Taxi-Sharing: Parameterized Complexity and Approximability of the Dial-a-ride problem with money as an incentive

Dimitri Watel, Alain Faye

Taxi-Sharing: Parameterized Complexity and Approximability of the Dial-a-ride problem with money as an incentive.

Alain Faye[a,c], Dimitri Watel[a,b,]

[a] *ENSIIE, 1 square de la Résistance, 91025, Evry, FRANCE*
[b] *SAMOVAR, Telecom SudParis, 9 Rue Charles Fourier, 91000, Évry, FRANCE*
[c] *CEDRIC, CNAM, 2 rue Conté, 75003, Paris, FRANCE*

Abstract

We study, in this paper, a taxi-sharing problem, called Dial-a-ride problem with money as an incentive (DARP-M). This problem consists in defining a set of taxis that will be shared by different clients in order to reduce their bill by a given factor $\alpha < 1$. To achieve this, each client shares the cost of the ride with other passengers. More precisely, the fragments of the ride in which the client is alone is fully paid by this client and, for each fragment in which the client shares the taxi with other passengers, the cost is equally divided between the passengers. In addition to this cost constraint, the taxi must satisfy a time window constraint for each passenger and a capacity constraint.

We define three versions of the problem: max-DARP-M where the objective is to drive the maximum number of clients with an arbitrarily large number of taxis; max-1-DARP-M in which we want to drive the maximum number of clients with one taxi; and 1-DARP-M which consists in deciding whether it is possible to drive at least one client while satisfying the constraints. We study the parameterized complexity and approximability of those problems with respect to four parameters: the factor α, the capacity *capa* of the taxis, the maximum size W of the time windows of the clients, and the value S of an optimal solution.

Among other results, we prove that 1-DARP-M is NP-Complete and max-DARP-M and max-1-DARP-M cannot be approximated in polynomial time to within any variable ratio even if α, *capa* and W are fixed and if the road network is a planar graph. We also give a polynomial algorithm for max-1-DARP-M for the case where *capa* and W are fixed and where the network does not contain a circuit. This algorithm implies a $\frac{1}{\sqrt{n}}$-polynomial approximation for max-DARP-M.

Keywords: Parameterized complexity, Approximability, Dial-a-ride problem, Taxi-Sharing

1. Introduction

The Dial-a-Ride problem (DARP) consists in the search for an optimal route for many vehicles in order to drive people from their respective origin to their respective destination. This model is used, for example, to determine an optimized route for taxis in order to pick up passengers. We focus in this article on the complexity of a version of this taxi-sharing problem in which the price paid by each passenger is shared. Such a version, called Dial-a-ride problem with money as an incentive, was previously introduced and studied in [19, 20].

Ride-sharing, including Taxi-sharing, has been massively studied for the last fifteen years due to the economical impact and the ecological impact of such a research. Indeed, optimizations reducing the number of vehicles or the number of travels is an obvious way to reduce the costs and the greenhouse gas emissions. DARP can be seen as a subproblem of the general pickup and delivery problem (GPDP) described in [21] in which the goal is to transport a resource from different pickup locations to drop off locations. In DARP, we consider a human resource (the clients) and each pickup or drop off location is associated with exactly one client. The consequence of this specific resource is that one must be aware of the user inconvenience.

1.1. Related work on the DARP problem

DARP can hardly be defined as a unique problem. The feasible and optimal solutions of a Dial-a-ride problem depend on the measure, the fleet parameters and the clients constraints. Thus, the variety of studies about DARP is not surprising.

Considering the measure, one may optimize the vehicle travel cost, see for example [3, 15, 18], the total travel time [10] or the profit [7]. Another option is to maximize the number of satisfied requests or a combination of all those parameters [19, 20, 22].

Some constraints modelize the user convenience. A usual option is to search for a feasible solution considering time windows [4, 10, 18, 22] as it has been done for the more general pickup and devivery problem [8]. This last problem is solved with a column generation scheme where columns define admissible routes. In [10, 18], the authors develop a similar approach merging a branch-and-cut algorithm with column generation. In [4, 22], the problem is solved using a Tabu search heuristic. Another option to modelize the user convenience is to tend to minimize the excess ride time [2, 11, 13].

Finally one can consider either the static problem in which all the requests are known in advance or the dynamic version in which the requests may occur at any time [1, 6, 11, 19, 20], this problem is usually solved using a local search heuristic.

A recent review about the Dial-a-ride problem and some of its generalizations may be found in [14]. We refer the reader to [5, 10] for a more specific review about DARP.

1.2. DARP with Money as an incentive

We focus on a problem where the goal is to find a feasible solution satisfying a client cost constraint. Few paper focused on that constraint. In [19, 20], the authors study the version of the problem in which each client, traveling by taxi, may share the cost of the ride with other passengers. More precisely, the fragments of the ride in which the client is alone is fully paid by this client. On the contrary, for each fragment in which the client shares the taxi with other passengers, the cost is equally divided between the passengers. The problem consists in the search for a ride in which every client does not pay more than the cost he would pay alone in a taxi traveling directly from his origin to his destination. Note that a client can be served by being affected to a private ride but each client must also satisfy a time window constraint. The objective is to maximize the number of served clients. This problem is called Dial-a-Ride problem with Money as Incentive and is denoted by DARP-M.

In [20], the authors give a reduction from the Traveling salesman problem to DARP-M, based on the sole time windows constraint. However, no taxi is shared, all the clients are driven in a private ride. It proves that serving all the clients and satisfying a time windows constraint is NP-Complete. Considering this reduction, DARP-M can be seen as a generalization of TSP in which we add a sharing cost constraint. Although this reduction clearly shows that DARP-M is strongly NP-Complete, it does not reflect the hardness of determining if at least two clients can be served by sharing a taxi while satisfying the cost constraint. That simpler question is not insignificant as it leads to a natural greedy algorithm for DARP-M in which we group clients who can share a taxi until all of them have to be affected to a private rides.

Furthermore, it was shown by [17] that searching for a (not elementary) shortest path between a source and a sink satisfying a time windows constraint is weakly NP-Complete as it can be solved in polynomial time if the width of the time windows is polynomially bounded. Consequently, as the reduction of [20] uses only the time windows constraint and as it is from the strongly NP-Complete problem TSP, it seems that it cannot be easily adapted to prove the hardness of determining if at least two clients can share a taxi.

1.3. Our contributions

We focus on the parameterized complexity and the parameterized approximability of three problems derived from DARP-M defined by [19, 20]. The purpose of this paper is mainly to investigate on how hard the cost constraint is. Particularly, we point out the fact that every hardness result we give is true even if we do not take into account the time windows.

We now formally define the problems we study. We work in a directed graph $G = (V, A)$. We are given a set of n clients arbitrarily numbered in $[\![1; n]\!]$. The i-th *client* is attached to two nodes v_i and v'_i, which are respectively the origin and the destination of the client. We respectively define V_c and V'_c as $\{v_i, i \leq n\}$ and $\{v'_i, i \leq n\}$. A route P of a taxi is defined by a list $(u_1, u_2, \ldots, u_{2 \cdot s(P)})$ of nodes in $V_c \cup V'_c$ where $s(P)$ is an integer. A taxi must satisfy three constraints.

Precedence constraint. For each client i, $v_i \in P$ if and only if $v'_i \in P$. In that case, if $v_i = u_j$ and $v'_i \in u_k$, then $j < k$. We say the client i *travels* in that taxi, or that the taxi *drives* the client i. The value of $s(P)$ can be seen as the number of clients traveling in P.

Capacity constraint. We consider that each taxi has the same number of seats. This number is defined as the *capacity capa* of the taxis. This capacity is at least 2 and is no more than n. For each node $u_j \in P$, let $n_j = \#\{u_k \in V_c, k \leq j\} - \#\{u_k \in V'_c, k \leq j\}$. This value is the number of clients in the taxi immediately after u_j. For every j, $n_j \leq capa$. In addition, if $j \neq 2 \cdot s(P)$, $n_j \geq 1$: a taxi cannot be emptied before the end of the ride.

Time constraint. Each arc $a = (u, v) \in A$ is weighted with a non-negative integer $t(a)$, corresponding to the time that a taxi spends to go from u to v. We extend this function to every couple of nodes in G: $t(u, v)$ is the weight of a shortest path in G from u to v. Each client i is associated with two moments b_i and e_i. A taxi must drive that client between times b_i and e_i. The taxi can start at any moment of the time window of its first client. We respectively define B_c and E_c as the sets containing all the values b_i and e_i for all the clients.

Cost constraint. Each arc $a = (u, v) \in A$ is weighted with a non-negative integer $\omega(a)$, corresponding to the cost that a client would pay alone in a taxi driving from u to v. We extend this function to every couple of nodes in G: $\omega(u, v)$ is the cost of a shortest path in G from u to v. We define the *desired gain* $\alpha < 1$ as the minimum factor reducing the bill of each client. The cost paid is divided between the passengers traveling on the same arc: for each client i traveling in P, if $v_i = u_j$ and $v'_i = u_k$, the cost paid by that client is $\omega_i = \sum_{l=j}^{k-1} \frac{\omega(u_l, u_{l+1})}{n_l}$. This cost must satisfy $\omega_i \leq \alpha \cdot \omega(v_i, v'_i)$. In that case, we say the taxi P *satisfies* the client i.

Note that there would not be any feasible solution if the capacity of the taxi is 1. This is why this case is forbidden.

Remark 1. A taxi P is only defined by waypoints in the road network: the origins where it picks up clients and the destinations where it delivers them. In order to draw the route of the taxi in the network, we follow the shortest paths in G from u_j to u_{j+1} for every $j < 2 \cdot s(P)$. (We assume that a shortest path over the costs ω is also a shortest path over the weights t.) That route is a path of G that can contain intermediate nodes that are neither an origin nor a destination of a client driven by P.

We can now define the problems max-DARP-M, max-1-DARP-M and 1-DARP-M.

Definition 1. Given a directed graph $G = (V, A)$ with non-negative weights ω over the arcs, n clients with their origin V_c, their destinations V'_c and their time windows B_c and E_c, a capacity $capa \leq n$ of the taxis, a desired gain $\alpha < 1$,

- the max-DARP-M problem consists in finding a set $\mathcal{P}$ of taxis satisfying the precedence constraint, the capacity constraint the time constraint and

the cost constraint maximizing $\sum_{P\in\mathcal{P}} s(P)$ such that for each client there is a unique taxi of $\mathcal{P}$ in which that client travels;

- the max-1-DARP-M problem consists in finding a taxi P satisfying the four constraints and maximizing $s(P)$;
- the 1-DARP-M problem consists in deciding whether a taxi P satisfying the four constraints exists.

We define also define two decision problems max-DARP-M$_=$ and max-1-DARP-M$_=$ in which, given an instance of max-DARP-M or max-1-DARP-M and an integer S, we search for a solution for which the objective value equals S. We finally define the parameter W as $\max_{i\in[\![1;n]\!]}(e_i - b_i) + 1$, the maximum width of a time window.

The results are summarized in Table 1.

Table 1: This table summarizes the set of results in the paper. The parameters column specifies which parameter is fixed or polynomialy bounded. An hyphen in the approximability column means that the cell does not make sense, either because the problem is polynomial (or XP), or because the line is about a decision problem. The last column indicates in which theorem/corollary the result is proven.

Graphs	Problem	Parameters	Comp.	Approx.	Result
Planar graphs	max-DARP-M	α, W, *capa*	NP-H	No approx.	Cor. 2.1
	max-1-DARP-M	α, W, *capa*	NP-H	No approx.	Cor. 2.1
	1-DARP-M	α, W, *capa*	NP-C	-	Th. 2.1
All graphs	max-DARP-M$_=$	S	XP	-	Sect. 3.1.3
	max-1-DARP-M$_=$	S	XP	-	Sect. 3.1.3
DAG	max-DARP-M	W	NP-H	No approx.	Th 3.3
		W (poly), *capa*	NP-H	$\frac{1}{\sqrt{n}}$-approx	Cor. 3.2
		α, W, *capa*	NP-H	APX-H	Th. 3.1
		α, *capa*	NP-H	No approx.	Cor. 3.1
	max-DARP-M$_=$	S, α, W, *capa*	W[1]-H	-	Th. 3.3
	max-1-DARP-M	W	NP-H	No approx.	Th 3.3
		W (poly), *capa*	XP	-	Th. 3.4
		α, *capa*	NP-H	No approx.	Cor. 3.1
	max-1-DARP-M$_=$	S, α, W, *capa*	W[1]-H	-	Th 3.3
	1-DARP-M	W	NP-C	-	Th. 3.3
		W (poly), *capa*	XP	-	Th. 3.4
		α, *capa*	NP-C	-	Th. 3.2
		α, W, *capa*	W[1]-H	-	Th. 3.3

Remark 2. We studied the problems in two main cases: planar and acyclic graphs. If the first may seem relevant considering the application, the second is clearly not as a road network hardly is acyclic. However, there exists parameterized algorithms for the three problems in that case. Its could be used if the circuits of the graph are removed (by defining a priority order over the nodes of the graph using, for example, the time windows of the clients). Of course, this restriction removes feasible solutions but according to Table 1, none of those feasible solutions may be build in polynomial or FPT time by any parameterized algorithm.

Remark 3. Note that, on every hardness result where W is fixed, the reduction first consists in setting the durations t to 0 and the time windows to $[0, 0]$, so that the time constraint is trivially satisfied and that $W = 1$. This means that the same hardness results occur even if we remove the time constraint.

2. Planar graphs

This section is dedicated to proving that 1-DARP-M is NP-Complete and that max-DARP-M and max-1-DARP-M are NP-Hard and cannot be approximated to within any constant or variable ratio, even if *capa*, α and W are fixed and if the graph is planar.

