ECAI 2014
T. Schaub et al. (Eds.)

doi:10.3233/978-1-61499-419-0-831

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

Gilles Simonin and **Barry O'Sullivan** [1]

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 starting and finishing the journey, 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,[2] Lyft,[3] Uber,[4] Sidecar,[5] and Wingz.[6] 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 [10]. From a data mining perspective, there is a significant amount of work on mining transport patterns from GPS trajectory data, which can used to establish typical travel plans of citizens [16, 7]. A comprehensive review of the ride-sharing problem, its variants, solution techniques, and challenges is available [5].

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 [5]. 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,[7] 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:

- $\mathbf{O_1}$ 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.
- $\mathbf{O_2}$ Maximise the extent to which riders are shared equally amongst cars, which in 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.

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 [10], 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, $d^\circ(v)$ denotes the degree of v, and $N(v)$ the neighbourhood

[1] Insight Centre for Data Analytics, University College Cork, Ireland. {gilles.simonin|barry.osullivan}@insight-centre.org
[2] https://car.ma
[3] http://www.lyft.me
[4] https://www.uber.com
[5] http://www.side.cr
[6] https://tickengo.com
[7] http://www.avego.com

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 set of edges between elements (v, w) such that $v \in V$ and $w \in W$. A k-star is a tree of size $k+1$ composed of a central vertex and k leaves. A k-Dstar is a k-star where the centre is a driver.

In Theorem 1, we present a polynomial reduction from the **EXACTONESSAT** problem [13], which is defined as follows:

EXACTONESSAT
Instance: A classical SAT formula, ϕ, in conjunctive normal form, and constant α.
Question: Does these exist a satisfying assignment to ϕ in which exactly α variables are assigned to true.

General Problem Formulation. The general ride-sharing problem setting that we study in this paper can be described as follows. Let R be the set of n riders, and D the set of m drivers. Drivers and riders share a one-time trip close to their desired departure times. Each rider $r_i \in R$ has a time window (TW) within which to complete the journey, $[es_{r_i}, ls_{r_i}]$ (earliest start and latest start) for pick-up time and $[ef_{r_i}, lf_{r_i}]$ (earliest finish and latest finish) for the time by which he/she must reach the destination. Similarly, each driver $d_j \in D$ may have an associated time window $[es_{d_j}, ls_{d_j}]$ for his journey start time s_{d_j}, and a time window $[ef_{d_j}, lf_{d_j}]$ for his arrival time. However, in practice, drivers tend to specify a fixed start, s_{d_j}, and arrival times, f_{d_j}, for journeys. It is this setting we study in this paper: drivers specify time points, while riders specify time windows.

We assume that we can partition the riders and drivers into location-centric clusters each located at a vertex in V, i.e. $\mathcal{V} = \{V_1, V_2, \ldots, V_m\}$, 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 $\mathcal{P}^{d_j} = \{V_1^{d_j}, V_2^{d_j}, \ldots, V_l^{d_j}\}$ be the path of length $(l-1)$ from the start location $V_1^{d_j}$ of the driver j to his destination $V_l^{d_j}$. The driver spends a fixed time duration to travel from one vertex to the next one throughout his path; one can deduce a function $Dur : \mathcal{V} \times \mathcal{V} \rightarrow \mathbb{R}$ that calculates the distance between each set of $\mathcal{V}$. Each city contains several riders and/or drivers. Without lost of generality, each vertex $V_h^{d_j}$ can be seen as 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 give an outline for an algorithm to build G:

- For each driver $d_j \in D$, we have $\mathcal{P}^{d_j} = \{V_1^{d_j}, \ldots, V_l^{d_j}\}$.
- For each set $V_h^{d_j}$ and for each rider (or driver) $r_i \in V_h^{d_j}$, if the destination of r_i (denoted $dest(r_i)$), the time windows of r_i and the time associated with driver d_j are compatible, we add the edge $\{r_i, d_j\}$ in E. The constraint to satisfy is the following:
$$\Big(dest(r_i) \in \mathcal{P}^{d_j} \setminus \{V_z^{d_j} \mid z \leq h\}\Big)$$
$$\wedge \Big(es_{r_i} \leq s_{d_i} + Dur(V_1^{d_j}, V_h^{d_j}) \leq ls_{r_i}\Big)$$
$$\wedge \Big(ef_{r_i} \leq s_{d_i} + Dur(V_1^{d_j}, dest(r_i)) \leq lf_{r_i}\Big)$$

The graph G has a number of important properties. The set of vertices R associated to riders forms an independent set. In the setting where drivers *do not* have the option of becoming riders the set of vertices D associated to drivers forms an independent set. This relaxation leads to a bipartite graph $G_b = (D, R, E)$.

