq j \leq a, i \in R\) do
\item \(v_3(\gamma_j(n), i) = w(d_n, s_i, t_i)\)
\item end for
\item Let \(G_3 \equiv \left(\bigcup_{j=1}^a \Gamma_j \cup R, v_3\right)\) be the complete bipartite graph with left vertex-set \(\bigcup_{j=1}^a \Gamma_j\), right vertex-set \(R\), and edges \(v_3(x, i)\) for \(x \in \bigcup_{j=1}^a \Gamma_j\) and \(i \in R\).
\item Find a minimum weight perfect matching \(M_3\) in \(G_3 \equiv \left(\bigcup_{j=1}^a \Gamma_j \cup R, v_3\right)\) with weight \(v_3(M_3)\).
\item Output allocation \(\text{TA} = M_3 \equiv \{(k, \{i_1, \ldots, i_a\}) : (\gamma_1(k), i_1), \ldots, (\gamma_a(k), i_a) \in M_3\}\).
\end{enumerate}
\end{algorithm}

We refer to the solution found by the extended transportation algorithm for the general car-sharing problem with minimum travel time as \(\text{TA}(a)_{\text{sum}}\). For each \((k, \{i_1, \ldots, i_a\}) \in \text{TA}(a)_{\text{sum}}\), we order the requests \(\{i_1, \ldots, i_a\}\) such that \(w(d_k, t_{i_j}) \geq w(d_k, t_{i_{j-1}})\) for all \(1 < j \leq a\). The crucial points of algorithm \(\text{TA}(a)_{\text{sum}}\) are found in lines 3 to 10 in Algorithm 3 where a travel time is defined; a resulting quantity is \(v_3(M_3)\). For every request set \(\{i_1, i_2, \ldots, i_a\}\) that are assigned to a car \(k\), \(v_3(M_3)\) chooses the path \(d_k \rightarrow s_{i_1} \rightarrow t_{i_1} \rightarrow d_k \rightarrow s_{i_2} \rightarrow t_{i_2} \rightarrow \ldots \rightarrow d_k \rightarrow s_{i_a} \rightarrow t_{i_a}\).

Using \(\text{cost}(M) = \sum_{k=1}^n \text{cost}(k, R_k)\) and \(\text{cost}(k, \{i_1, i_2, \ldots, i_a\}) \leq \sum_{j=1}^a w(d_k, s_{i_j}, t_{i_j}, d_k) - \max_{1 \leq i_a \leq a} w(d_k, t_{i_a})\), we have:

\[
\text{cost}(\text{TA}(a)_{\text{sum}}) \leq v_3(M_3). \quad (16)
\]

We now prove two lemma’s concerning this quantity \(v_3(M_3)\), which will be of use in Lemma 14.
Lemma 12. For any \( a \geq 2 \), we have:

\[
v_3(M_3) = \sum_{(k,(i_1,i_2,...,i_a)) \in M_4} \left( \sum_{j=1}^{a} w(d_k, s_{i_j}, t_{i_j}, d_k) - \max_{1 \leq i_k \leq a} w(d_k, t_{i_k}) \right).
\]

Proof. We claim that \( v_3(M_3) \) is minimized if and only if for each car \( k \in D \), car \( k \) serves request \( i_a \in R_k \) after serving all serves requests in \( R_k \setminus \{i_a\} = \{i_1, i_2, ..., i_{a-1}\} \). If this claim holds, then based on line 3 to line 10 in Algorithm \( 3 \),

\[
v_3(k, \{i_1, i_2, ..., i_a\}) = \sum_{j \neq k} w(d_k, s_{i_j}, t_{i_j}, d_k) - \max_{1 \leq i_k \leq a} w(d_k, t_{i_k}) \quad \text{and thus}
\]

\[
v_3(M_3) = \sum_{(k,(i_1,i_2,...,i_a)) \in M_4} \left( \sum_{j=1}^{a} w(d_k, s_{i_j}, t_{i_j}, d_k) - \max_{1 \leq i_k \leq a} w(d_k, t_{i_k}) \right).
\]

It remains to prove the claim. Consider any car \( k \in D \) and \( i_x \in R_k \setminus \{i_a\} \). We will prove that \( v_3(M_3) \) is minimized when serving request \( i_x \) earlier than request \( i_a \) if and only if \( w(d_k, t_{i_a}) > w(d_k, t_{i_x}) \).

**Necessary condition** If request \( i_x \in R_k \setminus \{i_a\} \) is served earlier than request \( i_a \), then, when minimizing \( v_3(k, R_k) \), it is true that \( w(d_k, t_{i_a}) > w(d_k, t_{i_x}) \) holds. If \( w(d_k, s_{i_x}, t_{i_x}) + w(d_k, t_{i_x}) + w(d_k, s_{i_a}, t_{i_a}) \leq w(d_k, s_{i_a}, t_{i_a}) + w(d_k, t_{i_a}) + w(d_k, s_{i_x}, t_{i_x}) \), then \( w(d_k, t_{i_x}) \leq w(d_k, t_{i_a}) \).

**Sufficient condition** If \( w(d_k, t_{i_a}) > w(d_k, t_{i_x}) \), then request \( i_x \) must be served earlier than request \( i_a \) for minimizing \( v_3(k, \{i_1, i_2, ..., i_{a-1}\}) \). If \( w(d_k, t_{i_a}) \geq w(d_k, t_{i_x}) \), then \( w(d_k, s_{i_x}, t_{i_x}) + w(d_k, s_{i_a}, t_{i_a}) + w(d_k, s_{i_a}, t_{i_a}) + w(d_k, t_{i_a}) + w(d_k, s_{i_x}, t_{i_x}) \), that means \( v_3(M_3) \) is smaller when serving request \( i_x \) earlier than request \( i_a \).