In this proof, we fix $capa = 2$ and $\alpha \in]0.5, 1[$. Note that it is possible to adapt the result for any fixed values of *capa* and α. This adaptation is not trivial and make the proof harder to read. That is why we present in this section only the simple case. We also consider that we remove the time constraint by setting $t(a) = 0$ for every arc $a \in A$ and $b_i = e_i = 0$ for all i and, in that case, $W = 1$.

Belonging to NP. We consider the decision version of max-1-DARP-M and max-DARP-M in which, given an instance of the optimization problems and an integer K, we search for a set of taxis or a unique taxi satisfying at least K clients.

Those problems and 1-DARP-M belong to NP as, given a taxi, we can easily determine whether the capacity, the cost and the precedence constraints are satisfied for every client in the taxi and count how many clients are satisfied by the taxi.

NP-hardness: the reduction. In this part, we prove a reduction from the 3-partition problem to 1-DARP-M. We then deduce the hardness of approximation results for the optimization problems.

Given n positive integers $X = [x_1, x_2, \ldots, x_n]$, with $n = 3m$, the 3-partition problem consists in the search for a partition $S_1 \uplus S_2 \uplus \cdots \uplus S_m$ of X such that $|S_j| = 3$ and $m \cdot \sum_{x \in S_j} x = \sum_{x \in X} x$ for all $j \leq m$. Let $B = \frac{1}{m} \sum_{x \in X} x$. The 3-partition problem is NP-Complete even if $x_i \in]B/4; B/2[$ for each $i \leq n$ [9].

We define two real values $1 \leq \phi \leq \Omega$. We set those variables later in this proof. Let $n \geq 7$ be an integer and $X = [x_1, x_2, \ldots, x_n]$ be an instance of 3-partition such that $x_i \in]B/4; B/2[$ for each $i \leq n$. We build an instance $\mathcal{J} =$

$(G, (V_c, V'_c, B_c, E_c), t, \omega, capa, \alpha)$ of 1-DARP-M as follows. As it was previously said, we fix $capa = 2$, $\alpha \in]0.5, 1[$, $t(a) = 0$ for every a and $b_i = e_i = 0$ for every client i.

There are 3 categories of clients :

- the *main* clients: m clients c_j, for $j \in [\![1; m]\!]$, going from u_j to u'_j;
- 2 clients a_1 and a_m, going respectively from v_1 to v'_1 and v_m to v'_m.
- $m * n$ clients d^i_j, for $i \in [\![1; n]\!]$, $j \in [\![1; m]\!]$ going from w^i_j to $w^{i\prime}_j$.

Figure 1 illustrates the graph G and the costs ω. Note that, for each client c_j, the cost of a private ride is 2Ω. For the clients a_1 and a_m, the cost is Ω. For the clients d^i_j for $j \neq 1$, the cost is ϕ and for the clients d^i_1 the cost is $\phi + x_i$. Note also that the graph is planar.

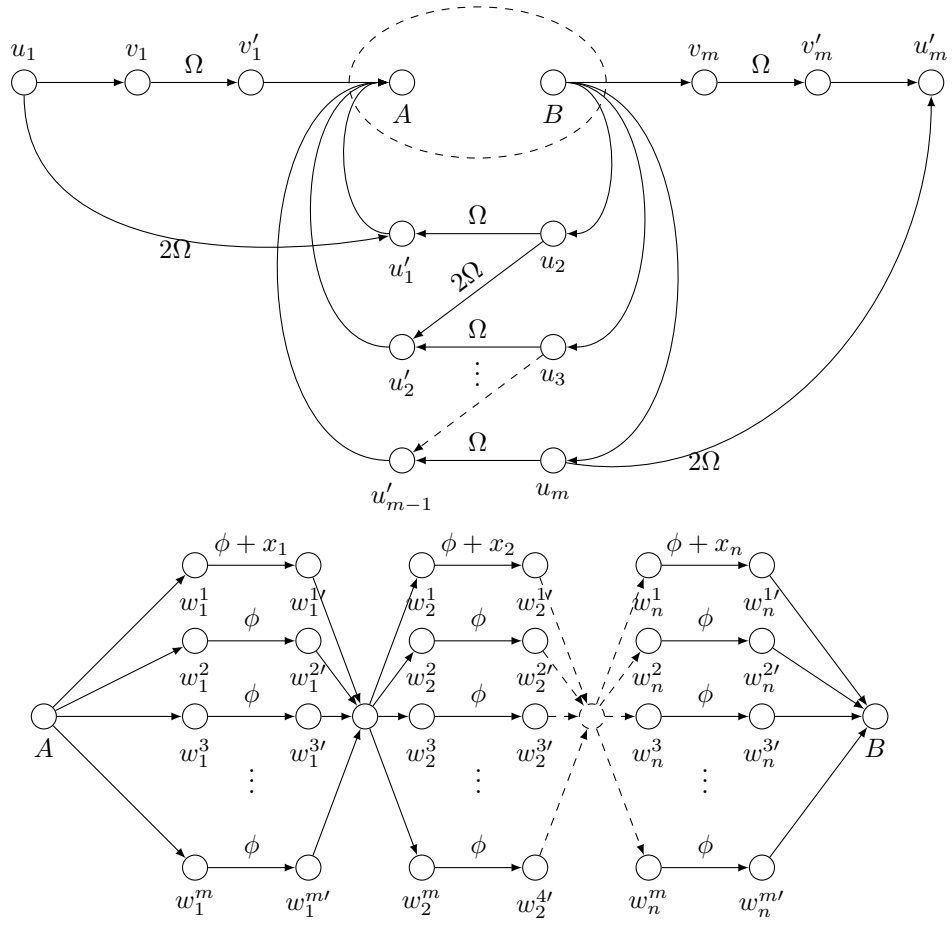

Figure 1: A reduction from 3-partition to max-1-DARP-M. Note that this graph can be drawn planar if we move u_1 and v_1 above u_2 and v'_1 above u'_1.

We set Ω and ϕ as follows:

$$\phi \geq B + 1 \tag{1}$$

$$\Omega = \frac{n\phi + B + \frac{1}{2}}{4\alpha - 2} \tag{2}$$

We start by proving useful properties on ϕ and Ω.

Lemma 2.1. *Ω and ϕ satisfy the following properties :*

$$\frac{2\Omega + n\phi + B + 1}{2} > 2\alpha\Omega \tag{3}$$

$$\frac{2\Omega + n\phi + B}{2} \leq 2\alpha\Omega \tag{4}$$

$$\frac{3}{2}\Omega + \frac{n\phi}{2} > 2\alpha\Omega \tag{5}$$

$$\frac{\Omega}{2} > \alpha \cdot (\phi + B) \tag{6}$$

Proof. Equation (2) proves that

$$\frac{n\phi + B}{4\alpha - 2} \leq \Omega < \frac{n\phi + B + 1}{4\alpha - 2}$$
$$n\phi + B + 2\Omega \leq 4\alpha\Omega < n\phi + B + 1 + 2\Omega$$

and this proves Equations (3) and (4).

We now prove Equation (6). As $\alpha < 1$,

$$4\alpha(4\alpha - 2) < 8 \text{ and } 2\alpha(4\alpha - 2) < 4$$

We recall that $n \geq 7$ and $B > 0$,

$$0 < (n + 1 - 4\alpha(4\alpha - 2))B + (n - 2\alpha(4\alpha - 2)) + \frac{1}{2}$$
$$0 < (n - 2\alpha(4\alpha - 2))(B + 1) + (1 - 2\alpha(4\alpha - 2))B + \frac{1}{2}$$

By Equation (1)

$$0 < (n - 2\alpha(4\alpha - 2))\phi + (1 - 2\alpha(4\alpha - 2))B + \frac{1}{2}$$
$$2\alpha(4\alpha - 2)(\phi + B) < n\phi + B + \frac{1}{2}$$

By Equation (2)

$$\alpha \cdot (\phi + B) < \frac{\Omega}{2}$$

Finally, we can similarly prove Equation (5). As $n \geq 7$, $\alpha < 1$ and $B > 0$,

$$0 < (n - (4\alpha - 3))(B + \frac{1}{2})$$

By Equation (1)

$$0 < n\phi + (3 - 4\alpha)(B + \frac{1}{2})$$
$$0 < (4\alpha - 2)n\phi + (3 - 4\alpha)(n\phi + B + \frac{1}{2})$$

By Equation (2)

$$0 < n\phi + (3 - 4\alpha)\Omega$$
$$2\alpha\Omega < \frac{n\phi}{2} + \frac{3}{2}\Omega$$

□

NP-hardness: from X to $\mathcal{J}$.

Lemma 2.2. *We now assume X is a YES-instance, then, $\mathcal{J}$ is a YES-instance.*

Proof. Let $I_1 \uplus I_2 \uplus \cdots \uplus I_m$ be a partition of $[\![1; n]\!]$ such that $\sum_{i \in I_j} x_i = B$ for all $j \leq m$. The taxi picks up c_1 at u_1 and drives the client a_1 from v_1 to v'_1. Then, for $j \in [\![1; m]\!]$ it drives each client c_j from his origin to his destination such that, when the taxi drives c_j from position A to position B, it drives the three clients d_i^1 for $i \in I_j$ and one client d_i^k for some $k > 1$ and every $i \notin I_j$. Finally, while driving the client c_m from u_m to u'_m, it picks up the client a_m at v_m and delivers them at v'_m. In that case, the client c_j pays $\frac{\Omega}{2} + \frac{n\phi+B}{2} + \frac{\Omega}{2}$. By Equation (4), the client c_j satisfies his cost constraint. Each other client pays exactly half of the price he would have pay alone. As $\alpha > 0.5$, it is a feasible solution for $\mathcal{J}$ satisfying every client. □

NP-hardness: from $\mathcal{J}$ to X. In order to prove the converse of Lemma 2.2, we first prove six intermediates results from Lemma 2.3 to 2.8.

Lemma 2.3. *The three following cases are not possible :*

- *The taxi drives a client d_j^i to a node which is not $w_j^{i\prime}$.*
- *The taxi drives the client a_1 to a node which is not v'_1.*
- *The taxi drives the client a_m to a node which is not v'_m.*

Proof. If the taxi drives the client d_j^i to any node different from $w_j^{i\prime}$, that client must go through at least one arc of cost Ω in order to reach his destination. By equation (6), even if there is another client in the taxi with d_j^i, he has to pay at least $\alpha \cdot (\phi + B)$. However, in a private drive, d_j^i pays ϕ or $\phi + x_i$ depending on whether $j = 1$ or not. As $x_i \in]\frac{B}{4}; \frac{B}{2}[$ for every i, d_j^i cannot satisfy his cost constraint. A similar argument proves the two other cases. □

Lemma 2.4. *A feasible solution must start at u_1.*

Proof. By enumerating every case, we prove that a taxi starting at any other position should pick up a client that could not satisfy the cost constraint.

1. If the taxi starts at w_1^i, the client d_1^i must pay $\phi + x_i$ alone, and then, cannot satisfy his cost constraint. Similarly the taxi cannot start at w_j^i for any $j > 1$.
2. If the taxi starts at v_1, the client a_1 must pay Ω alone, and then, cannot satisfy his cost constraint. Similarly, the taxi cannot start at v_m.
3. If the taxi starts at u_i, for any $i \geq 2$, the client c_i must firstly pay at least Ω to reach position A. Then he pays at least $\frac{n\phi+\Omega}{2}$ from A to his destination. By Equation (5), he cannot satisfy his cost constraint.

□

Lemma 2.5. *All the following cases are not possible :*

1. *the taxi drives through an arc (u_i, u_i')*
2. *the taxi drives c_i to the node u_j, for $j \neq i + 1$*
3. *the taxi drives c_i to the node u_i' but does not deliver it*
4. *the taxi drives c_i to the node u_i' alone, for $i \neq m$*
5. *the taxi never picks up a_1 while driving c_1*
6. *the taxi never picks up a_m while driving c_m*

Proof. If the first statement is true, then, by Lemma 2.3, the taxi can drive only one client c_j or two clients c_j and c_k through that arc. Because a client alone in the taxi would have to pay at least $2\Omega > 2\alpha\Omega$, he would not satisfy his cost constraint. Consequently, there are two clients c_j and c_k. One of them is not c_i. Without loss of generality, we assume that $k \neq i$. That client would have then to pay at least Ω through the arc (u_i, u_i'), then to reach position A, then to pay at least $\frac{n\phi}{2}$ to reach position B and, finally, to pay at least $\frac{\Omega}{2}$ to reach u_k'. By Equation (5), he cannot satisfy his cost constraint.

If the taxi goes to u_j with the client c_i, he pays at least $\frac{\Omega}{2} + \frac{n\phi}{2}$ to go to u_j from u_i. In order to reach his destination, he has to pay at least twice $\frac{\Omega}{2}$, through the arc (u_j, u_{j-1}') and through the arc (u_{i+1}, u_i'). By Equation (5), he cannot satisfy his cost constraint, this proves that the case 2 is not possible. The cases 3 to 6 can be similarly proven. □

Lemma 2.6. *While driving the client c_i from his origin to his destination, the taxi drives exactly n clients d_j^k from their respective origin to their respective destination.*

Proof. By the first statement of Lemma 2.5, in order to reach u_i', the client c_i must go from position A to position B. If the taxi reaches position A with c_i, that client is alone. Indeed, by Lemma 2.3, clients a_1, a_m and d_j^k cannot be driven to position A, and if the taxi drives another client c_j to position A from u_i, this contradicts one of the four first statements of Lemma 2.5. If we assume that strictly less than n clients d_j^k are driven from their respective origin to their respective destination, c_i pays at least $\phi + \frac{(n-1)\phi}{2}$ in order to reach position B from position A. As the taxi goes through at least one arc of cost Ω to reach position A from u_i and to reach u_i' from position B, his ride costs at least $\frac{\Omega}{2} + \phi + \frac{(n-1)\phi}{2} + \frac{\Omega}{2}$. Note that the same occurs if more than n such clients are driven.

