Complexity of Routing Problems with Release Dates

Claudia Archetti
Dominique Feillet
M. Grazia Speranza

WPDEM 3/2015
Complexity of routing problems with release dates

Claudia Archetti\(^{(1)}\)  Dominique Feillet\(^{(2)}\)  M. Grazia Speranza\(^{(1)}\)

\(^{(1)}\) Department of Economics and Management, University of Brescia, Italy, claudia.archetti@unibs.it, grazia.speranza@unibs.it
\(^{(2)}\) Ecole des Mines de Saint-Etienne, Department of Manufacturing Sciences and Logistics, Gardanne, France, feillet@emse.fr

Abstract

In this paper we consider a routing problem where uncapacitated vehicles are loaded with goods, requested by the customers, that arrive at the depot over time. The arrival time of a product at the depot is called its release date. We consider two variants of the problem. In the first one a deadline to complete the distribution is given and the total distance traveled is minimized. In the second variant no deadline is given and the total time needed to complete the distribution is minimized. While both variants are in general NP-hard, we show that they can be solved in polynomial time if the underlying graph has a special structure.

1 Introduction

Many carriers consolidate distribution by introducing a distribution center in the vicinity of final customers. Goods that are delivered to the centers usually by trucks are unloaded, sorted, consolidated and delivered to final customers by, usually smaller, trucks that perform routes, each visiting several final customers. A typical example of such distribution system is encountered in fast parcel delivery. Other examples are found in less-than-truckload distribution systems. Also, in the context of city logistics, often goods are consolidated and distribution in the city center is carried out by means of electric vehicles.

Usually, the phase of receiving goods arriving to these centers and the distribution to final customers are decoupled and treated independently from each other. When trucks reach the center, goods are received and temporarily stocked. At a later time, deliveries to final customers are organized. Goods are picked up, loaded into trucks and distributed. The two phases can be handled separately because of inventory that is built in the centers. Due to the pressure towards cost reduction in both inventory and transportation management,
third party logistics have grown in number and size. The management of distribution centers is becoming more and more dynamic. Goods arrive at any time during the day, and routes are continuously organized and started for delivery.

The classical vehicle routing problems (see the recent book by [9]) consider situations where the goods to be distributed are all available at the depot at the time the distribution starts. This is true also when time is considered such as in the problems with time windows (see [7]), in the periodic vehicle routing problems (see [8]) and in inventory routing problems (see [2] and [5]). A vehicle routing problem taking into account time issues recently introduced in the literature is the multi-period vehicle routing problem with due dates (see [1]) where goods have to be delivered within a certain time period. Also, city logistic problems deals with the distribution of freight over a planning horizon where different means of transportation have to be synchronized (see [6]).

Taking into account the arrival times of goods means considering a more dynamic organization of the distribution. In this case one has to organize the delivery routes with the additional constraint that goods are not all available at the depot at the start of the distribution. The routing problems must consider the additional issue of whether it is better to wait for additional goods to arrive and have a better loaded vehicle, or to start a route of the vehicle with the currently available goods. In the following, we call the arrival time of a product at the depot its release date. Problems with release dates, inspired by real applications, are introduced in [4, 3].

Most of the classical vehicle routing problems can be extended to consider release dates. We call this new class Vehicle Routing Problems with release dates (VRP-rd). The classical performance measure in vehicle routing problems is the distance. When release dates are considered, the minimization of the distance is achieved through waiting for all goods to arrive at the depot and solving a classical routing problem. This, however, may mean that the time required to complete the distribution is longer than desired. The completion time becomes a different relevant performance measure. Practical motivations for minimizing the completion time are the willingness to make the drivers available as early as possible for further tasks, the goal of improving the customer service or of avoiding rush hours. Motivations to minimize the distance are more classical and include the minimization of operational costs and related measures, such as pollution.

In this paper we focus on the novelty implied by the presence of release dates and analyze basic problems in the class of VRP-rd where vehicles are uncapacitated. As commonly done in routing problems, we assume
that the distance traveled coincides with the traveling time. We consider the case where a single vehicle is allowed to perform several trips during the time horizon (say, the day), one after the other, and the case where a fleet of vehicles is limited to a single route each. We consider two different objectives, the minimization of the total traveling distance within a deadline, and the minimization of the completion time, that is the sum of the total traveling time and the waiting time. We investigate the computational complexity of the optimization problems when the graph describing the locations of depot and customers has special structures, namely is a star or a line. The star models in an abstract way situations where the depot is at the center of the distribution area, whereas the line situations where customers are all located along a road. These special structures allow us to explore characteristics and properties implied by the presence of the release dates. We will show that, while the problems are in general NP-hard, they can be solved in polynomial time on stars and lines.

The paper is organized as follows. The following section provides the problem definition, summarizes the problem variants we study and summarizes the main contributions of the paper. Section 3 focuses on the Traveling Salesman Problem with release dates while Section 4 deals with the Uncapacitated Vehicle Routing Problem with release dates. Finally, conclusions are drawn in Section 5.