From this graph, the study of different cases of the ride-sharing matching problem become much easier. For example, we can consider the preferences of users by adding weights to the edges or vertices. We can study settings where we seek the best matching that maximises the number of users that are satisfied, or by the matching where every car is completely full or well balanced. Due to the set of riders, which forms an independent set, a matching is simple to find and many optimal solutions can be computed efficiently. In the next two sections we will present different complexity results depending on the objective we wish to optimise and whether or not drivers have the option of playing the role of riders in the final matching.

3 DISTINCT DRIVER AND RIDER GROUPS

In this section we consider the setting where the set of drivers and riders form distinct groups, i.e. drivers do not have the option of becoming riders in a matching, even if this would give a better quality solution. Therefore, the problem involves finding a feasible matching between the sets D and R according to the objective function. In the following we will present complexity results for both objective functions O_1 and O_2, as defined in the Introduction.

3.1 Maximizing the Number of Satisfied Users (O_1)

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 π_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 π_1 can be solved in polynomial time.*

Proof. This problem is equivalent to covering a maximum number of vertices in the graph G_b, 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 $G'_b = (kD, R, E')$ 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'_b, 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 (O_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 α is given.

Definition 2. *Let $\boldsymbol{\pi_2}$ 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 α k-Dstars.*

Theorem 2. *The decision problem π_2 is $\mathcal{NP}$-complete.*

Proof. One can observe that this problem is $\mathcal{NP}$. We will show that π_2 is $\mathcal{NP}$-complete by presenting a polynomial reduction from the EXACTONESSAT to π_2 as follows.

There are n different users (driver or rider) and γ different full car configurations containing $(k+1)$ users. For each user u_i, we define a variable $\boldsymbol{\chi_{u_i}^{c_j}}$ for each full car configuration c_j containing u_i, and for each configuration c_j a variable $\boldsymbol{C_j}$ 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 $(\chi_{u_a}^{c_j} \wedge \chi_{u_b}^{c_j} \wedge \ldots \wedge \chi_{u_z}^{c_j} \models C_j)$ from which, to get a CNF, we define for each configuration c_j the following clauses ($u_a^{c_j}, u_b^{c_j}, \ldots, u_z^{c_j}$ are the users in this particular configuration c_j): $(\neg\chi_{u_a}^{c_j} \vee \neg\chi_{u_b}^{c_j} \vee \ldots \vee \neg\chi_{u_z}^{c_j} \vee C_j) \wedge (\neg C_j \vee \chi_{u_a}^{c_j}) \wedge (\neg C_j \vee \chi_{u_b}^{c_j}) \wedge \ldots \wedge (\neg C_j \vee \chi_{u_z}^{c_j})) \wedge (C_j \vee \neg\chi_{u_a}^{c_j}) \wedge (C_j \vee \neg\chi_{u_b}^{c_j}) \wedge \ldots \wedge (C_j \vee \neg\chi_{u_z}^{c_j}))$. The set of all possible configurations is denoted Ⓐ, and the equivalences between the variables associated with each configuration are denoted Ⓑ.

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 in configuration C_j). The set of all possible configurations is denoted Ⓒ.

The number of car configurations is bounded by the structure of the graph; for each riders r_i this number is equal to the number of possible matchings with each neighbours (drivers in this case), therefore we have $\sum_{j \in N(r_i)} \begin{pmatrix} d^\circ(j)-1 \\ k-1 \end{pmatrix}$. For the drivers d_i this number is equal to the number of configurations from d_i, then we have $\begin{pmatrix} d^\circ(d_i) \\ k \end{pmatrix}$.

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

Ⓐ
$$\left[\begin{array}{ll} & (\neg\chi_{u_1}^{c_1} \vee \neg\chi_{u_2}^{c_1} \vee \ldots \vee \neg\chi_{u_{k+1}}^{c_1} \vee C_1) \\ \wedge & (\neg\chi_{u_1}^{c_2} \vee \neg\chi_{u_2}^{c_2} \vee \ldots \vee \neg\chi_{u_8}^{c_2} \vee C_2) \\ \vdots & \vdots \\ \wedge & (\neg\chi_{u_n}^{c_\gamma} \vee \neg\chi_{u_6}^{c_\gamma} \vee \ldots \vee \neg\chi_{u_{11}}^{c_\gamma} \vee C_\gamma) \end{array}\right]$$
$\wedge$