By Equation (1)

$$\Omega + \phi + \frac{(n-1)\phi}{2} \geq \Omega + \frac{(n\phi + B + 1)}{2}$$

By Equation (3)

$$\Omega + \phi + \frac{(n-1)\phi}{2} > 2\alpha\Omega$$

Consequently, the taxi must picks up and delivers exactly n clients d_j^k from position A to position B.

□

Lemma 2.7. *In a feasible solution for $\mathcal{J}$, every client is satisfied.*

Proof. By Lemmas 2.4, 2.5, 2.6, if there is a feasible solution, the taxi must start at u_1 and drive each client c_i from his origin to his destination, each one goes from position A to position B and then is driven with exactly n clients d_j^i. As there are m clients c_i and $n*m$ clients d_j^i, every client d_j^i is satisfied. Finally, a_1 and a_m must be picked up and delivered by Lemma 2.5. □

Lemma 2.8. *While driving c_i from his origin to his destination, it must drive exactly three clients $d_{i_1}^1$, $d_{i_2}^1$ and $d_{i_3}^1$ such that $x_{i_1} + x_{i_2} + x_{i_3} \leq B$.*

Proof. By Lemma 2.6, the set containing the clients d_i^1 is partitionned into m subsets, one subset S_j for each client c_j, each client of S_j is driven with c_j. Then the client c_j pays at least $\frac{\Omega}{2} + \frac{n\phi + \sum\limits_{i \in S_j} x_i}{2} + \frac{\Omega}{2}$. If $\sum\limits_{i \in S_j} x_i \geq B + 1$, by Equation (3), the client c_j cannot satisfy his cost constraint.

Moreover, if $|S_j| < 3$, as $n = 3m$, there is a client c_k such that $|S_k| > 3$. As $x_i \in]\frac{B}{4}; \frac{B}{2}[$, $\sum\limits_{i \in S_k} x_i \geq B + 1$ and c_k cannot satisfy his cost constraint. □

Lemma 2.9. *We now assume J is a YES-instance then X is a YES-instance.*

Proof. There exists a taxi satisfying at least one client. By lemma 2.7, every client c_i is satisfied by that taxi. By Lemma 2.8, such a solution proves the existence of a partition $S_1 \uplus S_2 \uplus \cdots \uplus S_m$ such that $\sum_{x \in S_j} x \leq B$. As $\sum_{x \in X} x = mB$, then $\sum_{x \in S_j} x = B$ for all j. Consequently, X is a YES-instance. □

By Lemma 2.2 and 2.9, we can deduce the following theorem.

Theorem 2.1. *1-DARP-M is NP-Complete even if G is planar and if α, capa and W are fixed.*

Corollary 2.1. *max-DARP-M and max-1-DARP-M are NP-Hard and, unless $P = NP$, cannot be approximated in polynomial time to within any variable ratio, even if G is planar and if capa, α and W are fixed.*

Proof. If we assume there exists a polynomial r-approximation algorithm $\mathcal{A}$ for max-DARP-M, where r is a function from $\mathbb{N}$ to $\mathbb{Q}^+$ satisfying $0 < r(p) < 1$. Let $\mathcal{J}$ be an instance of 1-DARP-M and max-DARP-M. If no client can be satisfied, the optimal solution of $\mathcal{J}$ is 0. Thus $\mathcal{A}$ returns a solution of value $r(|\mathcal{J}|) \cdot 0 = 0$. If some client is satisfied, the optimal solution of $\mathcal{J}$ is greater than 1 and $\mathcal{A}$ returns a solution of value greater than $r(|\mathcal{J}|) \cdot 1 > 0$. We can then decide in polynomial time whether at least one client can be satisfied or not. There is then a contradiction with Theorem 2.1. The same result occurs for max-1-DARP-M.

□

3. Directed acyclic graph

In the previous section, we proved hardness results for all the DARP-M problems even if we strongly restrict the instance. The reduction from 3-partition produced an instance in which any taxi must satisfy all the clients by cycling in the graph, driving multiple time through the same roads. Consequently, we study, in this section, the directed acyclic graph case, in order to establish the influence of directed cycles on the complexity of DARP-M.

Obviously, such a case hardly ever occurs on real road networks and finding a polynomial time algorithm does not seem relevant. However, we can arbitrarily order the nodes of the graph. For example, each client i is associated with a time window $[b_i, e_i]$; we can order the origins and the destinations of the clients using the values of b_i for the origins and e_i for the destination. For instance, the taxi could drive from the origin v_i to the origin v_j only if $b_i < b_j$.

In this section, we show some hardness results for the acyclic case, a parameterized algorithm for 1-DARP-M and max-1-DARP-M with respect to *capa* and W, and an parameterized $\frac{1}{\sqrt{n}}$-approximation algorithm for max-DARP-M in *capa* and W.

3.1. Hardness results

3.1.1. Hardness of approximation for max-DARP-M

In this section, we prove a hardness of approximation result for max-DARP-M when G is a DAG and when the parameters α, *capa* and W are fixed.

The reduction. We prove a reduction from the 3-Dimentional Matching problem (3DM). Given three finite disjoint sets X,Y and Z, and a subset S of triplets of $X \times Y \times Z$, (3DM) consists in the search for a maximum size subset M of S such that for every couple (m_1, m_2) of M, m_1 and m_2 are disjoint. (3DM) is NP-Complete and APX-Complete. [12]

From an instance $\mathcal{I} = (X, Y, Z, S)$ of (3DM), we now build an instance $\mathcal{J} = (G, (V_c, V'_c, B_c, E_c), \omega, t, capa, \alpha)$ of max-DARP-M where *capa* and α are fixed such that $\mathcal{I}$ has a feasible solution of size K if and only if $\mathcal{J}$ has a feasible solution with K taxis satisfying $7K$ clients. An example is given in Figure 2. We consider that we remove the time constraint by setting $t(a) = 0$ for every arc $a \in A$ and $b_i = e_i = 0$ for all i and, in that case, $W = 1$.

For each set $s = (x, y, z) \in S$, we define four clients c_s, c_s^x, c_s^y and c_s^z going respectively from v_s, v_s^x, v_s^y and v_s^z to v'_s, $v_s^{x\prime}$, $v_s^{y\prime}$ and $v_s^{z\prime}$. For each element x of X (respectively y of Y and z of Z), we add a client d_x, (respectively d_y and d_z) going from w_x to w'_x (respectively w_y to w'_y and w_z to w'_z).

We add an arc (w_x, w'_x) of cost 1 for each $x \in X$. We add similar arcs for each $y \in Y$ and each $z \in Z$. For each set $s = (x, y, z) \in S$, we add four arcs (v_s, v_s^x), $(v_s^{x\prime}, v_s^y)$, $(v_s^{y\prime}, v_s^z)$ and $(v_s^{z\prime}, v'_s)$ of cost 0. We also add two arcs (v_s^x, w_x) and $(w'_x, v_s^{x\prime})$ of cost 0. We similarly link the nodes of c_s^y and c_s^z to the origin and destination of d_y and d_z.

Finally, we set $capa = 3$ and $\alpha = \frac{1}{3}$. Consequently, there must be 3 clients in the taxi when it drives through an arc of cost non-zero.

Note that G is a DAG.

NP-Hardness. .

Theorem 3.1. *max-DARP-M is NP-Hard and APX-Hard, even if G is a DAG and if α and capa are fixed.*

Proof. As $\alpha = \frac{1}{3}$ and $capa = 3$, there must be 3 clients in the taxi while it is driving through an arc of cost 1. Any taxi must then start at a node v_s for some $s = (x, y, z) \in S$ and end at v'_s, otherwise, there cannot be enough client in the taxi to satisfy any cost constraint. Let P_s be such a taxi. We now show that there is only one possible path P_s going from v_s to v'_s. The taxi must go either to v_s^x or w_x as it cannot reach another node with an arc of cost 0. If the taxi goes to w_x directly, it would have to drive through the arc (w_x, w'_x) of cost at least 1 with at most 2 clients and thus, would not satisfy the cost constraint of the clients c_s and d_x. Consequently, the taxi P_s must pick up the clients c_s, c_s^x and d_x, and goes to w'_x and $v_s^{x\prime}$ to deliver c_s^x and d_x. Similarly, P_s must satisfy c_s^y, d_y, c_s^z and d_z before reaching v'_s.

A feasible solution can contain two taxis P_{s_1} and P_{s_2} if and only if $s_1 \cap s_2 \neq \emptyset$. Indeed, if, for instance, $s_1 \cap s_2 = \{x\}$, the two taxis would have to pick up the

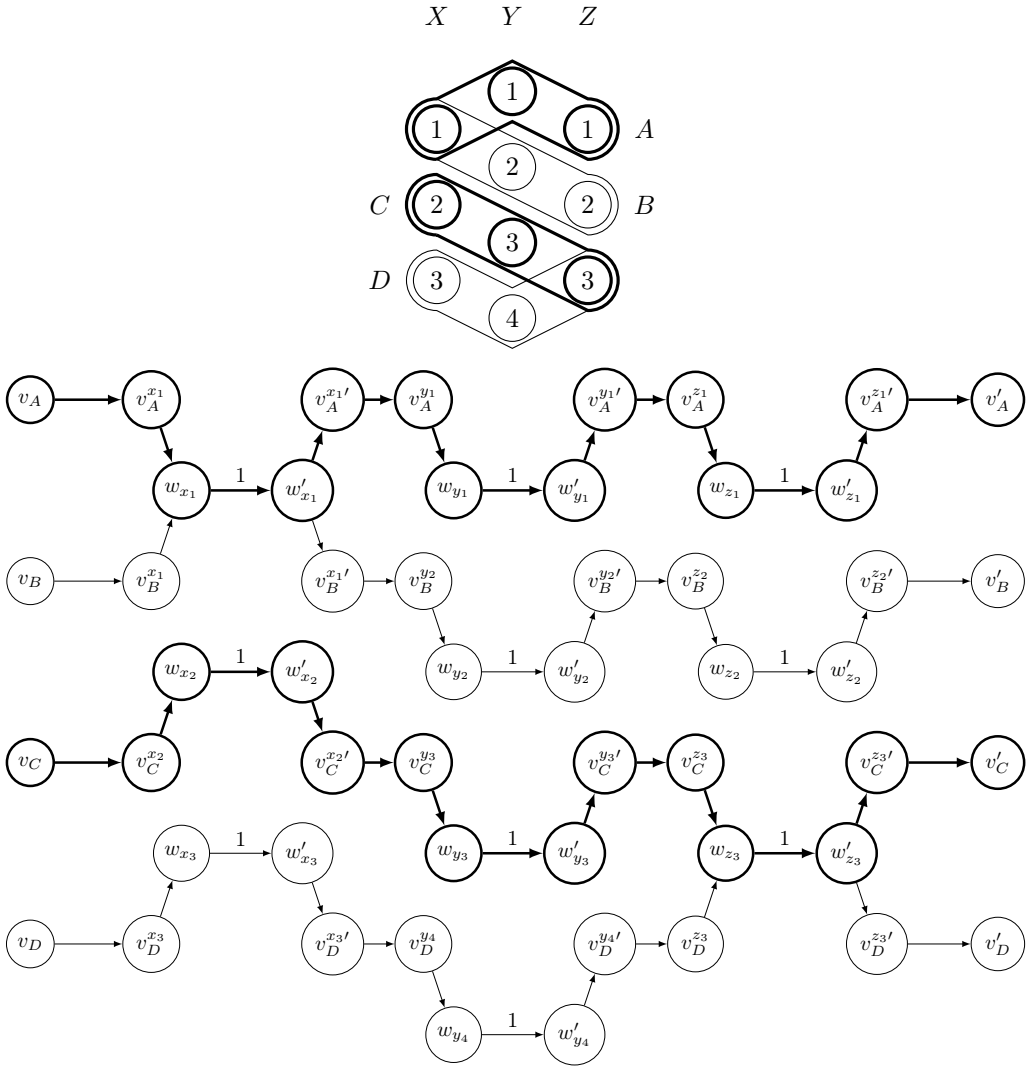

Figure 2: Example of reduction from (3DM). Every unspecified cost is 0. An optimal solution of (3DM) is 2: for example the sets A and C. An optimal solution for max-DARP-M is 14 with two taxis, for example the taxi starting at v_A and ending at v'_A and the taxi from v_C to v'_C.

same client d_x. Consequently, there is a feasible solution for $\mathcal{I}$ of size K if and only if there is a feasible solution of $\mathcal{J}$ with K taxis. As each taxi satisfies 7 clients, the solution satisfies $7K$ clients.

Let $M^* = \{s_1^*, s_2^*, \ldots, s_{K^*}^*\}$ be an optimal solution for $\mathcal{I}$ of size K^*. Then, an optimal solution $\mathcal{P}^* = (P_{s_1^*}^*, P_{s_2^*}^*, \ldots, P_{s_{K^*}^*}^*)$ for $\mathcal{J}$ has K^* taxis and satisfies $7K^*$ clients. If we assume there is a polynomial α-approximation for max-DARP-

M, such an algorithm would return a feasible solution $\mathcal{P} = (P_{s_1}, P_{s_2}, \ldots, P_{s_K})$ satisfying $7K$ clients such that $\alpha 7K^* \leq 7K$. Consequently, we could build in polynomial time a feasible solution $\{s_1, s_2, \ldots, s_K\}$ for $\mathcal{I}$ of size K such that $\alpha K^* \leq K$. Thus, there is a polynomial α-approximation for (3DM). As (3DM) is NP-Complete and APX-Complete and as G is a DAG in the reduction, this concludes the proof.