Lemma 13. For \( a \geq 2 \) and for each allocation \( M \), we have:

\[
v_3(M_3) \leq \sum_{(k,(i_1,i_2,...,i_a)) \in M^*} \left( \sum_{j=1}^{a} w(d_k, s_{i_j}, t_{i_j}, d_k) - \max_{1 \leq i_k \leq a} w(d_k, t_{i_k}) \right).
\]

This lemma is obvious since \( M_3 \) is a minimum weight perfect matching.

Lemma 14. \( TA_{\text{sum}}(a) \) is a \((2a - 1)\)-approximation algorithm for \( CS_{\text{sum}} \).

Proof.

\[
cost(TA_{\text{sum}}(a)) \leq v_3(M_3) \quad \text{(by [16])}
\]

\[
\leq \sum_{(k,(i_1,i_2,...,i_a)) \in M^*} \left( \sum_{j=1}^{a} w(d_k, s_{i_j}, t_{i_j}, d_k) - \max_{1 \leq i_k \leq a} w(d_k, t_{i_k}) \right) \quad \text{(by Lemma 13)}
\]

\[
= \sum_{(k,(i_1,i_2,...,i_a)) \in M^*} \left( \sum_{j=1}^{a} w(d_k, s_{i_j}, t_{i_j}) + \sum_{j=1}^{a} w(t_{i_j}, d_k) - \max_{1 \leq i_k \leq a} w(d_k, t_{i_k}) \right)
\]

\[
\leq \sum_{(k,R_k) \in M^*} (2a - 1) \cdot cost(k, R_k) \quad \text{(by cost(k, R_k) \leq \max_{i_k \in R_k} w(d_k, s_{i_k}, t_{i_k}))}
\]

\[
= (2a - 1) \cdot cost(M^*)
\]
Lemma 15. Let $v_3(\gamma_i(k), i) = (a - j + 1) \cdot w(d_k, s_i, t_i) + (a - j) \cdot w(t_i, d_k)$.

Proof. We claim that $v_3(M_3)$ is minimized if and only if for each car $k \in D$, car $k$ serves requests $R_k = \{i_1, i_2, ..., i_a\}$, where $\{(k, i_1), (k, i_2), ..., (k, i_a)\} \in M_3$ in order of non-decreasing travel time $w(d_k, s_i, t_i) + w(d_k, t_i)$ ($i_k \in R_k$). If this claim holds, then based on line 5 in Algorithm 4, $v_3(k, i_1, i_2, ..., i_a) = \sum_{j=1}^a ((a - j + 1) \cdot w(d_k, s_i, t_i) + (a - j) \cdot w(d_k, t_i))$ and thus $v_3(M_3) = \sum_{(k, i_1, i_2, ..., i_a) \in M_3} \sum_{j=1}^a ((a - j + 1) \cdot w(d_k, s_i, t_i) + (a - j) \cdot w(d_k, t_i))$.

It remains to prove the claim. Consider any car $k \in D$ and $i_x, i_y \in R_k$. We will prove that $v_3(M_3)$ is minimized when serving request $i_x$ earlier than request $i_y$ if and only if $w(d_k, s_{i_x}, t_{i_x}) + w(d_k, t_{i_x}) < w(d_k, s_{i_y}, t_{i_y}) + w(d_k, t_{i_y})$.

Necessary condition If request $i_x \in R_k$ is served earlier than request $i_y \in R_k$, then, when minimizing $v_3(k, R_k)$, it is true that $w(d_k, s_{i_x}, t_{i_x}) + w(d_k, t_{i_x}) < w(d_k, s_{i_y}, t_{i_y}) + w(d_k, t_{i_y})$.
holds. If \((a-x+1) \cdot w(d_k, s_{i_x}, t_{i_x}) + (a-x) \cdot w(d_k, t_{i_y}) + (a-y+1) \cdot w(d_k, s_{i_y}, t_{i_y}) + (a-y) \cdot w(d_k, t_{i_y}) \leq (a+y+1) \cdot w(d_k, s_{i_y}, t_{i_y}) + (a-y) \cdot w(d_k, t_{i_y})\) with \(x < y\), then \((y-x) \cdot w(d_k, s_{i_y}, t_{i_y}) + w(d_k, t_{i_y}) \leq (y-x) \cdot w(d_k, s_{i_x}, t_{i_y}) + w(d_k, t_{i_y})\), and thus we have \(w(d_k, s_{i_x}, t_{i_y}) + w(d_k, t_{i_y}) < w(d_k, s_{i_y}, t_{i_y}) + w(d_k, t_{i_y})\).

Sufficient condition If \(w(d_k, s_{i_x}, t_{i_x}) + w(d_k, t_{i_y}) < w(d_k, s_{i_y}, t_{i_y}) + w(d_k, t_{i_y})\), then request \(i_x\) must be served earlier than request \(i_y\) when minimizing \(v_3(k, \{i_1, i_2, \ldots, i_a\})\).

If \(w(d_k, s_{i_x}, t_{i_x}) + w(d_k, t_{i_y}) < w(d_k, s_{i_y}, t_{i_y}) + w(d_k, t_{i_y})\) with \(a \leq y \geq x > 0\), then
\[
(y-x) \cdot w(d_k, s_{i_y}, t_{i_y}) + w(d_k, t_{i_y}) \leq (y-x) \cdot w(d_k, s_{i_x}, t_{i_y}) + w(d_k, t_{i_y})
\]
and thus
\[
(a-x+1) \cdot w(d_k, s_{i_x}, t_{i_x}) + (a-x) \cdot w(d_k, t_{i_y}) + (a-y+1) \cdot w(d_k, s_{i_y}, t_{i_y}) + (a-y)w(d_k, t_{i_y}) \leq (a-y+1) \cdot w(d_k, s_{i_x}, t_{i_x}) + (a-y)w(d_k, t_{i_y}) + (a-x+1) \cdot w(d_k, s_{i_y}, t_{i_y}) + (a-x)w(d_k, t_{i_y}).
\]
This means that \(v_3(M_3)\) is smaller when serving request \(i_x\) earlier than request \(i_y\).

