Optimisation for the Ride-Sharing Problem: a Complexity-based Approach

Paper-ID 241

Abstract. The dial-a-ride problem is a classic challenge in transportation and continues to be relevant across a large spectrum of applications, e.g. door-to-door transportation services, patient transportation, etc. Recently a new variant of the dial-a-ride problem, called ride-sharing, has received attention due to emergence of the use of smartphone-based applications that support location-aware transportation services. The general dial-a-ride problem involves complex constraints on a time-dependent network. In ride-sharing riders (resp. drivers) specify transportation requests (resp. offers) between journey origins and destinations. The two sets of participants, namely riders and drivers, have different constraints; the riders have time windows for pickup and delivery, while drivers have a starting time window, a destination, and a vehicle capacity. The challenge is to maximise the overall utility of the participants in the system which can be defined in a variety of ways. In this paper we study variations of the ride-sharing problem, under different notions of utility, from a computational complexity perspective, and identify a number of tractable and intractable cases. These results provide a basis for the development of efficient methods and heuristics for solving problems of real-world scale.

1 INTRODUCTION

With the growth in the use of smartphones, social networks, and personal GPS, as well as increasing transportation and fuel costs, congestion, and other environmental concerns, there has been a growing level of interest in ride-sharing services. One-time ride-sharing services opportunistically match riders with drivers who are travelling along their route. This is a significant commercial growth area, and many commercial services are already in place such as Carma,1 Lyft,2 Uber,3 Sidecar,4 and Wingz.5 These services are enabled through smart-phone applications that help match riders and drivers, thereby providing taxi-like services into commuters at a fraction of the costs. In most ride-sharing scenarios riders pay a modest distance-based fee to the driver of their car, with a small commission for the service provider.

From an AI perspective, ride-sharing is a source of complex, possibly online, optimisation problems subject to preferences and uncertainty [11]. From a data mining perspective, there is a significant amount of work on mining transport patterns from GPS trajectory data, which can be used to establish typical travel plans of citizens [17, 8]. A comprehensive review of the ride-sharing problem, its variants, solution techniques, and challenges is available [6].

We focus on the matching problem between riders and drivers. We study a variant of the “inclusive ride-sharing” problem in which both the origin and destination of a passenger are on the route of the matched driver [6]. This matching is often framed as an optimisation problem in which a distance- or cost-based objective is minimised.

Contributions. We focus on a variant of inclusive ride-sharing that has been considered as part of an ongoing collaboration with an industry partner, Avego,6 who implement the Carma service mentioned earlier. We assume that each car has a capacity of \( k + 1 \) partners – one driver and \( k \) passengers. The objective is to maximise the satisfaction of the all users, specifically that every passenger finds a matching driver, and each driver finds at least one matching passenger. More specifically, we consider two different objective functions:

\[
\begin{align*}
    & \text{O}_1 \text{ Maximise the number of satisfied participants by including a maximum number of satisfied users in the matching. In practice, matchings in which some cars contain only one rider along with the driver are acceptable.} \\
    & \text{O}_2 \text{ Maximise the extent to which riders are shared equally amongst cars, in which the extreme can be used to perfectly balance passengers across participating cars. In some cities, such as San Francisco, drivers need a specified number of passengers in a car in order to benefit from carpooling incentives. This objective allows us to maximise the extent to which participants can benefit from those incentives. Thus, the problem becomes a matching with the same number of users per car.}
\end{align*}
\]

We study the computational complexity of these problems under two scenarios: one in which the set of drivers is specified, and another in which some drivers are willing to participate as riders should this be beneficial to the objective function. We present a novel theoretical analysis of these ride-sharing problems. In the cases where we find polynomial-time complexities, the corresponding algorithms are of practical value in the real-world setting. Our approach provides novelty over the heuristic approaches proposed in the artificial intelligence literature [11], as well as the general optimisation variants studied in the operations research literature [1].

