**TRANSPORTATION SCIENCE**
Vol. 00, No. 0, Xxxxx 0000, pp. 000–000
ISSN 0041-1655 | EISSN 1526-5447 | 00 | 0000 | 0001

**INFORMS**
DOI 10.1287/xxxx.0000.0000


# Feasibility of the Pickup and Delivery Problem with Fixed Partial Routes: A Complexity Analysis

Gerardo Berbeglia
HEC Montréal, 3000, chemin de la Côte-Sainte-Catherine, Montreal, Canada, H3T 2A7, gerardo.berbeglia@hec.ca,

Gilles Pesant
Département de génie informatique et génie logiciel, École Polytechnique de Montréal, C.P. 6079 Succursale Centre Ville, Montreal, Canada, H3C 3A7, pesant@polymtl.ca,

Louis-Martin Rousseau
Département de mathématiques et de génie industriel, École Polytechnique de Montréal, C.P. 6079 Succursale Centre Ville, Montreal, Canada, H3C 3A7, louis-martin.rousseau@polymtl.ca,

In the *Pickup and Delivery Problem* (PDP) a fleet of vehicles must serve customers requests which consist of transporting objects from their origins to their destinations. We introduce the *PDP with Fixed Partial Routes* (PDP-FPR), in which some partial routes are given, and the problem consists in obtaining a solution (a set of routes) which include those partial routes. We have analyzed the complexity of determining whether or not a feasible solution exists for this problem as well as for some relaxations of it. Checking the feasibility of the PDP-FPR and some of its relaxations is shown to be NP-complete, while for other relaxations, the problem was proved to be polynomial-time solvable.

*Key words*: pickup and delivery, complexity, partial routes, scheduling, dial-a-ride;

## 1. Introduction

*Pickup and Delivery Problems* (PDPs) constitute an important class of vehicle routing problems in which objects or people have to be collected and distributed. These problems arise in many areas such as robotics, courier services, and ambulatory services. PDPs can be classified into three different groups. In *many-to-many* PDPs, any vertex can serve as a source or as a destination for any commodity. The second group consists of *one-to-many-to-one* PDPs in which some commodities are available at the depot and are destined to the customer vertices while others are available at the customers are destined to the depot. Finally, in *one-to-one* PDPs, each commodity (also known as a request) has a given origin and a given destination. This problem is sometimes simply called the *Pickup and Delivery Problem* (PDP). Examples of the *one-to-one* PDP are courier operations and door-to-door transportation services offered for the elderly and handicapped people in many cities. The problem can have several constraints such as time windows, maximum ride times and other quality-of-service related restrictions. For a survey and classification of PDPs see e.g., Berbeglia et al. (2007).

In this paper, we introduce an extension of the *one-to-one PDP* called the *Pickup and Delivery Problem with Fixed Partial Routes* (PDP-FPR). Informally, in this problem we are given some partial routes and we need to construct a solution (i.e., a set of routes) that includes those partial routes. Our main motivation to study this problem and some of its relaxations is the development of a constraint programming algorithm to solve the Dial-a-Ride Problem. Under the constraint programming paradigm, solutions are constructed through the interaction between filtering methods and search techniques (Apt 2003). Along the way of finding a feasible solution or proving that none exists, partial solutions consisting of partial routes are obtained. It is here that it becomes

relevant to determine whether the actual partial solution can be 'extended' into a complete solution or if backtracking must be performed. Many of the results derived in this paper are used in a constraint programming approach to determine the feasibility of Dial-a-Ride Problem instances (Berbeglia et al. 2011). In addition, the study of the PDP-FPR can also be useful in other contexts where a feasible solution must respect some fixed partial routes, such as the study of arc-exchange procedures like *k-opt* (De Backer et al. 2000) as well as routing problems in which certain sequences of customers must be respected due to particular constraints.

The main contribution of this article is to prove that the problem of determining a feasible solution for the PDP-FPR is strongly NP-complete. This contrasts with the fact that the problem is polynomial-time solvable when no partial routes need to be respected. We have also studied the computational complexity of some of its relaxations and we have found some of them to be polynomial time solvable, while others remain NP-complete.

The PDP-FPR is not the first known routing problem for which determining the existence of a feasible solution is hard. Three other examples are the *Traveling Salesman Problem with Time Windows* (Savelsbergh 1985), the *Black and White Traveling Salesman Problem* (Ghiani et al. 2006), and the One-commodity Traveling Salesman problem with Pickups and Deliveries (Hernández-Pérez and Salazar-González 2003).

Similar results, this time in the field of scheduling, have been obtained by Mascis and Pacciarelli (2002) for a version of the job-shop scheduling problem. In their paper, the authors have proved that the problem of finding a feasible schedule that extends a given set of partial schedules is NP-complete, contrasting with the fact that the problem is polynomial time solvable when no partial schedules are imposed. Later, Meloni et al. (2004) have developed heuristics for the same problem based on the extension of partial schedules.

## 2. The Pickup and Delivery Problem with Fixed Partial Routes

The *Pickup and Delivery Problem with Fixed Partial Routes* can be defined as follows. Let $G = (V, A)$ be a complete and directed graph with vertex set $V = D \cup R$, where vertices $D$ represent the depot, and $R$ ($|R| = 2n$) represents the customer vertices. The set of depot vertices is partitioned into sets $Start = \{start(i) : i = 1, \ldots, m\}$ and $End = \{end(i) : i = 1, \ldots, m\}$ where $m$ represents the number of available vehicles.

The set $R$ is partitioned into sets $R^+$ (pickup vertices) and $R^-$ (delivery vertices). Let $H$ be the set of requests. Request $i \in H$ has pickup vertex $i^+ \in R^+$ whose load $q_{i^+} > 0$ and delivery vertex $i^- \in R^-$ whose load is $q_{i^-} = -q_{i^+}$. The loads of depot vertices are equal to zero, i.e., $q_i = 0$ for $i \in D$ and each vehicle has a capacity $Q > 0$. A *route* (not necessarily feasible) is a path over some vertices, that starts at any vertex $start(i) \in Start$ with $i \in \{1, \ldots, m\}$ (called the *starting depot* of vehicle $i$ and finishes at vertex $end(i) \in End$ (called the *ending depot* of vehicle $i$) such that the other vertices of the path belong to $R$. The maximum load of a route $r = (start(i), w_1, \ldots, w_t, end(i))$ is equal to $\max\{\sum_{j=1}^{h} q_{w_j} | 1 \leq h \leq t\}$.

A *partial route* is a sequence of vertices of $V$ which do not repeat and that can constitute a subsequence of a route. A *partial solution* consists of a set $P$ of partial routes such that each vertex $i \in D \cup R$ appears exactly in one partial route $p \in P$. The PDP-FPR consists of constructing $m$ vehicle routes such that:

($c1$) every vertex $v \in V$ appears in exactly one of the routes;
($c2$) for every request $r$, the pickup vertex and the delivery vertex are visited by the same route;
($c3$) for every request $r$, the pickup vertex $r^+$ is visited before its delivery vertex $r^-$;
($c4$) the maximum load of each route does not exceed $Q$;
($c5$) every partial route $p \in P$ is a subsequence of some of the $m$ routes.

If such a solution exists, the PDP-FPR instance is said to be *feasible*, otherwise it is *infeasible*. Note that this problem generalizes the feasibility problem of the *one-to-one Pickup and Delivery Problem*, since the latter can be seen as a special case in which each partial route in $P$ is composed of just one vertex. Observe also that the *one-to-one Pickup and Delivery Problem* is obtained if we do not impose constraint $(c5)$.

## 3. Definitions

Before studying the complexity of the PDP-FPR, we provide some notation and definitions regarding partial routes. Consider the partial route $p = (p_0, \ldots, p_u)$. We define $\alpha(p) = \max \{\sum_{i=0}^{j} q(p_i) : 0 \leq j \leq u\}$, $\delta(p) = \sum_{i=0}^{u} q(p_i)$ and $\gamma(p) = \max \{-\sum_{i=0}^{j} q(p_i) : 0 \leq j \leq u\}$. Thus, $\alpha(p)$ records how much more load will the vehicle attain along the path $p$ with respect to the load it had at the beginning. The value $\delta(p)$ is the difference between the vehicle load after and before the partial route $p$. Finally, $\gamma(p)$ records for how much less load the vehicle will lose along the path $p$ with respect to the load it began (see Figure 1 for an illustration). Observe that $-\gamma(p) \leq \delta(p) \leq \alpha(p)$ for any partial route $p$. Given partial routes $p_1$ and $p_2$, the partial route $p$ that consists of the concatenation of the partial routes $p_1$ and $p_2$ (in this order) is written $p = (p_1, p_2)$.

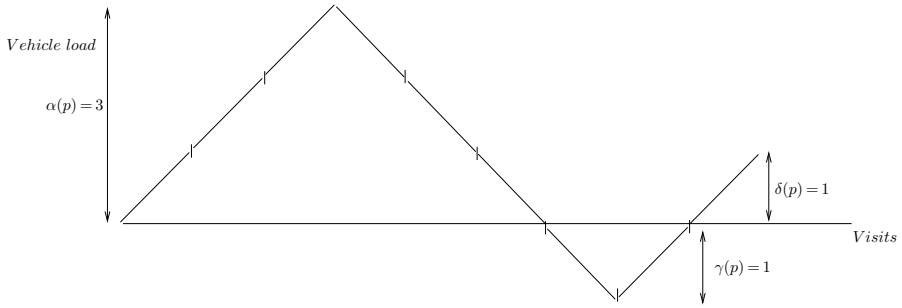


**Figure 1** **An example of a partial route which has a sequence of three pickups, followed by four deliveries and finishing with two pickups. All requests have unitary loads.**

A *vehicle route* or simply *route* $r = (v_0, \ldots, v_k)$ is said to be *empty* if $k = 1$ since in this case the route does not contain pickup nor delivery vertices. We say that a vertex $i$ belongs to the route $r = (v_0, \ldots, v_k)$ and we write it, $i \in r$, if $i = v_j$ for some $1 \leq j \leq k$. We say that a request $i \in H$ belongs to the route $r$ if $i^+ \in r$ and $i^- \in r$.