□

Remark 4. The previous proof can be adapted to any fixed values of *capa* and α such that $capa \geq \frac{1}{\alpha} \geq 3$ by replacing every client d_x, d_y or d_z by $\lceil \frac{1}{\alpha} \rceil - 2$ clients. The nodes w_x and w'_x (similarly w_y and w'_y or w_z and w'_z) would be replaced by two paths respectively containing the $\lceil \frac{1}{\alpha} \rceil - 2$ origins and the $\lceil \frac{1}{\alpha} \rceil - 2$ destinations of those new clients. Every arc of those paths would have a cost 0. And an arc of cost 1 would link the last origin to the first destination.

3.1.2. NP-Hardness when W is not bounded

We give, in this part, a proof that 1-DARP-M is NP-Complete, even if the graph is a DAG and if α and *capa* are fixed.

The reduction. We prove a reduction from the partition problem (PART). Given a finite set of integers $X = \{x_1, x_2, \ldots, x_n\}$, is it possible to part $[\![1; n]\!]$ into two parts $I \uplus J$ such that $\sum_{i \in I} x_i = \sum_{i \in J} x_i$. (PART) is weakly NP-Complete [9].

From an instance $\mathcal{I} = (X)$ of (PART), we now build an instance $\mathcal{J} = (G, (V_c, V'_c, B_c, E_c), \omega, t, capa, \alpha)$ of 1-DARP-M where G is a DAG and where *capa* and α are fixed. Let $B = \sum_{x \in X} x/2$. We fix $capa = 2$ and $\alpha = \frac{1}{2}$. We define three main clients c_1, c_2 and c_3 going respectively from v_1 to v'_1, v_2 to v'_2 and v_3 to v'_3. We also define $2n$ clients $d^1_1, d^1_2, \ldots, d^1_n$ and $d^2_1, d^2_2, \ldots, d^2_n$. The client d^j_i goes from w^j_i to $w^{j\prime}_i$.

The graph G, the costs ω and the times t are illustrated on Figure 3.

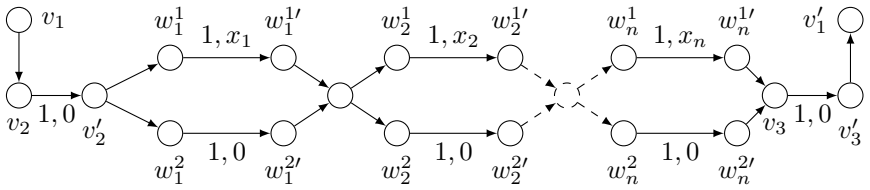

Figure 3: Example of reduction from (PART) to 1-DARP-M. On each arc a, we write the values $\omega(a)$ and $t(a)$ in that order. If no number is given, the two values are 0.

The time window of c_2 is $[0, 0]$. The time window of c_3 is $[B, B]$. For all the other clients, the time window is $[0, B]$.

Theorem 3.2. *1-DARP-M is weakly NP-Complete even if G is a DAG and if α and capa are fixed.*

Proof. As $\alpha = \frac{1}{2}$ and $capa = 2$, there must be 2 clients in the taxi while it is driving through an arc of cost 1. Consequently, the taxi must pick up the client c_1. In order to go to v'_1, the taxi must drive through (v_2, v'_2) and, then, must also pick up c_2. Similarly it must pick up c_3 and, for each $i \in [\![1; n]\!]$, either d_i^1 or d_i^2. Let I be the subset of $[\![1; n]\!]$ such that $i \in I$ if d_i^1 is picked up.

Due to the time window of c_2, the taxi must arrive at v_2 at time 0 otherwise the time constraint cannot be satisfied. Similarly it must arrive at v_3 at time B. The moment when the taxi reaches v_3 is $\sum_{i \in I} t(w_i^1, w_i^{1\prime}) + \sum_{i \notin I} t(w_i^2, w_i^{2\prime}) = \sum_{i \in I} x_i$.

Thus, there exists a feasible taxi if and only if $\sum_{i \in I} x_i = B = \sum_{i \notin I} x_i$. This concludes the proof. □

From Theorem 3.2, we can deduce the following results.

Corollary 3.1. *max-DARP-M and max-1-DARP-M are weakly NP-Hard and, unless W is polynomialy bounded or P = NP, cannot be approximated in polynomial time to within any variable ratio, even if G is a DAG and if capa and α are fixed.*

3.1.3. NP-Hardness and Parameterized Hardness with respect to the number of satisfied clients

If we search for a taxi satisfying exactly S clients, we can enumerate every subset of S clients and check if a taxi only satisfying that subset exists, and this give an XP algorithm for max-1-DARP-M$_=$ and max-DARP-M$_=$ with respect to S. We prove in this subsection that a better algorithm is hardly to exist as max-1-DARP-M$_=$ and max-DARP-M$_=$ are W[1]-hard in S, capa, α and W.

We also demonstrate that the three problems, max-1-DARP-M, max-DARP-M and 1-DARP-M are NP-Complete and cannot be approximated in polynomial time even if W is fixed and that 1-DARP-M is W[1]-hard in *capa*, α and W.

The reduction. We describe an FPT-Reduction from the partitioned clique problem. Given an undirected graph $G = (V = V_1 \uplus V_2 \uplus \cdots \uplus V_k, E)$ where V is partitioned into k independent sets, the partitioned clique problem consists in the search for a clique of size k. Any such clique contains exactly one node in each part V_i. This problem is NP-Complete and is W[1]-hard with respect to k [16].

From a parameterized instance G of the partitioned clique problem, we build an instance $\mathcal{J}$ of max-1-DARP-M such that the graph is a DAG. We consider that we remove the time constraint by setting $t(a) = 0$ for every arc $a \in A$ and $b_i = e_i = 0$ for all i and, in that case, $W = 1$. While describing $\mathcal{J}$, we explain how the reduction works on a simple example, given in Figure 4.

Our goal is to create a directed acyclic graph $H = (W \cup X, A)$.

W contains two nodes w_u^v and $w_u^{v\prime}$ for each edge (u, v) of G. It is partitioned into k layers and each layer is partitioned into $k - 1$ sublayers. We write W_i for layer i. The sublayers of layer i are numbered from 1 to k except i and we write W_i^j for sublayer j of layer i. For each edge $\{u, v\}$ in E such that $u \in V_i$

and $v \in V_j$ and $i < j$, we add a client c_u^v with the origin $w_u^v \in W_i^j$ and the destination $w_v^u \in W_j^i$.

Each sublayer is a stable set. A node $w_{u_1}^{v_1} \in W_i^j$ is linked to a node $w_{u_2}^{v_2}$ of the next sublayers of W_i (W_i^{j+1} if $j \neq i-1$, W_i^{i+1} otherwise) if $u_1 = u_2$. Note that this common node is necessarily in V_i. Note also that each layer is an acyclic graph.

X is a set of $k-1$ paths. For each $i \in [\![1; k-1]\!]$, we add to H a path $(x_i^1, x_i^2, \ldots, x_i^{d_i}, x_i^{1\prime}, x_i^{2\prime}, \ldots, x_i^{d_i\prime})$ where $d_i = \frac{(k-1)\cdot k}{2} - i \cdot (k-i)$. We also add d_i clients $D_i = \{d_i^1, d_i^2, \ldots, d_i^{d_i}\}$. Each client d_i^l goes from x_i^l to $x_i^{l\prime}$. Let X_i be the set of origins of those clients and X_i' be the set of destinations. All the nodes of the last sublayer of each layer W_i, for $i \neq k$, are linked to x_i^1, and, similarly, $x_i^{d_i\prime}$ is linked to all the nodes of the first layer of W_{i+1}.

We can easily see that H does not contain a circuit as each layer W_i is acyclic and as it is only connected to the following layer with the path $X_i \cup X_i'$.

The cost of every arc $(x_i^{d_i}, x_i^{1\prime})$ is 1. For every other arc, the cost is null. Finally, we set $S = \frac{(k-1)\cdot k}{2} + \sum_{p=1}^{k-1} d_p$, $capa = \frac{(k-1)\cdot k}{2}$ and $\alpha = \frac{1}{capa}$. Any taxi driving through an arc of cost non-zero must contains *capa* clients otherwise the cost constraint cannot be satisfied.

We only give, in this section, the key idea of the reduction. The formal proof is given in Appendix A.

Key idea of the reduction. Firstly, we assume we search for a taxi satisfying S clients. That taxi must satisfy exactly one client per sublayer and all the clients of D_i. The way H is built make the taxi do a choice: while traversing the $k-1$ sublayers of W_i, the taxi must choose a node v_i of V_i because each such node is associated with a connected component of the layer W_i. For example, the component of 4 in Figure 4, represented with dotted nodes, is $\{w_4^1, w_4^2, w_4^6\}$. While driving through a node containing 4 in W_2^1, it is not possible anymore to go to a node containing 3 or 5 in W_2^1 or W_2^3. Thus, it is not possible to drive to a node containing 3 or 5 in any other set, because this would mean that a client is picked up and not delivered. Thus, the taxi must choose one node v_i per set V_i. Consequently, the taxi build a set $C = \{v_i \in V_i, i \in [\![1; n]\!]\}$. While crossing the $k-1$ sublayers of W_i, it must also choose $k-1$ edges of G, one incident edge to v_i for each of the $k-1$ sets corresponding to the index i. Each of those edges selects a node of a set V_j, for $j \neq i$. This node is v_j for each j if and only if the set C is a clique. For example, a valid taxi corresponding to the clique $\{2, 5, 7\}$ is drawn in Figure 4 with bold arcs and nodes. If, on the other hand, the taxi chooses to start with the nodes w_2^4 and continue with w_2^7, then it must continue to $w_2^{4\prime}$ in order to satisfy the client c_2^4. It is then not possible to satisfy three clients: either it drives c_2^7 to his destination or forget this client and drives c_4^6. This is due to the fact that the taxi must choose a node of W_2^3 containing 4 but the edge $\{4, 7\}$ does not exists in G thus, the taxi must choose another node.

Secondly, in order to go from one layer W_i to the following layer, a taxi must

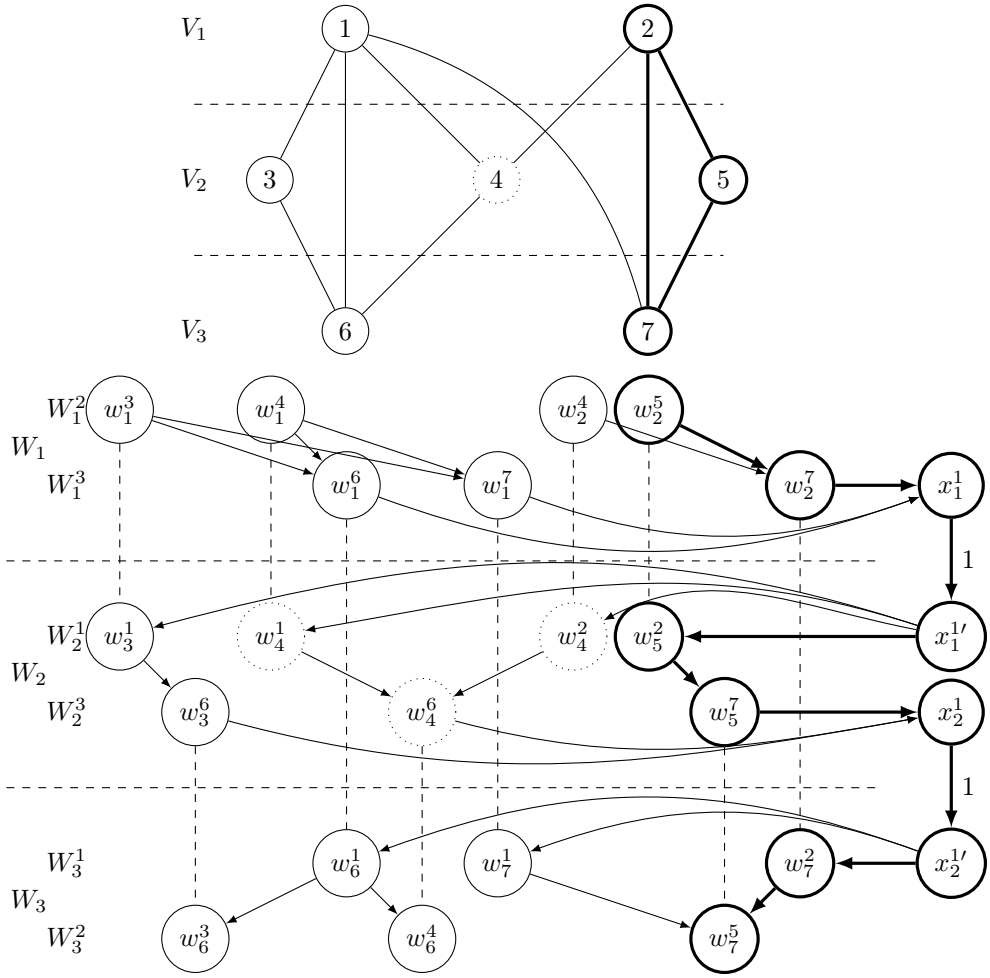

Figure 4: Example of reduction from the Partitioned Clique problem. There are 3 stable sets in G: V_1, V_2 and V_3; and 6 sublayers in H: W_1^2, W_1^3, W_2^1, W_2^3, W_3^1 and W_3^2. Each layer is separated from the other with an horizontal dashed line. The two paths of X are on the right. Note that there are dashed lines joining the origin and the destination of each client c_u^v for information. Those lines are not edges or arcs of the graph. Every cost which is not specified is 0.

go through the path $X_i \cup X_i'$ and thus it must go through an arc of cost 1. When this happens, as $\alpha = \frac{1}{capa}$, the taxi must be full: it must picks up every clients of D_i. For example, in Figure 4, $capa = 3$. Any taxi must picks up the clients d_1^1 and d_2^1 in order to go through $(x_1^1, x_1^{1\prime})$ and $(x_2^1, x_2^{1\prime})$. Thus there cannot be more than one taxi in a feasible solution otherwise the two taxis must satisfy some same clients and this is not allowed. Consequently, a feasible solution satisfying S clients can contain only one taxi.