2 PRELIMINARIES

Notation. We use various standard notation from graph theory, as well as a number well-known problems. We denote by \( G = (V, E) \) a graph with a set of vertices, \( V \), and the set of edges, \( E \). For each vertex \( v \), \( d^+(v) \) denotes the degree of \( v \), and \( N(v) \) the neighbourhood of \( v \). In the bipartite case, we define \( G_b = (V, W, E) \) where \( V \) (resp. \( W \)) represents the first set of vertices (resp. the second set) and \( E \) the

\[\text{https://car.ma} \]
\[\text{http://www.lyft.me} \]
\[\text{https://www.uber.com} \]
\[\text{http://www.side.cr} \]
\[\text{https://tickengo.com} \]
\[\text{http://www.avego.com} \]
We give an outline for an algorithm to build a graph we simply need to check for each driver the time constraints and riders/drivers; the latter is the case when we allow the possibility of drivers discussing earlier. Thus, let \( V \) be the set of vertices \( V_1, V_2, \ldots, V_m \), each of which can be associated with a city, a neighbourhood, etc. We assume there is negligible travel time to move within the cluster relative to inter-cluster travel. Let \( \mathcal{P}^d = \{ V_1^d, V_2^d, \ldots, V_k^d \} \) be the path of length \((l - 1)\) from the start location \( V_1 \) of the driver \( j \) to his destination \( V_k \). The driver spends a fixed time duration to travel from one vertex to the next one throughout his path; one can deduce a function \( Dur : V \times V \rightarrow R \) that calculates the distance between each vertex of \( V \). Each city contains several riders and/or drivers. Without loss of generality, each vertex \( V_k \) can be assimilated to a set containing riders and/or drivers.

Our approach involves building, in polynomial-time, a graph amongst drivers and riders with an edge between them when a matching is possible. Based on this graphical structure we can study the complexity of potential problems according to the various objectives discussed earlier. Thus, let \( G = (V, E) \) be this graph where \( V = R \cup D \) and \( E \) is the set of feasible matchings between drivers and riders/drivers; the latter is the case when we allow the possibility of drivers opting to ride in a particular matching. To construct this graph we simply need to check for each driver the time constraints between him and the riders/drivers contained in the sets of the path. We study the case where drivers are allowed to pick-up at most \( k \) riders. We consider the optimisation problem in which the objective is to minimise the number of unsatisfied users (driver or riders).

**Definition 1.** Let \( \pi_1 \) be the matching problem where drivers have a fixed travel route, fixed departure and arrival times, cannot select to participate as riders, and can collect at most \( k \) riders in their car. The objective is to minimise the number of unsatisfied users.

**Theorem 1.** The optimisation problem \( \pi_1 \) can be solved in polynomial time.

**Proof.** This problem is equivalent to covering a maximum number of vertices in the graph \( G_1 \), which we will define below, with stars of size less than \( k \). The existence of a solution to this problem (each driver can have at most \( k \) riders with him) leads to suppose that only vertices of \( D \) can be the center of stars of size less than \( k \).

We define a new bipartite graph \( G_2 = (kD, R, E') \) where the first independent set is \( k \) times the set \( D \) and \( E' \) is \( k \) times the set of edges \( E \). To simplify, each vertex associated to a driver is cloned \( k - 1 \) times with the same neighbours. From \( G_2 \), a maximum matching gives an optimal cover with Dstars of size less than \( k \) in polynomial time. Indeed, the \( k \) edges from identical vertices (clones) give the associated \( k \)-stars for each driver.

**3.2 Balancing Riders across Cars (\( \Omega_2 \))**

We consider the case where the goal is to satisfy a maximum of users with only full cars (exactly \( k \) riders and one driver). The decision problem is equivalent to finding \( k \alpha \) users satisfied using only full cars, where \( \alpha \) is given.