Lemma 16. For \(a \geq 2\) and for each allocation \(M = \{(k, \{i_1, i_2, \ldots, i_a\}) : k \in D, \{i_1, i_2, \ldots, i_a\} \in R\}\), suppose \(w(d_k, s_{i_j}, t_{i_j}) \geq w(d_k, s_{i_{j-1}}, t_{i_{j-1}})\) for all \(1 < j \leq a\), we have:
\[
v_3(M_3) \leq \sum_{(k, \{i_1, i_2, \ldots, i_a\}) \in M} \sum_{j=1}^{a} ((a-j+1) \cdot w(d_k, s_{i_j}, t_{i_j}) + (a-j) \cdot w(d_k, t_{i_j})).
\]

This lemma is obvious since \(M_3\) is a minimum weight perfect matching.

Lemma 17. \(TA_{lat}(a)\) is an \(a\)-approximation algorithm for \(CS_{lat}\).

Proof.
\[
wait(TA_{lat}(a)) \leq v_3(M_3) \quad \text{(by (17))}
\]
\[
\leq \sum_{(k, \{i_1, i_2, \ldots, i_a\}) \in M} \sum_{j=1}^{a} ((a-j+1) \cdot w(d_k, s_{i_j}, t_{i_j}) + (a-j) \cdot w(d_k, t_{i_j})) \quad \text{(by Lemma 16)}
\]
\[
= \sum_{(k, \{i_1, i_2, \ldots, i_a\}) \in M} \sum_{j=1}^{a} (a \cdot w(d_k, s_{i_j}, t_{i_j}) - (j-1) \cdot w(d_k, s_{i_j}, t_{i_j}) + (a-j) \cdot w(d_k, t_{i_j}))
\]
\[
\leq \sum_{(k, \{i_1, i_2, \ldots, i_a\}) \in M} a \cdot \sum_{j=1}^{a} w(d_k, s_{i_j}, t_{i_j})
\]
\[
\leq \sum_{(k, \{i_1, i_2, \ldots, i_a\}) \in M} a \cdot wait(k, R_k) \quad \text{(by wait(k, R_k) \leq \sum_{i_n \in R_k} w(d_k, s_{i_n}, t_{i_n}))}
\]
\[
= a \cdot wait(M^*)
\]
The third inequality holds since we ordered the requests in \(R_k (k \in D)\) with non-decreasing \(w(d_k, s_{i_j}, t_{i_j})\) for all \(j \in R_k\), and by triangle inequality.

C Proofs for the rest results in Table 1

Recall the results in Table 1. We will prove the rest of the results that were not proven in the main text.

C.1 Remaining proofs of results for \(CS_{sum}\)

In \(CS_{sum}\), for each \((k, \{i, j\})\), we have
\[
cost(k, \{i, j\}) = \min\{w(d_k, s_i) + w_k, w(d_k, s_j) + w_{ji}\}, \quad \text{and (18)}
\]
\[
cost(k, \{i, j\}) \geq \max\{w(d_k, s_i, t_i), w(d_k, s_j, t_j), w(d_k, t_i, w(d_k, t_j))\}. \quad \text{(19)}
\]
Lemma 18. For each request pair \( \{i,j\} \in R \), we have

\[
\frac{\mu_{ij} + \mu_{ji}}{2} + w(s_i, s_j) \leq \min\{3u_{ij}, 3u_{ji}\}
\]

and

\[
\frac{\mu_{ij} + \mu_{ji}}{2} + w(s_i, s_j) \leq \min\{2\mu_{ij}, 2\mu_{ji}\}.
\]

Proof. We first prove \( \frac{\mu_{ij} + \mu_{ji}}{2} + w(s_i, s_j) \leq 3u_{ij} \) and \( \frac{\mu_{ij} + \mu_{ji}}{2} + w(s_i, s_j) \leq 2\mu_{ij} \) by distinguishing three cases.

Case 1: \( u_{ij} = w(s_i, t_i, s_j, t_j) \) (resp. \( \mu_{ij} = 2w(s_i, t_i) + w(t_i, s_j, t_j) \)) Observe that \( \mu_{ij} \leq 2w(s_i, t_i) + w(t_i, s_j, t_j) \) (resp. \( u_{ij} \leq w(s_i, t_i, s_j, t_j) \)) and \( \mu_{ji} \leq 2w(s_j, t_j) + w(t_j, s_i, t_i) \). By the triangle inequality \( w(s_i, t_i) \leq w(s_i, t_i, s_j, t_j) \) and \( w(s_i, s_j) \leq w(s_i, t_i, s_j, t_j) \), we have \( \frac{\mu_{ij} + \mu_{ji}}{2} + w(s_i, s_j) \leq 3w(s_i, t_i) + 2w(t_i, s_j) + 2w(s_j, t_j) \leq 3u_{ij} \) (resp. \( \frac{\mu_{ij} + \mu_{ji}}{2} + w(s_i, s_j) \leq 3w(s_i, t_i) + 2w(t_i, s_j) + 2w(s_j, t_j) \leq 2\mu_{ij} \)).

Case 2: \( u_{ij} = w(s_i, s_j, t_i, t_j) \) (resp. \( \mu_{ij} = 2w(s_i, s_j) + w(t_i, s_j, t_j) \)). Observe that \( \mu_{ij} \leq 2w(s_i, s_j) + w(t_i, s_j, t_j) \) (resp. \( u_{ij} \leq w(s_i, s_j, t_i, t_j) \)) and \( \mu_{ji} \leq 2w(s_j, t_j) + w(t_j, s_i, t_i) \). By the triangle inequality \( w(s_i, s_j) \leq w(s_i, t_i) \) and \( w(s_i, t_i) \leq w(s_i, t_i, s_j, t_j) \), we have \( \frac{\mu_{ij} + \mu_{ji}}{2} + w(s_i, s_j) \leq 3w(s_i, s_j) + 3w(s_j, t_j) + w(t_j, t_j) \leq 3u_{ij} \) (resp. \( \frac{\mu_{ij} + \mu_{ji}}{2} + w(s_i, s_j) \leq 3w(s_i, s_j) + 3w(s_j, t_j) + w(t_j, t_j) \leq 2\mu_{ij} \)).