2 The Vehicle Routing Problem with release dates

Let $G = (V,A)$ be a complete graph. A traveling time and a traveling distance are associated with each arc $(i,j) \in A$. For the sake of simplicity, these two values are assumed identical and are denoted by $t_{ij}$. It is also assumed that the triangle inequality is satisfied. The set of vertices $V$ is composed by vertex 0 which identifies the depot and the set $N$ of customers, with $|N| = n$. The release date for customer $i \in N$ is denoted by $r_i$, $r_i \geq 0$. This means that the goods for customer $i$ arrive to the depot at time $r_i$. If goods for customer $i$ are already at the depot at the beginning of the time horizon, because they arrived overnight or the day before or earlier, then $r_i = 0$. We consider two different objectives. First, a deadline to complete the distribution is given and we aim at minimizing the total traveling distance. Then, no deadline is given and we aim at minimizing the total time needed to complete the distribution.

We consider two special cases: a case where a single vehicle is allowed to perform several trips during the time horizon (say, the day), one after the other, and a case where a fleet of vehicles is limited to a single route each. In both cases, capacity constraints are not considered. We call the first case the Traveling Salesman
Problem with release dates (TSP-rd) while the second is called the Uncapacitated Vehicle Routing Problem with release dates (UVRP-rd). For each case we consider the two above mentioned variants, where in the first variant the objective is to minimize the total traveling distance in such a way that the distribution is completed within a deadline $T$. We will refer to these problems as TSP-rd(distance) and UVRP-rd(distance), respectively. In the second variant we minimize the completion time, that is the sum of the total traveling time and the waiting time. We refer to these problems as TSP-rd(time) and UVRP-rd(time), respectively.

Note that, since we assume that the traveling time and the traveling distance associated with each arc $(i, j)$ are identical, the objective function of the TSP-rd(distance) and the UVRP-rd(distance) corresponds to the minimization of the sum of $t_{ij}$ of the arcs traversed. When considering the TSP-rd(time) and the UVRP-rd(time), the objective function minimizes the sum of $t_{ij}$ of the arcs traversed plus the waiting time at the depot. The waiting time is due to the fact that a vehicle has to wait at the depot until the latest release date of a customer that it is going to serve.

We focus on the study of the above mentioned problems on special graphs, namely the line and the star. The main contribution of this paper is to show that all the studied variants are polynomially solvable on these two special graphs. While the proof is straightforward for some of the cases considered, it is not for the case of the TSP-rd on the line. Tables 1 and 2 summarizes the results that we present in the following sections. We provide a short hint about how the optimal solution is built when the solution of the problem is straightforward, while we refer to the proof provided in the related section when it is not.
3 The Traveling Salesman Problem with release dates

In the *Traveling Salesman Problem with release dates* (TSP-rd) a single vehicle is considered. We study the TSP-rd(distance) and the TSP-rd(time). For both variants, if all release dates all equal to zero, due to the triangle inequality, it is optimal to load all goods on the vehicle and perform a single route. In this case the TSP-rd is equivalent to the TSP. This also implies that the TSP-rd is NP-hard, having the TSP as a special case. Note that if in the general case the deadline is very large, the TSP-rd(distance) becomes equivalent to the TSP, as it is optimal to wait until all goods are available and perform an optimal TSP tour. This is not true for the TSP-rd(time).

TSP-rd(time) always admits feasible solutions. For the TSP-rd(distance) to have a feasible solution, it must be possible to serve each customer within the deadline, that is a necessary feasibility condition is that 

\[ r_i + 2t_{0i} \leq T, \quad i = 1, \ldots, n. \]

It is not trivial, however, to guarantee that a feasible solution exists as it depends on the complete set of release dates and on the traveling times. Consider, for example, two customers, with release dates equal to zero and assume that time is measured in hours. The traveling time from the depot to each customer is equal to 4 and the traveling time between them is equal to 2. Suppose the deadline is 8. It is feasible to serve each customer individually but not to serve both within the deadline.

In the following we consider the special cases of the TSP-rd on the star and on the line. While the solution of both variants of the TSP-rd on the star is straightforward, the variants on the line are more complex and we propose to solve them through dynamic programming.

### 3.1 The TSP-rd on the star

In this section we consider the special case where graph \( G \) is a star, with the depot 0 located at the center of the star. It is easy to see that, whatever are the values of the release dates, the solution is formed by \( n \) routes going from the depot to each customer. Thus, the total distance traveled to serve all customers is a constant value equal to \( \sum_{i \in N} 2t_{0i} \) and this corresponds to the value of the optimal solution of the TSP-rd(distance). When considering the TSP-rd(time), as the total traveling time is constant, the minimization of the completion time corresponds to the minimization of the waiting time. Then, the best solution visits each customer in the order of non-decreasing release dates.