**Definition 2.** Let \( \pi_2 \) be the matching problem where drivers have a fixed travel route, fixed departure and arrival times, cannot select to participate as riders, and must have exactly \( k \) passengers. The decision problem involves finding a cover with \( \alpha \) \( k \)-Dstars.
Theorem 2. The decision problem \( \pi_2 \) is \( \mathcal{NP} \)-complete.

Proof. One can observe that this problem is \( \mathcal{NP} \). We will show that \( \pi_2 \) is \( \mathcal{NP} \)-complete by presenting a polynomial reduction from the ExactOneSAT to \( \pi_2 \) as follows.

There are \( \alpha \) different users (driver or rider) and \( \gamma \) different full car configurations containing \( (k+1) \) users. For each user \( u_{i} \), we define a variable \( x_{u_{i}}^{c_{j}} \) for each full car configuration \( c_{j} \) containing \( u_{i} \), and for each configuration \( c_{j} \) a variable \( C_{j} \) is defined. For each \( c_{j} \) if the corresponding passenger configuration is in the solution then variable \( C_{j} \) and all variables associated with the users in the corresponding car must be assigned to true \((k+1) \) users. We have \( (x_{u_{i}}^{c_{j}} \land x_{u_{i}}^{c_{j}} \land \ldots \land x_{u_{i}}^{c_{j}}) \equiv C_{j} \) from which, to get a CNF, we define for each configuration \( c_{j} \) the following clauses \( (u_{i}^{c_{j}}, u_{i}^{c_{j}}, \ldots, u_{i}^{c_{j}}) \) are the users in this particular configuration \( c_{j} \): \( (\neg x_{u_{i}}^{c_{j}} \lor \neg x_{u_{i}}^{c_{j}} \lor \ldots \lor \neg x_{u_{i}}^{c_{j}} \lor C_{j}) \land (\neg C_{j} \lor \neg x_{u_{i}}^{c_{j}}) \land (\neg C_{j} \lor \neg x_{u_{i}}^{c_{j}}) \land \ldots \land (\neg C_{j} \lor \neg x_{u_{i}}^{c_{j}}) \). The set of all possible configurations is denoted \( \mathcal{A} \), and the equivalences between the variables associated with each configuration are denoted \( \mathcal{B} \).

Each user \( u_{i} \) can be included in at most one car configuration, thus we state an AtMostOne constraint on the set of \( C_{j} \) variables associated with configurations containing \( u_{i} \), thus we have AtMostOne\( (C_{j} | u_{i} \text{ in configuration } C_{j}) \). The set of all possible configurations is denoted \( \mathcal{C} \).

The number of car configurations is bounded by the structure of the graph; for each riders, this number is equal to the number of possible matchings with each neighbors (drivers in this case), therefore we have \( \sum_{j \in N(i)} d(j) \). For the users \( d_{i} \) this number is equal to the number of configurations from \( d_{i} \), then we have \( \left( \frac{d(d_{i})}{k} \right) \).

We have \( n \) users and the objective is to fill \( \alpha \) cars with \((k+1)\alpha \) users. The polynomial transformation from this instance to the ExactOneSAT problem is the following CNF with \( \alpha' = (k+2)\alpha \) for the number of variables which must be assigned to true:

\[
\mathcal{A} \equiv \left( x_{u_{i}}^{c_{1}} \lor \neg x_{u_{i}}^{c_{2}} \lor \ldots \lor \neg x_{u_{i}}^{c_{k+1}} \lor C_{1} \right) \land \left( x_{u_{i}}^{c_{2}} \lor \neg x_{u_{i}}^{c_{2}} \lor \ldots \lor \neg x_{u_{i}}^{c_{k+1}} \lor C_{2} \right) \land \ldots \land \left( x_{u_{i}}^{c_{k+1}} \lor \neg x_{u_{i}}^{c_{k+1}} \lor \ldots \lor \neg x_{u_{i}}^{c_{k+1}} \lor C_{k} \right)
\]