Ⓑ
$$\left[\begin{array}{ll} & (\neg C_1 \vee \chi_{u_1}^{c_1}) \wedge \ldots \wedge (\neg C_1 \vee \chi_{u_{k+1}}^{c_1})) \\ \wedge & (C_1 \vee \neg\chi_{u_1}^{c_1}) \wedge \ldots \wedge (C_1 \vee \neg\chi_{u_{k+1}}^{c_1})) \\ \vdots & \vdots \\ \wedge & (\neg C_\gamma \vee \chi_{u_n}^{c_\gamma}) \wedge \ldots \wedge (\neg C_\gamma \vee \chi_{u_{11}}^{c_\gamma})) \\ \wedge & (C_\gamma \vee \neg\chi_{u_n}^{c_\gamma}) \wedge \ldots \wedge (C_\gamma \vee \neg\chi_{u_{11}}^{c_\gamma})) \end{array}\right]$$
for C_1 … for C_γ

$\wedge$

Ⓒ
$$\left[\begin{array}{ll} & (\text{ATMOSTONE}(C_j|\ u_1 \text{ in configuration } C_j)) \\ \wedge & (\text{ATMOSTONE}(C_j|\ u_2 \text{ in configuration } C_j)) \\ \vdots & \vdots \\ \wedge & (\text{ATMOSTONE}(C_j|\ u_n \text{ in configuration } C_j)) \end{array}\right]$$

$\Rightarrow$ Let us suppose that there exists a solution to the decision problem π_2. Then we have α full cars and $(k+1)\alpha$ satisfied users. For each full car c_j, the configuration variable C_j is assigned at true, and from Ⓐ and Ⓑ this leads to having, for each user u_i associated with c_j, the variable χ_{u_i} assigned to true. By the same argument all remaining C_j variables, and the χ_{u_i} ones, are assigned to false. The set of constraints in Ⓒ ensures 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 EXACTONESSAT with $\alpha' = (k+2)\alpha$.

$\Leftarrow$ Let us suppose that there exists a solution of the decision problem EXACTONESSAT with exactly $\alpha' = (k+2)\alpha$ variables assigned to true. First, we will prove that exactly α variables C_j must be assigned to true. Let us suppose that exactly $\alpha+1$ variables C_j are true. Then from Ⓐ and Ⓑ this leads to having $(k+1)\alpha+(k+1)$ number χ_{u_i} variables assigned to true, and unique by Ⓒ, 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 Ⓐ and Ⓑ this leads to having $(k+1)\alpha-(k+1)$ χ_{u_i} variables assigned to true and unique by Ⓒ, 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 Ⓐ and Ⓑ all χ_{u_i} variables remaining are assigned to false. Therefore, it is impossible to obtain a solution for π_2 and thus exactly α C_j variables are assigned to true.

From these α true configurations, one can deduce first that $(k+1)\alpha$ unique variables χ_{u_i} are assigned to true by Ⓐ, Ⓑ and Ⓒ, second that $(\gamma-\alpha)$ C_j and $[n-(k+1)\alpha]$ χ_{u_i} are assigned to false. The true clauses from Ⓐ give a solution to the problem π_2. □

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 (O_1)

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 π_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 [11]. 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 π_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 [17] where $k = 2$.

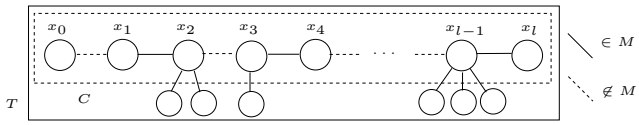


Figure 1. A "backbone" T associated with an M-alternating path C.

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 $d^\circ_M(x_i)$ 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 [2].

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 $i = 0, \ldots, \lceil \frac{l}{2} \rceil - 1$, $x_0 \in NS(M)$, $\{x_{2i}, x_{2i+1}\} \notin M$, and if $k \neq (2i+1)$ $\{x_{2i+1}, x_{2i+2}\} \in M$. 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 extremity of these edges (see Figure 1). Note that for each edge of M in C, 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 C 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_k 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 $d^\circ_M(x_l) = 1$.*

Definition 10 (Augmenting M-alternating path)**.** *Let $C = x_0, \ldots, x_l$ 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 $C = x_0, \ldots, x_l$ 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 $d^\circ_M(x_{2i-1}) \neq k$, or of type rider such that $d^\circ_M(x_{2i-1}) \neq 1 \wedge d^\circ_M(x_{2i}) \neq 1$.*

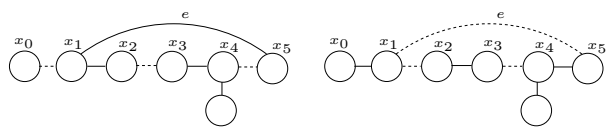
Proof. Proof by contradiction.