Let $I$ be an instance of the PDP-FPR. We say that a route is *feasible* if it respects the capacity and the precedence constraints. Formally, a route $r = (v_0, \ldots, v_k)$ is *feasible* if (1) for each $i \in H$ $i^+ \in r \Leftrightarrow i^- \in r$; (2) for each $i \in H$ such that $i^+ = v_l$ for some $1 \leq l \leq k-1$ then $i^- = v_j$ with $l < j \leq k$ and (3) $\sum_{i=1}^{j} q_{v_i} \leq Q$ for all $j = 1, \ldots, k$. Note that an empty route is by definition feasible.

## 4. Problem complexity

Determining whether a feasible solution exists for an instance of the *one-to-one Pickup and Delivery Problem* (i.e., without the constraint of fixed partial routes) is simple. One has just to verify if the inequality $Q \geq \max\{q_i | i \in R^+\}$ holds. If it does, it is possible to construct a feasible route that performs one at a time the pickup and the delivery of each request. If it does not, there is at least one request whose load is greater than the capacity of the vehicles and therefore no feasible solution exists. In this section, we prove that checking the feasibility of the PDP-FPR is strongly NP-complete, even for the single vehicle case ($m = 1$) and unitary loads.

The plan for the proof is the following. After giving a short definition and a lemma, we describe a problem, called the *Restricted M-optimal Scheduling Problem* which, as we will see later, is strongly NP-complete. We then describe a procedure that transforms, in polynomial time, instances of the

*Restricted M-optimal Scheduling Problem* into instances of the PDP-FPR, and we give an example. Finally, we prove that this transformation is such that the transformed instance of the PDP-FPR is feasible if and only if the original instance of the *Restricted M-optimal Scheduling Problem* is feasible, which proves that checking the feasibility of the PDP-FPR is strongly NP-complete.

DEFINITION 1. Two instances, $I$ and $I'$, of the PDP-FPR are said to be *equivalent* if either both are feasible or both are infeasible.

Consider an $n$ requests instance $I$ of the PDP-FPR in which requests loads are positive rational numbers $\{a_1/b_1, \ldots, a_n/b_n\}$ with $a_i, b_i \in \mathbb{N}$ and $gcd(a_i, b_i) = 1$ for $i = 1, \ldots, n$ where $gcd(x, y)$ denotes the greatest common divisor of $x$ and $y$. Assume also that the capacity of the vehicles is a rational number $q_1/q_2$ with $q_1, q_2 \in \mathbb{N}$ and $gcd(q_1, q_2) = 1$. Let $x$ be the least common multiple of $\{b_1, \ldots, b_n, q_2\}$. The following lemma will be needed for the NP-completeness proof that follows.

LEMMA 1. *It is possible to transform in polynomial time the instance $I$, into another equivalent instance $I'$ such that the vehicle's capacity is a natural number and with $\sum_{i=1}^{n} xa_i/b_i$ unitary requests.*

*Proof* The transformation begins by first multiplying the load of every request as well as the capacity of the vehicles by $x$. The request loads and the vehicle's capacity are now integers and since this can simply be seen as a change of units of the loads, this new instance is equivalent to $I$. Now every request $r \in H$ will be split into as many requests as the size of the load of request $r$. The pickup vertex $r^+$ is replaced with a new partial route $f(v) = (v_1, \ldots, v_h)$ consisting of the pickups of unitary load requests with $h = xa_v/b_v$, where $a_v/b_v$ is the original load of the request $r$. The same procedure is executed for the delivery vertex $r^-$, but the load of each of the vertices of the new partial route is $-1$ instead of 1. Finally, every partial route $p = (w_1, \ldots, w_f)$ of the original instance is replaced $f'(p) = (f(w_1), \ldots, f(w_f))$ which is the concatenation of the partial routes obtained for each of the vertices of partial route $p$. Under this transformation, passing through a partial route $f(v)$ simulates performing the pickup or delivery operation at the single vertex $v$. We can then conclude that the new instance is equivalent to the original instance $I$. □

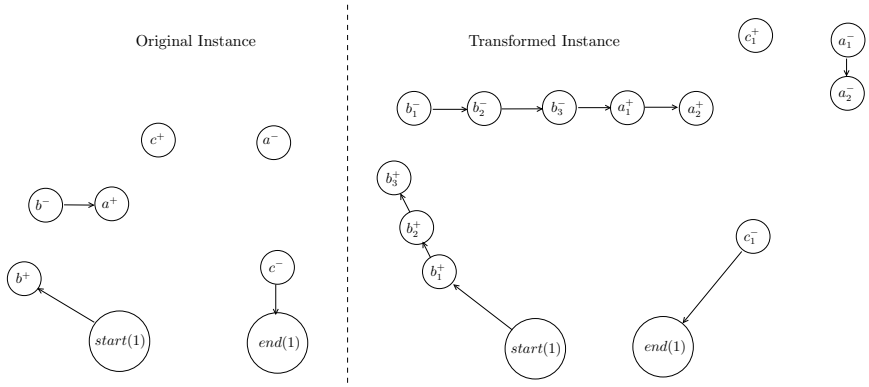


**Figure 2** **The instance of the left has requests $\{a, b, c\}$ with loads $1/2, 3/4, 1/4$ respectively and the vehicle capacity is 3/4. The equivalent instance on the right, with unitary requests and a vehicle capacity of 3, is obtained following the described transformation.**

We proceed to define the *Restricted M-optimal Scheduling Problem.*

**Name**: *Restricted M-optimal Scheduling Problem.*
**Input**: A natural number $k$ and a directed, acyclic and connected graph $G' = (V', A')$. Each node $w \in V'$ has associated a value $\pi(v)$ which is equal to $-1$ if the out degree $\delta^+(v) = 0$ and 1 otherwise.
**Question**: Does there exist a permutation $s = (s_1, \ldots, s_{n'})$ of the vertices in $V'$ such that (i) no

path exists from $s_i$ to $s_j$ in $G'$ whenever $i > j$; and (ii) $\sum_{i=1}^{j} \pi(s_i) \leq k$, for $1 \leq j \leq n'$?

As its name indicates, this problem is a restricted version of the *M-optimal Scheduling Problem* defined by Abdel-Wahab (1976) in which the function $\pi$ can take any integer values. See Figure 3 for an example of an instance of the *Restricted M-optimal Scheduling Problem* and a feasible solution. Abdel-Wahab (1976) have shown that the *M-optimal Scheduling Problem* is NP-complete by showing how to polynomially reduce the NP-complete *Register Allocation Problem* (Sethi 1973). In their proof, the authors transform every instance of the *Register Allocation Problem* into an instance of the *M-optimal Scheduling Problem* which is also an instance of the *Restricted M-optimal Scheduling Problem*. Therefore, the *Restricted M-optimal Scheduling Problem* is NP-complete.

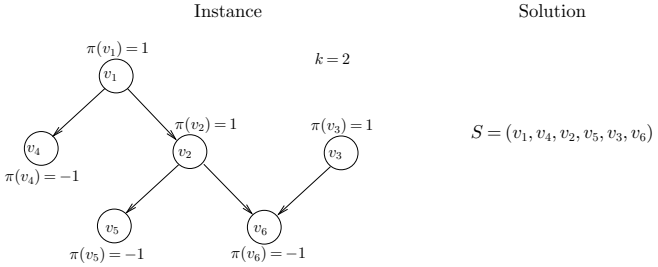


**Figure 3 An instance of the Restricted M-optimal Scheduling Problem and a solution.**

Observe that since the input graph is assumed to be acyclic, the set of arcs $A'$ induces a partial order in $V'$ as follows. Vertex $x$ *precedes* vertex $y$ (written $x < y$) if there is a path $p$ in $G'$ that starts at $x$ and finishes at $y$. Thus, property $(i)$ required in the *Restricted M-optimal Scheduling Problem* can be rephrased stating that the permutation must extend the mentioned partial order.

Assume that the vertex set of $G'$ is $V' = \{v_1, \ldots, v_{n'}\}$ and let $V^+ = \{v \in V' : \pi(v) = 1\}$ and $V^- = \{v \in V' : \pi(v) = -1\}$. Given an instance $I'$ of *Restricted M-optimal Scheduling Problem*, i.e., a graph $G' = (V', A')$ and a natural number $k$, we will construct an instance $I$ of the PDP-FPR. For illustration, the transformation of the instance shown in Figure 3 is given in Figure 4.

The instance $I$ has $n'+4$ partial routes $P = \{1, \ldots, n'+4\}$. Each $v_i \in V' = \{v_1, \ldots, v_{n'}\}$ of instance $I'$ has associated the partial route $i \in P$. Partial routes $n'+3$ and $n'+4$ are the starting depot and the ending depot respectively and they do not contain any request vertex. Finally two additional partial routes $n'+1$ and $n'+2$ are part of the new instance $I$. The request vertices of instance $I$ are placed following Algorithm 1. The function $AddRequest(i, j)$ in the algorithm means that a request whose pickup vertex is at partial route $i$ and whose delivery vertex is at partial route $j$ is inserted.

We can see from Algorithm 1 that the number of pickup vertices in each partial route $i$, with $1 \leq i \leq n'$ is equal $d_{out}(v_i) + 1$, i.e., the 'out' degree of vertex $v_i$ in the graph $G'$ plus one, because of the request whose delivery is at partial route $n'+2$. The number of delivery vertices in each partial route $i$, with $1 \leq i \leq n'$ is equal to $d_{in}(v_i)$, i.e., the 'in' degree of vertex $v_i$ in $G'$. Observe that each partial route $i$, with $1 \leq i \leq n'$, has no pickup vertices and $d_{in}(v_i) + 1$ delivery vertices, i.e. the 'in' degree of vertex $v_i$ in the graph $G'$ plus one because of the request whose pickup is in the partial route $n'+1$. In each partial route, it is assumed that the delivery vertices, in case there are some, are placed before the pickup vertices.

We now define the load for each request. Let $\epsilon$ be a positive real number, whose value will be given later. Table 1 shows the load of each request, in terms of $\epsilon$, according to the partial routes where their pickup and delivery vertices are located.