\[
\mathcal{B} \equiv \left( x_{u_{i}}^{c_{1}} \lor \neg x_{u_{i}}^{c_{2}} \lor \ldots \lor \neg x_{u_{i}}^{c_{k+1}} \lor C_{1} \right) \land \left( C_{1} \lor \neg x_{u_{i}}^{c_{1}} \lor \ldots \lor \neg x_{u_{i}}^{c_{k+1}} \right) \land \ldots \land \left( C_{k} \lor \neg x_{u_{i}}^{c_{k+1}} \lor \ldots \lor \neg x_{u_{i}}^{c_{k+1}} \right)
\]

\[
\mathcal{C} \equiv \left( \text{AtMostOne}(C_{j} | u_{i} \text{ in configuration } C_{j}) \right) \land \left( \text{AtMostOne}(C_{j} | u_{i} \text{ in configuration } C_{j}) \right) \land \ldots \land \left( \text{AtMostOne}(C_{j} | u_{i} \text{ in configuration } C_{j}) \right)
\]

\( \Rightarrow \) Let us suppose that there exists a solution to the decision problem \( \pi_2 \). Then we have \( \alpha \) full cars and \((k+1)\alpha \) satisfied users. For each full car \( c_{j} \), the configuration variable \( C_{j} \) is assigned to true, and by \( \mathcal{A} \) and \( \mathcal{B} \) this leads to having, for each user \( u_{i} \) associated with \( c_{j} \), the variable \( x_{u_{i}} \) assigned to true. By the same argument all remaining \( C_{j} \) variables, and the \( x_{u_{i}} \) ones, are assigned to false. The set of constraints in \( \mathcal{C} \) assures the uniqueness of each variable assigned to true. Therefore, we have exactly \((k+2)\alpha \) variables set to true over all sets of variables defined in the problem. To conclude, there exists a solution to the instance of ExactOneSAT with \( \alpha' = (k+2)\alpha \).

\( \Leftarrow \) Let us suppose that there exists a solution of the decision problem ExactOneSAT with exactly \( \alpha' = (k+2)\alpha \) variables are assigned to true. First, we will prove that exactly \( \alpha \) variables \( C_{j} \) must be assigned to true. Let us suppose that exactly \( \alpha+1 \) variables \( C_{j} \) are true. Then by \( \mathcal{A} \) and \( \mathcal{B} \) this leads to having \((k+1)\alpha+(k+1) \) number \( x_{u_{i}} \) variables assigned to true, and unique by \( \mathcal{C} \), so \((k+2)\alpha+(k+2) \) in all, which is impossible. Let us suppose that exactly \( \alpha-1 \) \( C_{j} \) variables are true. Then by \( \mathcal{A} \) and \( \mathcal{B} \) this leads to having \((k+1)\alpha-(k+1) \) \( x_{u_{i}} \) variables assigned to true and unique by \( \mathcal{C} \), so there are \((k+2)\alpha-(k-2) \) variables set to true in all. By supposition all the other \( C_{j} \) variables are assigned to false, so again by \( \mathcal{A} \) and \( \mathcal{B} \) all \( x_{u_{i}} \) remaining are assigned to false. Therefore, it is impossible to obtain a solution for \( \pi_2 \) and thus exactly \( \alpha \) \( C_{j} \) variables are assigned to true.

From these \( \alpha \) true configurations, one can deduce first that \((k+1)\alpha \) unique variables \( x_{u_{i}} \) are assigned to true by \( \mathcal{A} \), \( \mathcal{B} \) and \( \mathcal{C} \), second that \((\gamma-\alpha) \) \( C_{j} \) and \((n-(k+1)\alpha) \) \( x_{u_{i}} \) are assigned to false. The true clauses from \( \mathcal{A} \) give a solution to the problem \( \pi_2 \).

\[ \square \]

4 WHEN DRIVERS MAY CHOOSE TO RIDE

We consider the case where some drivers can opt to become riders if they do not have to drive in an optimal matching. For this case, we cannot use a bipartite graph for the problem model because the set \( D \) is no longer an independent set. Thus we work on a graph \( G = (V, E) \) where \( V = D \cup R \).