Case 3: \( u_{ij} = w(s_i, t_i, s_j, t_j) \) (resp. \( \mu_{ij} = 2w(s_i, t_i) + w(t_i, s_j, t_j) \)). Observe that \( \mu_{ij} \leq 2w(s_i, t_i) + w(t_i, s_j, t_j) \) (resp. \( u_{ij} \leq w(s_i, t_i, s_j, t_j) \)) and \( \mu_{ji} \leq 2w(s_j, t_j) + w(t_j, s_i, t_i) \). By the triangle inequality \( w(s_i, t_i) \leq w(s_i, t_i, s_j, t_j) \) and \( w(s_i, t_i) \leq w(s_i, t_i, s_j, t_j) \), we have \( \frac{\mu_{ij} + \mu_{ji}}{2} + w(s_i, s_j) \leq 3w(s_i, t_i) + 3w(s_j, t_j) + w(t_j, t_i) \leq 3u_{ij} \) (resp. \( \frac{\mu_{ij} + \mu_{ji}}{2} + w(s_i, s_j) \leq 3w(s_i, t_i) + 3w(s_j, t_j) + w(t_j, t_i) \leq 2\mu_{ij} \)).

The proofs of \( \frac{\mu_{ij} + \mu_{ji}}{2} + w(s_i, s_j) \leq 3u_{ji} \) and \( \frac{\mu_{ij} + \mu_{ji}}{2} + w(s_i, s_j) \leq 2\mu_{ji} \) are symmetric, thus the claim is proved.

Lemma 19. MA(2, \( \mu \)) is a 3-approximation algorithm for \( \text{CS}_{\text{sum}} \).

Proof.

\[
\text{cost}(\text{MA}(2, \mu)) = \sum_{(k, \{i,j\}) \in \text{MA}} \min\{w(d_k, s_i) + u_{ij}, w(d_k, s_j) + u_{ji}\} \quad \text{(by (3))}
\]

\[
\leq \sum_{(k, \{i,j\}) \in \text{M}^*} (2w(d_k, s_i) + 2w(d_k, s_j) + \frac{\mu_{ij} + \mu_{ji}}{2})
\]

\[
\leq \sum_{(k, \{i,j\}) \in \text{M}^*} (2w(d_k, s_i) + w(s_j, s_j) + \frac{\mu_{ij} + \mu_{ji}}{2}) \quad \text{(by the triangle inequality)}
\]

\[
\leq \sum_{(k, \{i,j\}) \in \text{M}^*} (2w(d_k, s_i) + 3\min\{u_{ij}, u_{ji}\}) \quad \text{(by Lemma 18)}
\]

\[
\leq \sum_{(k, \{i,j\}) \in \text{M}^*} 3\text{cost}(k, \{i,j\}) \quad \text{(by (18))}
\]

\[
\leq 3 \text{cost}(\text{M}^*)
\]

(20)

The first inequality follows from Lemma 2.

Lemma 20. TA(2) is a 4-approximation algorithm for \( \text{CS}_{\text{sum}} \).
Proof.  

\[
\text{cost}(\text{TA}(2)) = \sum_{(k,\{i,j\}) \in \text{TA}} \min \{w(d_k, s_i) + u_{ij}, w(d_k, s_j) + u_{ji}\} \quad \text{(by } (3)\text{)}
\]

\[
\leq \sum_{(k,\{i,j\}) \in M^*} \min \{2w(d_k, s_i, t_i) + w(t_i, d_k, s_j, t_j), 2w(d_k, s_j, t_j) + w(t_j, d_k, s_i, t_i)\}
\]

\[
\leq \sum_{(k,\{i,j\}) \in M^*} 4\text{cost}(k, \{i,j\}) \quad \text{(by } (19)\text{)}
\]

\[
\leq 4 \text{cost}(M^*) \quad \text{(21)}
\]

The first inequality holds since TA is a minimum weight perfect matching in algorithm TA(2). 

\[\blacksquare\]

\[\blacktriangleleft\]

\section*{Theorem 1.} CA(2, µ) is a 3-approximation algorithm for CS\text{sum}.

According to (20) and (21), we have: \(\text{cost}(\text{CA}(I)) = \min \{\text{cost}(\text{MA}(I)), \text{cost}(\text{TA}(I))\} \leq 3 \text{cost}(M^*(I))\) for any instance \(I\). 

\section*{C.2 Remaining proofs of results for CS\text{sum}_{s=t}}

In CS\text{sum}_{s=t}, for each \((k,\{i,j\})\), by (1) and the triangle inequality, we have

\[
\frac{3}{2}\text{cost}(k, \{i,j\}) \geq \frac{1}{2}(w(d_k, s_i) + w(d_k, s_j)) + w(s_i, s_j) \quad \text{and,} \quad \text{cost}(k, \{i,j\}) \geq \max\{w(d_k, s_i), w(d_k, s_j)\}. \quad \text{(22)}
\]

\[
\text{cost}(k, \{i,j\}) \geq \max\{w(d_k, s_i), w(d_k, s_j)\}. \quad \text{(23)}
\]

\[\blacktriangleleft\]

\[\blacktriangleleft\]

\section*{Lemma 1.} MA(1, u) is a 3/2-approximation algorithm for CS\text{sum}_{s=t}. Moreover, there exists an instance \(I\) for which \(\text{cost}(\text{MA}(I))=3/2 \text{cost}(M^*(I))\).