Finally, a taxi must satisfy S clients. Indeed, it must pick up a client from

W_1^k and delivers it at W_k^1. Otherwise there is no way for it to pick up *capa* clients before going though an arc of cost 1. Thus it must satisfy exactly one client per set W_i^j and all the clients of every set D_i.

Theorem 3.3. *Even if G is a DAG,*

- *1-DARP-M is W[1]-hard with respect to capa, α and W;*
- *max-DARP-M$_=$ and max-1-DARP-M$_=$ are W[1]-hard with respect to S, capa, α and W;*
- *1-DARP-M is NP-Complete and max-DARP-M, max-1-DARP-M are NP-Hard and cannot be approximated in polynomial time to within any variable ratio even if W is fixed.*

The proof of this theorem is given in Appendix A.

3.2. Parameterized algorithms

In this section, we first give an algorithm to solve max-1-DARP-M in a DAG in pseudopolynomial time when *capa* is fixed. We then deduce a $\sqrt{n}$-approximation algorithm for max-DARP-M in a DAG in pseudopolynomial time when *capa* is fixed.

3.2.1. A parameterized algorithm for max-1-DARP-M

We consider an instance $\mathcal{I} = (G, (V_c, V_c', B_c, E_c), t, \omega, capa, \alpha)$ with n clients and where G is a DAG. We assume that, in G, there is a path from any origin $v_i \in V_c$ to the corresponding destination $v_i' \in V_c'$: there is no path from v_i' to v_i. We finally consider that there is no intermediate point: $V = V_c \cup V_c'$. Every arc (u, v) corresponds to a shortest path from u to v in the road network if such a path exists.

Definition 2. We now define an *auxiliary graph* $\mathcal{S}(\mathcal{I})$ in which each node is associated with a state corresponding to the taxi leaving a node $u \in V$ at time t with a set S of at most *capa* clients and such that κ clients already entered the taxi (including the clients who have left the taxi and the clients who have not); we write that state $w(u, t, S, \kappa)$, $u \in V$, $t \in [b_i, e_i]$ if $u = v_i$ or if $u = v_i'$, $S \subset [\![1; n]\!]$, $|S| \leq capa$, $\kappa \in [\![1; n]\!]$. An arc is a transition between two states: we add an arc $(w(u_1, t_1, S_1, \kappa_1), w(u_2, t_2, S_2, \kappa_2))$ in $\mathcal{S}(\mathcal{I})$ if and only if all the following three properties are true:

1. $S_1 \neq \emptyset$
2. there is a path from u_1 to u_2 in G;
3. $t_2 - t_1 = t(u_1, u_2)$
4. - either u_2 is the origin v_i of client i, $S_2 = S_1 \uplus \{i\}$ and $\kappa_2 = \kappa_1 + 1$
   - or u_2 is the destination v_i' of client i, $S_1 = S_2 \uplus \{i\}$ and $\kappa_2 = \kappa_1$

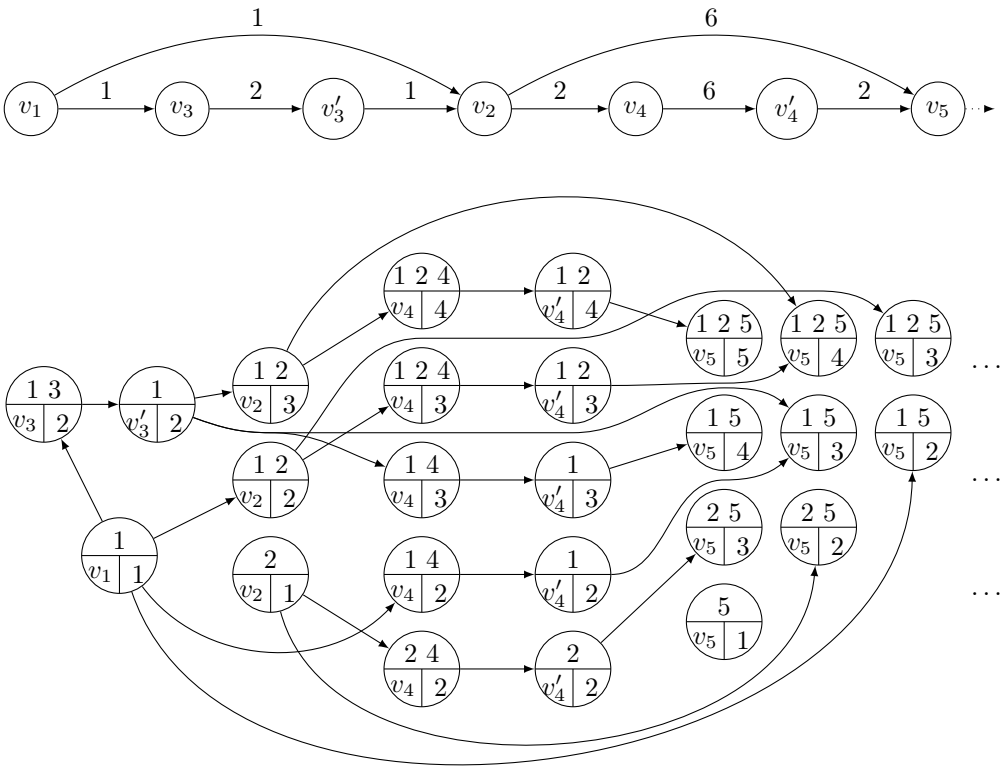

Figure 5: Example of transformation from $\mathcal{I}$ to the auxiliary graph $\mathcal{S}(\mathcal{I})$. For readability, we do not consider, in this figure, the time windows, and some states like $w(v_3, t, 3, 1)$ or $w(v_4, t, 4, 1)$ are missing. The weight of each arc a on the upper graph is the cost $\omega(a)$. For each state $w(u, t, S, \kappa)$ contains u, S and κ respectively on the lower left part, the upper part and the lower right part of the node. The time t is not given.

An example is given in Figure 5.

As the existence of an arc between two states $w(u_1, t_1, S_1, \kappa_1)$ and $w(u_2, t_2, S_2, \kappa_2)$ implies that there is a path from u_1 to u_2 in G and as G is a DAG, we can deduce the following property.

Property 1. $\mathcal{S}(\mathcal{I})$ is a DAG.

We now introduce Algorithm 1, which solves max-1-DARP-M using the auxiliary graph $\mathcal{S}(\mathcal{I})$. We then prove the polynomial time complexity and the correctness of the algorithm.

For each node $w = (v, t, S, \kappa) \in \mathcal{S}(\mathcal{I})$, we define a set $\mathcal{P}(w)$ of mappings associating to each client of S a non negative real: each mapping represents a

possible taxi driving the clients of S through v and $p(i)$ is the cost that the client i has already paid from its origin to v in that taxi.

Definition 3. Let $w = (v, t, S, \kappa) \in \mathcal{S}(\mathcal{I})$, and p and p' be two mappings of $\mathcal{P}(w)$. We say p *dominates* p' if, for every client i in S, $p(i) \leq p'(i)$. We write $p \preceq p'$.

We use Algorithm 1 to compute all the sets of mappings of the auxilliary graph and to deduce a feasible solution for $\mathcal{I}$.

Each set $\mathcal{P}(w)$ is built recursively using the sets of mappings of all predecessors of w. In order to simplify Algorithm 1, we define, for each arc $(w_1, w_2) \in \mathcal{S}(\mathcal{I})$, a set of intermediate mappings $\mathcal{P}(w_1, w_2)$ which can be seen as the subset of $\mathcal{P}(w_2)$ built from the state w_1 with useful additional information. This set is built with the SUBMAP function, described in Algorithm 2. In addition, a BUILD function is given in Algorithm 3 to build a solution. Table 2 illustrates some iterations of the algorithm on the example given in Figure 5.

Algorithm 1 Main algorithm

Require: an instance $\mathcal{I} = (G, (V_c, V_c', B_c, E_c), t, \omega, capa, \alpha)$ of max-1-DARP-M
Ensure: an optimal solution for $\mathcal{I}$

Build the auxiliary graph $\mathcal{S}(\mathcal{I})$
For each client $i \in [\![1; n]\!]$ and $t \in [b_i, e_i]$ **Do**
    $p_i \leftarrow$ a mapping associating 0 to the client i
    $\mathcal{P}(w(v_i, t, \{i\}, 1)) \leftarrow \{p_i\}$
    $pred(p_i) \leftarrow null$
$L \leftarrow$ a topological ordering of $\mathcal{S}(\mathcal{I}) \backslash \{w(v_i, \{i\}, 1) | i \in [\![1; n]\!]\}$
For each $w = w(v, t, S, \kappa) \in L$ **Do**
    $\mathcal{P}(w) \leftarrow \emptyset$
    For each predecessor $w^- = w(v^-, t^-, S^-, \kappa^-)$ of w **Do**
        $\mathcal{P}(w^-, w) \leftarrow$ SUBMAP$(\mathcal{I}, \mathcal{S}(\mathcal{I}), (w^-, w), \mathcal{P}(w^-))$
        For each $(p, w^-, p^-) \in \mathcal{P}(w^-, w)$ **Do**
            If For all $p' \in \mathcal{P}(w)$, $p' \not\preceq p$ **Then**
                remove from $\mathcal{P}(w)$ every mapping p' such that $p \preceq p'$
                add p to $\mathcal{P}(w)$
                $pred(p) \leftarrow (w^-, p^-)$
$T \leftarrow \{w = w(v_i', t, \emptyset, \kappa) | i, \kappa \in [\![1; n]\!], \mathcal{P}(w) \neq \emptyset\}$
If $T = \emptyset$ **Then Return** no solution.
$\tau \leftarrow \text{argmax}\{\kappa | w(v_i', t, \emptyset, \kappa) \in T\}$
$p \leftarrow$ a mapping of $\mathcal{P}(\tau)$
Return BUILD$(\mathcal{I}, \mathcal{S}(\mathcal{I}), pred, \tau, p)$

Due to the length of this part, we put the proof of correctness of Algorithm 1 in Appendix B.

The end of this part is dedicated to proving that, for every node $w = (v, t, S, \kappa) \in \mathcal{S}(\mathcal{I})$, the size of $\mathcal{P}(w)$ is polynomial if *capa* is fixed and if W is

Algorithm 2 SUBMAP Function

Require: an instance $\mathcal{I} = (G, (V_c, V_c', B_c, E_c), t, \omega, capa, \alpha)$ of max-1-DARP-M, the auxiliary graph $\mathcal{S}(\mathcal{I})$, an arc $(w_1 = w(u_1, t_1, S_1, \kappa_1), w_2 = w(u_2, t_2, S_2, \kappa_2)) \in \mathcal{S}(\mathcal{I})$, a set $\mathcal{P}(w_1)$ of mappings from S_1 to $\mathbb{R}^+$

Ensure: a set $\mathcal{P}(w_1, w_2)$ of mappings from S_2 to $\mathbb{R}^+$

1: **function** SUBMAP($\mathcal{I}, \mathcal{S}(\mathcal{I}), (w_1, w_2), \mathcal{P}(w_1)$)
2: $\mathcal{P}(w_1, w_2) \leftarrow \emptyset$
3: **For each** mapping $p_1 \in \mathcal{P}(w_1)$ **Do**
4: Initialize a mapping p_2 of $S_2 \to \mathbb{R}^+$
5: **For each** client $i \in S_1$ **Do**
6: **If** $p_1(i) + \frac{\omega(u_1,u_2)}{|S_1|} \leq \alpha \cdot \omega(v_i, v_i')$ **Then**
7: **If** $i \in S_2$ **Then** $p_2(i) \leftarrow p_1(i) + \frac{\omega(u_1,u_2)}{|S_1|}$
8: **Else**
9: **Continue** loop **For** at Line 3
10: **If** S_2 contains a client i not in S_1 **Then**, $p_2(i) = 0$
11: Add (p_2, w_1, p_1) to $\mathcal{P}(w_1, w_2)$
12: **Return** $\mathcal{P}(w_1, w_2)$

Algorithm 3 BUILD Function : Build a partial solution from a selected node.

