# Local Search Algorithms for the Maximum Carpool Matching Problem

**Gilad Kutiel**[1] **and Dror Rawitz**[*2]

1 **Department of Computer Science, Technion, Haifa, Israel**
`gkutiel@cs.technion.ac.il`

2 **Faculty of Engineering, Bar Ilan University, Ramat Gan, Israel**
`dror.rawitz@biu.ac.il`

## Abstract

The Maximum Carpool Matching problem is a *star packing* problem in directed graphs. Formally, given a directed graph $G = (V, A)$, a capacity function $c : V \to \mathbb{N}$, and a weight function $w : A \to \mathbb{R}^+$, a *carpool matching* is a subset of arcs, $M \subseteq A$, such that every $v \in V$ satisfies:

**(i)** $d^{\text{in}}_M(v) \cdot d^{\text{out}}_M(v) = 0$,
**(ii)** $d^{\text{in}}_M(v) \le c(v)$, and
**(iii)** $d^{\text{out}}_M(v) \le 1$.

A vertex $v$ for which $d^{\text{out}}_M(v) = 1$ is a *passenger*, and a vertex for which $d^{\text{out}}_M(v) = 0$ is a *driver* who has $d^{\text{in}}_M(v)$ passengers. In the Maximum Carpool Matching problem the goal is to find a carpool matching $M$ of maximum total weight. The problem arises when designing an online carpool service, such as Zimride [4], which tries to connect between users based on a similarity function. The problem is known to be NP-hard, even in the unweighted and uncapacitated case. The Maximum Group Carpool Matching problem, is an extension of Maximum Carpool Matching where each vertex represents an unsplittable group of passengers. Formally, each vertex $u \in V$ has a size $s(u) \in \mathbb{N}$, and the constraint $d^{\text{in}}_M(v) \le c(v)$ is replaced with $\sum_{u:(u,v)\in M} s(u) \le c(v)$.

We show that Maximum Carpool Matching can be formulated as an unconstrained submodular maximization problem, thus it admits a $\frac{1}{2}$-approximation algorithm. We show that the same formulation does not work for Maximum Group Carpool Matching, nevertheless, we present a local search $(\frac{1}{2} - \varepsilon)$-approximation algorithm for Maximum Group Carpool Matching. For the unweighted variant of both problems when the maximum possible capacity, $c_{\max}$, is bounded by a constant, we provide a local search $(\frac{1}{2} + \frac{1}{2c_{\max}} - \varepsilon)$-approximation algorithm. We also show that the problem is APX-hard, even if the maximum degree and $c_{\max}$ are at most 3.

**1998 ACM Subject Classification** F.2.2 Nonnumerical Algorithms and Problems

**Keywords and phrases** approximation algorithms, local search, star packing, submodular maximization

**Digital Object Identifier** 10.4230/LIPIcs.ESA.2017.55

## 1 Introduction

As traveling costs become higher and parking becomes sparse it is only natural to share rides or to *carpool*. Originally, carpooling was an arrangement among a group of people by which they take turns driving the others to and from a designated location. However,

[*] Supported in part by the Israel Science Foundation (grant no. 497/14).

taking turns is not essential, instead passengers can share the cost of the ride with the driver. Carpooling has social advantages other than reducing the costs: it reduces fuel consumption and road congestion and frees parking space. While in the past carpooling was usually a fixed arrangement between friends or neighbors, the emergence of social networks has made carpooling more dynamic and wide scale. These days applications like Zimride [4], BlaBlaCar [1], Moovit [2] and even Waze [3] are matching passengers to drivers.

The matching process of passengers to drivers entails more than matching the route. Passenger satisfaction also needs to be taken into account. Given several riding options (including taking their own car), passengers have preferences. For example, a passenger may prefer to ride with a co-worker or a friend. She may have an opinion on a driver that she rode with in the past. She may prefer a non-smoker, someone who shares her taste in music, or someone who is recommended by others. Moreover, the matching process may take into account driver preferences. For instance, we would like to minimize the extra distance that a driver has to take. Preferences may also be computed using past information. Knapen et al. [16] described an automatic service to match commuting trips. Users of the service register their personal profile and a set of periodically recurring trips, and the service advises registered candidates on how to combine their commuting trips by carpooling. The service estimates the probability that a person $a$ traveling in person's $b$ car will be satisfied by the trip. This is done based on personal information and feedback from users on past rides.

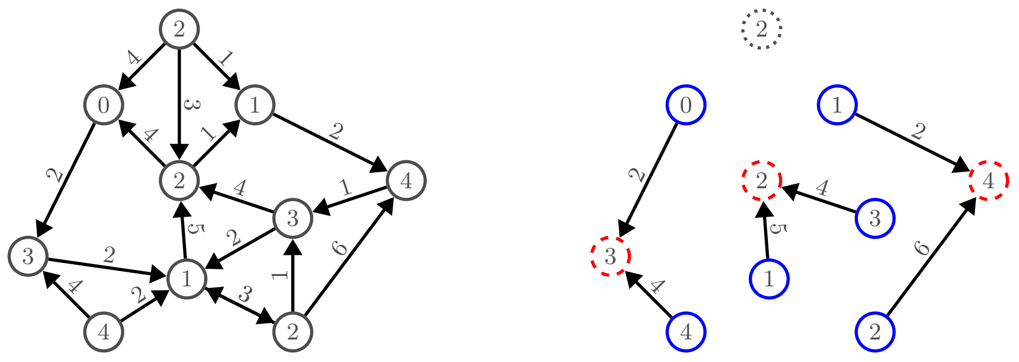
In this paper we assume that potential passenger-driver satisfactions are given as input and the goal is to compute an assignment of passengers to drivers so as to maximize the global satisfaction. More formally, we are given a directed graph $G = (V, A)$, where each vertex $v \in V$ corresponds to a user of the service, and an arc $(u, v)$ exists if the user corresponding to vertex $u$ is willing to commute with the user corresponding to vertex $v$. We are given a capacity function $c : V \to \mathbb{N}$ which bounds the number of passengers each user can drive if she is selected as a driver. A non-negative weight function $w : A \to \mathbb{R}^+$ is used to model the amount of satisfaction $w(u, v)$ of assigning $u$ to $v$. If $(u, v) \in A$ implies that $(v, u) \in A$ and $w(u, v) = w(v, u)$, the instance is *undirected.* If $w(v, u) = 1$, for every $(v, u) \in A$, then the instance is *unweighted.* If $c(v) = \deg(v)$, for every $v$, then the instance is *uncapacitated.*