3.2 The TSP-rd on the line

In this section we consider the special case where graph $G$ is a line. The depot 0 is located at the point of abscissa 0. The set of customers $N$ is composed of two subsets $N_L$ and $N_R$ such that customers in $N_L$ are on the left side of the line (negative abscissa) and customers in $N_R$ on the right side (positive abscissa). On the right side, customers $i$ ($1 \leq i \leq n_R$) are placed at (positive) abscissa $t_R^i$ in non-decreasing order ($i < j$ implies $t_R^i \leq t_R^j$). Ties are broken arbitrarily. On the left side, customers $i$ ($1 \leq i \leq n_L$) are placed at (negative) abscissa $t_L^i$ in non-increasing order ($i < j$ implies $t_L^i \geq t_L^j$). Ties are broken arbitrarily. Traveling times (distances) between customers or between a customer and the depot are set to their (positive) difference in abscissa. Release date for customer $i$ is denoted $r^R_i$ if $i$ is on the right side, and $r^L_i$ if $i$ is on the left side. However, in order to ease the notation, superscripts $R$ and $L$ will be forgotten on release dates and abscissas when the side on which the customer is located is either clear from the context or not relevant.

Given a solution to any variant of the TSP-rd, we call route a trip starting and ending at the depot and not visiting the depot in between. A solution to the TSP-rd on the line is a sequence of routes $R_1, R_2, ..., R_X$, where $X$ is the total number of routes. In the following, without loss of generality, we assume that the routes are ordered on the basis of their non-decreasing departure time from the depot. Thus, $R_1$ is the first route performed by the vehicle and $R_X$ is the last one. We use the notation $R_r$ to indicate the set of customers served by the route. Note that a route cannot contain customers belonging to both sides of the line. We say that a route is on the right if the customers it serves are located on the right side of the line, otherwise the route is said to be on the left.

We investigate the complexity of the TSP-rd(distance) and of the TSP-rd(time) on the line.

For the sake of illustrating the algorithms and the proofs, we use the simple example of a line with two customers on each side depicted in Figure 1.

![Figure 1](attach:figure1.png)

Figure 1: Example of a TSP-rd on the line
3.2.1 Properties

Property 1. Let us consider two customers $i$ and $j$ on the same side of the line with $i < j$ (and thus $|t_i| \leq |t_j|$) such that $r_i \leq r_j$. An optimal solution to the TSP-rd(distance) and the TSP-rd(time) exists where $i$ and $j$ are in the same route.

Proof: Let us consider an optimal solution and the route $R$ in which customer $j$ is served. The departure time of $R$ is not less than $r_j$. If $i$ is not in this route, it is possible to transfer it to this route (as $r_i \leq r_j$). Furthermore, as $i$ is located in the segment defined by 0 and $j$, this transfer is done with no additional cost both in terms of completion time and traveled distance. The new solution is not worse and thus optimal.

A consequence of Property 1 is that the presence of a customer $i$ has no impact on the solution cost as long as there exists another customer $j$ on the same side of the line such that $|t_i| < |t_j|$ and $r_i < r_j$. Thus, in what follows, we consider only instances where $r_i > r_j$ for $i < j$ on the same side of the line. Therefore, on each side of the line, customers are ordered, starting from the depot and following the line, in non-decreasing order of traveling time from the depot and in non-increasing order of release date.

Definition 1. We say that a solution to the TSP-rd on the line is basic if the following condition holds:

\[ |t_i| \geq |t_j| \quad (i \in R_r, j \in R_{r'}, 1 \leq r < r' \leq X, r \text{ and } r' \text{ on the same side}). \]  

Thus, in a basic solution, each route goes on one side of the line, serves a subset of the farthest customers from the depot that have not yet been served by previous routes, and returns to the depot (see Figure 2).

Property 2. There exists a basic optimal solution to the TSP-rd(time) on the line. If there exists a feasible solution to the TSP-rd(distance) on the line, then there exists an optimal basic solution.

Proof: The proof is the same for the TSP-rd(time) and the TSP-rd(distance). In both cases we assume that an optimal solution exists and we assume that this optimal solution $s$ is not basic. We show that we can transform $s$ into a basic solution $s'$ with the same cost whatever the objective function is (completion time or distance).

If $s$ is not basic, it contains at least a pair of routes on the same side $R_r$ and $R_{r'}$ with $r < r'$ such that $R_r$ serves a customer $i$ and $R_{r'}$ serves a customer $j$ with $|t_i| < |t_j|$. Note that, since $i$ is on the segment between
Route $R_1$: 
(starting time 2, distance 20)

Route $R_2$: 
( starting time 22, distance 8)

Route $R_3$: 
( starting time 30, distance 16)

Figure 2: Example of a basic solution for the graph of Figure 1, with completion time 46 and traveled distance 44

0 and $j$, we have $r_i > r_j$. As the departure time of route $R_{r'}$ is greater than the departure time of route $R_r$, customer $i$ can be moved to route $R_{r'}$ with no additional cost (neither completion time nor distance). This procedure can be repeated iteratively until condition (1) is satisfied.