$\Rightarrow$ Suppose that C is an augmenting M-alternating path, and suppose that an odd vertex x_{2i-1} of type driver such that $d^\circ_M(x_{2i-1}) \neq k$ 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 $\{x_1, x_2\}$ in M, 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 $d^\circ_M(x_{2_i-1}) \neq k$.

If we suppose that an odd vertex x_{2i-1} of type rider such that $d^\circ_M(x_{2i-1}) \neq 1$ and $d^\circ_M(x_{2i}) \neq 1$ 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). Otherwise, 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 $d^\circ_M(x_{2i-1}) \neq k$ or of type rider such that $d^\circ_M(x_{2i-1}) \neq 1$ and $d^\circ_M(x_{2i}) \neq 1$. We will show that C is an augmenting M-alternating path. Let $x_j = x_{2i-1}$, $i \in \mathbb{N}^*$, be the first odd vertex on the M-alternating path with $d^\circ_M(x_j) < k$. We have 3 cases:



(a) Example: cycle in the backbone T

(b) Example: augmenting operation with a cycle in T

Figure 2. Illustrations for Remark 1.

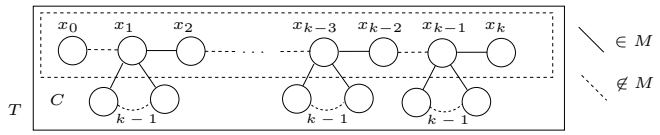


Figure 3. The backbone T associated with C.

1. $d^\circ_M(x_j)=0$ where x_j is of any type, C ends with an non-covered vertex. So C is an augmenting M-alternating path (see illustration in Figure 4.a).
2. $d^\circ_M(x_j) = 1$ and $d^\circ_M(x_{j+1}) = 1$ where x_j 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.b).
3. $d^\circ_M(x_j) = 1$ and $1 < d^\circ_M(x_{j+1}) \leq k$ 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 [17] and is related, although not covered by the one in [11]. 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.*

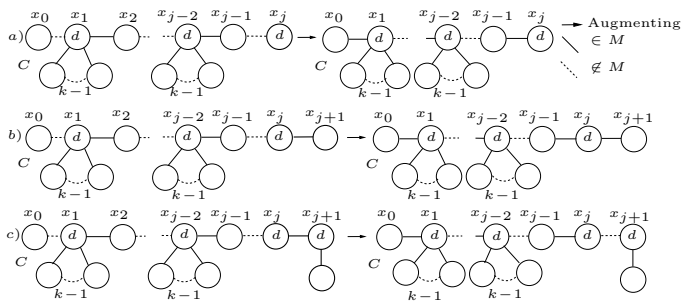


Figure 4. Augmenting of the three cases with $j = 2i - 1$.

From Theorem 3, we can now introduce the algorithm which gives a maximum k-Dstar cover, and thus an optimal solution to the problem π_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 the 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: Creating 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 its 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 α full cars.

Definition 12. *Let π_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 α k-Dstars.*

Corollary 1. *From the Theorem 2, we can assume that the decision problem π_4 is $\mathcal{NP}$-complete. Indeed, the only difference between the two problems is the number γ 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 $\sum_{j \in N(d_i)} \begin{pmatrix} d^\circ(j) - 1 \\ k - 1 \end{pmatrix} + \begin{pmatrix} d^\circ(d_i) \\ k \end{pmatrix}$. The proof and the number of variables set to true remains the same.*

5 RELATED WORK

There are two complementary research communities 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 [7, 15, 8, 16,

19]. Closely related to this is the study of the evolution of networks over time [3], 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 be studied. Kamar 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 [10]. 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 [21]. 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 [12] and genetic algorithm approaches [9]. These are complementary to our work since different objective functions are at play.

In the operations research literature there is a significant body of work on the dial-a-ride problem [1, 4], 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 closely related to dynamic pick-up and delivery problem [6]. Solving dial-a-ride problems is typically very challenging for systematic optimisation methods. Therefore, these problems are typically solved using heuristic methods [14] or variants of local search [18]. Studies of the problem in the presence of dynamic requests, time-windows, and uncertainty have also been reported [20].

6 CONCLUSION AND FUTURE WORK

We have presented a novel theoretical analysis of the ride-sharing problem. We proposed a mathematic 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 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, more interesting situations arise when preferences provide an implicit ranking over matching proposals.