Given a directed graph $G$ and a subset $M \subseteq A$, define $d^{\text{in}}_M(v) \triangleq |\{u : (u, v) \in M\}|$ and $d^{\text{out}}_M(v) \triangleq |\{u : (v, u) \in M\}|$. A feasible *carpool matching* is a subset of arcs, $M \subseteq A$, such that every $v \in V$ satisfies: (i) $d^{\text{in}}_M(v) \cdot d^{\text{out}}_M(v) = 0$, (ii) $d^{\text{in}}_M(v) \le c(v)$, and (iii) $d^{\text{out}}_M(v) \le 1$. A feasible carpool matching $M$ partitions $V$ as follows:

$$P_M \triangleq \{v : d^{\text{out}}_M(v) = 1\} \quad D_M \triangleq \{v : d^{\text{in}}_M(v) \ge 1\} \quad Z_M \triangleq \{v : d^{\text{out}}_M(v) = d^{\text{in}}_M(c) = 0\}$$

where $P_M$ is the set of *passengers*, $D_M$ is the set of *active drivers*, and $Z_M$ is the set of *solo drivers.* In the Maximum Carpool Matching problem the goal is to find a matching $M$ of maximum total weight, namely to maximize $w(M) \triangleq \sum_{(v,u)\in M} w(v, u)$. In other words, the Maximum Carpool Matching problem is about finding a set of (directed toward the center) vertex disjoint stars that maximizes the total weight of the arcs. Figure 1 contains an example of a Maximum Carpool Matching instance. Note that in the unweighted case the goal is to find a carpool matching $M$ that maximizes $|P_M|$. Moreover, observe that if $G$ is undirected, $D_M \cup Z_M$ is a *dominating set.* Hence, in this case, an optimal carpool matching induces an optimal dominating set and vice versa. Since Minimum Dominating Set is NP-hard, it follows that Maximum Carpool Matching is NP-hard even if the instance is undirected, unweigthed, and uncapacitated.

We also consider an extension of Maximum Carpool Matching, called Maximum Group Carpool Matching, in which each vertex represents a group of passengers, and each group may have a different size. Such a group may represent a family or two friends



**(a)** An instance containing a directed graph with capacities on the vertices and weights on the arcs.

**(b)** A feasible carpool matching with total weight of 23. $P_M$ is the set of blue vertices, and $D_M$ is the set of red, dashed vertices, and $Z_M$ contains only the dotted, black vertex.

**Figure 1** A Maximum Carpool Matching example.

traveling together. Formally, each vertex $u \in V$ has a size $s(u) \in \mathbb{N}$, and the constraint $d_M^{\text{in}}(v) \le c(v)$ is replaced with the constraint $\sum_{u:(u,v)\in M} s(u) \le c(v)$. Notice that Knapsack is the special case where only arcs directed at a single vertex have non-zero (integral) weights.

**Related work.** Agatz et al. [5] outlined the optimization challenges that arise when developing technology to support ride-sharing and survey the related operations research models in the academic literature. Hartman et al. [15] designed several heuristic algorithms for the Maximum Carpool Matching problem and compared their performance on real data. Other heuristic algorithms were developed by Knapen et al. [17]. Hartman [14] proved that the Maximum Carpool Matching problem is NP-hard even in the case where the weight function is binary and $c(v) \le 2$ for every $v \in V$. In addition, Hartman presented a natural integer linear program and showed that if the set of drivers is known, then an optimal assignment of passengers to drivers can be found in polynomial time using a reduction to Network Flow (see also [18].) Kutiel [18] presented a $\frac{1}{3}$-approximation algorithm for Maximum Carpool Matching that is based on a Minimum Cost Flow computation and a local search $\frac{1}{2}$-approximation algorithm for the unweighted variant of Maximum Carpool Matching. The latter starts with an empty matching and tries to improve the matching by turning a single passenger into a driver.

Nguyen et al. [19] considered the Spanning Star Forest problem. A *star forest* is a graph consisting of vertex-disjoint star graphs. In the Spanning Star Forest problem, we are given an undirected graph $G$, and the goal is to find a spanning subgraph which is a star forest that maximizes the weight of edges that are covered by the star forest. Notice that this problem is equivalent to Maximum Carpool Matching on undirected and uncapacitated instances. We also note that if all weights leaving a vertex are the same, then the instance is referred to as vertex-weighted. Nguyen et al. [19] provided a PTAS for unweighted planner graphs and a polynomial-time $\frac{3}{5}$-approximation algorithm for unweighted graphs. They gave an exact optimization algorithm for weighted trees, and used it on a maximum spanning tree of the input graph to obtain a $\frac{1}{2}$-approximation algorithm for weighted graphs. They also shows that it is NP-hard to approximate unweighted Spanning Star Forest within a ratio of $\frac{259}{260} + \varepsilon$, for any $\varepsilon > 0$. Chen et al. [13] improved the approximation ratio for unweighted graphs from $\frac{3}{5}$ to 0.71 and gave a 0.64-approximation algorithm for vertex weighted graphs.

They also showed that the edge- and vertex-weighted problem cannot be approximated to within a factor of $\frac{19}{20}+\varepsilon$, and $\frac{31}{32}+\varepsilon$, resp., for any $\varepsilon > 0$, assuming that P $\neq$ NP. Chakrabarty and Goel [11] improved the lower bounds to $\frac{10}{11}+\varepsilon$ and $\frac{13}{14}$.

Athanassopoulos et al. [7] improved the ratio for the unweighted case to $\frac{193}{240} \approx 0.804$. They considered a natural family of *local search* algorithms for Spanning Star Forest. Such an algorithm starts with the solution where all vertices are star centers. Then, it repeatedly tries to turn $t \leq k$ from leaves to centers and $t+1$ centers to leaves. A change is made if it results in a feasible solution, namely if each leave is adjacent to at least one center. The algorithm terminates when such changes are no longer possible. Athanassopoulos et al. [7] showed that, for any $k$ and $\varepsilon \in (0, \frac{1}{2(k+2)}]$, there exists an instance $G$ and a local optima whose size is smaller than $(\frac{1}{2}+\varepsilon)\text{OPT}$, where OPT is the size of the optimal spanning star forest. We note that, for a given $k$, the construction of the above result requires that the maximum degree of $G$ is at least $2(k+2)$. Hence, this result does not hold in graphs with maximum degree $\Delta$.

Arkin et al. [6] considered the Maximum Capacitated Star Packing problem. In this problem the input consists of a complete undirected graph with non-negative edge weights and a capacity vector $c = \{c_1, \ldots, c_p\}$, where $\sum_{i=1}^{p} c_i = |V| - p$. The goal is to find a set of vertex-disjoint stars in $G$ of size $c_1, \ldots, c_p$ of maximum total weight. Arkin et al. [6] provided a local search algorithm whose approximation ratio is $\frac{1}{3}$, and a matching-based $\frac{1}{2}$-approximation algorithm for the case where edge weights satisfy the triangle inequality.