Require: an instance $\mathcal{I} = (G, (V_c, V_c', B_c, E_c), t, \omega, capa, \alpha)$ of max-1-DARP-M, the auxiliary graph $\mathcal{S}(\mathcal{I})$, a predecessor function *pred*,a node $w = w(u, t, S, \kappa) \in \mathcal{S}(\mathcal{I})$ and a mapping p of $\mathcal{P}(w)$

Ensure: an path ending at u in G

1: **function** BUILD($\mathcal{I}, \mathcal{S}(\mathcal{I}), pred, w, p$)
2: $P \leftarrow \{u\}$
3: **If** $pred(p) \neq null$ **Then**
4: $(w^-, p^-) \leftarrow pred(p)$
5: $P \leftarrow P \cup$ BUILD($\mathcal{I}, \mathcal{S}(\mathcal{I}), pred, w^-, p^-$)
6: **Return** P

Table 2: Example of iterations of the For loop at Line 7 of Algorithm 1 on the instance given in Figure 5. We assume that every duration $t(a)$ is 0 and that every time window contains 0. The first part ends with a state containing two pareto optimum mappings whereas the second part ends with an iteration where a mapping is not inserted in $\mathcal{P}(w)$ because it is dominated by an existing mapping.-

w	w^-	$\mathcal{P}(w^-, w)$	$\mathcal{P}(w)$
$w(\{1,3\},0,v_3,2)$	$w(\{1\},0,v_1,1)$	$1\to 1, 3\to 0$	$1\to 1, 3\to 0$
$w(\{1\},0,v'_3,2)$	$w(\{1,3\},0,v_3,2)$	$1\to 2$	$1\to 2$
$w(\{1,2\},0,v_2,3)$	$w(\{1\},0,v'_3,2)$	$1\to 3, 2\to 0$	$1\to 3, 2\to 0$
$w(\{1,2\},0,v_2,2)$	$w(\{1\},0,v_1,1)$	$1\to 1, 2\to 0$	$1\to 1, 2\to 0$
$w(\{1,2,4\},0,v_4,3)$	$w(\{1,2\},0,v_2,2)$	$1\to 2, 2\to 1, 4\to 0$	$1\to 2, 2\to 1, 4\to 0$
$w(\{1,2\},0,v'_4,3)$	$w(\{1,2,4\},0,v_4,3)$	$1\to 4, 2\to 3$	$1\to 4, 2\to 3$
$w(\{1,2,5\},0,v_5,4)$	$w(\{1,2\},0,v_2,3)$	$1\to 6, 2\to 3, 5\to 0$	$1\to 6, 2\to 3, 5\to 0$
$w(\{1,2,5\},0,v_5,4)$	$w(\{1,2\},0,v'_4,3)$	$1\to 5, 2\to 4, 5\to 0$	$1\to 6, 2\to 3, 5\to 0$ $1\to 5, 2\to 4, 5\to 0$
$w(\{1,4\},0,v_4,2)$	$w(\{1\},0,v_1,1)$	$1\to 3, 4\to 0$	$1\to 3, 4\to 0$
$w(\{1\},0,v'_4,2)$	$w(\{1,4\},0,v_4,2)$	$1\to 6$	$1\to 6$
$w(\{1,5\},0,v_5,3)$	$w(\{1\},0,v'_4,2)$	$1\to 8, 5\to 0$	$1\to 8, 5\to 0$
$w(\{1,5\},0,v_5,3)$	$w(\{1\},0,v'_3,2)$	$1\to 9, 5\to 0$	$1\to 8, 5\to 0$

bounded by polynomial in $|\mathcal{I}|$, and deduce that Algorithm 1 is XP with respect to *capa* when W is polynomially bounded.

Definition 4. Let $w = (v, t, S, \kappa) \in \mathcal{S}(\mathcal{I})$, for every subset $I \subset S$, we define $p_{|I}$ as the subvector of p restricted to every client of I. The set $\mathcal{P}(w, I)$ is the subset of pareto optimal vectors of $\{p_{|I}, p \in \mathcal{P}(w)\}$, *i.e.* for any two distinct mappings p and p' of $\mathcal{P}(w, I)$, $p \not\preceq p'$ and $p' \not\preceq p$.

Note that $\mathcal{P}(w, S) = \mathcal{P}(w)$ due to Lines 12 and 13 of Algorithm 1.

We want to prove the following properties:

Property 2. Let $w = (u, t, I \uplus J, \kappa) \in \mathcal{S}(\mathcal{I})$, with $|I| \geq 2$, and if u is the origin of a client in I, then $|P(w, I)| \leq n^{(capa-1)\cdot(|I|-2)} \cdot W^{|I|-2}$.

Property 3. Let $w = (u, t, I \uplus J, \kappa) \in \mathcal{S}(\mathcal{I})$, with $|I| \geq 1$, and if u is not the origin of any client in I then $|P(w, I)| \leq n^{(capa-1)\cdot(|I|-1)} \cdot W^{|I|-1}$.

We prove the two properties by induction on the size of $|I|$. Each property alternatively proves the other one. The following lemmas proves that Property 3 is true when $|I| = 1$, that if Property 3 is true when $|I| \leq s$ for some constant s, then Property 2 is true when $|I| = s+1$, and, finally, when Property 2 is true when $|I| \leq s$ for some constant s, then Property 3 is true when $|I| = s$.

Lemma 3.1. *Property 3 is true when* $|I| = 1$.

Proof. Let $I = \{i\}$. In that case, every mapping of $|P(w, I)|$ maps some positive real to the client i. There can be only only one pareto optimal mapping: the one associating the smallest real to i. □

Lemma 3.2. *If, for some constant $s \leq capa$, Property 3 is true when $1 \leq |I| \leq s$, then Property 2 is true when $|I| = s + 1$.*

Proof. Let $w = w(u, t, I \uplus J, \kappa)$, we assume $I = \{i_1, i_2, \ldots, i_{s+1}\}$. We assume that u is the origin of some client in I and, in any topological ordering of G, v_{i_j} is *before* v_{i_k} if and only if $j \leq k$. Then $u = v_{i_{s+1}}$ and for any mapping p of $\mathcal{P}(w)$, $p(i_{s+1}) = 0$. If we consider two mappings p and p' of $\mathcal{P}(w)$, $p_{|I\setminus i_{s+1}} \preceq p'_{|I\setminus i_{s+1}}$ if and only if $p_{|I} \preceq p'_{|I}$. Thus, $|\mathcal{P}(w, I)| = |\mathcal{P}(w, I\setminus i_{s+1})|$. By the hypothesis, $|P(w, I\setminus i_{s+1})| \leq n^{(capa-1)\cdot(s-1)} \cdot W^{s-1} = n^{(capa-1)\cdot(|I|-2)} \cdot W^{|I|-2}$. Thus Property 2 is proved for I. □

In order to prove the last lemma, we first prove an intermediate result.

Definition 5. Using the *pred* array, we can define a *precedence tree* of mappings in which p^- is linked to p if $pred(p) = (w^-, p^-)$ for some state w^-. Let $w_1 = w(u_1, t_1, I \uplus J_1, \kappa_1)$ and $w_2 = w(u_2, t_2, I \uplus J_2, \kappa_2)$ such that there is a path from w_1 to w_2 in $\mathcal{S}(\mathcal{I})$. We say that a mapping p_1 of $\mathcal{P}(w_1)$ *generates* p_2 of $\mathcal{P}(w_2)$ if there is a path of mappings from (w_1, p_1) to (w_2, p_2) in the precedence tree.

Lemma 3.3. *Let $w_1 = w(u_1, t_1, I \uplus J_1, \kappa_1)$ and $w_2 = w(u_2, t_2, I \uplus J_2, \kappa_2)$ such that there is a path from w_1 to w_2 in $\mathcal{S}(\mathcal{I})$. Let q_1 be a mapping of $\mathcal{P}(w_1, I)$ and p_1 and p'_1 be two mappings of $\mathcal{P}(w_1)$ such that $p_{1|I} = q_1$ and $p'_{1|I} = q_1$. Let p_2 and p'_2 be two mappings of $\mathcal{P}(w_2)$ such that p_1 generates p_2 and p'_1 generates p'_2. Let finally q_2 and q'_2 be $p_{2|I}$ and $p'_{2|I}$. Then $q_2 \preceq q'_2$ or $q'_2 \preceq q_2$.*

Proof. We first assume that w_1 is a predecessor of w_2. Note that, due to Algorithm 2 at Line 7, $p_2(i) - p_1(i) = p'_2(i) - p'_1(i)$ equals $\frac{\omega(u_1,u_2)}{|I\uplus J_1|}$ for every clients $i \in I \subset S_1$. Since $p_{1|I} = p'_{1|I} = q_1$, $p_1(i) = p'_1(i) = q_1(i)$ and then $q_2(i) = p_2(i) = p'_2(i) = q'_2(i)$.

We can similarly show the same property if w_1 is an ancestor of w_2 instead of just a predecessor. In that case, there are intermediates mappings between p_1 and p_2, and between p'_1 and p'_2. There is a path $(w_1 = x_1, x_2, \ldots, w_2 = x_l)$ such that p_1 generates a mapping p_{x_2} of x_2 which generates a mapping p_{x_3} of x_3, ... There is also a path $(w_1 = x'_1, x'_2, \ldots, w_2 = x'_k)$ such that p'_1 generates a mapping p'_{x_2} of x'_2 which generates a mapping p'_{x_3} of x'_3, ... Let $x_i = w(u_i^x, t_i^x, S_i^x, \kappa_i^x)$ and $x'_i = w(u_i^{x'}, t_i^{x'}, S_i^{x'}, \kappa_i^{x'})$. Due to Algorithm 2 at Line 7, $p_2(i) - p_1(i) = \sum_{i=1}^{l-1} \frac{\omega(u_{i+1}^x - u_i^x)}{|S_i^x|}$ and $p'_2(i) - p'_1(i) = \sum_{i=1}^{l-1} \frac{\omega(u_{i+1}^{x'} - u_i^{x'})}{|S_i^{x'}|}$. Since $p_{1|I} = p'_{1|I} = q_1$, for all $i \in I$, $p_1(i) = p'_1(i) = q_1(i)$ and then depending whether $\sum_{i=1}^{l-1} \frac{\omega(u_{i+1}^x - u_i^x)}{|S_i^x|} \geq \sum_{i=1}^{l-1} \frac{\omega(u_{i+1}^{x'} - u_i^{x'})}{|S_i^x|}$ or not, either for all $i \in I$ $q_2(i) \geq q'_2(i)$ or for all $i \in I$ $q_2(i) < q'_2(i)$. □

Lemma 3.4. *If, for some constant $s \leq capa$, Property 2 is true when $2 \leq |I| \leq s$, then Property 3 is true when $|I| = s$.*

Proof. Let $w = w(u, t, I \uplus J, \kappa)$, we assume $I = \{i_1, i_2, \ldots, i_s\}$ and u is not the origin of any client in I.

Let

$$A = \bigcup_{\substack{t' \in [e_{i_s}, b_{i_s}] \\ J' \subset [\![1;n]\!] \setminus I \\ |J'| \leq capa - |I| \\ \kappa' \in [\![1;n]\!]}} \mathcal{P}(w(v_{i_s}, t', I \uplus J', \kappa'), I)$$

. We want to prove that $|\mathcal{P}(w, I)| \leq |A|$.

We define a function *anc* associating to each mapping of $\mathcal{P}(w, I)$ a mapping of A. Let q be a mapping of $\mathcal{P}(w, I)$. There exists a mapping p of $\mathcal{P}(w)$ such that $q = p_{|I}$. Thus there exists an ancestor $w_s = w(v_{i_s}, t', I \uplus J', \kappa')$ of w in $\mathcal{S}(\mathcal{I})$ and a mapping p_s of $\mathcal{P}(w_s)$ such that p_s generates p. Let $anc(q) = p_{s|I}$.

If $|\mathcal{P}(w, I)| > |A|$, there exist two mappings q_1 and q_2 in $\mathcal{P}(w, I)$ such that $anc(q_1) = anc(q_2)$. By Lemma 3.3, $q_1 \prec q_2$ or $q_2 \prec q_1$. There is a contradiction because every mapping of $\mathcal{P}(w, I)$ is Pareto optimal. Consequently $|\mathcal{P}(w, I)| \leq |A|$.

$$\begin{aligned} |\mathcal{P}(w, I)| \leq |A| &\leq \sum_{\substack{t' \in [e_{i_s}, b_{i_s}] \\ J' \subset [\![1;n]\!] \setminus I \\ |J'| \leq capa - |I| \\ \kappa' \in [\![1;n]\!]}} |\mathcal{P}(w(v_{i_s}, t', I \uplus J', \kappa'), I)| \\ &\leq (W \cdot n^{capa-2} \cdot n) \cdot (n^{(capa-1)\cdot(|I|-2)} \cdot W^{|I|-2}) \\ &\leq n^{(capa-1)\cdot(|I|-1)} \cdot W^{|I|-1} \end{aligned}$$

Thus Property 3 is proved for I. □

Lemma 3.5. *Algorithm 1 is pseudo XP with respect to capa.*

Proof. We assume *capa* are fixed and want to prove that the algorithm is polynomial in $|I|$ and W. Note firstly that the size of $\mathcal{S}(\mathcal{I})$ is $O(n \cdot W \cdot n^{capa} \cdot n)$.

Secondly, Lemmas 3.1, 3.2 and 3.4 prove by induction that Properties 2 and 3 are true. Thus, for each node $w \in \mathcal{S}(\mathcal{I})$, the size of $\mathcal{P}(w)$ is at most $W^{capa} \cdot n^{capa \cdot capa}$.

The number of iterations of the loops of Algorithm 1 and the complexity of each operation inside the loops depend polynomially on n and W, $\mathcal{S}(\mathcal{I})$ or $|\mathcal{P}(w)|$ for some node $w \in \mathcal{S}(\mathcal{I})$. Thus, the algorithm is polynomial. □

By Lemma B.1 (proved in Appendix B) and 3.5, we prove the following theorem.

Theorem 3.4. *If G is a DAG, max-1-DARP-M is pseudo XP in capa.*

3.2.2. A parameterized approximation for max-DARP-M

Corollary 3.2. *If G is a DAG, there is a $\frac{1}{\sqrt{n}}$-approximation for max-DARP-M in time pseudo XP with respect to capa.*

Proof. We use Algorithm 4.