ACKNOWLEDGEMENT. This publication has emanated from research supported in part by a research grant from Science Foundation Ireland (SFI) under Grant Number SFI/12/RC/2289, and from EU FP7-FET Grant 284715 (ICON).

REFERENCES

[1] Niels Agatz, Alan Erera, Martin Savelsbergh, and Xing Wang, 'Optimization for dynamic ride-sharing: A review', *European Journal of Operational Research*, **223**(2), 295 – 303, (2012).
[2] C. Berge, 'Two Theorems in Graph Theory', *Proceedings of the National Academy of Science*, **43**, 842–844, (September 1957).
[3] Michele Berlingerio, Francesco Bonchi, Björn Bringmann, and Aristides Gionis, 'Mining graph evolution rules', *Proceedings of ECML/PKDD*, 115–130, (2009).
[4] Jean-François Cordeau and Gilbert Laporte, 'The dial-a-ride problem (darp): Variants, modeling issues and algorithms', *4OR*, **1**(2), 89–101, (2003).
[5] M. Furuhata, M. Dessouky, F. Ordónez, M-E Brunet, X. Wang, and S. Koenig, 'Ridesharing: the state-of-the-art and future directions', *Preprint submitted to Transportation Research Part B: Methodological (available from the authors)*, (2013).
[6] Berbeglia Gerardo, Jean-François Cordeau, and Gilbert Laporte, 'Dynamic pickup and delivery problems', *European Journal of Operational Research*, **202**(1), 8–15, (2010).
[7] Fosca Giannotti, Mirco Nanni, and Dino Pedreschi, 'Efficient mining of temporally annotated sequences', in *Proc. of SIAM Int. Conf. on Data Mining (SDM 2006)*. SIAM, (2006).
[8] Danhuai Guo, 'Mining traffic condition from trajectories', *Proc. of the Fifth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2008)*, 256–260, (2008).
[9] W. Herbawi and M. Weber, 'The ride-matching problem with time windows in dynamic ridesharing: A model and a genetic algorithm', *Proceedings of the ACM Genetic and Evolutionary Computation Conference (GECCO)*, 1–8, (2012).
[10] Ece Kamar and Eric Horvitz, 'Collaboration and shared plans in the open world: Studies of ridesharing', *IJCAI*, 187, (2009).
[11] Alexander K. Kelmans, 'Optimal packing of induced stars in a graph', *Discrete Mathematics*, **173**(13), 97 – 127, (1997).
[12] A. Kleiner, B. Nebel, and V. A. Ziparo, 'A mechanism for dynamic ride sharing based on parallel auctions', *IJCAI*, 266–272, (2011).
[13] Stefan Kratsch, Dániel Marx, and Magnus Wahlström, 'Parameterized complexity and kernelizability of max ones and exact ones problems', *Proceedings of Mathematical Foundations of Computer Science*, 489 – 500, (2010).
[14] Diana Marco and Dessouky Maged, 'A new regret insertion heuristic for solving large-scale dial-a-ride problems with time windows', *Transportation Research Part B: Methodological*, **38**(6), 539–557, (July 2004).
[15] Anna Monreale, Fabio Pinelli, Roberto Trasarti, and Fosca Giannotti, 'WhereNext: a location predictor on trajectory pattern mining', *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 637–646, (2009).
[16] Mirco Nanni and Dino Pedreschi, 'Time-focused clustering of trajectories of moving objects', *J. Intell. Inf. Syst.*, **27**(3), 267–289, (2006).
[17] Gilles Simonin, Rodolphe Giroudeau, and Jean-Claude König, 'Extended matching problem for a coupled-tasks scheduling problem', *TMFCS'09 : International Conference on Theoretical and Mathematical Foundations of Computer Science, Orlando, Floride*, 082–089, (July 2009).
[18] Parragh Sophie, Doerner Karl, and Hartl Richard, 'Variable neighborhood search for the dial-a-ride problem', *Comput. Oper. Res.*, **37**(6), 1129–1138, (June 2010).
[19] Roberto Trasarti, Fabio Pinelli, Mirco Nanni, and Fosca Giannotti, 'Mining mobility user profiles for car pooling', in *The 17th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, San Diego, CA, USA, (2011).
[20] Zhihai Xiang, Chengbin Chu, and Haoxun Chen, 'The study of a dynamic dial-a-ride problem under time-dependent and stochastic environments', *European Journal of Operational Research*, **185**(2), 534 – 551, (2008).
[21] Jamal Yousaf, Juanzi Li, Lu Chen, Jie Tang, Xiaowen Dai, and John Du, 'Ride-sharing: A multi source-destination path planning approach', *Proceedings of the Australasian Conference on AI*, 815–826, (2012).