Bar-Noy et al. [8] considered the Minimum 2-Path Partition problem. In this problem the input is a complete graph on $3k$ vertices with non-negative edge weights, and the goal is to partition the graph into disjoint paths of length 2. This problem is the special case of the undirected carpool matching where $c(v) = 2$, for every $v \in V$. They presented two approximation algorithms, one for the weighted case whose ratio is 0.5833, and another for the unweighted case whose ratio is $\frac{3}{4}$.

Another related problem is $k$-Set Packing, where one is given a collection of weighted sets, each containing at most $k$ elements, and the goal is to find a maximum weight subcollection of disjoint sets. Chandra and Halldórsson [12] presented a $\frac{3}{2(k+1)}$-approximation algorithm for this problem. Maximum Carpool Matching can be seen as a special case of $k$-Set Packing with $k = c_{\max} + 1$. Consider a subset of vertice $U$ of size at most $k$. Observe that each subset of vertices has an optimal internal assignment of passenger to drivers. Let the weight of this assignment be the profit of $U$, denoted by $p(U)$. If $k = O(1)$, $p(U)$ can be computed for every $U$ of size at most $k$ in polynomial time. The outcome is a $k$-Set Packing instance. This leads to a $\frac{3}{2(c_{\max}+2)}$-approximation algorithm when $c_{\max} = O(1)$.

**Our contribution.** Section 2 contains approximation algorithms for Maximum Carpool Matching. First, in Section 2.1 we show that Maximum Carpool Matching can be formulated as an unconstrained submodular maximization problem, thus it has a $\frac{1}{2}$-approximation algorithm due to [10, 9]. We present a local search algorithm for Maximum Carpool Matching which repeatedly checks whether the current carpool matching can be improved by means of a star centered at a vertex, and it terminates when such a step is not possible. The approximation ratio of this algorithm is $\frac{1}{2}$ if weights are polynomially bounded, and its ratio is $\frac{1}{2} - \varepsilon$ in general.

In Section 3 we consider Maximum Carpool Matching with bounded maximum capacity. In Section 3.1 we show that Maximum Carpool Matching is APX-hard even for undirected and unweighted instances with $\Delta \leq b$, for any $b \geq 3$. In Section 3.2 we provide another local search algorithm, whose approximation ratio is $\frac{1}{2} + \frac{1}{2c_{\max}} - \varepsilon$, for any

$\varepsilon > 0$, for unweighted Maximum Carpool Matching, where $c_{\max} \triangleq \max_{v \in V} c(v)$. Given a parameter $k$, our algorithm starts with the empty carpool matching. Then, it repeatedly tries to find a better matching by replacing $t \leq k$ arcs in the current solution by $t+1$ arcs that are not in the solution. We show that our analysis is tight. We also note that our algorithm falls within the local search family defined in [7]. However, on undirected and uncapaciated instances we have that $c_{\max} = \Delta$, and as mentioned above the result from [7] does not hold in bounded degree graphs.

Finally, Section 4 discusses Maximum Group Carpool Matching. We show that the unconstrained submodular maximization formulation for Maximum Carpool Matching does not work for Maximum Group Carpool Matching. We show, however, that this problem still admits a $(\frac{1}{2} - \varepsilon)$-approximation algorithm by extending our first local search algorithm. In addition, we show that the second local search algorithm generalizes to unweighted Maximum Group Carpool Matching with the same approximation ratio.

## 2 Approximation Algorithms

We present two algorithms for Maximum Carpool Matching: a $\frac{1}{2}$-approximation algorithm that is based on formulating the problem as an unconstrained submodular maximization problem and a local search $(\frac{1}{2} - \varepsilon)$-approximation algorithm. While the latter does not improve upon the former, it will be shown (in Section 4) that it can be generalized to Maximum Group Carpool Matching without decreasing the approximation ratio.

### 2.1 Submodular Maximization

In this section we show that the Maximum Carpool Matching problem can be formulated as an unconstrained submodular maximization problem, and thus it has a $\frac{1}{2}$-approximation algorithm due to Buchbinder et al. [10, 9].

Given a Maximum Carpool Matching instance $(G = (V, A), c, w)$, consider a subset $S \subseteq V$. Let $M(S)$ be a maximum weight carpool matching satisfying $D_{M(S)} \subseteq S \subseteq V \setminus P_{M(S)}$, namely $M(S)$ is the best carpool matching whose drivers belong to $S$ and whose passengers belong to $V \setminus S$. In other words, $M(S)$ is the maximum weight carpool matching that is a subset of $A \cap (V \setminus S) \times S$. Given $S$, the carpool matching $M(S)$ can be computed in polynomial time by computing a maximum $b$-matching in the bipartite graph $B = (V \setminus S, S, A \cap (V \setminus S) \times S)$ which can be done using an algorithm for Minimum Cost Flow as shown in [18].

Consider the function $\bar{w} : 2^V \to \mathbb{R}$, where $\bar{w}(S) \triangleq w(M(S)) = \sum_{e \in M(S)} w(e)$. Observe that $\bar{w}(\emptyset) = \bar{w}(V) = 0$, and that $\bar{w}$ is not monotone. In the next lemma we prove that $\bar{w}$ is a *submodular set function*. Recall that a function $f$ is submodular if $f(S) + f(T) \geq f(S \cup T) + f(S \cap T)$ for every two sets $S$ and $T$ in the domain of $f$.

▶ **Lemma 1.** *$\bar{w}$ is submodular.*

**Proof.** Consider any two subsets $S, T \subseteq V$. We show that $\bar{w}(S) + \bar{w}(T) \geq \bar{w}(S \cup T) + \bar{w}(S \cap T)$. Let $M(S \cup T)$ and $M(S \cap T)$ be optimal carpool matchings with respect to $S \cup T$ and $S \cap T$. To prove the lemma we construct two feasible carpool matchings $M_S$ and $M_T$ such that $M_S \subseteq (V \setminus S) \times S$, $M_T \subseteq (V \setminus T) \times T$, and $M_S \cup M_T = M(S \cup T) \cup M(S \cap T)$. The lemma follows, since $\bar{w}(S) \geq w(M_S)$ and $\bar{w}(T) \geq w(M_T)$.