Proof.  

\[
\text{cost}(\text{MA}(1, u)) = \sum_{(k,\{i,j\}) \in \text{MA}} \min \{w(d_k, s_i) + u_{ij}, w(d_k, s_j) + u_{ji}\} \quad \text{(by } (3)\text{)}
\]

\[
\leq \sum_{(k,\{i,j\}) \in M^*} (w(s_i, s_j) + \frac{1}{2}(w(d_k, s_i) + w(d_k, s_j)))
\]

\[
\leq \sum_{(k,\{i,j\}) \in M^*} 3/2\text{cost}(k, \{i,j\}) \quad \text{(by } (22)\text{)}
\]

\[
\leq 3/2 \text{cost}(M^*) \quad \text{(24)}
\]

The first inequality holds since MA is a minimum weight perfect matching in algorithm MA(1, u) and \(u_{ij} = u_{ji} = w(s_i, s_j)\). To see that equality may hold in (24), consider the subgraph induced by the nodes \((k_1, s_3), (s_1, s_2), \text{ and } (k_2, s_4)\) in Figure 2. 

\[\blacksquare\]

\[\blacktriangleleft\]

\section*{Lemma 2.} TA(1) is a 3-approximation algorithm for CS\text{sum}_{s=t}. Moreover, there exists an instance \(I\) for which \(\text{cost}(\text{TA}(I))=3 \text{cost}(M^*(I))\). 

\[\blacktriangleleft\]
Approximation algorithms for car-sharing problems

Proof.

\[
\begin{align*}
\text{cost}(TA(1)) &= \sum_{(k,\{i,j\}) \in TA} \min\{w(d_k, s_i) + u_{ij}, w(d_k, s_j) + u_{ji}\} \quad \text{(by (3))} \\
&\leq \sum_{(k,\{i,j\}) \in TA} \min\{2w(d_k, s_i) + w(d_k, s_j), 2w(d_k, s_j) + w(d_k, s_i)\} \\
&\leq \sum_{(k,\{i,j\}) \in M^*} \min\{2w(d_k, s_i) + w(d_k, s_j), 2w(d_k, s_j) + w(d_k, s_i)\} \\
&\leq \sum_{(k,\{i,j\}) \in M^*} 3\text{cost}(k, \{i, j\}) \quad \text{(by (23))} \\
&\leq 3 \text{cost}(M^*) \\
\end{align*}
\]

The first inequality holds by \( u_{ij} = u_{ji} = w(s_i, s_j) \) and the triangle inequality, and the second inequality holds since \( TA \) is a minimum weight perfect matching in algorithm \( TA(1) \). To see that equality may hold in (25), consider the subgraph induced by the nodes \((s_5, s_6), (k_3, k_4), \) and \((s_7, s_8)\) in Figure 2.

\[\blacklozenge \]

**Lemma 21.** \( MA(2, \mu) \) is a 3/2-approximation algorithm for \( CS_{\text{sum}, s=t} \).

Proof.

\[
\begin{align*}
\text{cost}(MA(2, \mu)) &= \sum_{(k,\{i,j\}) \in MA} \min\{w(d_k, s_i) + u_{ij}, w(d_k, s_j) + u_{ji}\} \quad \text{(by (3))} \\
&= \sum_{(k,\{i,j\}) \in MA} (w(s_i, s_j) + \min\{w(d_k, s_i), w(d_k, s_j)\}) \\
&\leq \sum_{(k,\{i,j\}) \in M^*} (w(s_i, s_j) + \frac{1}{2}(w(d_k, s_i) + w(d_k, s_j))) \\
&\leq \frac{3}{2} \sum_{(k,\{i,j\}) \in M^*} \text{cost}(k, \{i, j\}) \quad \text{(by (22))} \\
&\leq \frac{3}{2} \text{cost}(M^*)
\end{align*}
\]

The first inequality holds because \( u_{ij} = u_{ji} = w(s_i, s_j) \), and the second inequality inequality follows from Lemma 2.

\[\blacklozenge \]

**Lemma 22.** \( TA(2) \) is a 4-approximation algorithm for \( CS_{\text{sum}, s=t} \).

Proof.

\[
\begin{align*}
\text{cost}(TA(2)) &= \sum_{(k,\{i,j\}) \in TA} \min\{w(d_k, s_i) + u_{ij}, w(d_k, s_j) + u_{ji}\} \quad \text{(by (3))} \\
&\leq \sum_{(k,\{i,j\}) \in TA} \min\{3w(d_k, s_i) + w(d_k, s_j), 3w(d_k, s_j) + w(d_k, s_i)\} \\
&\leq \sum_{(k,\{i,j\}) \in M^*} \min\{3w(d_k, s_i) + w(d_k, s_j), 3w(d_k, s_j) + w(d_k, s_i)\} \\
&\leq \sum_{(k,\{i,j\}) \in M^*} 4\text{cost}(k, \{i, j\}) \quad \text{(by (23))} \\
&\leq 4 \text{cost}(M^*)
\end{align*}
\]

The first inequality holds by \( u_{ij} = u_{ji} = w(s_i, s_j) \) and triangle inequality, and the second inequality holds since \( TA \) is a minimum weight perfect matching in algorithm \( TA(2) \).

\[\blacklozenge \]
Theorem 23. CA(2, μ) is a 10/7-approximation algorithm for CS_{sum,s=t}.