4.1 Maximizing the Number of Satisfied Users (O₁)

We consider the case where the objective is to minimise the number of unsatisfied users. Due to the fact that the graph \( G \) is not bipartite, the optimal solution is to match a maximum of users with Dstars of size less than \( k \). Therefore, we propose a polynomial-time maximum cover with Dstars of size less than \( k \) called a \( k \)-Dstar cover in order to solve the following optimisation problem.

Definition 3. Let \( \pi_{3} \) be the ride-sharing matching problem where drivers have a fixed travel route, fixed departure and arrival times, can decide to participate as drivers or riders, and must have at most \( k \) passengers. The aim is to minimise the number of unsatisfied users (driver or riders alone).

A problem similar to the \( k \)-Dstar cover has previously been studied in the literature [12]. However, we present the idea of finding alternating paths with properties that we need for the algorithm to compute an optimal solution for the problem \( \pi_{3} \). In this case only drivers can be the centre of stars. Therefore, we present this new version of \( k \)-star cover with new properties. This results is a generalisation of the special case studied in [18] where \( k = 2 \).

Definition 4 (\( k \)-Dstar cover). Let \( G = (V, E) \) be a graph, a \( k \)-Dstar cover \( M \) is a set of edges such that the connected components of the partial graph induced by \( M \) are either simple vertices, or any Dstars of size less than \( k \).
Definition 5 (M-covered vertex). An M-covered vertex (resp. M-non-covered) is a vertex which belongs (resp. does not belong) to at least one edge in M. The set of vertices covered by M (resp. non-covered by M) will be denoted by $S(M)$ (resp. $NS(M)$). An edge of M between two riders does not exist, by definition.

Definition 6 (Maximum k-Dstar cover). In a maximum k-Dstar cover, the number of covered vertices is maximum, therefore the number of non-covered vertices is minimum.

Definition 7 (Vertex degree in relation to M). Let $M$ be a k-Dstar cover in a graph $G = (V,E)$. For each $i = 1,\ldots,n$, let $d_M^-(x_i)$ be the number of edges of $M$ which are incident to $x_i$.

We will now give the definition of an alternating path in a k-Dstar cover which is similar to the classical alternating path in a maximum matching [3].

Definition 8 (M-alternating path). Let $M$ be a k-Dstar cover in a graph $G = (V,E)$, an M-alternating path $C = x_0, x_1, \ldots, x_l$ is a path in $G$ such that for $i = 0,\ldots,\left\lfloor \frac{l}{2} \right\rfloor - 1$, $x_0 \in NS(M)$, $(x_i, x_{i+1}) \notin M$, and if $k \neq (2i + 1)$ then $(x_{2i+1}, x_{2i+2}) \in M$. Note that for each edge in $M$, one of these vertices is of type driver by Definition 5.

Definition 9 (“Backbone” of an M-alternating path). Let $M$ be a k-Dstar cover in a graph $G = (V,E)$, and $C = x_0, x_1, \ldots, x_l$ an M-alternating path in $G$. The “backbone”, denoted by $T$, associated with the path $C$ is composed of $C$, the edges of $M$ which are incident to $C$, and eventually the extremities of these edges (see Figure 1). Note that for each edge in $M$, only one extremity can be incident at one or more vertices in $T$. And this extremity must be of type driver.

Remark 1. If $T$ contains a cycle, there exists $e \in M$ that links two vertices of $C$. If one of these vertices is not an extremity of $T$ then we will have a path of length three in $M$; all the vertices are covered by edges of $C$ except, eventually, the extremities. By definition, $x_0 \in NS(M)$, thus $T$ contains a cycle when the last vertex $x_l$ of $C$ is connected to another vertex of $C$ by an edge $e \in M$ and $e \notin C$; see the illustration in Figure 2(a). Note that $d_M^-(x_l) = 1$.