First, add all the edges in $M(S \cup T)$ entering $S \setminus T$ to $M_S$. Similarly, add all the edges in $M(S \cup T)$ entering $T \setminus S$ to $M_T$. Observe that $d^{\text{in}}_{M_S}(v) = d^{\text{in}}_{M(S \cup T)}(v) \leq c(v)$, for every $v \in S \setminus T$ and that $d^{\text{in}}_{M_T}(v) = d^{\text{in}}_{M(S \cup T)}(v) \leq c(v)$, for every $v \in T \setminus S$. Next, add the edges in $M(S \cap T)$ leaving $T \setminus S$ to $S$ and add the edges in $M(S \cap T)$ leaving $S \setminus T$ to $T$. It remains to distribute the edges leaving $V \setminus (S \cup T)$ and entering $S \cap T$ in both $M(S \cup T)$ and $M(S \cap T)$. Note that there may exist edges $(v, u)$, where $v \notin S \cup T$, and $u \in S \cap T$ such that $(v, u) \in M(S \cup T)$ and $M(S \cap T)$. We refer to this edges as *duplicate* edges. We add all edges leaving $V \setminus (S \cup T)$ and entering $S \cap T$ in $M(S \cap T)$ to $M_S$. Notice that this is possible, since after this addition we have that $d^{\text{in}}_{M_S}(v) \leq d^{\text{in}}_{M(S \cap T)}(v) \leq c(v)$, for every vertex $v \in S \cap T$. Then we add all duplicate edges in $M(S \cup T)$ to $M_T$. The remaining edges are distributed between $M_S$ and $M_T$ without violating capacities. This can be done, since $d^{\text{in}}_{M(S \cup T)}(v) + d^{\text{in}}_{M(S \cap T)}(v) \leq 2c(v)$, for every $v \in S \cap T$. ◀

Buchbinder et al. [10, 9] presented a general $\frac{1}{2}$-approximation algorithm for unconstrained submodular maximization, thus we have the following theorem.

▶ **Theorem 2.** *There exists a polynomial time $\frac{1}{2}$-approximation algorithm for* Maximum Carpool Matching.

## 2.2 A Star Improvement Algorithm

In this section we give a local search $(\frac{1}{2} - \varepsilon)$-approximation algorithm for Maximum Carpool Matching. This algorithm repeatedly checks whether the current carpool matching $M$ can be improved by means of a star centered at a vertex $v$. The profit from this star is the total weight of the arcs in the star, and the cost is the total weight of lost arcs (e.g., arcs from passengers to drivers that became passengers of $v$). If the profit is larger than the cost, then an improvement step is performed. The algorithm terminates when such a step is not possible. We remind the reader that this algorithm will be extended to Maximum Group Carpool Matching in Section 4.

We need a few definitions before presenting our algorithm. Given a directed graphs $G = (V, A)$, define $N^{\text{in}} \triangleq \{u : (u, v) \in A\}$ and $N^{\text{out}} \triangleq \{u : (v, u) \in A\}$. Let $M$ be a feasible carpool matching. The weight $w_M(v)$ of a vertex $v$ with respect to $M$ is the sum of the weights of the arcs in $M$ that are incident on $v$, namely

$$w_M(v) \triangleq w(M \cap N^{\text{in}}) + w(M \cap N^{\text{out}}) = \textstyle\sum_{(u,v) \in M} w(u, v) + \sum_{(v,u) \in M} w(v, u) \ .$$

For a subset of vertices $U \subseteq V$ we define $w_M(U) \triangleq \sum_{v \in U} w_M(v)$.

We now argue that, with respect to any carpool matching $M$, the total weight of all the vertices is equal to twice the weight of the matching.

▶ **Observation 3.** *$w_M(V) = 2w(M)$.*

**Proof.** $\displaystyle\sum_{v \in V} w_M(v) = \sum_{v \in V} \sum_{(u,v) \in M} w(u, v) + \sum_{v \in V} \sum_{(v,u) \in M} w(v, u) = 2 \sum_{e \in M} w(e).$ ◀

Denote by $\delta(u, v)$ the difference between the weight of the arc and the weight of its source vertex, that is: $\delta_M(u, v) \triangleq w(u, v) - w_M(u)$. For a subset $S \subseteq A$ of arcs define $\delta(S) \triangleq \sum_{(u,v) \in S} \delta(u, v)$.

A subset $S_v$ of arcs entering a vertex $v$, whose size is not greater than the capacity of $v$, is called an *improvement* to vertex $v$ if $\delta(S_v)$ is greater than the value of $v$. More formally,

▶ **Definition 4.** A subset $S_v \subseteq A \cap (V \times \{v\})$ is an *improvement* with respect to a carpool matching $M$, if $|S_v| \leq c(v)$ and $\delta_M(S_v) > w_M(v)$. Furthermore, if there exists an improvement for a vertex $v$, we say that vertex $v$ can be *improved.*

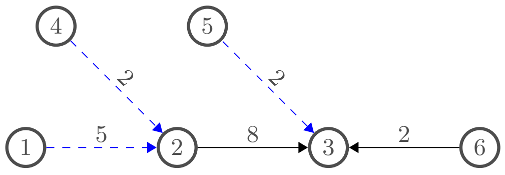


**Figure 2** In this example $M$ is the set of the blue, dashed arcs. In this case $w_M(2) = 7$, $w_M(5) = 2$, and $w_M(6) = 0$. Also, $\delta_M(2,3) = 1$ and $\delta_M(6,3) = 2$. The set $\{(2,3),(6,3)\}$ is an *improvement* to vertex 3 and $\Gamma(2,3) = \{(1,2),(4,2),(3,5),(6,3)\}$.

**Algorithm 1: StarImprove**$(G, c)$

```
1 $M \leftarrow \emptyset$
2 repeat
3     done $\leftarrow$ TRUE
4     for $v \in V$ do
5         if there exists an improvement $S_v$ then
6             $M \leftarrow M \setminus \Gamma(S_v) \cup S_v$
7             done $\leftarrow$ FALSE
8 until done;
```

Given an arc $(u,v) \in A$, let $\Gamma(u,v)$ be the set of arcs that incident $(u,v)$, namely define $\Gamma(u,v) \triangleq (N^{\text{in}}(u) \times \{u\}) \cup (\{v\} \times N^{\text{out}}(v))$. If $S$ is a set of arcs, then $\Gamma(S) \triangleq \bigcup_{(u,v)\in S} \Gamma(u,v)$. Figure 2 depicts all the above definitions.

We are now ready to describe our local search algorithm, which is called **StarImprove** (Algorithm 1). It starts with an empty carpool matching $M$, and in every iteration it looks for a vertex that can be improved. If there exists such a vertex $v$, then the algorithm removes the arcs that are incident on it from $M$, and adds the arcs in $S_v$. The algorithm terminates when no vertex can be improved. Figure 3 depicts an improvement step.