Proof.

\[ 7\text{cost}(CA(2, \mu)) \leq 6\text{cost}(MA(2, \mu)) + \text{cost}(TA(2)) \]

\[ \leq \sum_{(k,\{i,j\}) \in M^*} \left( 6w(s_i, s_j) + \min\{2w(d_k, s_i) + w(d_k, s_j)\} \right) \]

\[ \leq \sum_{(k,\{i,j\}) \in M^*} \left( 10w(s_i, s_j) + \min\{w(d_k, s_i), w(d_k, s_j)\} \right) \quad \text{(by the triangle inequality)} \]

\[ \leq \sum_{(k,\{i,j\}) \in M^*} 10\text{cost}(k, \{i,j\}) \quad \text{(by [1])} \]

\[ \leq 10 \text{cost}(M^*) \]

The second inequality holds since MA is a minimum weight perfect matching in algorithm MA(2, μ) and TA is a minimum weight perfect matching in algorithm TA(2).

C.3 Remaining proofs of results for CS_{lat}

In CS_{lat}, for each (k, {i, j}), we have:

\[ \text{wait}(k, \{i, j\}) = \min\{2w(d_k, s_i) + \mu_{ij}, 2w(d_k, s_j) + \mu_{ji}\} \quad \text{and,} \]

\[ \text{wait}(k, \{i, j\}) \geq \max\{w(d_k, s_i, t_i) + w(d_k, s_j, t_j), w(d_k, t_i) + w(d_k, t_j)\} \quad \text{(by the triangle inequality)} \]

Lemma 24. MA(2, μ) is a 2-approximation algorithm for CS_{lat}. Moreover, there exists an instance I for which \text{wait(MA(I))}=2 \text{wait(M^*(I))}.

Proof.

\[ \text{wait}(MA(2, \mu)) = \sum_{(k,\{i,j\}) \in MA} \min\{2w(d_k, s_i) + \mu_{ij}, 2w(s_k, s_j) + \mu_{ji}\} \quad \text{(by [7])} \]

\[ \leq \sum_{(k,\{i,j\}) \in M^*} \frac{2w(d_k, s_i) + \mu_{ij} + 2w(d_k, s_j) + \mu_{ji}}{2} \]

\[ \leq \sum_{(k,\{i,j\}) \in M^*} \frac{4\min\{w(d_k, s_i), w(d_k, s_j)\} + \mu_{ij} + 2w(s_i, s_j) + \mu_{ji}}{2} \]

\[ \leq \sum_{(k,\{i,j\}) \in M^*} \min\{4w(d_k, s_i) + 2\mu_{ij}, 4w(d_k, s_j) + 2\mu_{ji}\} \quad \text{(by Lemma [18])} \]

\[ \leq \sum_{(k,\{i,j\}) \in M^*} 2\text{wait}(k, \{i,j\}) \quad \text{(by [26])} \]

\[ \leq 2 \text{wait}(M^*) \quad \text{(28)} \]

The first inequality holds since MA is a minimum weight perfect matching in algorithm MA(2, μ), and by the triangle inequality. To see that equality may hold in (28), consider the subgraph induced by the nodes (k_1, s_3), (s_1, s_2), and (k_2, s_4) in Figure 3.

Lemma 25. TA(2) is a 2-approximation algorithm for CS_{lat}. Moreover, there exists an instance I for which \text{wait(TA(I))}=2 \text{wait(M^*(I))}.
Approximation algorithms for car-sharing problems

Proof.

\[
\text{wait}(\text{TA}(2)) = \sum_{(k,i,j) \in \text{TA}} \min\{2w(d_k, s_i) + \mu_{ij}, 2w(s_k, s_j) + \mu_{ji}\} \text{ (by (7))}
\]

\[
\leq \sum_{(k,i,j) \in M^*} \min\{2w(d_k, s_i) + w(t_i, d_k, s_j, t_j), 2w(d_k, s_j) + w(t_j, d_k, s_i, t_i)\}
\]

\[
\leq \sum_{(k,i,j) \in M^*} \frac{3w(d_k, s_i) + 3w(d_k, s_j) + w(t_i, d_k) + w(t_j, d_k)}{2}
\]

\[
\leq \sum_{(k,i,j) \in M^*} 2\text{wait}(k, \{i, j\}) \text{ (by (27))}
\]

\[
\leq 2 \text{wait}(M^*)
\]

The first inequality holds since TA is a minimum weight perfect matching in algorithm TA(2) and by the triangle inequality. To see that equality may hold in (29), consider the subgraph induced by the nodes \((s_5, s_6), (k_3, k_4), (s_7, s_8)\) in Figure 3.

\[\blacktriangleleft\]

Lemma 26. For each request pair \(\{i, j\} \in R\), we have

\[
u_{ij} + u_{ji} + 2w(s_i, s_j) \leq \min\{4\mu_{ij}, 4\mu_{ji}\}.
\]

Proof. We first prove \(u_{ij} + u_{ji} + 2w(s_i, s_j) \leq 4\mu_{ij}\) by distinguishing three cases.

Case 1 \(\mu_{ij} = 2w(s_i, t_i) + w(t_i, s_j, t_j)\). Observe that \(u_{ij} \leq w(s_i, t_i, s_j, t_j)\) and \(u_{ji} \leq w(s_j, t_j, s_i, t_i)\). By the triangle inequality \(w(s_i, t_j) \leq w(s_i, t_i, s_j, t_j)\) and \(w(s_i, s_j) \leq w(s_i, t_i, s_j)\), we have \(u_{ij} + u_{ji} + 2w(s_i, s_j) \leq 5w(s_i, t_i) + 4w(t_i, s_j) + 2w(s_j, t_j) \leq 4\mu_{ij}\).

Case 2 \(\mu_{ij} = 2w(s_i, s_j, t_j) + w(t_i, s_j)\). Observe that \(u_{ij} \leq w(s_i, s_j, t_j, t_i)\) and \(u_{ji} \leq w(s_j, t_j, s_i, t_i)\). By the triangle inequality \(w(s_i, t_i) \leq w(s_i, s_j, t_j, t_i)\), we have \(u_{ij} + u_{ji} + 2w(s_i, s_j) \leq 5w(s_i, s_j) + 2w(s_j, t_j) + w(t_i, t_j) \leq 4\mu_{ij}\).