Since each request load must be positive, we need

$$\epsilon(d_{in}(w) - d_{out}(w)) + 1 > 0$$

**Algorithm 1** Insertion of request vertices on the partial routes of instance $I$

1: Input: Graph $G' = (V' = \{v_1, \ldots, v_{n'}\}, A')$
2: **for** each $v_i \in V^-$ **do**
3: AddRequest$(n'+1, i)$
4: **end for**
5: **for** each $v_i \in V^+$ **do**
6: AddRequest$(i, n'+2)$
7: **end for**
8: **for** each $(v_i, v_j) \in A'$ **do**
9: AddRequest$(i, j)$
10: **end for**
11: AddRequest$(n'+1, n'+2)$

| Requests | | Load |
|---|---|---|
| Partial route of the pickup | Partial route of the delivery | |
| $p \in \{1, \ldots, n'\}$ | $q \in \{1, \ldots, n'\}$ | $\epsilon$ |
| $n'+1$ | $n'+2$ | $\epsilon$ |
| $j \in \{i : v_i \in V^+\}$ | $n'+2$ | $\epsilon(d_{in}(v_j) - d_{out}(v_j)) + 1$ |
| $n'+1$ | $j \in \{i : v_i \in V^-\}$ | $1 - d_{in}(v_j)\epsilon$ |

**Table 1** **Load of the requests according to the partial routes where their pickup and delivery vertices are located in terms of $\epsilon$.**

for each $w \in V^+$, and

$$1 - d_{in}(v)\epsilon > 0$$

for each $v \in V^-$.

From the first condition, we conclude that $\epsilon(d_{out}(w) - d_{in}(w)) < 1$. Note that this is satisfied by any positive value of $\epsilon$ if $d_{out}(w) - d_{in}(w) \leq 0$. Therefore, to satisfy condition $(i)$ it is sufficient to take $\epsilon \geq \theta = \min\ (\{1/(d_{in}(w) - d_{out}(w) + 1) : w \in V^+, d_{out}(w) - d_{in}(w) > 0\} \cup \{1\})$. To satisfy the second condition, we should have $\epsilon < 1/d_{in}(v)$ for all $v \in V^-$. It is therefore sufficient to take $\epsilon \geq \varpi = \min\ (\{1/(d_{in}(v) + 1) : v \in V^-\} \cup \{1\})$. Since we need to satisfy both conditions we set $\epsilon = \min\ \{\theta, \varpi\}$.

Finally, the capacity of the vehicle is set to $k + \phi$, where $\phi = \epsilon + \sum_{v \in V^-} 1 - d_{in}(v)\epsilon$, i.e., the sum of the loads of the requests whose pickup vertices are in the partial route $n+1$.

We now show as an example, how the instance of the *Restricted M-optimal Scheduling Problem* shown in Figure 3, is transformed into an instance of the PDP-FPR depicted in Figure 4.

In the instance shown in Figure 3, $V^+ = \{v_1, v_2, v_3\}$ and $V^- = \{v_4, v_5, v_6\}$. The *in* and *out* degrees of the vertices are: $d_{in}(v_1) = 0$, $d_{out}(v_1) = 2$, $d_{in}(v_2) = 1$, $d_{out}(v_2) = 2$, $d_{in}(v_3) = 0$, $d_{out}(v_3) = 1$, $d_{in}(v_4) = 1$, $d_{out}(v_4) = 0$, $d_{in}(v_5) = 1$, $d_{out}(v_5) = 0$, $d_{in}(v_6) = 2$, $d_{out}(v_6) = 0$.

Thus, considering that $\epsilon = \min\ \{\theta, \varpi\}$, where $\theta = \min\ \{1/(d_{out}(w) - d_{in}(w) + 1) : w \in V^+, d_{out}(w) - d_{in}(w) > 0\} \cup \{1\}$ and where $\varpi = \min\ \{1/(d_{in}(v) + 1) : v \in V^-\} \cup \{1\}$ we deduce that $\epsilon = 1/3$. The capacity of the vehicle is set to $Q = k + \epsilon + \sum_{v \in V^-} 1 - d_{in}(v)\epsilon = 2 + \epsilon + |V^-| - 4\epsilon = 5 - 3\epsilon = 5 - 3/3 = 4$. The resulting instance of the PDP-FPR, which consists of 10 partial routes and a set $H = \{1, \ldots, 5, v_1, \ldots, v_6, \kappa\}$ of 12 requests, is given in Figure 4. Each requests of the form $v_i$, with $i = 1, \ldots, 6$, represents the request associated to the vertex $v \in V'$, which was added in the lines 2-7 of Algorithm 1. Each request $i$, with $i = 1, \ldots, 6$, represents a request associated with an arc $a' \in A'$ which was added in the lines $8 - 10$ of Algorithm 1. Finally, the request $\kappa$ is the one added in line 11. Above each partial route is written its number. The request loads are given in Table 2.

The described transformation is used to prove the following theorem.

| Request | Load |
|---|---|
| $v_1$ | $\epsilon(d_{in}(v_1) - d_{out}(v_1)) + 1 = 1/3$ |
| $v_2$ | $\epsilon(d_{in}(v_2) - d_{out}(v_2)) + 1 = 2/3$ |
| $v_3$ | $\epsilon(d_{in}(v_3) - d_{out}(v_3)) + 1 = 2/3$ |
| $v_4$ | $1 - d_{in}(v_4)\epsilon = 2/3$ |
| $v_5$ | $1 - d_{in}(v_5)\epsilon = 2/3$ |
| $v_6$ | $1 - d_{in}(v_6)\epsilon = 1/3$ |
| $\kappa, 1, \ldots, 5$ | $1/3$ |

**Table 2 Load of the requests in the example**

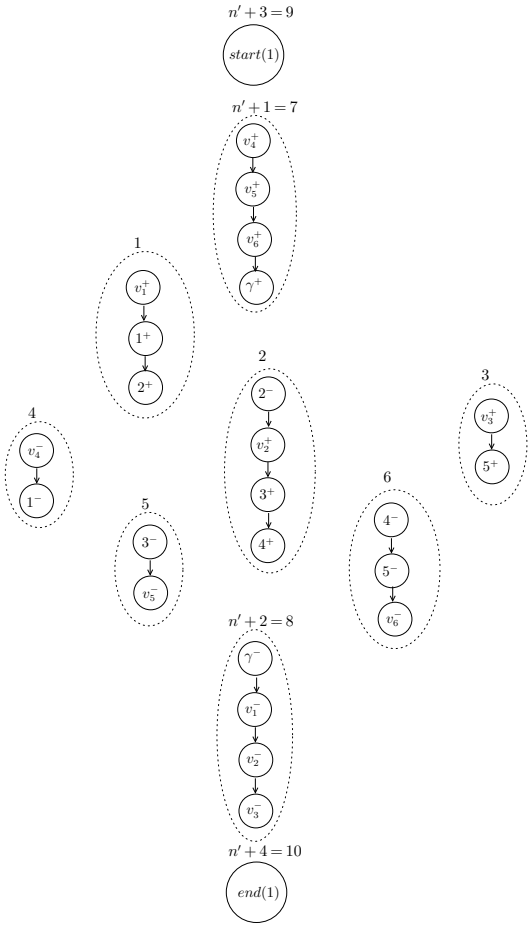


**Figure 4 The corresponding instance of the PDP-FPR**

THEOREM 1. *The Pickup and Delivery Problem with Fixed Partial Routes is strongly NP-complete, even when $m = 1$ and request loads are unitary.*

*Proof* The problem belongs to NP, since whenever a problem instance is feasible, a complete feasible route is a certificate which can be verified in polynomial time. Given an instance $I'$ of the *Restricted M-optimal Scheduling Problem*, i.e. a directed graph $G' = (V', A')$, with $V' = \{1, \ldots, n'\}$, and a natural number $k$, we construct an instance $I$ of the PDP-FPR as described above. Observe that this construction can be completed in polynomial time, even if $k$ is written in unary notation. We will now prove that the instance of *Restricted M-optimal Scheduling Problem* has positive answer if and only if the constructed instance $I$ is feasible.

Assume that the instance $I'$ is feasible. Thus, there exists a sequence $s = (v_{s_1}, \ldots, v_{s_{n'}})$ containing all the vertices of $V'$ such (i) it extends the partial order given by the arcs in $A'$ and (ii) $\sum_{i=1}^{j} \pi(v_{s_i}) \leq k$, for $1 \leq j \leq n'$. We will show that the route that traverses the partial routes of the constructed instance $I'$ in the order $s' = (n'+3, n'+1, s_1, \ldots, s_{n'}, n'+2, n'+4)$ is feasible. We first show that all precedence constraints between pickups and deliveries are respected using the given order of the partial routes. The partial route $n'+1$ has no delivery vertices and therefore it is possible to place it right after the partial route $n'+3$ which only contains the starting depot. The partial route $n'+2$ has no pickup vertices and therefore no precedence constraint is violated by placing it just before the partial route $n'+4$ which only contains the ending depot. Suppose now that there is a request whose delivery is in partial route $s_i$ and whose pickup is in the partial route $s_j$, with $i < j$. This would imply that $(v_{s_j}, v_{s_i}) \in A'$ and therefore $s = (v_{s_1}, \ldots, v_{s_{n'}})$ would not respect the precedence constraints in the *Restricted M-optimal Scheduling Problem*, which leads to a contradiction.