Algorithm 4

Require: An instance $\mathcal{I} = (G, (V_c, V_c', B_c, E_c), t, \omega, capa, \alpha)$ of max-DARP-M
Ensure: A feasible solution for $\mathcal{I}$

1: $\mathcal{P} \leftarrow \emptyset$
2: **loop**
3: $\quad \mathcal{I}_1 \leftarrow$ the instance of max-1-DARP-M with the same parameters as $\mathcal{I}$
4: $\quad P \leftarrow$ an optimal solution for the instance $\mathcal{I}_1$ of max-1-DARP-M
5: $\quad$ **If** $P = \emptyset$ **Then Return** $\mathcal{P}$
6: $\quad$ **Else**
7: $\quad\quad$ Insert P into $\mathcal{P}$
8: $\quad\quad$ Remove every client satisfied by P from $\mathcal{I}$.

If G is a DAG and as *capa* is a fixed parameter, we can compute the taxi P at Line 4 in pseudopolynomial time by Theorem 3.4. Consequently, Algorithm 4 is pseudopolynomial.

We define $s(P)$ as the number of clients that are driven by the taxi P. Let $\mathcal{P}^* = (P_1^*, P_2^*, \ldots, P_q^*)$ be an optimal solution for $\mathcal{I}$ and let $\mathcal{P} = (P_1, P_2, \ldots, P_r)$ be the solution returned by Algorithm 4. We now show the following property :

Property 4. Either $\mathcal{P}^*$ and $\mathcal{P}$ are empty or $\frac{\sum_{i=1}^{q} s(P_i^*)}{\sum_{i=1}^{r} s(P_i)} \leq \sqrt{\sum_{i=1}^{q} s(P_i^*)}$.

Note that $\mathcal{P}$ is empty if and only if $\mathcal{P}^*$ is empty. We just have to assume that $\mathcal{P} \neq \emptyset$ and prove the second part of the property. Finally, note that Property 4 implies that Algorithm 4 is a $\frac{1}{\sqrt{n}}$-approximation algorithm as $\sum_{i=1}^{q} s(P_i^*) \leq n$.

We prove Property 4 by induction on n, the number of clients.

Basis : if there are 2 clients, then, there cannot be more than one taxi in a feasible solution for $\mathcal{I}$: $q = r = 1$. Consequently, the optimal solutions for $\mathcal{I}$ and for $\mathcal{I}_1$ are the same. Thus, $\frac{s(P_1^*)}{s(P_1)} = 1 \leq \sqrt{2} = \sqrt{s(P_1^*)}$. Consequently, Property 4 is proved in that case.

Inductive Step : We now assume that the property is true for every instance with n clients or less. Let $\mathcal{I}$ be an instance with $n+1$ clients. Let l be the number of taxis in $\mathcal{P}^*$ with a non empty intersection with P_1. Without loss of generality, we renumber those taxis $(P_1^*, P_2^*, \ldots, P_l^*)$. Note that $l \leq s(P)$ because a client cannot be satisfied by two taxis in $\mathcal{P}^*$, thus, there cannot be more than $s(P)$ taxis intersecting P. In addition, note that $l \leq q$.

As P_1 is an optimal solution of $\mathcal{I}_1$ and as every taxi in $\mathcal{P}^*$ is a feasible solution of $\mathcal{I}_1$,

$$\begin{aligned}\frac{\sum_{i=1}^{l} s(P_i^*)}{s(P_1)} &\leq \frac{l \cdot s(P_1)}{s(P_1)} \\ &\leq l \\ &\leq s(P_1)\end{aligned}$$

Consequently,

$$\sqrt{\sum_{i=1}^{l} s(P_i^*)} \leq s(P_1) \tag{7}$$

$$\frac{\sum_{i=1}^{l} s(P_i^*)}{s(P_1)} \leq \sqrt{\sum_{i=1}^{l} s(P_i^*)}$$

If $l = q$, then the property is proved.

Otherwise, let $\mathcal{J}$ be the instance $\mathcal{I}$ where every client satisfied by P_1 is removed. Note that, this instance is exactly the instance Algorithm 4 is working on at the beginning of the second of iteration. Consequently, if we directly run Algorithm 4 on instance $\mathcal{J}$, it returns $(P_2, P_3, \ldots, P_r)$.

Let $\mathcal{Q}^*$ be an optimal solution of $\mathcal{J}$. Note that, as $l \neq q$, $(P_{l+1}^*, P_{l+2}^*, \ldots, P_q^*)$ is not empty, and as it is a feasible solution for $\mathcal{J}$, $\mathcal{Q}^*$ is not empty. In addition, $r \geq 2$. By the inductive hypothesis,

$$\frac{\sum_{Q^* \in \mathcal{Q}^*} s(Q^*)}{\sum_{i=2}^{r} s(P_i)} \leq \sqrt{\sum_{Q^* \in \mathcal{Q}^*} s(Q^*)}$$

$$\sqrt{\sum_{Q^* \in \mathcal{Q}^*} s(Q^*)} \leq \sum_{i=2}^{r} s(P_i)$$

As $(P_{l+1}^*, P_{l+2}^*, \ldots, P_q^*)$ is a feasible solution for $\mathcal{J}$,

$$\sqrt{\sum_{i=l+1}^{q} s(P_i^*)} \leq \sum_{i=2}^{r} s(P_i)$$

By equation (7),

$$\sqrt{\sum_{i=1}^{l} s(P_i^*)} + \sqrt{\sum_{i=l+1}^{q} s(P_i^*)} \leq \sum_{i=1}^{r} s(P_i)$$

Finally, note that if $A > 0$ and $B > 0$, then $\sqrt{A+B} \leq \sqrt{A} + \sqrt{B}$,

$$\sqrt{\sum_{i=1}^{q} s(P_i^*)} \leq \sum_{i=1}^{r} s(P_i)$$

The inductive step is proved. Consequently, Property 4 is proved too and this concludes the proof of the corollary. □

Remark 5. The proof of Corollary 3.2 proves also that the smaller the optimal solution is, the better the approximation ratio is.

4. Conclusion

We have studied a taxi sharing problem in which the price of a trip is evenly shared between the passengers of the trip. The bill of the passengers must be reduced by a given factor α. In addition, the taxi must satisfy a capacity constraint and a time window constraint. We defined two optimization problems, max-DARP-M and max-1-DARP-M, and a decision problem 1-DARP-M and studied the parameterized complexity and approximability of those problems. It seems that the cost constraint affects the complexity of the problem more than the time constraint. Note that the time constraint make the problems weakly hard: when the width of the time windows are polynomially bounded and when the cost constraint is removed, the problems are polynomial. On the contrary, even if the time constraint is removed and if all the parameters are fixed, the problems are hard to solve.

We showed that there exists a pseudopolynomial algorithm for max-1-DARP-M and 1-DARP-M if the capacity *capa* of the taxis is fixed and if the road network is acyclic. This algorithm makes it possible to build a $\frac{1}{\sqrt{n}}$-approximation for max-DARP-M. However, considering its time complexity, this algorithm seems unpractical without any implementation improvement.

Some questions remain open: what is the parameterized complexity and approximability of the three problems with respect to α or to α and W?, and is there a constant factor parameterized approximation for max-DARP-M in *capa*?

To conclude, max-1-DARP-M seems too hard to be solved in practice and it looks like the cost constraint is the main cause of that. We think this constraint is hard because it is independently defined for each client. A way to simplify it could be to define a unique constraint for all the clients or for all the clients of a same taxi. In the current model, every client cannot pay more than α multiplied by the cost of a private ride. Instead of that constraint, we could ask all the clients of a same taxi to not pay more than the sum of all their private rides multiplied by α. If we then fairly divide the cost of the ride, no client would pay more than α multiplied by the cost of a private ride. Note that some of the clients would not pay exactly the cost of their own ride but also a part of the rides of the other clients.

[1] Attanasio, A., Cordeau, J., Ghiani, G., and Laporte, G. Parallel tabu search heuristics for the dynamic multi-vehicle dial-a-ride problem. *Parallel Computing 30*, 3 (2004), 377–387.

[2] Chan, S. *Metaheuristics for solving the dial-a-ride problem.* PhD thesis, North Carolina State University, 2004.

[3] Cordeau, J. A branch-and-cut algorithm for the dial-a-ride problem. *Operations Research 54*, 3 (2006), 573–586.

[4] Cordeau, J., and Laporte, G. A tabu search heuristic for the static multi-vehicle dial-a-ride problem. *Transportation Research Part B: Methodological 37*, 6 (2003), 579–594.

[5] Cordeau, J., and Laporte, G. The dial-a-ride problem: models and algorithms. *Annals of Operations Research 153*, 1 (may 2007), 29–46.

[6] Coslovich, L., Pesenti, R., and Ukovich, W. A two-phase insertion technique of unexpected customers for a dynamic dial-a-ride problem. *European Journal of Operational Research 175*, 3 (2006), 1605–1615.

[7] Crainic, T., and Malucelli, F. Meta-heuristics for a class of demand-responsive transit systems. *INFORMS Journal on Computing 17*, 1 (2005), 10–24.

[8] Dumas, Y., Desrosiers, J., and Soumis, F. The pickup and delivery problem with time windows. *European Journal of Operational Research 54*, 1 (1991), 7–22.

[9] Garey, M., and Johnson, D. *Computers and intractability: a guide to the theory of NP-completeness.* W. H. Freeman & Co., New York, NY, USA, 1979.

[10] Hu, T., and Chang, C. Exact Algorithm for Dial-A-Ride Problems with Time-Dependent Travel Cost. *Journal of the Eastern Asia Society for Transportation Studies 10* (2013), 916–933.

[11] Jung, J., Jayakrishnan, R., and Park, J. Dynamic Shared-Taxi Dispatch Algorithm with Hybrid-Simulated Annealing. *Computer-Aided Civil and Infrastructure Engineering* (jun 2015).

[12] Kann, V. Maximum bounded 3-dimensional matching is MAX SNP-complete. *Information Processing Letters 37*, 1 (1991), 27–35.

[13] Ma, S., Zheng, Y., and Wolfson, O. Real-time city-scale taxi ridesharing. *Knowledge and Data Engineering, IEEE Transactions on 27*, 7 (2015), 1782–1795.

[14] Nedregård, I. *The Integrated Dial-a-Ride Problem-Balancing Costs and Convenience.* PhD thesis, Norges teknisk-naturvitenskapelige universitet, Trondheim, 2015.

[15] Parragh, S., Doerner, K., and Hartl, R. Variable neighborhood search for the dial-a-ride problem. *Computers & Operations Research 37*, 6 (2010), 1129–1138.

[16] Pietrzak, K. On the parameterized complexity of the fixed alphabet shortest common supersequence and longest common subsequence problems. *Journal of Computer and System Sciences 67*, 4 (2003), 757–771.

[17] Powell, W. B., and Chen, Z.-l. A generalized threshold algorithm for the shortest path problem with time windows. In *Network Design: Connectivity and Facilities Location, Proceedings of a DIMACS Workshop, Princetin, New Jersey, USA, April 28-30, 1997* (1997), pp. 303–318.

[18] Ropke, S., and Cordeau, J. Branch and cut and price for the pickup and delivery problem with time windows. *Transportation Science 43*, 3 (2009), 267–286.

[19] Santos, D., and Xavier, E. Dynamic taxi and ridesharing: A framework and heuristics for the optimization problem. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence* (2013), IJCAI 13', AAAI Press, pp. 2885–2891.

[20] Santos, D., and Xavier, E. Taxi and Ride Sharing: A Dynamic Dial-a-Ride Problem with Money as an Incentive. *Expert Systems with Applications 42*, 19 (2015), 6728–6737.

[21] Savelsbergh, M., and Sol, M. The General Pickup and Delivery Problem. *Transportation Science 29*, 1 (feb 1995), 17–29.

[22] Wang, X., Dessouky, M., and Ordonez, F. A Pickup and Delivery Problem for Ridesharing Considering Congestion. *Transportation letters* (2015).

Appendix A. Proof of the reduction of Theorem 3.3

This appendix is dedicated to the proof of Theorem 3.3 by formally proving the reduction given in Subsubsection 3.1.3. We first show two intermediates lemmas.

Lemma A.1. *A feasible taxi picks up a client in W_i^j, for some $i < j$, and delivers it in W_j^i if and only if it goes through every path $X_l \cup X_l'$ including the arc $(x_l^{d_l}, x_l^{1\prime})$ for $l \in [\![i; j-1]\!]$.*

Proof. We first demonstrate that the taxi picks up a client in W_i^j, for some $i < j$, and delivers it in W_j^i if and only if it goes through every path $X_l \cup X_l'$ including the arc $(x_l^{d_l}, x_l^{1\prime})$ for $l \in [\![i; j-1]\!]$. The necessary condition follows from the fact that any path from W_i^j to W_j^i goes through those paths.

We now assume there is a feasible route of a taxi in H not containing any node of W_i^j and that this taxi goes through $(x_l^{d_l}, x_l^{1\prime})$ for some $l \in [\![i; j-1]\!]$. The clients that can go through that arc $(x_i^{d_i}, x_i^{1\prime})$ are those for which the origin is before $x_i^{d_i}$ and the destination is after $x_i^{1\prime}$ in a topological ordering of H. There are firstly the d_i clients of D_i. There are secondly the clients coming from sublayer W_p^q to sublayer W_q^p where $p \leq l$ and $q > l$ except if $p = i$ and $q = j$ because the taxi does not drive any client from W_i^j. There are $l \cdot (k-l) - 1$ such couples of sublayers. As there is no more than one client per sublayer in a path of H, there cannot be more than $d_i + i \cdot (k-i) - 1$ clients in a route of a taxi going through $(x_i^{d_i}, x_i^{1\prime})$. As $d_i = \frac{(k-1)\cdot k}{2} - i \cdot (k-i)$ and $capa = \frac{(k-1)\cdot k}{2}$, there can be at most $capa - 1$ clients in the taxi. However $\alpha = \frac{1}{capa}$, the taxi cannot go through an arc of cost 1 with less than $capa$ clients. Thus, the taxi cannot drive through any arc $(x_l^{d_l}, x_l^{1\prime})$ for $l \in [\![i; j-1]\!]$. □

Lemma A.2. *If there is a clique C of size k in G, there is a feasible solution for $\mathcal{J}$ satisfying S clients.*

Proof. If there is a clique C of size k in G, then, let u_i be the node of $C \cap V_i$. We define the subgraph P of H such that, for each $i < j \in [\![1; k]\!]$, P contains one node per sublayer, the origin and the destination of the client $c_{u_i}^{u_j}$, and all the paths $X_i \cup X_i'$. There is always in P an arc linking two nodes w_u^v and w_x^y of two consecutive sublayers of W_i because $u = x = u_i$. Thus, P is a path.