Case 3 \(\mu_{ij} = 2w(s_i, s_j, t_i) + w(t_j, t_i)\). Observe that \(u_{ij} \leq w(s_i, s_j, t_i, t_j)\) and \(u_{ji} \leq w(s_j, t_i, s_i, t_i)\). By the triangle inequality \(w(s_i, t_j) \leq w(s_i, s_j, t_i, t_j)\), we have \(u_{ij} + u_{ji} + 2w(s_i, s_j) \leq 5w(s_i, s_j) + 2w(s_j, t_j) + 2w(t_j, t_i) \leq 4\mu_{ij}\).

The proof of \(u_{ij} + u_{ji} + 2w(s_i, s_j) \leq 4\mu_{ij}\) follows from symmetry.

\[\blacktriangleleft\]

Lemma 27. MA(1, \(u\)) is a 3-approximation algorithm for CSlat.

Proof.

\[
\text{wait}(\text{MA}(1, u)) = \sum_{(k,\{i,j\}) \in \text{MA}} \min\{2w(d_k, s_i) + \mu_{ij}, 2w(s_k, s_j) + \mu_{ji}\} \text{ (by (7))}
\]

\[
\leq \sum_{(k,i,j) \in M^*} \max\{\mu_{ij}, \mu_{ji}\} + 2w(d_k, s_i, s_j)\}
\]

\[
\leq \sum_{(k,i,j) \in M^*} (u_{ij} + u_{ji} + w(d_k, s_i) + w(d_k, s_j))
\]

\[
\leq \sum_{(k,i,j) \in M^*} (u_{ij} + u_{ji} + 2\min\{w(d_k, s_i), w(d_k, s_j)\} + w(s_i, s_j))
\]

\[
\leq \sum_{(k,i,j) \in M^*} (2\min\{w(d_k, s_i), w(d_k, s_j)\} + 3\min\{\mu_{ij}, \mu_{ji}\}) \text{ (by Lemma 26)}
\]

\[
\leq \sum_{(k,i,j) \in M^*} 3\text{wait}(k, \{i, j\}) \text{ (by (32) and (33))}
\]

\[
\leq 3 \text{wait}(M^*)
\]
The second inequality follows from Lemma \ref{lem:triangle-inequality} and the second inequality holds by the triangle inequality.

\begin{lemma}
\textsc{Lemma 28.} TA(1) is a 3-approximation algorithm for \textsc{CS}_{\text{lat}}.
\end{lemma}

\begin{proof}
\begin{align*}
\text{wait(TA(1))} &= \sum_{(k,\{i,j\}) \in \text{TA}} \min\{2w(d_k, s_i) + \mu_{ij}, 2w(s_k, s_j) + \mu_{ji}\} \quad \text{(by (7))} \\
&\leq \sum_{(k,\{i,j\}) \in M^*} 2 \min\{w(d_k, s_i, t_i, d_k, s_j, t_j), w(d_k, s_j, t_j, d_k, s_i, t_i)\} \\
&\leq \sum_{(k,\{i,j\}) \in M^*} 3 \text{wait}(k, \{i, j\}) \quad \text{(by the triangle inequality and case analysis)} \\
&\leq 3 \text{wait}(M^*) \quad (31)
\end{align*}

The first inequality holds since TA is a minimum weight perfect matching in algorithm TA(1) and by \(\mu_{ij} = w(s_i, s_j)\). To see that equality may hold in (34), consider the subgraph induced by the nodes \((k_1, s_3), (s_1, s_2),\) and \((k_2, s_4)\) in Figure 3.
\end{proof}

\begin{theorem}
\textsc{Theorem 29.} CA(1, u) is a 3-approximation algorithm for \textsc{CS}_{\text{lat}}.
\end{theorem}

According to (30) and (31), we have \(\text{wait(CA(I))} = \min\{\text{wait(MA(I))}, \text{wait(TA(I))}\} \leq 3 \text{wait}(M^*(I))\) for any instance \(I\).

\section{C.4 Remaining proofs of results for \textsc{CS}_{\text{lat}, s=t}}

In \textsc{CS}_{\text{lat}, s=t}, for each \((k,\{i, j\})\), we have:

\begin{align*}
\text{wait}(k, \{i, j\}) &\geq 2 \min\{w(d_k, s_i), w(d_k, s_j)\} + w(s_i, s_j) , \quad \text{and} \quad (32) \\
\text{wait}(k, \{i, j\}) &\geq w(d_k, s_i) + w(d_k, s_j) . \quad (33)
\end{align*}

\begin{lemma}
\textsc{Lemma 30.} MA(2, \mu) is a 2-approximation algorithm for \textsc{CS}_{\text{lat}, s=t}. Moreover, there exists an instance \(I\) for which \(\text{wait(MA(I))} = 2 \text{wait}(M^*(I))\).
\end{lemma}

\begin{proof}
\begin{align*}
\text{wait(MA(2, \mu))} &= \sum_{(k,\{i,j\}) \in \text{MA}} \min\{2w(d_k, s_i) + \mu_{ij}, 2w(s_k, s_j) + \mu_{ji}\} \quad \text{(by (7))} \\
&\leq \sum_{(k,\{i,j\}) \in M^*} (w(s_i, s_j) + w(d_k, s_i) + w(d_k, s_j)) \\
&\leq \sum_{(k,\{i,j\}) \in M^*} 2 \text{wait}(k, \{i, j\}) \quad \text{(by (32) and (33))} \\
&\leq 2 \text{wait}(M^*) \quad (34)
\end{align*}