We will now show that the capacity of the vehicle is never exceeded. After visiting all vertices of the partial route $n+1$, the vehicle load will be equal to the sum of the loads of all pickup vertices in this partial route. This amount, which was already stated before, is $\phi = \epsilon + \sum_{v \in V^-} 1 - d_{in}(v)\epsilon$. Consider now a partial route $s_i$ whose associated vertex $v_{s_i}$ in $G'$ has $\pi(v_{s_i}) = +1$. What is the difference in the vehicle load between before and after traversing all the pickup and delivery vertices in the partial route $s_i$? To calculate this, we need to sum up the load of all requests whose pickup vertices are in $s_i$ and subtract the sum of the loads of all requests whose delivery vertices belong to the same partial route. This is exactly $\delta(s_i)$. The load of the request whose delivery is in the partial route $n+2$ is $\epsilon(d_{in}(v_{s_i}) - d_{out}(v_{s_i})) + 1$. All the other pickup vertices have a load of $\epsilon$ and there are $d_{out}(v_{s_i})$ of them. The number of delivery vertices is equal to $d_{in}(v_{s_i})$ and each of them has a load of $\epsilon$. Therefore

$$\delta(s_i) = \epsilon d_{out}(v_{s_i}) + \epsilon(d_{in}(v_{s_i}) - d_{out}(v_{s_i})) + 1 - \epsilon d_{in}(v_{s_i}) = 1.$$

Applying the same reasoning, but for the partial routes $s_j$ whose associated vertex in $G'$ has $\pi(v_{s_j}) = -1$, yields

$$\delta(s_j) = 0 - (1 - d_{in}(v_{s_j})\epsilon + d_{in}(v_{s_j})\epsilon) = -1.$$

Thus, for each partial route $i \in \{1, \ldots, n'\}$, $\delta(i) = \pi(v_i)$. The load of the vehicle after visiting the partial route $s_j$ with $1 \leq j \leq n'$ is

$$\delta(n'+1) + \sum_{i=1}^{j} \delta(s_j) = \phi + \sum_{i=1}^{j} \pi(v_{s_j}).$$

Since the sequence $(s_1, \ldots, s_n)$ is a feasible sequence for the *Restricted M-optimal Scheduling Problem*, it follows that

$$\phi + \sum_{i=1}^{j} \pi(v_{s_j}) \leq \phi + k = Q$$

and therefore the vehicle capacity $Q$ is respected at the end of visiting each partial route (note that the last partial route $n+2$ has only deliveries). The fact that in each partial route, all deliveries are done before the pickups, ensures that the capacity is respected along all the vertices of route. We conclude that the route which traverses the partial routes in the order $s' = (n+1, s_1, \ldots, s_n, n+2)$ is a feasible solution for the instance $I$ of the PDP-FPR.

Assume now that there is a feasible route $s'$ for the constructed instance $I$. The route $s'$, which must visit exactly once each of the $n'+4$ partial routes, can be represented as $s' = (n'+3, x_1, \ldots, x_h, n'+1, s_1, \ldots, s_l, n'+2, w_1, \ldots, w_f, n'+4)$. The partial route $n'+1$ must be before the

partial route $n'+2$, since both share a request whose pickup is in the partial route $n'+1$. Observe that $v_{w_i} \in V^-$ for all $1 \leq i \leq f$ because any $\beta \in V^+$ has a pickup which must be delivered at a vertex in the partial route $n+2$. Therefore, $\delta(w_i) = \pi(v_{w_i}) = -1$. Similarly, for all $1 \leq i \leq h$, $v_{x_i} \in V^+$, since any $\beta \in V^-$ has associated a delivery whose pickup is in the partial route $n+1$. This implies that $\delta(x_i) = \pi(v_{x_i}) = 1$. We show now that the sequence $s = (v_{x_1}, \ldots, v_{x_h}, v_{s_1}, \ldots, v_{s_l}, v_{w_1}, \ldots, v_{w_f})$ is valid for the instance $I'$ of the *Restricted M-optimal Scheduling Problem*.

The sequence $s$ respects the precedence constraints. This is easy to see, since for each arc $(v_x, v_y) \in A'$ in the instance $I'$ of the *Restricted M-optimal Scheduling Problem*, we have created a request whose pickup is in the partial route $x$ and whose delivery is in the partial route $y$.

We now prove that the sequence $s$ respects that each partial sum is at most $k$. Since $s'$ is a feasible route, we know that

$$\sum_{i=1}^{h} \delta(x_i) + \delta(n+1) = h + \phi \leq Q = k + \phi.$$

Therefore, we have that $h \leq k$, which implies that

$$\sum_{i=1}^{j} \pi(v_{x_i}) \leq h \leq k$$

for all $1 \leq j \leq h$.

Using again the fact that $s'$ is a feasible route for instance $I$, we know that

$$\sum_{i=1}^{h} \delta(x_i) + \delta(n+1) + \sum_{i=1}^{j} \delta(s_i) = \sum_{i=1}^{h} \delta(x_i) + \phi + \sum_{i=1}^{j} \delta(s_i) \leq Q = \phi + k.$$

for $1 \leq j \leq l$. Thus, it holds that

$$\sum_{i=1}^{h} \pi(v_{x_i}) + \sum_{i=1}^{j} \pi(v_{s_i}) \leq k$$

for $1 \leq j \leq l$.

Finally, taking into account that $s'$ is a feasible route and that $\delta(w_j) = -1$ for $1 \leq j \leq f$, it holds for every $1 \leq j \leq f$ that

$$\sum_{i=1}^{h} \delta(x_i) + \sum_{i=1}^{l} \delta(s_i) + \sum_{i=1}^{j} \delta(w_i) \leq k.$$

Therefore we conclude now that the sequence $s = (v_{x_1}, \ldots, v_{x_h}, v_{s_1}, \ldots, v_{s_l}, v_{w_1}, \ldots, v_{w_f})$ is valid for the *Restricted M-optimal Scheduling Problem*.

Taking into account how $\epsilon$ was set, we know that $\epsilon \in A \cup B$ where $A = \{1/(d_{out}(w) - d_{in}(w) + 1) : w \in V^+, d_{out}(w) - d_{in}(w) > 0\} \cup \{1\}$ and $B = \{1/(d_{in}(v) + 1) : v \in V^-\} \cup \{1\}$. We can conclude that $\epsilon = 1/t$ where $t$ is a natural number bounded by the number of vertices of the original graph $G$ plus one.

The load of the requests in the constructed instance are either of the form (i)$\epsilon = 1/t$ or (ii) $1 + z\epsilon = (t+z)/t$ where $z \in \mathbb{Z}$ and $|z|$ is bounded by the number of vertices in $G'$. Therefore the load of each request $i$ is of the form $a_i/b_i$ with $a_i = 1$ and $b_i = t$; or $a_i = t + z/gcd(t+z, t)$ and $b_i = t/gcd(t+z, t)$. Therefore, by Lemma 1, it is possible to transform in polynomial time the instance $I$ to one which has $\sum_{i=1}^{n} t a_i/b_i \leq \sum_{i=1}^{n} t + z = n(t+z)$ unitary requests. This number is at most $2n$ times the maximum degree of the original graph $G'$ plus one, which is polynomial in the size of the instance $I'$. □

## 5. Complexity of some relaxations

In this section, we define four relaxations of the PDP-FPR and we study the complexity of determining whether or not there exists a feasible solution. We also study a relaxation of the PDP-FPR with an additional restriction which imposes a maximum ride time for each request. This constraint is common in pickup and delivery problems where people are transported, called Dial-a-Ride Problems (Cordeau and Laporte 2007).

### 5.1. Elementary PDP-FPR

The *Elementary* PDP-FPR is a relaxation of the PDP-FPR that only takes into account constraints ($c1$), ($c2$), and ($c5$). Therefore, an instance $I$ of the PDP-FPR is said to be elementary feasible if it is possible to construct $m$ routes such that ($i$) each vertex is visited exactly once, ($ii$) the pickup vertex and the delivery vertex of the same request are visited by the same vehicle and ($iii$) every partial route $p \in P$ is a subsequence of some of the $m$ routes.

Let $I$ be an instance of the PDP-FPR. The following definitions are needed to characterize the instances which are elementary feasible.

Definition 2. The instance $I$ is said to be a *complete solution* (feasible or not) if for every partial route $p \in P$ and every request $h \in H$, $p$ is a route and $h^+ \in p \Leftrightarrow h^- \in p$.

Note that any instance which is a complete solution is elementary feasible.

Definition 3. A route $r$ (with $1 \leq r \leq m$) is said to be *open* in $I$ if there is no partial route $p \in P$ such that $start(r) \in p$ and $end(r) \in p$.

Definition 4. A request $h \in H$ is said to be *divided* in $I$ if there is a partial route $p \in P$ which is a route and exactly contains one of the vertices $h^+$ and $h^-$.

Given an instance $I$ of the PDP-FPR, we consider the undirected *auxiliary graph* $\tilde{G} = (\tilde{V}, \tilde{E})$ defined as follows. $\tilde{V}$ consists of the set of partial routes $P$ of $I$. An edge $i, j$ (with $i \neq j$) belongs to the edge set $\tilde{E}$ if one of these two properties hold: (1) there is a request $r \in H$ such that $r^+ \in i$ and $r^- \in j$, (2) there is a vehicle $v$ such that $start(v) \in i$ and $end(v) \in j$.

For simplifying notation, we say that a vertex $x$ of the PDP-FPR instance $I$ *belongs* to a vertex $w \in \tilde{G}$ and we write it $x \in w$, when $x$ belongs to the partial path of $I$ associated to the vertex $w$.

We are now ready to characterize the instances which are elementary feasible.

Theorem 2. *An instance $I$ of the PDP-FPR is elementary feasible if and only if (1) it is a complete solution; or (2) for any pair of different vehicles $v_1, v_2$, the partial routes associated to the vertices $start(v_1)$ and $end(v_2)$ do not belong to the same connected component in auxiliary graph $\tilde{G}$; and there is at least one open route; and no request $h \in H$ is divided in $I$.*

*Proof* Suppose $I$ is elementary feasible and it is not a complete solution, which means that either there is partial route $p'$ that is not a route, or that there is a divided request in $I$. In case there is a divided request, constraint $c2$ can never be respected and therefore $I$ can't be elementary feasible. We can thus assume there exists a partial route $p'$ that is not a route. Suppose that there is no open route, which implies that for each vehicle $i$ with $1 \leq i \leq m$ there is a partial route $p = (start(i), \ldots, end(i)) \in P$. Therefore, it follows that it is impossible for the partial route $p'$ to be potentially part of a *route* and therefore the constraint $c5$ is not respected. This means that $I$ is not elementary feasible, which is a contradiction. Suppose now that there are two vehicles $v_1$ and $v_2$ such that there exists a path $\tilde{p} = (w_1, \ldots, w_k)$ in $\tilde{G}$ such that $start(v_1) \in w_1$ and $end(v_2) \in w_k$. Note now that whenever there is an edge $\{i, j\}$ in $\tilde{G}$ then the partial routes $i$ and $j$ must be part of the same route in any solution which is feasible for the Elementary PDP-FPR. Therefore, the existence of the path $\tilde{p}$ implies that any extension of the instance $I$ must have the partial routes $w_1$ and $w_k$ in the same route. This implies that any solution must have the vertices $start(v_1)$ and $end(v_2)$ in the same route, in which case the instance $I$ is not elementary feasible, and this is a contradiction.