We proceed to bound the approximation ratio of the algorithm, assuming termination.

For a vertex $v$ and a set $S$ of edges entering $v$, let $N_S^{\text{in}}(v) = \{u : (u,v) \in S\}$ be the set in-neighbors corresponding to $S$.

▶ **Lemma 5.** *Let $M$ be a matching computed by **StarImprove**. Let $v$ be a vertex with no improvement, and let $S \subseteq N_M^{in}(v)$, such that $|S| \le c(v)$, then $w(S) \le w_M(v) + w_M(N_S^{in}(v))$.*

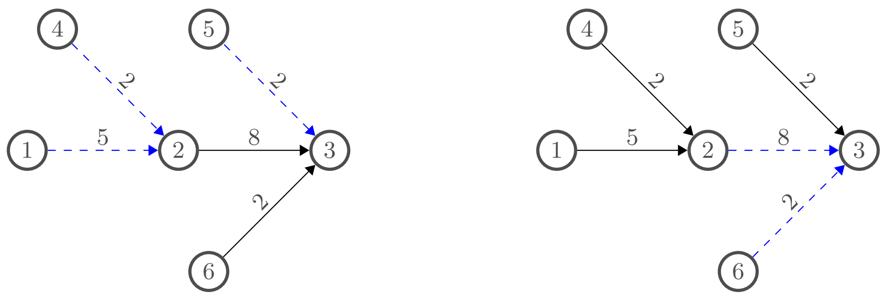
**Proof.** If no improvement exists, then we have that
$w(S) - w_M(N_S^{\text{in}}(v)) = \sum_{(u,v)\in S}(w(u,v) - w_M(u)) = \delta_M(S) \le w_M(v)$ . ◀

To bound the approximation ratio of the algorithm, we use a charging scheme argument.

▶ **Lemma 6.** *If **StarImprove** terminates, then the computed solution is $\frac{1}{2}$-approximate.*

**Proof.** Let $M$ be the matching produced by the algorithm, and let $M^*$ be an optimal matching. We load every vertex $v$ with an amount of money equal to $w_M(v)$, and then we show that this is enough to pay for every arc in the optimal matching. Due to Observation 3 the total amount of money that we use is exactly twice the weight of $M$.

Consider a driver $v \in D_{M^*}$, and let $S = (V \times \{v\}) \cap M^*$. By lemma 5 we know that $w(S) \le w_M(v) + w_M(N_S^{\text{in}}(v))$, thus we can pay for $S$, using the money on $v$ and on $N_S^-(v)$. Clearly, these vertices will not be charged again. ◀



**(a)** A matching that can be improved. **(b)** The matching after improving vertex 3.

**Figure 3** An improvement example.

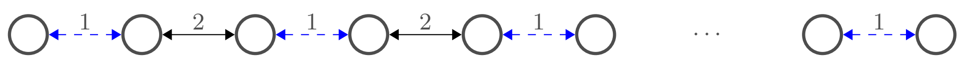


**Figure 4** Consider a path with $2n+1$ arcs, and alternating arc weights (2 and 1), if **StarImprove** selects all arcs of weight 1, then no further improvement can be done and the value of the matching is $n+1$, while the optimal matching has value of $2n$.

We show that our analysis is tight using in Figure 4.

It remains to consider the running time of the algorithm.

▶ **Theorem 7.** *Algorithm* ***StarImprove*** *is a $\frac{1}{2}$-approximation algorithm for* Maximum Carpool Matching*, if edge weights are integral and polynomially bounded.*

**Proof.** First, observe that determining if a vertex $v$ can be improved can be done efficiently by considering the incoming arcs to $v$ in a non-increasing order of their $\delta_M$s, and only ones with positive values. A vertex $v$ can be improved, then, if the $\delta$s of the first $c(v)$ (or less) arcs sum up to more than $w_M(v)$. It follows that the running time of an iteration of the for-loop is polynomial. Since the edge weights are integral and polynomially bounded, the weight of an optimal carpool matching is polynomially bounded. The algorithm runs in polynomial time, because in each iteration the algorithm improves the weight of the matching by at least one or otherwise it terminates. ◀

It remains to consider the case of general weights. It can be shown that one can use standard scaling and rounding to ensure a polynomial running time in the cost of a $(1+\varepsilon)$ factor in the approximation ratio. The proof is omitted for lack of space.

▶ **Theorem 8.** *There exists a $(\frac{1}{2}-\varepsilon)$-approximation algorithm for* Maximum Carpool Matching*, for every $\varepsilon \in (0, \frac{1}{2})$.*

## 3 Constant Maximum Capacity

In this section we study the Maximum Carpool Matching problem when the maximum capacity is constant, i.e., when $c_{\max} = O(1)$. We show that this variant of the problem is APX-hard even for unweighted and undirected instances. We also describe and analyze a local search algorithm for the unweighted variant of the problem, and show that the algorithm achieves a $\frac{1}{2} + \frac{1}{2c_{\max}} - \varepsilon$ approximation ratio, for any $\varepsilon > 0$.

## 3.1 Hardness

As we mentioned earlier, Spanning Star Forest has a lower bound of $\frac{10}{11}+\varepsilon$ for any $\varepsilon > 0$, unless P=NP [11], and this bound applies to Maximum Carpool Matching. The result, however, does not hold for the case where $\Delta = O(1)$ (and $c_{\max} = O(1)$). In this section we show that the problem remains APX-hard even in this case.

Formally, the (unweighted) Minimum Dominating Set problem is defined as follows. The input is an undirected graph $G = (V, E)$, and a feasible solution, or a *dominating set*, is a subset $D \subseteq V$ that dominates $V$ namely such that $D \cup \bigcup_{v \in D} N(v) = V$, where $N(v)$ is the neighborhood of $v$. The goal is to find a minimum cardinality dominating set. Minimum Dominating Set-$b$ is the special case of Minimum Dominating Set in which the maximum degree of a vertex in the input graph $G$ is bounded by $b$. The problem was shown to be APX-hard, for $b \geq 3$, by Papadimitriou and Yannakakis [20].

We now consider the unweighted and undirected special case of the Maximum Carpool Matching problem. In this case, the input consists of an undirected graph $G$ and a capacity function $c$, and the goal is to find a carpool matching $M$ that maximizes $|P_M|$.

Given an undirected graph $G$, let $D^*$ be a minimum cardinality dominating set, and let $M^*$ be an optimal carpool matching with respect to $G$ and the capacity function: $c(v) = \deg(v)$, for every $v \in V$.