The first inequality holds since MA is a minimum weight perfect matching in algorithm MA(2, \mu), and by \(\mu_{ij} = w(s_i, s_j)\). To see that equality may hold in (34), consider the subgraph induced by the nodes \((k_1, s_3), (s_1, s_2),\) and \((k_2, s_4)\) in Figure 3.
\end{proof}

\begin{lemma}
\textsc{Lemma 31.} TA(2) is a 2-approximation algorithm for \textsc{CS}_{\text{lat}, s=t}. Moreover, there exists an instance \(I\) for which \(\text{wait(TA(I))} = 2 \text{wait}(M^*(I))\).
\end{lemma}
Approximation algorithms for car-sharing problems

Proof. We have:

\[
\text{wait}(\text{TA}(2)) = \sum_{(k,\{i,j\}) \in \text{TA}} \min\{2w(d_k, s_i) + \mu_{ij}, 2w(s_k, s_j) + \mu_{ji}\} \quad \text{(by (7))}
\]

\[
\leq \sum_{(k,\{i,j\}) \in M^*} \min\{3w(d_k, s_i) + w(d_k, s_j), 3w(d_k, s_j) + w(d_k, s_i)\}
\]

\[
\leq \sum_{(k,\{i,j\}) \in M^*} 2\text{wait}(k, \{i,j\}) \quad \text{(by (33))}
\]

\[
\leq 2 \text{wait}(M^*). \quad (35)
\]

The first inequality holds since TA is a minimum weight perfect matching in algorithm TA(2), and by \(\mu_{ij} = w(s_i, s_j)\). To see that equality may hold in (35), consider the subgraph induced by the nodes \((s_5, s_6), (k_3, k_4), \text{and } (s_7, s_8)\) in Figure 3.

Lemma 32. MA(1, u) is a 2-approximation algorithm for \(CS_{lat,s=t}\).

Proof.

\[
\text{wait}(\text{MA}(1,u)) = \sum_{(k,\{i,j\}) \in \text{MA}} \min\{2w(d_k, s_i) + \mu_{ij}, 2w(d_k, s_j) + \mu_{ji}\} \quad \text{(by (7))}
\]

\[
\leq \sum_{(k,\{i,j\}) \in M^*} \min\{2w(d_k, s_i) + w(s_i, s_j), 2w(d_k, s_j) + w(s_i, s_j)\}
\]

\[
\leq \sum_{(k,\{i,j\}) \in M^*} (w(s_i, s_j) + w(d_k, s_i) + w(d_k, s_j))
\]

\[
\leq \sum_{(k,\{i,j\}) \in M^*} 2\text{wait}(k, \{i,j\}) \quad \text{(by (32) and (33))}
\]

\[
\leq 2 \text{wait}(M^*)
\]

The first inequality holds since MA is a minimum weight perfect matching in algorithm MA(1, u), and by \(\mu_{ij} = w_{ij} = w(s_i, s_j)\).

Lemma 33. TA(1) is a 2-approximation algorithm for \(CS_{lat,s=t}\).

Proof.

\[
\text{wait}(\text{TA}(1)) = \sum_{(k,\{i,j\}) \in \text{TA}} \min\{2w(d_k, s_i) + \mu_{ij}, 2w(s_k, s_j) + \mu_{ji}\} \quad \text{(by (7))}
\]

\[
\leq \frac{4}{3} \sum_{(k,\{i,j\}) \in \text{TA}} \min\{2w(d_k, s_i) + w(d_k, s_j), 2w(s_k, s_j) + w(d_k, s_i)\}
\]

\[
\leq \frac{4}{3} \sum_{(k,\{i,j\}) \in M^*} \min\{2w(d_k, s_i) + w(d_k, s_j), 2w(d_k, s_j) + w(d_k, s_i)\}
\]

\[
\leq \sum_{(k,\{i,j\}) \in M^*} \min\{8/3w(d_k, s_i) + 4/3w(d_k, s_j), 8/3w(d_k, s_j) + 4/3w(d_k, s_i)\}
\]

\[
\leq \sum_{(k,\{i,j\}) \in M^*} 2\text{wait}(k, \{i,j\}) \quad \text{(by (33))}
\]

\[
\leq 2 \text{wait}(M^*)
\]

The first inequality holds since \(\mu_{ij} = w(s_i, s_j)\) and the triangle inequality, and the second inequality holds since TA is a minimum weight perfect matching in algorithm TA(1).
Theorem 34. CA(1, u) is a 8/5-approximation algorithm for CSlat,s=t.

Proof.

\[ 5 \text{wait}(CA(1, u)) \leq 2 \text{wait}(MA(1, u)) + 3 \text{wait}(TA(1)) \]

\[ \leq \sum_{(k, \{i, j\}) \in MA} 2w(s_i, s_j) + 2 \min\{w(d_k, s_i), w(d_k, s_j)\} \]

\[ + 4 \sum_{(k, \{i, j\}) \in TA} \min\{2w(d_k, s_i) + w(d_k, s_j), 2w(d_k, s_j) + w(d_k, s_i)\} \]

\[ \leq \sum_{(k, \{i, j\}) \in M^*} 2w(s_i, s_j) + 2 \min\{w(d_k, s_i), w(d_k, s_j)\} \]

\[ + 4 \sum_{(k, \{i, j\}) \in M^*} \min\{2w(d_k, s_i) + w(d_k, s_j), 2w(d_k, s_j) + w(d_k, s_i)\} \]

\[ \leq \sum_{(k, \{i, j\}) \in M^*} (2w(s_i, s_j) + 2w(d_k, s_i) + 2w(d_k, s_j) + 8w(d_k, s_i) + 4w(d_k, s_j)) \]

\[ \leq \sum_{(k, \{i, j\}) \in M^*} 8 \text{wait}(k, \{i, j\}) \text{ (by (32) and (33))} \]

\[ \leq 8 \text{ wait}(M^*) \]

The first inequality holds since \( \mu_{ij} = w(s_i, s_j) \) and the triangle inequality, and the second inequality holds since MA is a minimum weight perfect matching in algorithm MA(1, u), and since TA is a minimum weight perfect matching in algorithm TA(1).