In the following we will prove that the TSP-rd(time) and the TSP-rd(distance) on the line are polynomially solvable.

**3.2.2 A polynomial time procedure for the solution of the TSP-rd(time) on the line**

We solve the problem by dynamic programming. We define a state-space graph that represents the possible states when the vehicle is located at the depot. A state $(i, j)$ indicates that all the vertices with an index higher than $i$ on the right side of the line and with an index higher than $j$ on the left side of the line are visited; furthermore, no other vertices are visited. The set of states with $0 \leq i \leq n_R$ and $0 \leq j \leq n_L$ defines the nodes of a graph $H$, where arcs are added to represent the possible transitions between these states. An arc is introduced from a state $(i_1, j_1)$ to another state $(i_2, j_2)$ if and only if $i_2 \leq i_1$, $j_2 \leq j_1$ and exactly one of these two inequalities is strict (exactly one thus being an equality).

Note that $H$ is acyclic and that the set of paths connecting state $(n_R, n_L)$ to state $(0, 0)$ in $H$ represents the set of basic solutions of the problem. Figure 3 depicts the state-space graph $H$ for the example of Figure 1 and the path corresponding to the basic solution of Figure 2.
(a) State-space graph $H$ for the example of Figure 1

(b) Path corresponding to the basic solution of Figure 2

Figure 3: Illustration of the state-space graph

state $(1, 2)$: the vehicle is at the depot and vertex 2 on the right has been visited
Thanks to Property 2, we know that at least one basic solution is optimal, that is, one of the paths between states \((n_R, n_L)\) and \((0, 0)\) in \(H\) represents an optimal solution. In order to find this path, we use the following recursion:

1. \(f(n_R, n_L) = 0\),
2. \(f_R(i, j) = \min_{i < k \leq n} (\max(f(k, j), r_{i+1}) + 2|t_k|)\),
3. \(f_L(i, j) = \min_{j < l \leq n} (\max(f(i, l), r_{j+1}) + 2|t_l|)\),
4. \(f(i, j) = \min(f_R(i, j), f_L(i, j))\).

In this recursion, \(f_R(i, j)\) gives the minimum completion time to reach state \((i, j)\) ending with a route on the right, \(f_L(i, j)\) gives the minimum completion time to reach state \((i, j)\) ending with a route on the left and \(f(i, j)\) gives the minimum completion time to reach state \((i, j)\).

**Theorem 1.** The completion time of the best basic solution is given by \(f(0, 0)\).

**Proof:** We proceed by induction, showing for each state \((i, j)\) that: (i) a path exists reaching state \((i, j)\) with completion time \(f(i, j)\), (ii) no path exists with a smaller completion time. These properties are true for state \((n_R, n_L)\). Let us consider a state \((i, j)\) and assume that they are true for all states before state \((i, j)\) in the topological order of the graph. If the last route performed to reach state \((i, j)\) is on the right, then the completion time cannot be lower than \(\min_{i < k \leq n} (\max(f(k, j), r_{i+1}) + 2|t_k|) = f_R(i, j)\). In fact, if the minimum completion time is lower than \(f_R(i, j)\), it means that there exists at least one state \((k, j)\), \(i < k \leq n\), for which the minimum completion time is lower than \(f(k, j)\), and this contradicts the assumption that the statement is true for all states before state \((i, j)\). The same holds for the case where the last route performed to reach state \((i, j)\) is a route on the left. As \(f(i, j) = \min(f_R(i, j), f_L(i, j))\), the result holds.

The optimal solution can be computed in polynomial time by computing values \(f(i, j)\) according to the topological order of graph \(H\). Given that the number of state is \((n_L + 1)(n_R + 1)\) and that each state is at most connected to \(n_L + n_R\) predecessor states, the following result holds.

**Theorem 2.** The TSP-\(rd(time)\) can be solved in \(O(n^3)\).

Figure 4 shows the values \(f(i, j)\) (underlined) and the optimal ingoing arcs for each state, for the example of Figure 1. Each arc connecting two states \((k, j)\) and \((i, j)\) is associated with a value equal to
max(f(k, j), r_{i+1}) + 2|t_k|$. The same for each arc connecting state \((i, l)\) with \((i, j)\). There are two optimal paths that reach state \((0, 0)\). The first one is the following: \((2, 2) \rightarrow (1, 2) \rightarrow (0, 2) \rightarrow (0, 0)\) and corresponds to the solution depicted in Figure 2. A second optimal path is \((2, 2) \rightarrow (1, 2) \rightarrow (1, 0) \rightarrow (0, 0)\) which corresponds to a solution where the vehicle first performs route \(R_1\) of Figure 2, then route \(R_3\) and, finally, route \(R_2\).