▶ **Observation 9.** $|P_{M^*}| + |D^*| = |V|$

**Proof.** Given a carpool matching $M$, observe that $D_M \cup Z_M$ is a dominating set. In the other direction, a dominating set $D$ induces a carpool matching of size $|V \setminus D|$. ◀

We use this duality to obtain a hardness result for Maximum Carpool Matching.

▶ **Theorem 10.** *The* Maximum Carpool Matching *problem is APX-hard, even for undirected and unweighted instances with maximum degree bounded by $b$, for $b \geq 3$.*

**Proof.** We prove the theorem by presenting an $L$-reduction from Minimum Dominating Set-$b$. (For details on $L$-reductions the reader is referred to [20].) We define a function $f$ from Minimum Dominating Set-$b$ instances to Maximum Carpool Matching instances as follows: $f(G) = (G, c)$, where $c(v) = \deg(v)$, for every $v \in V$. Next, we define a function $g$ that given a carpool matching computes a dominating set as follows: $g(M) = V \setminus P_M$. Both $f$ and $g$ can be computed in polynomial time.

Let $D^*$ be an optimal dominating set with respect to $G$, and let $M^*$ be an optimal carpool matching with respect to $G$ and $c$. Since $|D^*| \geq \frac{|V|}{b+1}$, it follows that $|P_{M^*}| \leq b|D^*|$, In addition, if $M$ is a carpool matching, we have that $|D_M \cup Z_M| - |D^*| = (|V| - |P_M|) - |D^*| = |P_{M^*}| - |P_M|$. Hence, there is an $L$-reduction from Minimum Dominating Set-$b$ to unweighted and undirected Maximum Carpool Matching with bounded capacity $b$. ◀

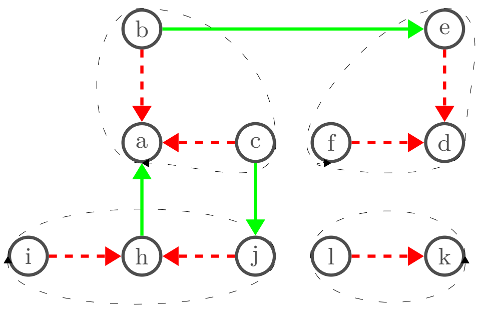
## 3.2 Local Search

In this section we present a local search $(\frac{1}{2} + \frac{1}{2c_{\max}} - \varepsilon)$-approximation algorithm for unweighted Maximum Carpool Matching whose running time is polynomial if $c_{\max} = O(1)$.

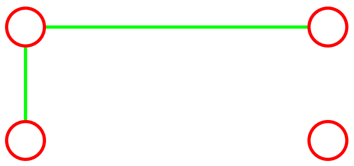
Let $k$ be a constant integer to be determined later. Algorithm **EdgeSwap** (Algorithm 2) maintains a feasible matching $M$ throughout its execution and operates in iterative manner where in each iteration it tries to find a better solution by replacing a subset of at most $k$ edges in the current solution with another (larger) subset of edges not in the solution. The algorithm halts when no improvement can be done.

**Algorithm 2: EdgeSwap**$(G, c, k)$

```
1  $M \leftarrow \emptyset$
2  repeat
3      $done \leftarrow$ true
4      forall $M' \subseteq M : |M'| \leq k$ do
5          forall $A' \subseteq A \setminus M : |A'| = |M'| + 1$ do
6              if $M \setminus M' \cup A'$ is feasible then
7                  $M \leftarrow M \setminus M' \cup A'$
8                  $done \leftarrow$ false
9  until $done$;
10 return $M$
```



**(a)** $M^*$ is depicted by the dashed red edges, and $M$ is depicted by the solid green edges. The optimal stars are outlined.



**(b)** The star graph: each vertex corresponds to a star in $G$. The upper left vertex corresponds to the star that contains the vertices a,b,c.

**Figure 5** An example of a star graph.

Algorithm **EdgeSwap** terminates in polynomial time, since in every non-final iteration it improves the value of the solution by one. Thus, after at most $n$ iterations the algorithm terminates. In every iteration the algorithm examines all subsets of edges of a fixed size and tests for feasibility, both these operations can be done in polynomial time.

Observe that a vertex $v \in D_M \cup Z_M$ is the center of a *directed star* whose leaves are the passengers in the set $P_M(v) = \{u : (u, v) \in M\}$ ($P_M(v) = \emptyset$, for $v \in Z_M$). Given a carpool matching $M$, we define $\mathcal{S}(M)$ to be the set of stars that are induced by $M$. Denote by $V(S)$ the set of vertices of a star, i.e., if $v$ is the center of $S$, then $V(S) = \{v\} \cup P_M(v)$. Also, let $A(S)$ be the arcs of $S$. For $\mathcal{T} \subseteq \mathcal{S}(M)$, define $V(\mathcal{T}) \triangleq \bigcup_{S \in \mathcal{T}} V(S)$ and $A(\mathcal{T}) \triangleq \bigcup_{S \in \mathcal{T}} A(S)$.

It remains to analyze the approximation ratio of **EdgeSwap**. Let $M^*$ be an optimal matching, and let $M$ be the matching computed by **EdgeSwap**. Given both matchings we build the *star graph* in which each vertex represents a star from the optimal solution, namely from $\mathcal{S}(M^*)$, and an edge exists between two vertices if there is a star in $\mathcal{S}(M)$ that intersects the two corresponding stars of the optimal solution. Formally $H = (\mathcal{S}(M^*), E)$ where $E = \left\{(S_i^*, S_j^*) : \exists S \in \mathcal{S}(M), V(S) \cap V(S_i^*) \neq \emptyset \wedge V(S) \cap V(S_j^*) \neq \emptyset\right\}$. Figure 5 depicts a star graph.

▶ **Lemma 11.** *The maximum degree of $H$ is $c_{\max}(c_{\max} + 1)$.*

**Proof.** Each star in $\mathcal{S}(M^*)$ contains at most $c_{\max} + 1$ vertices and each such vertex can belong to a star in $\mathcal{S}(M)$ containing additional $c_{\max}$ vertices, each of which is located in a different star in $\mathcal{S}(M^*)$. ◀

In what follows we compare $|M|$ and $|M^*|$ in maximal connected components of the star graph $H$. Intuitively, we show that $M$ is optimal on small maximal components, and that the approximation ratio on medium (non-necessarily) components can be bounded due to the termination condition of **EdgeSwap**. Large maximal components will be partitioned into medium components.

We first show that large connected graphs (or maximal connected components) can be partitioned into medium size components. The proof is omitted for lack of space.