We proceed with the inverse implication: in the case that $I$ is a complete solution, then $I$ is elementary feasible. Let us assume now that $I$ is not a complete solution and that for any pair of different vehicles $v_1, v_2$, the partial routes associated to the vertices $start(v_1)$ and $end(v_2)$ do not belong to the same connected component in auxiliary graph $\tilde{G}$. Let us assume also that there is at least one open route and that no request is divided. We will now show that $I$ must be elementary feasible. Let $\mathcal{P} = \{\tau_1, \ldots, \tau_\ell\}$, with $\ell \geq m$, be a partition of the vertices of $\tilde{G}$ associated to the equivalence relation $R$ defined as $aRb$ if and only if the vertices $a \in \tilde{V}$ and $b \in \tilde{V}$ are in the same connected component in $\tilde{G}$. Let $\mathcal{S} \subset \mathcal{P}$ be the set of connected components $\{c|$such that there is no partial route $p \in c$ that contains a starting depot or an ending depot$\}$. We construct now an elementary feasible solution as follows. Let $i$ be a route which is open in $I$. Let $\tau_i$ be the set of partial routes which are in the connected component of the partial route that possesses $start(i)$ (therefore, the partial route containing $end(i)$ is also in $\tau_i$). Since route $i$ is open, it is possible to close route $i$ by including all the partial routes in $\tau_i$ as well as all the partial routes $\mathcal{S}$. Since no request is divided in $I$, the route $i$ will not have a pickup vertex of a request without having its delivery vertex. Every other route $1 \leq r \leq m, r \neq i$ which is open is completed by visiting all the partial routes of the connected component of the partial route of $start(r)$. It is clear that the resulting solution is elementary feasible. □

This characterization of the elementary feasible instances shows that the problem of determining whether or not an instance is feasible for the Elementary PDP-FPR can be solved in polynomial time.

### 5.2. Uncapacitated PDP-FPR

The *Uncapacitated* PDP-FPR is a relaxation of the PDP-FPR which consists of adding the precedence constraint ($c3$) to the Elementary PDP-FPR. Thus, an instance of the PDP-FPR is said to be feasible for the Uncapacitated PDP-FPR if it is possible to construct $m$ routes such that ($i$) each vertex is visited once, ($ii$) the pickup and the delivery of the same request are in the same route and the pickup is visited before the delivery, and ($iii$) every partial route $p \in P$ is a subsequence of some of the $m$ routes.

We show in this section how to determine, in polynomial time, whether or not an instance of the PDP-FPR is feasible for the Uncapacitated PDP-FPR.

We define first the *precedence graph* $\widehat{G} = (\widehat{V}, \widehat{A})$ associated to instance $I$ as follows. The vertex set $\widehat{V}$ has as elements each of the partial routes of the partial solution $P$ of $I$. An arc $(i, j)$ belongs to the arc set $\widehat{A}$ if and only if either one of these two properties hold: (1) the partial route $i$ has a pickup vertex whose associated delivery vertex is in the partial route $j$; (2) there is a vehicle $v$ such that partial route $i$ has the starting depot vertex associate to the vehicle $v$ and the partial route $j$ has the ending depot vertex associated to vehicle $v$. The construction of the *precedence graph* can be completed in $O(n^2)$ time.

To proceed, we need to define a property of partial routes called *locally feasible*, which intuitively tells whether or not a partial route satisfies the precedence constraint.

Definition 5. A partial route $p = (p_1, \ldots, p_k)$ of an instance $I$ of the PDP-FPR is said to be *locally feasible*, if for every request $r \in H$, if $p_i = r^+$ for some $i = 2, \ldots, k$, then $p_j \neq r^-$ for all $j = 1, \ldots, i-1$.

The following theorem characterizes the instances of the PDP-FPR which are feasible for the Uncapacitated PDP-FPR.

Theorem 3. *An instance $I$ is feasible for the Uncapacitated PDP-FPR if and only if $I$ is elementary feasible, each partial route of $I$ is locally feasible and $\widehat{G}$ is acyclic.*

*Proof* First note that for $I$ to be feasible for the Uncapacitated PDP-FPR, the partial routes of $I$ must be locally feasible and $I$ must be elementary feasible. Assume now that instance $I$ is

feasible for the Uncapacitated PDP-FPR and suppose its precedence graph $\widehat{G}$ contains a cycle $c=(i_1,\ldots,i_k,i_1)$, with $k>1$.

Observe first that in any solution that respects the constraints of the Uncapacitated PDP-FPR, the pair of partial routes $i$ and $j$ associated to any arc $a=(i,j)\in\widehat{A}$ must be part of the same route. Moreover, partial route $i$ must be traversed before partial route $j$. By transitivity, the existence of the cycle $c=(i_1,\ldots,i_k,i_1)$ means that partial route $i_2$ must be visited before and after partial route $i_1$, which is a contradiction.

Assume now that the directed graph $\widehat{G}$ is acyclic, each partial route of $I$ is locally feasible and that the instance $I$ is elementary feasible. Consider a partition $\mathcal{P}$ of the vertices of $\widehat{G}$ into their connected components. First note that since $I$ is elementary feasible, for any vehicle $i$, the partial routes of its starting depot and of its ending depot must belong to the same set in the partition $\mathcal{P}$. In addition, the depot vertices of different vehicles must be in different sets of the partition. The opposite case would mean that there must be a route which visits two depots associated with different vehicles, which is impossible. The partition can then be written as $\mathcal{P}=\{C_1,\ldots,C_m,E_1,\ldots,E_\rho\}$ with $\rho\geq 0$. The set $C_i$ is the connected component of $\widehat{G}$ which contains the partial routes of the starting and ending depot of vehicle $i$. The sets $E_i$ with $1\leq i\leq\rho$, consists of the connected components which do not contain any partial route with a starting or ending depot. We now construct each of the vehicle routes. If the sets $\{C_1,\ldots,C_m\}$ contain exactly one partial route each, then it means that $\rho=0$ because $I$ is elementary feasible. Thus, there is nothing left to do because all routes are already closed, and since we assumed $I$ to be elementary feasible and each partial route to be locally feasible, the routes are feasible for the Uncapacitated PDP-FPR. Assume now that there is at least one partition set $C_i$ such that $|C_i|\geq 2$. We construct the following route: First we traverse the partial route of $C_i$ where the starting depot is. Then, we traverse one after another, all the partial routes of the partition sets $\{E_1,\ldots,E_\rho\}$, in topological order, i.e., never choosing a partial route $y$ such that there is another partial route $x$ which has not yet been chosen satisfying $(x,y)\in\widehat{A}$. Once all the partial routes of $\{E_1,\ldots,E_\rho\}$ have been visited, we finish the route by visiting the rest of partial routes of $C_i$ using a topological order, which ensures that all the precedence constraints are respected. The route for each of the other partition sets $C_j$, with $j\neq i$, is constructed by visiting only the partial routes in $C_j$ in a topological order, therefore meeting all the precedence constraints. □

Since the problems of determining whether or not a partial routes is locally feasible, checking whether $I$ is elementary feasible, and detecting a cycle in a graph are polynomial-time solvable, deciding whether or not an instance of the PDP-FPR has a solution satisfying the constraints of the Uncapacitated PDP-FPR can be solved in polynomial time.

### 5.3. PDP-FPR without Precedence

The PDP-FPR *without Precedence* is a relaxation of the PDP-FPR in which the only relaxed constraint is the precedence constraint i.e., constraint ($c3$). Thus, an instance of the PDP-FPR is feasible for the PDP-FPR without Precedence if it is possible to construct a set of $m$ routes such that the pickup and delivery vertices of each request are both in the same route and their maximum load does not exceed $Q$. It worth observing that under this relaxation the vehicle load could be negative.

In this section we present a polynomial time algorithm to solve this problem in the single vehicle case ($m=1$). For the multiple vehicle case, the question whether or not there exists a polynomial time algorithm remains open. It is worth saying, that the presented algorithm can also be used to determine the feasibility of the fixed partial route extension of the *Traveling Salesman Problem with Pickups and Deliveries* (Mosheiov 1994).

Let $\pi=\{p_1,\ldots,p_l\}$ be a non-empty set of partial routes. We define $h(\pi)=\min\{\alpha(x)|\ x=(p_{\sigma_l(1)},\ldots,p_{\sigma_l(l)})$ with $\sigma_l\in S_l\}$ where $S_l$ is the symmetric group on $\{1,\ldots,l\}$ (i.e., the set of permutations of $l$ elements). Therefore, $h(\pi)$ is the minimum value of $\alpha$ that can be obtained from a

partial route which is the concatenation of all the partial routes in the set $\pi$. We will later show how to compute this function in polynomial time by solving a single machine scheduling problem.

**5.3.1. The algorithm** Checking whether an instance is feasible for the single vehicle PDP-FPR without Precedence basically consists in verifying that the partial routes in $S$ can be ordered in such a way that the capacity constraint ($c4$) is satisfied. Algorithm 2 performs this task.