3.2.3 A polynomial time procedure for the solution of the TSP-rd(distance)
on the line

We solve the problem by dynamic programming. We define the same state-space graph \(H\) as in Section 3.2.2. We first define a non-polynomial solution approach and then prove that an efficient dominance rule can be introduced to obtain a polynomial-time complexity.

We first compute the latest allowed arrival time for each state \((i, j)\) in \(H\), i.e., the latest time at which
state \((i, j)\) can be reached without violating the deadline \(T\). We compute the latest arrival time by adapting the recursion of Section 3.2.2 and computing recursively the due dates starting from state \((0, 0)\). We denote this value \(\text{due}(i, j)\) and call it due date hereafter:

- \(\text{due}(0, 0) = T\),
- \(\text{due}_R(i, j) = \max\{\text{due}(k, j) - 2|t_i| \mid 0 \leq k < i\text{ such that } r_{k+1} \leq \text{due}(k, j) - 2|t_i|\} \) (\(-\infty\) if \(r_{k+1} > \text{due}(k, j)\) for all \(0 \leq k < i\)),
- \(\text{due}_L(i, j) = \max\{\text{due}(i, l) - 2|t_j| \mid 0 \leq l < j\text{ such that } r_{l+1} \leq \text{due}(i, l) - 2|t_j|\} \) (\(-\infty\) if \(r_{l+1} > \text{due}(i, l)\) for all \(0 \leq l < j\)),
- \(\text{due}(i, j) = \max(\text{due}_R(i, j), \text{due}_L(i, j))\).

Note that when \(r_{k+1} > \text{due}(k, j)\) then state \((k, j)\) is not a feasible state as the latest arrival time at which this state could be reached (and thus all locations from \(k + 1\) to \(n_R\) on the right and from \(j + 1\) to \(n_L\) on the left are visited) is lower than the release date of \(k + 1\). The same happens when \(r_{l+1} > \text{due}(i, l)\).

**Theorem 3.** *The TSP-rd(distance) admits a feasible solution if and only if \(\text{due}(n_R, n_L) \neq -\infty\).*

**Proof:** With a proof similar to the one used for Theorem 1, we can show that \(\text{due}(n_R, n_L)\) gives the latest arrival time at state \((n_R, n_L)\). When \(\text{due}(n_R, n_L) = -\infty\), we have that \(\text{due}_R(n_R, n_L) = \text{due}_L(n_R, n_L) = -\infty\), i.e., \(r_{k+1} > \text{due}(k, j)\) for all \(0 \leq k < n_R\) and \(r_{l+1} > \text{due}(i, l)\) for all \(0 \leq l < n_L\). This means that state \((n_R, n_L)\) cannot be reached on time. Moreover, when the TSP-rd(distance) has a feasible solution, this means that the state \((n_R, n_L)\) can be reached on time. This implies that \(\text{due}(n_R, n_L)\) is greater than or equal to 0. \(\square\)

Figure 5 illustrates the computation of due dates on the example of Figure 1 with \(T = 55\).

**A pseudo-polynomial time procedure for the solution of the TSP-rd(distance) on the line**

Section 3.2.2 describes a polynomial time procedure which can be used to check whether the TSP-rd(distance) admits a feasible solution. In fact, we can solve the TSP-rd(time) to calculate the minimum completion time needed to serve all customers. If the minimum completion time is not greater than \(T\), then the TSP-rd(distance) admits a feasible solution, otherwise the problem is infeasible. Thus, in the following we assume that the problem is feasible.
Every basic solution can be represented by a path connecting state \((n_R, n_L)\) to state \((0,0)\) in \(H\). Note that the opposite is not true as some paths do not correspond to a feasible solution. As we know that there exists an optimal basic solution, we know that one of the feasible paths in \(H\) represents an optimal solution. We compute this path with a labeling procedure.

Let us define a label \(L\) as a quadruplet \(L = (i, j, c, t)\) where \((i, j)\) is the state associated with the label, \(c\) is its distance and \(t\) its duration. Note that, as we assume that the traveling distance and the traveling time associated with each arc are identical, the distance corresponds to the total traveling time while the duration is the sum of the traveling time plus the waiting time. The distance and duration represent the accumulated distance and duration of the sequence of routes that are performed to reach state \((i, j)\). Note that different labels can be associated with the same state, with different distances and durations, as different sequences of routes can be used to attain this state.

The labeling procedure starts by labeling state \((n_R, n_L)\) with a single label \((n_R, n_L, 0, 0)\). The procedure then consists in selecting the nodes of graph \(H\) in topological order and extending all the labels associated with a node to all the neighbors of this node. The extension rule for a label is defined as follows.