▶ **Lemma 12.** *An undirected connected graph $G = (V, E)$ with maximum degree $\Delta$, can be decomposed into connected components of size at least $\ell$ and at most $\Delta\ell$, if $\ell \leq |V|$.*

Define $\deg_M(v) \triangleq d^{\text{in}}_M(v) + d^{\text{out}}_M(v)$. For a subset $U \subseteq V$ of vertices define $\deg_M(U) \triangleq \sum_{v \in U} \deg_M(u)$. Observe that $|M| = \frac{1}{2}\deg_M(V)$.

In the next lemma we bound the degree ratio in a component that contains stars with at most $k$ arcs.

▶ **Lemma 13.** *Let $\mathcal{T} \subseteq \mathcal{S}(M^*)$ that induces a connected subgraph of $H$. If $|A(\mathcal{T})| \leq k$, then*

$$\frac{\deg_M(V(\mathcal{T}))}{\deg_{M^*}(V(\mathcal{T}))} \geq \frac{1}{2} + \frac{1}{2c_{\max}} - \frac{1}{2c_{\max}|\mathcal{T}|} .$$

**Proof.** Consider the solution $M'$ obtained from $M$ by removing all the edges from $M$ that intersect $V(\mathcal{T})$ and adding all the edges from $M^*$ that intersect $V(\mathcal{T})$. Observe that if an edge $(u, v)$ in $M^*$ intersects $V(\mathcal{T})$, then $\{u, v\} \in V(\mathcal{T})$ by the definition of the graph $H$. Hence, $M'$ is feasible carpool matching.

Since $\mathcal{T}$ induces a connected subgraph of $H$, the removal of edges in $M$ that intersect $V(\mathcal{T})$ decreased $|M|$ by at most $\deg_M(V(\mathcal{T})) - |\mathcal{T}| + 1$. On the other hand, the increase in size is exactly $\frac{1}{2}\deg_{M^*}(V(\mathcal{T})) \leq c_{\max}|\mathcal{T}|$. Since $|A(\mathcal{T})| \leq k$, we know that this difference can not be positive, or else, EdgeSwap would not have terminated. Thus $\frac{1}{2}\deg_{M^*}(V(\mathcal{T})) \leq \deg_M(V(\mathcal{T})) - |\mathcal{T}| + 1$, and so

$$\frac{\deg_M(V(\mathcal{T}))}{\deg_{M^*}(V(\mathcal{T}))} \geq \frac{1}{2} + \frac{|\mathcal{T}| - 1}{\deg_{M^*}(V(\mathcal{T}))} \geq \frac{1}{2} + \frac{|\mathcal{T}| - 1}{2c_{\max}|\mathcal{T}|} = \frac{1}{2} + \frac{1}{2c_{\max}} - \frac{1}{2c_{\max}|\mathcal{T}|} ,$$

as required. ◀

It remains to bound the approximation ratio of **EdgeSwap**.

▶ **Lemma 14.** *If $k \geq c_{\max}$, then $|M| \geq (\frac{1}{2} + \frac{1}{2c_{\max}} - \frac{c_{\max}(c_{\max}+1)}{2k}) \cdot |M^*|$.*

**Proof.** Consider a maximal (with respect to set inclusion) connected component of $H$ induced by the vertices in $\mathcal{T} \subseteq \mathcal{S}(M^*)$. If $|M \cap A(\mathcal{T})| \leq k$, then it must be that $|M \cap A(\mathcal{T})| = |M^* \cap A(\mathcal{T})|$, since otherwise $M \cap A(\mathcal{T})$ could be improved.

It remains to consider a maximal component $\mathcal{T}$ such that If $|M \cap A(\mathcal{T})| > k$. Since the number of edges in $S \in \mathcal{S}(M^*)$ is at most $c_{\max}$, it must be that $|V(\mathcal{T})| > \frac{k}{c_{\max}}$. Due to Lemma 12 (with $\ell = \frac{k}{c^2_{\max}(c_{\max}+1)}$) we can partition $\mathcal{T}$ into connected components each of which contains between $\frac{k}{c^2_{\max}(c_{\max}+1)}$ and $\frac{k}{c_{\max}}$ vertices. Since each such vertex set $\mathcal{X}$ is connected and contains at most $\frac{k}{c_{\max}}$ stars, it follows that $|A(\mathcal{X})| \leq k$. Due to Lemma 13 we have that $\frac{\deg_M(V(\mathcal{X}))}{\deg_{M^*}(V(\mathcal{X}))} \geq \frac{1}{2} + \frac{1}{2c_{\max}} - \frac{c_{\max}(c_{\max}+1)}{2k}$. Since $\deg_{M^*}(V(\mathcal{T})) = \sum_{\mathcal{X}} \deg_{M^*}(V(\mathcal{X}))$ and $\deg_M(V(\mathcal{T})) = \sum_{\mathcal{X}} \deg_M(V(\mathcal{X}))$, it follows that $\frac{\deg_M(V(\mathcal{T}))}{\deg_{M^*}(V(\mathcal{T}))} \geq \frac{1}{2} + \frac{1}{2c_{\max}} - \frac{c_{\max}(c_{\max}+1)}{2k}$, and thus $|M \cap A(\mathcal{T})| \geq \left(\frac{1}{2} + \frac{1}{2c_{\max}} - \frac{c_{\max}(c_{\max}+1)}{2k}\right)|M^* \cap A(\mathcal{T})|$. ◀

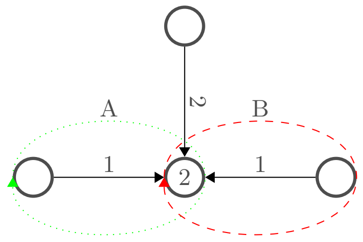


**Figure 6** An unweighted Maximum Carpool Matching instance. Capacities are written inside vertices, and arcs are labeled with their size. We have that $\bar{w}(A) + \bar{w}(B) = 2 < 3 = \bar{w}(A \cup B) + \bar{w}(A \cap B)$.

By setting $k = \lceil c_{\max}(c_{\max} + 1)/2\varepsilon \rceil$, we get the following result.

▶ **Corollary 15.** *There exists a $(\frac{1}{2} + \frac{1}{2c_{\max}} - \varepsilon)$-approximation algorithm for unweighted* Maximum Carpool Matching, *for every $\varepsilon > 0$.*

## 4 Group Carpool

We now consider Maximum Group Carpool Matching which a variant of Maximum Carpool Matching in which we are given a size function $s : V \to \mathbb{N}$, and the constraint $d^{\text{in}}_M(v) \le c(v)$ is replaced with the constraint $\sum_{u:(u,v)\in M} s(u) \le c(v)$.