Definition 10 (Augmenting M-alternating path). Let $C = x_0, x_1, \ldots, x_l$ be an M-alternating path, and $x_0 \in NS(M)$. $C$ is an augmenting M-alternating path if the cardinality of the k-Dstar cover given by $C$ can be increased by changing the membership in $M$ of all the edges of $C$, except possibly for the last one. After this modification, each edge of $M$ still contains a vertex of type driver.

Remark 2. From Remark 1, a path of length three or four can be created due to the augmenting operation used in Definition 10. Let $e$ be the edge of $M$ that creates the cycle in $T$ and thus creates the path of length three or four after the augmenting operation. Then, the edge $e$ can be removed from $M$ in order to increase the number of covered vertices by the k-Dstar cover (see Figure 2(b)).

Definition 11 (Even vertex and odd vertex). Let $C = x_0, x_1, \ldots, x_l$ be an M-alternating path, and $x_0 \in NS(M)$. A vertex $x_i$ with an index equal to an even number (resp. odd number) in $C$ is called an even vertex (resp. odd vertex).

We now present Lemma 1 about the augmenting M-alternating paths and the fundamental Theorem 3 of the k-Dstar cover with M-alternating paths.

Lemma 1. Let $M$ be a k-Dstar cover, $C = x_0, x_1, \ldots, x_l$ an M-alternating path with $x_0 \in NS(M)$, and let $T$ be the backbone associated with the M-alternating path $C$. $C$ is an augmenting M-alternating path if and only if there exists a vertex $x_{2i-1}$, $i \in \mathbb{N}^*$, of type driver such that $d_M^-(x_{2i-1}) \neq k$, or of type rider such that $d_M^-(x_{2i-1}) \neq 1$.

Proof. Proof by contradiction.

$\Rightarrow$ Suppose that $C$ is an augmenting M-alternating path, and suppose that an odd vertex $x_{2i-1}$ of type driver such that $d_M^-(x_{2i-1}) \neq k$ does not exist (so $T$ does not contain any cycle). Thus, $C$, and its backbone $T$ have the shape shown in Figure 3.

From Definition 10 and Remark 2, if $T$ does not contain any cycle, we can simply increase the cardinality of the covered vertices in the path $C$ by changing the membership in $M$ of all the edges of $C$. If we change the edge $\{x_0, x_1\}$ in $M$ in order to cover $x_0$, the edge $\{x_1, x_2\}$ must change, else $x_1$ will be a $(k+1)$-Dstar centre. In this way, we change the membership of $\{x_1, x_2\}$, which means that we must change $\{x_2, x_3\}$. Recursively, we will change the membership to $M$ of all $C$ edges. Thus, the last vertex $x_l$ will not be covered, and $C$ will not be an augmenting M-alternating path. This is inconsistent with the former assumptions. Therefore, there exists a vertex $x_{2i-1}$ such that $d_M^-(x_{2i-1}) \neq k$.

If we suppose that an odd vertex $x_{2i-1}$ of type rider such that $d_M^-(x_{2i-1}) \neq 1$ and $d_M^-(x_{2i}) \neq 1$ does not exist, with the same process we show that it is inconsistent.

$\Leftarrow$ If $T$ contains a cycle, then $C$ contains an augmenting M-alternating path (see Remark 1). Else, suppose that $T$ does not contain a cycle, and that there exists a vertex $x_{2i-1}$, $i \in \mathbb{N}^*$, of type driver such that $d_M^-(x_{2i-1}) \neq k$ or of type rider such that $d_M^-(x_{2i-1}) \neq 1$ and $d_M^-(x_{2i}) \neq 1$. We will show that $C$ is an augmenting M-alternating path. Let $x_i = x_{2i-1}, i \in \mathbb{N}^*$, be the first odd vertex on the M-alternating path with $d_M^-(x_i) < k$. We have 3 cases:

1. $d_M^-(x_i) = 0$ where $x_i$ is of any type, $C$ ends with a non-covered vertex. So $C$ is an augmenting M-alternating path (see illustration in Figure 4.a).
2. $d_M^-(x_i) = 1$ and $d_M^-(x_{i+1}) = 1$ where $x_{i+1}$ is type driver, the M-alternating path $C$ contains an edge $(x_j, x_{j+1}) \in M$ whose extremities have a degree equal to 1. We remove the part of the
path that is after this edge, this part is already covered. Thus, we have an $M$-alternating sub-path, in which all the vertices of odd index have a degree equal to $k$ and the sub-path end is an edge $(x_j, x_{j+1})$. It is easy to see that this sub-path is an augmenting $M$-alternating path by changing the membership in $M$ of the edges of $C$, except the last one $(x_j, x_{j+1})$. So $C$ is an augmenting $M$-alternating path (see illustration in figure 4).

3. $d_M^3(x_j) = 1$ and $1 < d_M^3(x_{j+1}) \leq k$ where $x_j$ is of any type, the $M$-alternating path $C$ owns an odd vertex with degree equal to 1 adjacent to an even vertex with degree strictly greater than 1. We change the membership in $M$ of the edge between these two vertices, and we remove the path part which is after the odd vertex with degree equal to 1, this part is already covered. Thus, we have an $M$-alternating sub-path, in which all the vertices of odd index have a degree equal to $k$, and the sub-path end is like a non-covered vertex. It is easy to see that this sub-path is an augmenting $M$-alternating path by changing the membership in $M$ of all $C$ edges as in the first case. So $C$ is an augmenting $M$-alternating path (see figure 4).

Finally, we give the theorem of the equivalence between $k$-Dstar cover and augmenting $M$-alternating path. The proof is a generalisation of the result in [18] and is related, although not covered by the one in [12]. Because of lack of space, the proof is omitted.

**Theorem 3.** Let $M$ be a $k$-Dstar cover in a graph $G$, $M$ admits a maximum cardinality if and only if $G$ does not possess an augmenting $M$-alternating path.

From Theorem 3, we can now introduce the algorithm which gives a maximum $k$-Dstar cover, and thus an optimal solution to the problem $\pi_3$. Let $M$ be a $k$-Dstar cover, and let $C$ be an augmenting $M$-alternating path. The algorithm substitutes covered edges for non-covered edges in $C$, except one of the edges at the end according to different cases seen in Lemma 1. We denote this operation Augmenting$(M, C)$, which results in a new $k$-Dstar cover which covers one or two vertices more than $M$. It is very important to start from a maximum matching, indeed each edge of $M$ will contains a vertex associated to a driver $d_i$. From this matching, the Augmenting$(M, C)$ algorithm will search augmenting $M$-alternating path where the centre of each star is a driver $d_i$. The algorithm that creates a maximum $k$-Dstar cover is:

**Data:** $G = (V, E)$

**Result:** A $k$-star cover $M$

begin
$M :=$ a Maximum Matching of $G$;
while there exists an augmenting $M$-alternating path $C$ do
$M :=$ Augmenting$(M, C)$
end
Return $M$;
end

**Algorithm 1:** Research of a maximum $k$-Dstar cover

The algorithm that searches an augmenting $M$-alternating path from a non-covered vertex $x_0$ is based on a “breadth first search tree” where the root is $x_0$. For each vertex, we check if the distance to $x_0$ is odd, and then we select the first vertex of type driver whose degree is less than $k$ or of type rider whose degree and the degree of its neighbour equal to 1 according to $M$.

The breadth first search has complexity $O((n + m))$ where $n$ (resp. $m$) is the number of vertices (resp. edges). In the worst case we search $n$ times an augmenting $M$-alternating path. The Algorithm 1 is performed in $O(n^2)$.

### 4.2 Balancing Riders across Cars ($O_2$)

We consider again the perfect balanced objective where $k\alpha$ users are satisfied with $\alpha$ full cars.