Let \(L_1 = (i_1, j_1, d_1, t_1)\) be a label of state \((i_1, j_1)\) and \((i_2, j_2)\) a successor of \((i_1, j_1)\) in \(H\). If \(i_2 < i_1\) (and \(j_2 = j_1\)), the extension of \(L_1\) to \((i_2, j_2)\) results in a label \(L_2 = (i_2, j_2, d_2, t_2)\) with \(d_2 = d_1 + 2t_1\) and
\[ t_2 = \max(t_1, r_{i_2+1}) + 2t_{i_1}. \] If \( t_2 \) exceeds \( \text{due}(i_2, j_2) \), label \( L_2 \) is discarded. If \( j_2 < j_1 \) (and \( i_2 = i_1 \)), the extension of \( L_1 \) to \((i_2, j_2)\) results in a label \( L_2 = (i_2, j_2, d_2, t_2) \) with \( d_2 = d_1 + 2t_{j_1} \) and \( t_2 = \max(t_1, r_{j_2+1}) + 2t_{j_1} \). If \( t_2 \) exceeds \( \text{due}(i_2, j_2) \), label \( L_2 \) is discarded.

Applying this extension rule, the set of labels reaching state \((0, 0)\) represents the set of basic solutions and \( d \) represents the distance traveled. Thus, the optimal solution is the one associated with the label with minimum distance. However, as many as \( O(T) \) labels can be associated with each node and the procedure is not polynomial. Figure 6 shows the first two iterations of the procedure on the example of Figure 1. Note that label \((2, 1, 16, 26)\) associated with state \((2, 1)\) in the graph representing the first iteration of the procedure is discarded as the duration, which is 26, is greater than the due date of state \((2, 1)\) which is 25.

Figure 6: First and second iteration of the pseudo-polynomial algorithm for the TSP-rd(distance) example of Figure 1

A polynomial time procedure for solution of the TSP-rd(distance) on the line

We now show how the algorithm described above can be modified to become polynomial. The modifi-
cation relies on the following property.

**Property 3.** Let two labels $L_1 = (i, j, d_1, t_1)$ and $L_2 = (i, j, d_2, t_2)$ be associated with the same state $(i, j)$ with $d_1 < d_2$. $L_2$ can be eliminated without loss of optimality.

Proof: We prove that there exists a path obtained by extending the path associated with label $L_1$ till the final state $(0,0)$ whose distance is lower than or equal to the distance of any path obtained by extending the path associated with label $L_2$.

If $t_1 \leq t_2$, $L_2$ is trivially dominated. Thus, let us assume $t_2 < t_1$.

We introduce the following notation. We denote by $P$ the path associated with a label $L$. $P_1$ and $P_2$ are the paths associated with labels $L_1$ and $L_2$, respectively. Let $P^{(0,0)}$ be the best feasible path that extends path $P$ associated with label $L$ till the final state $(0,0)$ (note that a feasible extension always exists since labels satisfy due dates). We denote as $P^{(i,j)\rightarrow(0,0)}$ the part of path $P^{(0,0)}$ that goes from state $(i,j)$ to state $(0,0)$. Let $P \oplus P'$ be the concatenation of two paths $P$ and $P'$ for which the final node of $P$ coincides with the first node of $P'$. Then, $P^{(0,0)} = P \oplus P^{(i,j)\rightarrow(0,0)}$. We finally denote as $d(P)$ the distance of a path and $t(P)$ the completion time of a path starting at state $(n_R, n_L)$ (extended from label $(n_R, n_L, 0,0)$).

Given that $d_1 < d_2$, we will show that $d(P_2^{(0,0)}) \geq d(P_1^{(0,0)})$. We introduce path $\tilde{P}_1 = P_1 \oplus P_2^{(i,j)\rightarrow(0,0)}$.

For the sake of clarity, Figure 7 depicts the paths used in the proof.

![Figure 7: Paths used in the proof](image-url)
Let us first consider the case where the extension of \(L_1\) in \(\tilde{P}_1\) does not imply additional waiting times. In this case, we will show that:

1. if \(\tilde{P}_1\) is feasible, then it is better than \(P_2^{+,(0,0)}\);
2. if \(\tilde{P}_1\) is infeasible, the distance of the path \(P_1^{(i,j)\rightarrow(0,0)}\) is lower than the distance of \(P_2^{(i,j)\rightarrow(0,0)}\).

Let us prove case 1. We have \(d(\tilde{P}_1) = d_1 + d(P_2^{(i,j)\rightarrow(0,0)})\). As \(d_1 + d(P_2^{(i,j)\rightarrow(0,0)}) < d_2 + d(P_2^{(i,j)\rightarrow(0,0)}) = d(P_2^{(0,0)})\), then \(\tilde{P}_1\) is better than \(P_2^{(0,0)}\). As \(P_1^{(0,0)}\) is better than \(\tilde{P}_1\) by definition, we conclude that \(d(P_1^{(0,0)}) \leq d(P_2^{(0,0)})\).