We start by showing that this variant of the problem does not fit the submodular maximization formulation as defined for Maximum Carpool Matching. Recall the submodular maximization formulation given in Section 2.1, namely $\bar{w} : 2^V \to \mathbb{R}$, where $\bar{w}(S) \triangleq w(M(S))$ and $M(S)$ is the maximum weight carpool matching that satisfies $D_{M(S)} \subseteq S \subseteq V \setminus P_{M(S)}$. Figure 6 contains an instance that shows that the function $\bar{w}$ is not submodular anymore.

We show that Maximum Group Carpool Matching has a $(\frac{1}{2} - \varepsilon)$-approximation algorithm by extending the algorithm from Section 2.2. The main concern when trying to adopt the algorithm to Maximum Group Carpool Matching is how to determine if a vertex can be improved. With Maximum Carpool Matching, if weights are polynomially-bounded, it was enough to consider the incoming arcs to a vertex $v$ in a non-increasing order of $\delta_M$ (see proof of Theorem 7). This does not work anymore, since in the Maximum Group Carpool Matching we have sizes. In fact, given $v$, finding the best star with respect to $\delta_M$ is a Knapsack instance where the size of the knapsack is $c(v)$. If weights are polynomially-bounded, then $\delta_M(e)$ is bounded for every arc $e \in A$, and therefore this instance of Knapsack can be solved in polynomial time using dynamic programming.

▶ **Theorem 16.** *Algorithm **StarImprove** is a $\frac{1}{2}$-approximation algorithm for* Maximum Group Carpool Matching, *if edge weights are integral and polynomially bounded.*

Using standard scaling and rounding we obtain the following result.

▶ **Theorem 17.** *There exists a $(\frac{1}{2} - \varepsilon)$-approximation algorithm for* Maximum Group Carpool Matching, *for every $\varepsilon \in (0, \frac{1}{2})$.*

Finally, we show that a variant of Algorithm **EdgeSwap** from Section 3.2 can be used to solve Maximum Group Carpool Matching while keeping the same approximation guarantees. The only difference is that when checking feasibility of a set of arcs we do not compare the number of passengers to the capacity of a driver, but rather compare the total size of the passengers to the capacity.

▶ **Theorem 18.** *There exists a* $(\frac{1}{2} + \frac{1}{2c_{\max}} - \varepsilon)$*-approximation algorithm for unweighted* Maximum Group Carpool Matching*, for every* $\varepsilon > 0$*.*

**Acknowledgements.** We thank David Adjiashvili and Reuven Bar-Yehuda for helpful discussions.

## References

1 Blablacar. `https://www.blablacar.com`.
2 Moovit carpool. `https://moovitapp.com/`.
3 Waze. `https://www.waze.com/`.
4 Zimride by enterprise. `https://zimride.com/`.
5 Niels A. H. Agatz, Alan L. Erera, Martin W. P. Savelsbergh, and Xing Wang. Optimization for dynamic ride-sharing: A review. *European Journal of Operational Research*, 223(2):295–303, 2012.
6 Esther M. Arkin, Refael Hassin, Shlomi Rubinstein, and Maxim Sviridenko. Approximations for maximum transportation with permutable supply vector and other capacitated star packing problems. *Algorithmica*, 39(2):175–187, 2004.
7 Stavros Athanassopoulos, Ioannis Caragiannis, Christos Kaklamanis, and Maria Kyropoulou. An improved approximation bound for spanning star forest and color saving. In *34th International Symposium on Mathematical Foundations of Computer Science*, pages 90–101, 2009.
8 Amotz Bar-Noy, David Peleg, George Rabanca, and Ivo Vigan. Improved approximation algorithms for weighted 2-path partitions. In *23rd Annual European Symposium on Algorithms*, volume 9294 of *LNCS*, pages 953–964, 2015.
9 Niv Buchbinder and Moran Feldman. Deterministic algorithms for submodular maximization problems. In *27th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 392–403, 2016.
10 Niv Buchbinder, Moran Feldman, Joseph Naor, and Roy Schwartz. A tight linear time (1/2)-approximation for unconstrained submodular maximization. *SIAM J. Comput.*, 44(5):1384–1402, 2015.
11 Deeparnab Chakrabarty and Gagan Goel. On the approximability of budgeted allocations and improved lower bounds for submodular welfare maximization and GAP. *SIAM J. Comput.*, 39(6):2189–2211, 2010.
12 Barun Chandra and Magnús M. Halldórsson. Greedy local improvement and weighted set packing approximation. *J. Algorithms*, 39(2):223–240, 2001.
13 Ning Chen, Roee Engelberg, C. Thach Nguyen, Prasad Raghavendra, Atri Rudra, and Gyanit Singh. Improved approximation algorithms for the spanning star forest problem. *Algorithmica*, 65(3):498–516, 2013.
14 Irith Ben-Arroyo Hartman. Optimal assignment for carpooling. submitted.
15 Irith Ben-Arroyo Hartman, Daniel Keren, Abed Abu Dbai, Elad Cohen, Luk Knapen, Ansar-Ul-Haque Yasar, and Davy Janssens. Theory and practice in large carpooling problems. In *5th International Conference on Ambient Systems, Networks and Technologies*, pages 339–347, 2014.
16 Luk Knapen, Daniel Keren, Ansar-Ul-Haque Yasar, Sungjin Cho, Tom Bellemans, Davy Janssens, and Geert Wets. Estimating scalability issues while finding an optimal assignment for carpooling. In *4th International Conference on Ambient Systems, Networks and Technologies*, pages 372–379, 2013.
17 Luk Knapen, Ansar-Ul-Haque Yasar, Sungjin Cho, Daniel Keren, Abed Abu Dbai, Tom Bellemans, Davy Janssens, Geert Wets, Assaf Schuster, Izchak Sharfman, and Kanishka

Bhaduri. Exploiting graph-theoretic tools for matching in carpooling applications. *J. Ambient Intelligence and Humanized Computing*, 5(3):393–407, 2014.

**18** Gilad Kutiel. Approximation algorithms for the maximum carpool matching problem. In *12th International Computer Science Symposium in Russia*, volume 10304 of *LNCS*, pages 206–216, 2017.

**19** C. Thach Nguyen, Jian Shen, Minmei Hou, Li Sheng, Webb Miller, and Louxin Zhang. Approximating the spanning star forest problem and its application to genomic sequence alignment. *SIAM J. Comput.*, 38(3):946–962, 2008.

**20** Christos Papadimitriou and Mihalis Yannakakis. Optimization, approximation, and complexity classes. In *12th Annual ACM Symposium on Theory of Computing*, pages 229–234, 1988.