**Algorithm 2** Feasibility checking algorithm for the single vehicle PDP-FPR without Precedence

Input: Instance $I$ of the PDP-FPR.
**if** $I$ is not elementary feasible **then**
**return** FALSE
**end if**
**if** $|P| = 1$ **then**
**return** $(\alpha(P) \leq Q)$
**end if**
Let $s_1 \in P$ be the partial route to which the starting depot belongs.
Let $s_2 \in P$ be the partial route to which the ending depot belongs.
**if** $(\alpha(s_1) > Q)$ or $(\sum_{i \in P \setminus \{s_2\}} \delta(i) + \alpha(s_2) > Q)$ or $(h(P \setminus \{s_1, s_2\}) > Q - \delta(s_1))$ **then**
**return** FALSE
**end if**
**return** TRUE

First observe that $I$ must be elementary feasible to be feasible for the PDP-FPR without Precedence. Second, if the set of partial routes $P$ has only one partial route (i.e., $|P| = 1$) then $P$ is already a complete route. Therefore, $P$ is feasible for the PDP-FPR without Precedence if and only if $\alpha(s) \leq Q$, with $s$ being the only partial route in $P$. If there are at least two partial routes in a partial solution $P$, observe first that the starting depot and the ending depot must belong to different partial routes, since we assume that the instance is elementary feasible. We then have to verify that ($i$) the partial route of the starting depot, $s_1$, does not exceed the vehicle's capacity $Q$; ($ii$) at the end of the route the partial route of the ending depot, $s_2$, does not exceeds the vehicle's capacity; and ($iii$) all the other partial routes, $P \setminus \{s_1, s_2\}$ can be ordered in such a way that they do not exceeds the value $Q - \delta(s_1)$ (see Algorithm 2). In the next section we show how to compute the function $h(R)$ in polynomial time. As a consequence, the single vehicle PDP-FPR without Precedence is solvable in polynomial time.

**5.3.2. Computing $h(P)$** The problem of computing $h(P)$ as it was defined is an optimization problem. The decision version of the problem can be stated as follows.

**Name**: *h(H,Q) decision problem.*
**Input**: A set $H = \{1, \ldots, k\}$ of partial routes and a natural number $Q$. Each partial route $p$ has associated two integer values denoted $\delta(p)$ and $\alpha(p) \geq 0$ such that $\alpha(p) \geq \delta(p)$.
**Question**: Does there exist an ordering $o = (o_1, \ldots, o_k)$ of the elements in $H$, such that $\sum_{i=1}^{j} \delta(o_i) + \alpha(o_{j+1}) \leq Q$ for $j = 0, \ldots, k-1$?

This problem can be seen as a *single machine scheduling problem with due dates and processing times* in which we need to determine whether or not there exists a schedule with no tardy tasks. The latter problem can be stated as follows.

**Name**: *Single machine scheduling decision problem with due dates and processing times.*
**Input**: A set $J = \{1, \ldots, k\}$ of jobs. Each job $i$ has associated a processing time $p(i)$ and a due date $d(i)$.

**Question**: Does there exist an ordering $o = (o_1, \ldots, o_k)$ of the elements in $J$, such that $\sum_{i=1}^{j} p(o_i) \leq d(o_j)$ for $j = 1, \ldots, k$?

Suppose we are given an instance of the $h(H, Q)$ *decision problem*, i.e., a set of partial routes $H$ and a natural number $Q$. The instance is then feasible if there exists an ordering $o = (o_1, \ldots, o_k)$ of the elements in $H$ such that

$$\begin{aligned}
&\sum_{i=1}^{j} \delta(o_i) + \alpha(o_{j+1}) \leq Q && \text{for } j = 0, \ldots, k-1 \\
\Leftrightarrow &\sum_{i=1}^{j+1} \delta(o_i) \leq Q + \delta(o_{j+1}) - \alpha(o_{j+1}) && \text{for } j = 0, \ldots, k-1 \\
\Leftrightarrow &\sum_{i=1}^{j} \delta(o_i) \leq Q + \delta(o_j) - \alpha(o_j) && \text{for } j = 1, \ldots, k.
\end{aligned}$$

Thus, if we consider each partial route $p \in P$ as a job and we let $\delta(p)$ be the processing time of job $p$ and $Q - \alpha(p) + \delta(p)$ be its due date, then $h(P) \leq Q$ if and only if there exists a schedule of the set of jobs such that all due dates are respected.

It is worth observing that after this transformation, some deadlines as well as some processing times of the tasks may be negative, and therefore the classical "earliest deadline first" algorithm cannot be applied. We present now a polynomial time algorithm to minimize the number of tardy tasks for the *Single machine scheduling decision problem with due dates and processing times* in which processing times and deadlines can be negative. The algorithm runs in $O(n^2)$ time and is to the best of our knowledge, the first one for this problem.

Consider an instance of the *Single machine scheduling decision problem with due dates and processing times* where $J = \{1, \ldots, n\}$ is the set of jobs. Given a schedule (an ordering) $S = (s_1, \ldots, s_n)$ of the jobs in $J$, a job $s_k$ is considered to be *tardy* in schedule $S$, if $\sum_{i=1}^{k} p(s_i) > d(s_k)$.

The problem consists of obtaining a schedule $S = (s_1, \ldots, s_n)$ of all jobs in $J$ such that the number of tardy jobs is minimized. Such a schedule is called *optimal*. A schedule is said to be *feasible* if it has no tardy tasks. Moore (1968) presented an $O(n^2)$ time algorithm to solve this problem but the algorithm requires the processing times of all tasks to be non-negative. To handle the case in which some jobs have negative processing time or negative deadlines, we decompose the problem into two subproblems, transform one of them, and apply Moore's algorithm to each subproblem. Before presenting the algorithm we need the following lemma.

LEMMA 2. *In any instance of the single machine scheduling with due dates, there exists an optimal schedule* $s = (s_1, \ldots, s_n)$ *such that* $p(s_i) \geq 0 \Rightarrow p(s_{i+1}) \geq 0$ *for* $i = 1, \ldots, n-1$.

*Proof* Consider an optimal schedule $s = (s_1, \ldots, s_n)$ such that there exists $i \in \{1, \ldots, n-1\}$ with $p(s_i) \geq 0$ and $p(s_{i+1}) < 0$. We now prove that the schedule $\tilde{s} = (s_1, \ldots, s_{i+1}, s_i, \ldots, s_n)$ is also optimal. It is clear that all jobs other than $s_i$ and $s_{i+1}$ are done on time in $\tilde{s}$ if and only if they are on time in $s$. Observe now that if $s_{i+1}$ is not late in $s$, it cannot be late in $\tilde{s}$. Note also that since $p(s_{i+1}) < 0$, $s_i$ cannot be late in $\tilde{s}$ if it is not in $s$. Applying this swapping procedure repetitively, we will obtain an optimal schedule respecting the desired property. □

Lemma 2 allows us to decompose the problem into two subproblems. The first one consists of optimally scheduling the jobs in $J^- = \{j \in J : p(j) < 0\}$. After scheduling the jobs in $J^-$, we will have 'extra time' (equal to $\sum_{j \in J^-} p(j)$) to schedule the jobs in $J^+ = J \setminus J^-$. We could add this extra time to each deadline of the jobs in $J^+$ and then obtain an optimal schedule for this second subproblem. Once both subproblems are solved, an optimal solution for the general problem can

be obtained by concatenating both schedules. In the second subproblem, all processing times are non-negative and therefore an optimal schedule can be obtained using Moore's algorithm.

We now describe how to find an optimal schedule of the tasks in $J^-$. Consider any schedule $S = (s_1, \ldots, s_k)$ of the jobs in $J^-$. Job $s_i$ is tardy if and only if

$$\sum_{j=1}^{i} p(s_j) > d(s_i) \Leftrightarrow \tag{1}$$

$$\sum_{j=1}^{k} p(s_j) - \sum_{j=i+1}^{k} p(s_j) > d(s_i) \Leftrightarrow \tag{2}$$

$$\sum_{j=i+1}^{k} -p(s_j) > d(s_i) + \sum_{j=1}^{k} -p(s_j) \Leftrightarrow \tag{3}$$

$$\sum_{j=i}^{k} -p(s_j) > d(s_i) - p(s_i) + \sum_{j=1}^{k} -p(s_j). \tag{4}$$

Note now that a job $s_i$ is tardy if and only if it is tardy in the schedule $S' = (s_k, \ldots, s_1)$ of the problem in which each job $s_l$ has processing time $-p(s_l)$ and a deadline equal to $d(s_l) - p(s_l) + \sum_{j=1}^{k} -p(s_j)$. Since in this new problem all processing times are non negative (because $-p(s_l) \geq 0$), we can apply Moore's algorithm, obtain an optimal schedule and reverse it for the original problem in which processing times are negative.

The algorithm for the general problem can then be stated as shown in Algorithm 3.

**Algorithm 3** Minimizing the number of tardy jobs in the single machine scheduling with due dates and negative processing times

Consider $J^- = \{i \in J : p(i) < 0\}$
Consider $J^+ = \{i \in J : p(i) \geq 0\}$;
Let $O^- = (v_1, \ldots, v_k)$ be an optimal sequence obtained by Moore's algorithm to schedule the tasks in $J^-$ with modified processing times and deadlines. A job $j_i$ with processing time $p(j_i)$ will have as new processing time $p'(j_i) = -p(j_i)$. Its new deadline will be $d'(j_i) = d(j_i) - p(j_i) + \sum_{l \in J^-} -p(s_l)$, where $d(h)$ and $p(h)$ are the original deadline and the original processing time of job $h$.
Let $O^+ = (w_{k+1}, \ldots, w_n)$ be an optimal sequence obtained by Moore's algorithm to schedule the tasks in $J^+$ with modified deadline. For each job $j_i \in J^+$, the new deadline is $d'(j_i) = d(j_i) - \sum_{h \in J^-} p(h)$ where $d(j_i)$ is the original deadline and $p(h)$ is the original processing time of the job $h \in J^-$.
Return the schedule $S = (v_k, \ldots, v_1, w_{k+1}, \ldots, w_n)$.

### 5.4. Positive Uncapacitated PDP-FPR without Precedence

The *Positive Uncapacitated* PDP-FPR *without Precedence* is a relaxation of the PDP-FPR that contains constraints $c1$, $c2$, $c5$ of the PDP-FPR and an additional constraint which can be stated as follows.

($c6$) On every route $r = (r_0, \ldots, r_q)$, the load cannot be negative, i.e., $\sum_{i=0}^{j} q_{r_i} \geq 0$ for $j = 0, \ldots, q$.