Let us now move to case 2. By combining the following assertions:

- \(t(P_1^{+,(0,0)}) \geq t_1 + d(P_1^{(i,j)\rightarrow(0,0)})\);
- \(t(P_1^{+,(0,0)}) < t(\tilde{P}_1)\) (that holds because \(P_1^{+,(0,0)}\) is feasible and \(\tilde{P}_1\) is not);
- \(t(\tilde{P}_1) = t_1 + d(P_1^{(i,j)\rightarrow(0,0)})\) (as there is no waiting time while extending \(L_1\) to \(\tilde{P}_1\));

we obtain that \(d(P_1^{(i,j)\rightarrow(0,0)}) \leq t(P_1^{+,(0,0)}) - t_1 < t(\tilde{P}_1) - t_1 = d(P_2^{(i,j)\rightarrow(0,0)})\). Finally, as \(d(P_1^{+,(0,0)}) = d_1 + d(P_1^{(i,j)\rightarrow(0,0)})\), \(d(P_2^{+,(0,0)}) = d_2 + d(P_2^{(i,j)\rightarrow(0,0)})\) and \(d_1 < d_2\), we have \(d(P_1^{+,(0,0)}) < d(P_2^{+,(0,0)})\).

Let us now consider the second case, when the extension of \(L_1\) in \(\tilde{P}_1\) implies additional waiting times. The basic idea is to show that, as additional waiting time will happen when extending \(L_1\) with \(P_2^{(i,j)\rightarrow(0,0)}\), the advantage of \(L_2\) compared to \(L_1\), that is \(t_2 < t_1\), will be lost.

We denote as \(P_1^{\text{nost}}\) the maximal subpath of \(P_2^{(i,j)\rightarrow(0,0)}\) starting at node \((i,j)\) and such that \(P_1 \oplus P_1^{\text{nost}}\) does not imply waiting time after \((i,j)\). We consider the state following the last node of \(P_1^{\text{nost}}\) in \(P_1^{(i,j)\rightarrow(0,0)}\) and \(P_1^{\text{nost}}\) the path obtained by extending \(P_1^{\text{nost}}\) to this state. Without loss of generality, assume that this state is reached with a route on the right. We denote as \((k,j)\) this state. Let us finally recall that \(d_1 < d_2\) and \(t_2 < t_1\).

Waiting time at state \((k,j)\) implies that the vehicle has to wait for the release time of location \(k+1\), thus \(t(P_1 \oplus P_1^{\text{nost}}) = r_{k+1}\). Considering \(P_2 \oplus P_2^{\text{nost}}\) either we have \(t(P_2 \oplus P_2^{\text{nost}}) = r_{k+1}\) or \(t(P_2 \oplus P_2^{\text{nost}}) > r_{k+1}\). Thus, we have \(t(P_1 \oplus P_1^{\text{nost}}) \leq t(P_2 \oplus P_2^{\text{nost}})\). If we continue with the extension of both paths until state \((0,0)\), we obtain \(t(P_1 \oplus P_2^{(i,j)\rightarrow(0,0)}) \leq t(P_2 \oplus P_2^{(i,j)\rightarrow(0,0)}) = t(P_2^{(0,0)})\). Furthermore, \(d(P_1 \oplus P_2^{(i,j)\rightarrow(0,0)}) < d_2 + d(P_2^{(i,j)\rightarrow(0,0)}) = d(P_2^{(0,0)})\). Path \(P_1 \oplus P_2^{(i,j)\rightarrow(0,0)}\) is thus feasible and better than \(P_2^{(0,0)}\). As \(P_1^{(0,0)}\) is better than \(P_1 \oplus P_2^{(i,j)\rightarrow(0,0)}\), we obtain that \(d(P_1^{(0,0)}) < d(P_2^{(0,0)})\). \(\square\)
Thanks to this property, we can apply the labeling algorithm keeping only labels of minimum distance at each node. This provides a polynomial-time algorithm to solve the TSP-rd(distance). Given that the number of states is \((n_L + 1)(n_R + 1)\) and that each state is at most connected to \(n_L + n_R\) successors, the following result holds.

**Theorem 4.** The TSP-rd(distance) can be solved in \(O(n^3)\).

### 4 The Uncapacitated Vehicle Routing Problem with release dates

In the *Uncapacitated Vehicle Routing Problem with release dates* (UVRP-rd) an unlimited fleet of vehicles is considered. Each vehicle can perform a single route at most. We study the UVRP-rd(time) and the UVRP-rd(distance).

Note that, as the fleet is unlimited, assuming that a vehicle can perform one route at most does not change the definition of the problem. This assumption is introduced to better match the definitions commonly used in vehicle routing problems and to prepare for further extensions, such as for the case with a limited fleet of capacitated vehicles.

The UVRP-rd(distance) is NP-hard, having the TSP as a special case when all the release dates are equal to zero.