**Definition 12.** Let $\pi_4$ be the ride-sharing matching problem where drivers have a fixed travel route, fixed departure and arrival times, can decide to participate as drivers or riders, and must have exactly $k$ passengers. The decision problem involves finding a cover with $\alpha$ $k$-Dstars.

**Corollary 1.** From the Theorem 2, we can assume that the decision problem $\pi_4$ is $NP$-complete. Indeed, the only difference between the two problems is the number $\gamma$ of configurations of full cars. For each driver the number of permutations is equal to the configurations associated with the drivers’ neighbours $N(d_i)$ and the ones from $d_i$, giving us $\sum_{j \in N(d_i)} \left( \binom{d'(j)}{k - 1} \right) + \binom{d'(d_i)}{k}$. The proof and the number of variables set to true remains the same.

### 5 RELATED WORK

There are two complementary research communities who are interested in the ride-sharing problem. Firstly in the artificial intelligence community there is a line of work that focuses on using data mining techniques to extract human mobility patterns from personal GPS data [8, 16, 9, 17, 20]. Closely related to this is the study of the evolution of networks over time [4], which can be useful to predict how mobility patterns will change into the future.
Ride-sharing viewed as a problem of creating and coordinating amongst shared plans has also been studied. Kumar et al. develop and evaluate computational methods for guiding collaboration that demonstrate how shared plans can be created in real-world settings, where agents can be expected to have diverse and varying goals, preferences, and availabilities [11]. This is a complex setting in which formal notions of fairness and efficiency are required, which can be achieved through concepts from mechanism design and game theory. Yousaf et al. focus on encouraging people to use a ride-sharing system by satisfying their demands in terms of safety, privacy, convenience and also provide enough incentives for drivers and riders [22]. They formulate the problem as a multi source-destination path planning problem. In contrast, we have studied a particular set of instances of ride matching that compiles all feasible matches into a graph over which we can find optimal matches in polynomial time for cars of known maximum capacity. A number of alternative approaches to ride-matching have also been proposed in the AI literature, such as an auction-based approach [13] and genetic algorithm approaches [10]. These are complementary to our approach in that different objective functions are at play.

In the operations research literature a significant body of work has been reported on the dial-a-ride problem [1, 5], in which a taxi-like service is provided to multiple users who have little or no access to public services. Such services are often used with a permanent or long term health issues, or who are unable to access public transport. Dial-a-ride problems are often viewed as a form of dynamic pick-up and delivery problem [7]. They typically involve the management of a single vehicle, but sometimes a multi-vehicle fleet is considered [2].

Solving dial-a-ride problems is typically very challenging for systematic optimisation methods. Therefore, these problems are typically solved using heuristic methods [15] or variants of local search [19]. Studies of the problem in the presence of dynamic requests, time-windows, and uncertainty have also been reported [21].

6 CONCLUSION AND FUTURE WORK

We have presented a novel theoretical analysis of ride-sharing problem. We proposed a mathematical model of the problem and a transformation into a compatible ride-sharing graph. This allowed us to study the problem according to several constraints and objectives.

Our complexity results show that maximising the number of satisfied ride-sharing users can be achieved in polynomial time when each car has a known maximum capacity that should not be exceeded, regardless of whether or not some drivers are willing to participate as riders. We believe that these results for this polynomial case can be extended to settings where cars have different capacities, which would make our results even more general in practical. However, if we require a solution that perfectly balances occupancy across cars then the problem becomes NP-complete.

Our ongoing work in this area is focusing on the use of preferences in the ride-matching problem. The approach we have presented in this paper can be immediately applied in the case where preferences are interpreted as ruling-out particular matches. However, the more interesting situations arise when preferences provide an implicit ranking over matching proposals. We are also interested in considering the problem of maximising the likelihood of finding an acceptable match in the setting where users have a probability of approving a proposed match.

REFERENCES