Thus, the Positive Uncapacitated PDP-FPR without Precedence relaxation resembles the PDP-FPR without Precedence except that there is a lower bound constraint (equal to zero) to the vehicle load, instead of an upper bound (equal to $Q$).

The following lemma shows that the problem of determining whether or not an instance of the Single vehicle PDP-FPR is feasible for the Positive Uncapacitated PDP-FPR without Precedence

can solved using the algorithm presented in Section 5.3 for the single vehicle PDP-FPR without Precedence.

LEMMA 3. *Checking whether an instance of the Single vehicle PDP-FPR is feasible for the Positive Uncapacitated PDP-FPR without Precedence can be done in polynomial time using Algorithm 2.*

*Proof* Consider an instance $I$ of the Single vehicle PDP-FPR and let $H = \{p_1, \ldots, p_k\}$ be the set of partial routes of $I$ with $p_1$ and $p_k$ being the partial routes associated to the starting depot and the ending depot respectively. For any instance $I$, we define a new instance $I'$ as follows. The pickup and delivery vertices are reversed. Thus, let $H' = \{p'_1, \ldots, p'_k\}$ be the set of partial routes of $I'$. For each $i \in \{1, \ldots, k\}$, partial route $p'_i$ has $\alpha(p'_i) = \gamma(p_i)$ and $\delta(p'_i) = -\delta(p_i)$. Finally, the vehicle capacity $Q'$ of instance $I'$ is set to zero.

We claim that instance $I$ is feasible for the Positive Uncapacitated PDP-FPR without Precedence if and only if instance $I'$ is feasible for the PDP-FPR without Precedence.

If there is only one partial route in $I$, then the instance $I$ is feasible for the Positive Uncapacitated PDP-FPR without Precedence if and only if

$$\begin{aligned} &\gamma(p_1) = 0 \Leftrightarrow && \\ &\gamma(p_1) \leq 0 \Leftrightarrow && [\text{ since } \gamma(p_1) \text{ is non-negative }] \\ &\alpha(p'_1) \leq 0 \Leftrightarrow && [\text{ by definition of } p'_1 \text{ }] \\ &\alpha(p'_1) \leq Q' && [\text{ since } Q' = 0 \text{ }] \end{aligned} \tag{5}$$

Thus, $I$ is feasible for the Positive Uncapacitated PDP-FPR without Precedence if and only if $I'$ is feasible for the PDP-FPR without Precedence.

We consider now the case in which there are at least two partial routes ($k > 1$). The first requirement for $I$ to be feasible is that $\gamma(p_1) = 0$ but as we have seen it is equivalent to have $\alpha(p'_1) \leq Q'$. This was one of the first three conditions for $I'$ to be feasible for the PDP-FPR without Precedence when $k > 1$. The second requirement is that the load cannot be negative at the last partial route $p_k$. Thus,

$$\begin{aligned} \sum_{i \in [k], i \neq k} \delta(p_i) - \gamma(p_k) &\geq 0 \Leftrightarrow && \\ \sum_{i \in [k], i \neq k} -\delta(p'_i) - \alpha(p'_k) &\geq Q' \Leftrightarrow && [\text{ definition of } I' \text{ }] \\ \sum_{i \in H, i \neq k} \delta(p'_i) + \alpha(p'_k) &\leq Q' && \end{aligned} \tag{6}$$

Therefore, the second requirement is satisfied if and only if the second requirement of Algorithm 2 is holds for instance $I'$.

Lastly, the third requirement is that there exists a permutation $(o_2, \ldots, o_{k-1})$ of the partial paths $\{p_2, \ldots, p_{k-1}\}$ such that the load is always non-negative along the partial routes $o_2, \ldots, o_{k-1}$ when the complete route $(p_1, o_2, \ldots, o_{k-1}, p_k)$ is traversed. Thus,

$$\begin{aligned} \delta(p_1) + \sum_{i=2}^{j} \delta(o_i) - \gamma(o_{j+1}) &\geq 0 && \text{for all } j = 0, \ldots, k-2 \Leftrightarrow && \\ -\delta(p_1) - \sum_{i=2}^{j} \delta(o'_i) - \alpha(o'_{j+1}) &\geq Q' && \text{for all } j = 0, \ldots, k-2 \Leftrightarrow && [\text{ definition of } I' \text{ }] \\ \sum_{i=2}^{j} \delta(o'_i) + \alpha(o'_{j+1}) &\leq Q' - \delta(p'_1) && \text{for all } j = 0, \ldots, k-2 && \end{aligned}$$

Therefore, the third requirement is equivalent to check whether $h(\{p'_2,\ldots,p'_{k-1}\}) \leq Q' - \delta(p'_1)$, which is exactly the third requirement of the Algorithm 2 (in the case in which $k=2$ the requirement trivially holds) $\square$.

### 5.5. Positive PDP-FPR without Precedence

The *Positive* PDP-FPR *without Precedence* is a relaxation of the PDP-FPR which contains all the constraints of the PDP-FPR without Precedence and the additional constraint ($c6$) described in the previous section which states that the vehicle load cannot be negative along any route. Therefore, an instance $I$ of the PDP-FPR is feasible for the Positive PDP-FPR without Precedence if it is possible to construct $m$ vehicles routes such that the pickup and delivery vertices of each request are both in the same route, the capacity of each vehicle is never exceeded and for each vehicle, the load along each point of its route cannot be negative. It worth observing that in the PDP-FPR, the load is never negative since the precedence constraints forbid the appearance of a delivery before its pickup in any route. Using the notation for partial routes, the Single Vehicle Positive PDP-FPR without Precedence can be stated as follows.

**Name**: *Single vehicle Positive PDP-FPR without Precedence.*
**Input**: A set $R=\{1,\ldots,k\}$ of partial routes and a natural number $Q$. Each partial route $p$ has associated three integer values denoted $\delta(p)$, $\alpha(p)\geq 0$ and $\gamma(p)\geq 0$ such that $-\gamma(p)\leq\delta(p)\leq\alpha(p)$. Also, $\sum_{i=1}^{k}\delta(i)=0$ and $\alpha(q)+\gamma(q)\geq 1$ for $2\leq q\leq k-1$.
**Question**: Does there exist an ordering $o=(o_1,\ldots,o_k)$ of the elements in $R$, such that: $(i)$ $o_1=1$; $(ii)$ $o_k=k$; $(iii)$ $\sum_{i=1}^{j}\delta(o_j)+\alpha(o_{j+1})\leq Q$; $(iv)$ $\sum_{i=1}^{j}\delta(o_j)-\gamma(o_{j+1})\geq 0$ for $j=0,\ldots,k-1$?

In the case in which each partial route $o_i$ with $2\leq i\leq k-1$ consists of exactly one pickup vertex or one delivery vertex and the partial routes $o_1$ (partial route associated with the starting depot) and $o_k$ (partial route associated with the ending depot) are empty, the problem is trivially equivalent to the *One Commodity Pickup and Delivery Traveling Salesman Problem* (1-PDTSP) which is NP-complete (Hernández-Pérez and Salazar-González 2003). As a consequence, determining whether or not there exists a feasible solution for the Single vehicle Positive PDP-FPR without Precedence is also NP-complete.

### 5.6. Uncapacitated PDP-FPR with ride times

An important case of the *one-to-one PDP* appears when the requests consist of users that need to be transported from an origin to a destination. In these cases, the problem is called the *Dial-a-Ride Problem* (DARP) whose main application is the transportation of elderly or disabled people. Generally, the DARP contains additional constraints which control user inconvenience, such as tight time windows and maximum ride times which impose an upper bound on the user's trip duration. Models and algorithms for the static and the dynamic versions of the DARP were surveyed by Cordeau and Laporte (2007).

In this section we study a relaxation called the *Uncapacitated* PDP-FPR *with ride times*, which is equivalent to the Uncapacitated PDP-FPR relaxation except that it also possess an additional constraint about the maximum ride times for each request.

We will see that the inclusion of this constraint, converts the polynomial time solvable Uncapacitated PDP-FPR into an NP-complete problem.

We denote by $t_{ij}$, the time taken to go from vertex $i$ to vertex $j$ in $G$. Consider a route $r=(1,v_1,\ldots,v_{2n},2)$, that respects the precedence constraint $c3$ and such that it serves the set $H=\{1,\ldots,n\}$ of $n$ requests, where vertices 1 and 2 represent the starting and the ending depot respectively. The route is assumed to respect the precedence constraint $(iii)$. Now consider a request $i$

and assume its pickup $i^+ \in R^+$ be $v_j$ and its delivery $i^- \in R^-$ be $v_k$. The *ride time* of request $i \in H$ on route $r$ is defined as

$$\sum_{l=j}^{k-1} t_{l,l+1}.$$

The maximum ride time constraint can be stated as follows.

($c7$) At each route $r$, the ride time of each request $i \in H$ does not exceed a preset value $L$.

Consider an instance $I$ of the single vehicle PDP-FPR. Let $R = \{1^+, 1^-, \ldots, n^+, n^-\}$ be the set of request vertices and 1(2) denote the starting (ending) depot vertex. Let $P$ denote a partial solution. The feasibility problem for the Uncapacitated PDP-FPR with ride times can be stated as follows.

**Name**: Uncapacitated PDP-FPR with ride times.

**Input**: $V = \{1, 2\} \cup R$, a non-negative number (maximum ride time) $L$. A partial solution $P$ and a distance function $t_{ij}$ defined over $V \times V$.

**Question**: Does there exist a permutation of $V$, $\rho = (v_1, \ldots, v_{2n+1}, v_{2n+2})$ such that $(i)$ $v_1 = 0$ and $v_{2n+2} = 2$; $(ii)$ for each $1 \leq i \leq n$, if $v_j = i^+$ and $v_k = i^-$ then $j < k$ and $\sum_{l=j}^{k-1} t_{l,l+1} \leq L$; $(iii)$ $\rho$ extends $P$.

By saying that the permutation $\rho$ extends $P$, we mean that every partial route $p \in P$ is a subsequence of the permutation (route) $\rho = (v_1, \ldots, v_{2n+1}, v_{2n+2})$.