In the UVRP-rd(time), as the fleet is unlimited, we can assign a vehicle to each customer. At the release date of a customer, the assigned vehicle starts the route and delivers the goods to the customer. This solution minimizes the completion time of all deliveries. Then, the problem is polynomially solvable. Furthermore, when the graph is a star, as in the case of the TSP-rd, the total traveling distance is a constant value equal to \(\sum_{i \in N} 2t_0i\) which is the value of the optimal solution of the UVRP-rd(distance). Thus, the only case that remains to be studied is the UVRP-rd(distance) on the line which is analyzed in the following section.
4.1 The UVRP-\(\text{rd}(\text{distance})\) on the line

We now consider the UVRP-\(\text{rd}\) with customers located on a line. The depot is located at the point of abscissa 0. We first notice that Property 1 still holds. Consequently, the presence of a customer \(i\) has no impact on the solution value as long as there exists another customer \(j\) such that \(|t_i| < |t_j|\) and \(r_i < r_j\). Thus, we assume, without loss of generality, that \(r_i > r_j\) for \(i < j\). Moreover, as done for the TSP-\(\text{rd}\), we assume that the routes are ordered on the basis of their non-decreasing departure time from the depot. Given this order, Property 2 is still valid. Thus, if there is a feasible solution to the UVRP-\(\text{rd}(\text{distance})\), then there exists an optimal basic solution. Note that, in order to check whether a feasible solution exists, it is sufficient to verify that \(r_i + 2t_{0i} < T\) as the fleet is unlimited. Finally, an optimal solution exists such that each vehicle performs a route that visits customers on the right side of the line only or the left only. Also, the departure time of each route is independent from the departure time of the other routes. Thus, the problem can be decomposed in two subproblems: the right problem, i.e., the problem of serving the customers on the right side of the line, and the left problem, i.e., the problem of serving the customers on the left side of the line.

We now propose Algorithm 1 to solve the right problem which can be also used to solve the left problem. We define by \(R\) the set of routes, \(k\) the index of the farthest customer (on the right) not served by the routes in \(R\) and \(R_r\) the current route.

### Algorithm 1 Solution of the UVRP-\(\text{rd}\) on the line

1. \(R \leftarrow \emptyset\)
2. \(k = n_R\)
3. \(R_r \leftarrow \emptyset\)
4. repeat
5. Insert in \(R_r\) all customers \(i\) for which \(r_i \leq T - 2d_k\)
6. \(R = R \cup \{R_r\}\)
7. \(R_r \leftarrow \emptyset\)
8. \(k = \text{argmax}\{i| r_i > T - 2d_k\}\)
9. until all customers are served

Note that Algorithm 1 has complexity \(O(n^2)\) as the number of created routes may be as large as the number of customers and for each new route the algorithm explores the possibility of including the remaining customers.

Theorem 5. The UVRP-\(\text{rd}(\text{distance})\) on the line is solvable in time \(O(n^2)\). An optimal solution is provided by Algorithm 1.
Proof: Let us suppose that there exists an optimal solution $s'$ with a cost which is lower than the cost of solution $s$ given by Algorithm 1. Thanks to Property 2, we can assume that solution $s'$ is a basic solution. Without loss of generality, we can assume that each route in $s'$ departs from the depot as late as possible, i.e., at time $T - 2d_k$ where $k$ is the farthest customer visited in the route. Let us consider the first route $R_j(s')$ in $s'$ which is different from the corresponding route $R_j(s)$ in $s$. As routes 1, ..., $j - 1$ are identical in $s$ and $s'$, the farthest customer served in $R_j(s)$ is the farthest customer served in $R_j(s')$. Let us denote this customer by $k_j$. Note that the fact that route $R_j(s)$ serves all customers $i$ for which $r_i \leq T - 2d_{k_j}$ implies that route $R_j(s')$ serves a subset of customers served by route $R_j(s)$. Customers in $R_j(s)$ and not in $R_j(s')$ can be moved from routes $R_l(s')$ with $l > j$ to $R_j(s')$ with no additional cost. This procedure can be repeated iteratively for all routes in $s'$. We obtain a solution which is identical to $s$, thus reaching a contradiction.

5 Conclusions and future research

The major contribution of this paper is the analysis of basic routing problems where products to be distributed arrive to the depot at different times. The presence of release dates implies that completion time becomes a relevant performance measure. One may be interested in minimizing the total distribution time or in minimizing the distance traveled, provided the distribution is completed within a deadline. We have shown that the uncapacitated problems, with one vehicle that can perform different routes or an unlimited fleet of vehicles where each vehicle can perform one route only, can be solved in polynomial time when the underlying graph is a star or a line. The analysis has brought to light important characteristics of the routing problems with release dates.

As the class of routing problems with release dates is new, several research directions remain open. We mention here only the directions closely related to the content of this paper. The algorithmic solution of the problems studied merits investigation. Also, as variants of the problems studied in this paper one might consider a deadline specific for each customer, or to replace release dates with time intervals during which goods will arrive.
References