P satisfies the precedence constraints. A similar argument to the one given in the proof of Lemma A.1 proves that P never drives more than $capa$ clients at the same time and that P satisfies the cost constraint. Finally, P satisfies exactly $S = \frac{k\cdot(k-1)}{2} + \sum_{1}^{k-1} d_i$ clients. □

Lemma A.3. *The route of a feasible taxi contains exactly one node in each sublayer and all the nodes of X.*

Proof. As every sublayer W_i^j is either linked to the next sublayer or connected to W_{i+1} with the path $X_i \cup X_i'$, there is no path connecting two nodes of W_i^j. Thus, there is at most one node of W_i^j is a path of H.

The taxi must go through at least one arc of cost 1, because, for every client, there is such an arc separating its origin to its destination. Thus, by Lemma A.1, it must satisfy at least one client from W_1^k. Consequently, again by Lemma A.1, it goes through every arc $(x_l^{d_l}, x_l^{1\prime})$ for $l \in [\![i; k-1]\!]$ and every node of X. Thus, again by Lemma A.1, the route of the taxi contains one node per sublayer. □

Lemma A.4. *If there is a feasible taxi for $\mathcal{J}$ satisfying S clients, there is a clique C of size k in G.*

Proof. We now assume that there is a taxi P satisfying S clients. Let C be the subgraph of G induced by the set of edges $\{\{u, v\} \in E | w_u^v \in P\}$. Note that $w_u^v \in P \Leftrightarrow w_v^u \in P$ because P satisfies the precedence constraint. By Lemma A.3, there is in P exactly one node per sublayer and P contains all the

nodes of X. As a consequence, for each $i < j$, there are at least one node $u_i \in V_i$ and one node $u_j \in V_j$ such that the client $c_{u_i}^{v_j}$ is satisfied, thus, such that the edge $\{u_i, u_j\} \in C$. In addition, for each $i \leq k$, $|V_i \cap C| \geq 1$. By proving that $|V_i \cap C| = 1$ for all i, we prove that C is a clique of size k of G.

If, for some i, $|V_i \cap C| > 1$, there would be two nodes $u_1 \neq u_2 \in V_i \cap C$ and two other nodes v_1, v_2 such that $\{u_1, v_1\} \in C$ and $\{u_2, v_2\} \in C$. We assume that $v_1 \in V_{j_1}$, $v_2 \in V_{j_2}$ and $i < j_1 \leq j_2$. Every other case can be similarly proven. There are two nodes $w_1 = w_{u_1}^{v_1} \in W_i^{j_1} \cap P$ and $w_2 = w_{u_2}^{v_2} \in W_i^{j_2} \cap P$. By construction, there is a path in H from w_1 to w_2 if and only if $u_1 = u_2$. As w_1 and w_2 belong to the path P, we deduce that $u_1 = u_2$ and that $|V_i \cap C| = 1$. And this conclude the proof. □

Hardness results. Lemma A.3 proves that any feasible solution satisfies exactly S clients and any taxi satisfies all the clients of X. Thus there cannot be two taxis in a feasible solution. Consequently, in that instance, an optimal solution of the problems max-1-DARP-M and max-DARP-M satisfies S clients if and only if the answer to the problems max-1-DARP-M$_=$, max-1-DARP-M and 1-DARP-M is YES. Otherwise, no client can be satisfied.

Lemma A.2 and A.4 proves then Theorem 3.3.

Appendix B. Proof of the correctness of Algorithm 1

This part is dedicated to proof the correctness of Algorithm 1. The key idea is to prove that any mapping of $\mathcal{P}(w)$, for some node $w = (u, t, S, \kappa)$, corresponds to the part of a taxi from the first origin of its route to u, hereinafter called a *partial taxi*.

Definition 6. For every node $w = (u, t, S, \kappa) \in \mathcal{S}(\mathcal{I})$, we define the set $pP(w)$ of *partial taxis* of w as the set of paths P in G such that:

(i) P starts at an origin v_i, for some client i, and ends at u

(ii) the capacity constraint is satisfied;

(iii) the time constraint is satisfied;

(iv) S is the set $\{i | v_i \in P \text{ and } v_i' \notin P\}$ and κ is the value of $|\{i | v_i \in P\}|$.

(v) if $v_i' \in P$, then the precedence and the cost constraints are satisfied for the client i;

(vi) if $i \in S$, the cost $\omega(i, P)$ paid by the client i from v_i to u in P is less than $\alpha \cdot \omega(v_i, v_i')$;

Remark 6. If P is a taxi starting at v_i, then any subpath of P starting at v_i is a partial taxi. However, there exists partial taxis such that no valid taxi contains them.

Lemma B.1. *Let $w = (u, t, S, \kappa) \in \mathcal{S}(\mathcal{I})$. We assume $p \in \mathcal{P}(w)$ is mapping that was just added at Line 14. Let P be the result of the BUILD function on w and p, then $P \in pP(w)$ and, for each $i \in S$, $p(i) = \omega(i, P)$.*

Proof. Let L' be the list starting with the nodes of $\{w(v_i, t, \{i\}, 1) | i \in [\![1; n]\!], t \in [b_i, e_i]\}$ and ending with the list L defined at Line 6 of Algorithm 1. Note that any node w of $\{w(v_i, t, \{i\}, 1) | i \in [\![1; n]\!], t \in [b_i, e_i]\}$ has no predecessor (otherwise, let $w^- = w(u^-, S^-, \kappa^-)$ be that predecessor, then, according to Definition 2, by the Rule 4 an arc of $\mathcal{S}(\mathcal{I})$ must satisfy, $S^- = \emptyset$ and this is not compatible with Rule 1). As a consequence, L' is a topological ordering of $\mathcal{S}(\mathcal{I})$.

We prove the lemma by induction on the index of w in L'.

Basis: We first prove the lemma for the nodes of $\{w(v_i, t, \{i\}, 1) | i \in [\![1; n]\!], t \in [b_i, e_i]\}$. Let $w = w(v_j, t_j, \{j\}, 1)$ be such a node. The set of mappings of w is initialized at Line 2 of Algorithm 1. The only mapping in $\mathcal{P}(w)$ is p_j, the mapping associating 0 to the client j. By Definition of $pP(w)$, a partial taxi P of that set ends in v_j and satisfies $\{i | v_i \in P \text{ and } v'_i \notin P\} = \{j\}$ and $\{i | v_i \in P\} = 1$ by (i) and (iv). Thus, for all $i \neq j$, $v'_i \in P$ otherwise, by (v), $v_i \in P$ and this would be a contradiction with the fact that $\{i | v_i \in P\} = 1$. Consequently, $pP(w)$ contains only one partial taxi $P = \{v_j\}$. In that taxi, the client j pays 0. The lemma is then proved for w.

Inductive Step: Let $w = w(u, t, S, \kappa) \in L = L' \backslash \{w(v_i, t, \{i\}, 1) | i \in [\![1; n]\!], t \in [b_i, e_i]\}$. We now assume that the lemma is true for every node before w in L'. Note that $\mathcal{P}(w)$ is build in the for loop from line 7 and 15 of Algorithm 1.

Without loss of generality, we assume u is the origin v_j of the client j.

Let p be a mapping of $\mathcal{P}(w)$, this mapping is added at Line 14 of Algorithm 1. Consequently, there is a predecessor $w^- = (u^-, t^-, S^-, \kappa^-)$ of w and a mapping $p^- \in \mathcal{P}(w^-)$ such that $(p, w^-, p^-) \in \mathcal{P}(w^-, w)$ at Line 11. By Definition 2, as u is the origin of the client j, $S = S^- \uplus \{j\}$. The SUBMAP function is called at Line 10 of Algorithm 1. In that function, when $p_1 = p^-$, the current iteration is not stopped at Line 9 of Algorithm 2, otherwise (p, w^-, p^-) would not be returned in $\mathcal{P}(w^-, w)$. Consequently, $p(j) = 0$ and, for each client $i \in S^-$,

$$p(i) = p^-(i) + \frac{\omega(u^-, v_j)}{|S^-|} \leq \alpha \cdot \omega(v_i, v'_i) \tag{B.1}$$

Note that the value of $pred(p^-)$ is never changed after Line 15 of Algorithm 1: if we call the BUILD function with p^- and w^- just after p^- is added to $\mathcal{P}(w^-)$ or if we call it later, the result is the same. By the inductive hypothesis, that function returns a partial taxi $P^- \in pP(w^-)$ such that, for each $i \in S^-$, $p^-(i) = \omega(v_i, P)$.

If we call BUILD after p is added to $\mathcal{P}(w)$, as $pred(p)$ is set to (w^-, p^-), the result P is the path P^- to which we add the node v_j. We now prove $P \in pP(w)$.

- (i) is obviously proved for the path P.
- As $P^- \in pP(w^-)$, the capacity constraint is satisfied from the first node of P to v^- by (ii). By (iv), there are $|S^-|$ clients in the taxi when it leaves

the node u^-. By Definition 2, since the arc (w^-, w) exists in $\mathcal{S}(\mathcal{I})$ and since u is an origin node, $|S^-| \leq capa - 1$. Thus, there are at most *capa* clients in the taxi when it leaves v_j and (ii) is proved for P.

- The time constraint is satisfied for P^-. The taxi leaves u^- at time t^- and reaches u at time $t^- + t(u^-, u)$. By (iii) Definition 2, t belongs to the time windows associated with u and, since the arc (w^-, w) exists in $\mathcal{S}(\mathcal{I})$, $t = t^- + t(u^-, u)$. Consequently, (iii) is proved for P.

- $S = S^- \uplus \{j\} = \{i | v_i \in P^- \text{ and } v'_i \notin P^-\} \uplus \{j\}$ and $\kappa^- = |\{i | v_i \in P^-\}$ by (iv). In G, there is no path from v'_j to v_j, thus $v'_j \notin P^-$. Consequently, $S = \{i | v_i \in P \text{ and } v'_i \notin P\}$. Moreover, again by Definition 2, since the arc (w^-, w) exists in $\mathcal{S}(\mathcal{I})$ and since u is an origin node, $\kappa = \kappa^- + 1 = |\{i | v_i \in P^-\} \uplus \{j\}| = |\{i | v_i \in P\}|$. Consequently, (iv) is proved for P.

- u is an origin node, then, as (v) is true for P^-, it is also true for P.

- The cost $\omega(i, P)$ paid by any client $i \in S^-$ in the path P from v_i to v_j is $\omega(i, P^-) + \frac{\omega(u^-, v_j)}{|S^-|}$. By the inductive hypothesis, $\omega(i, P^-) = p^-(i)$. Thus, $\omega(i, P) = p^-(i) + \frac{\omega(u^-, v_j)}{|S^-|} = p(i) \leq \alpha \cdot \omega(v_i, v'_i)$ by Equation (B.1). The cost $\omega(j, P)$ paid by the client j is 0, thus $\omega(j, P) = p(j) \leq \alpha \cdot \omega(v_j, v'_j)$. Consequently, (vi) is proved for the path P.

As a consequence, P belongs to $pP(w)$ and, for each client $i \in S$, $p(i) = \omega(i, P)$. Lemma B.1 is shown for w. By induction, Lemma B.1 is proved. □

Lemma B.2. *For every node $w = (u, t, S, \kappa) \in \mathcal{S}(\mathcal{I})$, for each partial taxi $P \in pP(w)$, there is a mapping $p \in \mathcal{P}(w)$ such that, for each $i \in S$, $p(i) \leq \omega(i, P)$.*

Proof. This proof is similar to the one of Lemma B.1. We do it by induction on the same list L'. The basis is exactly the same in the two proofs.

For the inductive step, a converse argument to the inductive step of Lemma B.1 proves that there is a mapping p built with the SUBMAP function such that for each client $i \in S$, $p(i) = \omega(i, P)$. When, Algorithm 1 reaches Line 14, either there is a mapping $p' \preceq p$ and, for each client $i \in S$, $p'(i) \leq p(i) = \omega(i, P)$ or no such mapping exists and p is added to $\mathcal{P}(w)$. Lemma B.2 is shown for w and by induction, Lemma B.2 is proved. □

Theorem B.1. *Algorithm 1 returns an optimal solution for $\mathcal{I}$.*

Proof. We prove that Lines 17 and 20 returns either no solution if no solution exists or an optimal solution.

If, for some $w = w(v'_j, t, \emptyset, \kappa) \in T$, there is no mapping in $\mathcal{P}(w)$, then, by Lemma B.2, there is no valid taxi from any origin to v'_j in G. If T is empty, at Line 17, there is no solution and Algorithm 1 correctly returns no solution.

If such a mapping exists, by Lemma B.1, the function BUILD at Line 20 returns a path $P \in pP(\tau)$. As $\{i | v_i \in P \text{ and } v'_i \notin P\} = \emptyset$, thus every client of P satisfies the precedence and the cost constraints: P is a valid taxi. In

addition, P satisfies $|\{i|v_i \in P\}| = \kappa$ clients. By definition of τ, P is an optimal solution. $\square$