We prove that this problem is NP-complete even in the Euclidean case by showing that the *Traveling Salesman Problem with Deadlines* can be polynomially reduced to the Uncapacitated PDP-FPR with ride times. Note that vertex $w_0$ represent the depot while the other vertices represent customers. We define below the *Traveling Salesman Problem with Deadlines*.

**Name**: *Traveling Salesman Problem with Deadlines.*

**Input**: A set $W = \{w_0, \ldots, w_n\}$, a set of non-negative numbers (deadlines) $T = \{t_1, \ldots, t_n\}$. An Euclidean distance function $t'_{ij}$ on $W \times W$.

**Question**: Does there exist a permutation of $W \setminus \{0\}$, $o = (o_1, \ldots, o_n)$ such that for each $k = 1, \ldots, n$, if $o_k = w_j$ then $t'_{0,o_1} + \sum_{i=1}^{k-1} t'_{o_i,o_{i+1}} \leq t_j$?

The *Traveling Salesman Problem with Deadlines* is NP-complete even in the Euclidean space, since in the case in which all deadlines $w_i$ are equal is equivalent to the problem of determining whether there is a tour over all vertices that is shorter than $w_i +$ max $\{t'_{i0} | 1 \leq i \leq n\}$, which is the Euclidean Traveling Salesman Problem.

THEOREM 4. *The Uncapacitated PDP-FPR with ride times is strongly NP-complete, even in the single vehicle case and with Euclidean distances.*

*Proof* Consider an instance $I$ of the *Traveling Salesman Problem with Deadlines*, i.e., a set $W = \{w_0, \ldots, w_n\}$, a set of non-negative numbers (deadlines) $T = \{t_1, \ldots, t_n\}$, and an Euclidean distance function $t'_{i,j}$ on $W \times W$. We assume, without loss of generality, that $t_1 \leq t_2 \leq \ldots \leq t_n$.

We construct the following instance $I'$ of the Uncapacitated PDP-FPR with ride time as follows. The vertex set is $V = \{1, 2, 1^+, 1^-, \ldots, n^+, n^-\}$ where for each $i = 1, \ldots, n$, vertex $i^-$ is located at the same place as $w_i$. Vertex $n^+$ is placed at the location of $w_0$. The rest of the vertices ($\{1, 1^+, \ldots, n - 1^+\}$) are located along a line in the order $(1, 1^+, \ldots, n - 1^+)$ respecting the following distances

$$t_{1,1^+} = \alpha$$

and

$$t_{j^+,j+1^+} = t_{j+1} - t_j$$

for $j = 1, \ldots, n-1$, where $\alpha$ is any non-negative constant.

The partial solution $P$ contains $n+2$ partial routes. The first partial route $p_1$ is composed of the starting depot (vertex 1) and of all the pickup vertices in the following order $p_1 = (1, 1^+, \ldots, n^+)$. The others partial routes are single vertex paths, one for each of the vertices in $\{2, 1^-, \ldots, n^-\}$. The maximum ride time is set to $L = t_n$. See Figure 5 for an illustration.

Since any feasible route must contain as a subsequence the partial route $p_1$ it is not necessary to specify how the line of the vertices in $p_1$ is placed with respect to the other vertices.

We claim that the instance $I'$ of the Uncapacitated PDP-FPR with ride times is feasible if and only if the instance $I$ of the Traveling Salesman Problem with Deadlines is.

Observe first that since any feasible solution of the instance $I'$ must extend the partial route $p_1 = (1, 1^+, \ldots, n^+)$, we need only to determine whether or not there exists a permutation $perm = (a_1, \ldots, a_n)$ of the delivery vertices, such that the route formed by the concatenation of the $p_1$ and $perm$, i.e., $r = (v_0, v_1, \ldots, v_n)(perm)$ is feasible.

By observing the way we have positioned the vertices of $p_1$, the time traveled from the pickup vertex $i^+$ up to the end of the partial route $p_1$ is equal to

$$\sum_{j=i}^{n-1} t_{i^+,i+1^+} = t_{i+1} - t_i + \ldots + t_n - t_{n-1} = t_n - t_i = L - t_i.$$

Thus, the time left after visiting vertex $n^+$ to visit the delivery vertex $i^-$ without exceeding the maximum ride time is $t_i$. Therefore, a feasible solution for $I'$ exists if and only if there exists a feasible solution for the instance $I$ of the Traveling Salesman Problem with Deadlines. □

Observe that in the reduction from the TSP with deadlines, the constructed instance of the Uncapacitated PDP-FPR with ride times always respects the precedence constraint because all the pickup vertices are put in the partial route of the starting depot. Thus, even by taking out the precedence constraint (constraint $c3$ of the PDP-FPR), the problem remains NP-complete.

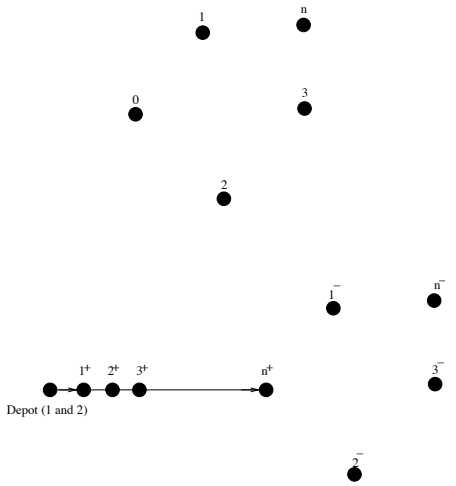


**Figure 5** **An instance of the TSP with deadlines and the constructed instance of the Uncapacitated PDP-FPR with ride times**

## 6. Conclusions

We have introduced an extension of the *one-to-one Pickup and Delivery Problem* in which certain fixed partial routes must be respected. Unlike what happens in the standard *one-to-one PDP*, we have proved that determining whether or not an instance of the PDP-FPR is feasible is strongly NP-complete. We have then studied the complexity of determining whether or not there exists a feasible solution to some relaxations of the problem. For some relaxations, polynomial time algorithms based on detecting a cycle in a graph and solving a scheduling problem with negative processing times, were found. For other relaxations, the problems remained NP-complete. A summary of these results is presented in Table 3. Constraints $c1$, $c2$, $c3$, $c4$ and $c5$ are those of the PDP-FPR, while constraints $c6$ and $c7$ are those extra constraints presented in sections 5.4 and 5.6 respectively. We write *'yes'* at a cell when the associate constraint is imposed, the symbol *'-'* when it is not imposed but it is always respected due to the other constraints, and *'no'* when it is not imposed and may not be respected. It is worth noting that for all the relaxations studied, the problem is trivially polynomial time solvable when partial routes are not considered.

| Problem name | Constraints | | | | | | | Complexity |
|---|---|---|---|---|---|---|---|---|
| | $c1$ | $c2$ | $c3$ | $c4$ | $c5$ | $c6$ | $c7$ | |
| PDP-FPR | yes | yes | yes | yes | yes | - | no | strongly NP-complete |
| Elementary PDP-FPR | yes | yes | no | no | yes | no | no | polynomial |
| Uncapacitated PDP-FPR | yes | yes | yes | no | yes | - | no | polynomial |
| PDP-FPR without precedence | yes | yes | no | yes | yes | no | no | unknown (poly for $m=1$) |
| Positive uncapacitated PDP-FPR without precedence | yes | yes | no | no | yes | yes | no | unknown (poly for $m=1$) |
| Positive PDP-FPR without precedence | yes | yes | no | yes | yes | yes | no | NP-complete |
| Uncapacitated PDP-FPR with ride times | yes | yes | yes | no | yes | - | yes | strongly NP-complete |

**Table 3 Summary of the complexity results**

**Acknowledgments**
We sincerely thank Jean-François Cordeau and Gilbert Laporte for their advices and comments that helped to improve the quality of this paper. The first author was supported by the Canada Research Chair in Distribution Management and the Canada Research Chair in Logistics and Transportation. This support is gratefully acknowledged.

## References

Abdel-Wahab, H. M. 1976. Scheduling with applications to register allocation and deadlock problems. Ph.D. thesis, University of Waterloo, Canada.

Apt, K. 2003. *Principles of Constraint Programming*. Cambridge University Press.

Berbeglia, G., J.-F. Cordeau, I. Gribkovskaia, G. Laporte. 2007. Static pickup and delivery problems: A classification scheme and survey. *TOP* **15** 1–31.

Berbeglia, G., G. Pesant, L.-M. Rousseau. 2011. Checking feasibility of dial-a-ride instances using constraint programming. *Transportation Science* Forthcoming.

Cordeau, J.-F., G. Laporte. 2007. The dial-a-ride problem: models and algorithms. *Annals of Operations Research* **153** 29–46.

De Backer, B., V. Furnon, P. Shaw. 2000. Vehicle routing problems using constraint programming and metaheuristics. *Journal of Heuristics* **6** 501–523.

Ghiani, G., G. Laporte, F. Semet. 2006. The black and white traveling salesman problem. *Operations Research* **54** 366–378.

Hernández-Pérez, H., J. J. Salazar-González. 2003. The one-commodity pickup-and-delivery traveling salesman problem. Michael Junger, Gerhard Reinelt, Giovanni Rinaldi, eds., *Eureka, You shrink! Lecture Notes in Computer Science*, vol. 2570. Springer, Berlin.

Mascis, A., D. Pacciarelli. 2002. Job-shop scheduling with blocking and no-wait constraints. *European Journal of Operational Research* **143** 498–517.

Meloni, C., D. Pacciarelli, M. Pranzo. 2004. A rollout metaheuristic for job scheduling problems. *Annals of Operations Research* **131** 215–235.

Moore, J. M. 1968. An $n$ job, one machine sequencing algorithm for minimizing the number of late jobs. *Management Science* **15** 102–109.

Mosheiov, G. 1994. The traveling salesman problem with pick-up and delivery. *European Journal of Operational Research* **79** 299–310.

Savelsbergh, M. W. P. 1985. Local search in routing problems with time windows. *Annals of Operations Research* **4** 285–305.

Sethi, R. 1973. Complete register allocations problems. *Proceedings of the Fifth Annual ACM Symposium on Theory of Computing* 183–195.