Computers and Operations Research 98 (2018) 24–37

Contents lists available at ScienceDirect

Computers and Operations Research

journal homepage: www.elsevier.com/locate/cor

An iterated local search for the Traveling Salesman Problem with release dates and completion time minimization

Claudia Archetti[a], Dominique Feillet[b], Andrea Mor[a,*], M. Grazia Speranza[a]

[a] University of Brescia, Department of Economics and Management, Brescia, Italy
[b] Ecole des Mines de Saint-Etienne and LIMOS UMR CNRS 6158, CMP Georges Charpak, Gardanne F-13541 France

ARTICLE INFO

Article history:
Received 4 October 2017
Revised 26 April 2018
Accepted 3 May 2018
Available online 16 May 2018

Keywords:
Traveling Salesman Problem with release dates
Heuristics
Matheuristics
Iterated local search

ABSTRACT

In the Traveling Salesman Problem (TSP) with release dates and completion time minimization an uncapacitated vehicle delivers to customers goods which arrive at the depot over time. A customer cannot be served before the demanded goods arrive at the depot. A release date is associated with each customer which represents the time at which the goods requested by the customer arrive at the depot. The vehicle may perform multiple routes, all starting and ending at the depot. The release dates of the customers served in each route must be not larger than the time at which the route starts. The objective of the problem is to minimize the total time needed to serve all customers, given by the sum of the traveling time and the waiting time at the depot. The waiting time is due to the fact that the vehicle has to wait at the depot until the latest release date of the customers it is going to serve in the next route. We introduce some properties, propose a mathematical programming formulation and present a heuristic approach based on an iterated local search where the perturbation is performed by means of a destroy-and-repair method. Two alternative repair operators, one simple and fast and the other based on a mathematical programming model, are proposed, which give rise to two variants of the heuristic. The mathematical formulation is used to find the optimal solution on instances with up to 20 customers, built from benchmark instances for the classical TSP. Comparison with optimal solutions shows that both algorithms provide high-quality solutions. Tests are also made on larger instances to compare the performance of the two variants of the heuristic.

© 2018 Elsevier Ltd. All rights reserved.

1. Introduction

A common trait of the classical Vehicle Routing Problems (VRP) is the assumption that the goods to be distributed are available at the depot when the distribution starts. This implies that all vehicle routes may start immediately to distribute goods to customers. However, there are different settings in which this assumption is not satisfied, i.e., goods are not all available at the depot when the distribution starts and arrive at the depot over time. In this case, vehicles need to wait at the depot for the goods to arrive before distributing them. One example is related to city logistics and city distribution centers. Such an application has been studied in Cattaruzza et al. (2016a) where the authors introduce the Multi-Trip Vehicle Routing Problem with Time Windows and Release Dates (MTVRPTW-R) in which the goods that have to be distributed arrive at the depot over time, i.e., they become available when the distribution has already started. This poses the additional question of whether it is better to wait for additional goods to arrive and have a better loaded vehicle, or to start a route with the currently available goods. The arrival time at the depot of the goods to be delivered to a customer is called its release date. Another example arises in same day delivery problems related to e-commerce logistics. In this case, customer orders arrive online when the distribution (of previously received orders) has already started. Thus, the newly received orders have to be integrated in the distribution plan by designing vehicle routes that perform multiple trips, i.e., vehicles return to the depot multiple times in order to pickup the newly arrived orders that need to be distributed.

The focus of this paper is to study a routing problem arising in the applications mentioned above. In particular, we consider the TSP with release date and completion time minimization (TSP-rd(time)), that is the problem where each customer is associated with a release date and a single uncapacitated vehicle is allowed to perform multiple trips during the time horizon (say, the day), one after the other. No restriction is imposed on the maximum time taken by the vehicle to serve all customers and the objective

* Corresponding author.
E-mail addresses: claudia.archetti@unibs.it (C. Archetti), feillet@emse.fr (D. Feillet), andrea.mor@unibs.it (A. Mor), grazia.speranza@unibs.it (M.G. Speranza).

https://doi.org/10.1016/j.cor.2018.05.001
0305-0548/© 2018 Elsevier Ltd. All rights reserved.

is to minimize the completion time of the service, that is the time at which the vehicle is back to the depot and has served all customers. The completion time is given by the sum of the traveling time and waiting time. We consider the static problem where all information about the customers and the associated release dates are known in advance. As the TSP-rd(time) is relatively new in the literature, we believe that a deeper understanding of the static and deterministic version of the problem can be useful when solving the problem where release dates are characterized by dynamicity and uncertainty.

The contributions of this paper are summarized as follows. We first devise some properties of the problem and describe an approximation algorithm derived from Christofides approximation algorithm for the TSP. Then, we propose a mathematical programming formulation and present a heuristic approach based on an iterated local search where the perturbation is performed by means of a destroy-and-repair operator. Two alternative repair operators are proposed, one simple and fast and the other based on a mathematical programming model, which gives rise to two variants of the proposed heuristic. Ad hoc neighborhoods for the local search are introduced which are based on the characteristics of the problem. The mathematical programming formulation is used to find the optimal solution on instances with up to 20 customers built from benchmark instances for the classical TSP. On all but one instances the heuristic with the simple repair operator finds the optimal solution.

The paper is organized as follows. In Section 2 we review the literature related to similar problems. In Section 3 the TSP-rd(time) is defined and the properties and the mathematical programming formulation are presented. The heuristic is described in Section 4, whereas the computational experiments are presented in Section 5. Finally, conclusions are drawn in Section 6.

2. Literature review

To the best of our knowledge, Cattaruzza et al. (2016a) is the first work where the concept of release date is introduced in a routing setting. The problem studied is a multi-vehicle routing problem with time windows. The authors propose a hybrid genetic algorithm to solve the problem and test it on instances generated from Solomon's instances for the VRP with time windows (Solomon, 1987). The vehicle routing problem with release dates with a single uncapacitated vehicle is introduced in Archetti et al. (2015). The authors call this problem the Travelling Salesman Problem with release dates (TSP-rd). Two variants of the TSP-rd are proposed: one takes into consideration a deadline for completing the distribution and minimizes the total traveling time, the other one has no deadline and seeks the minimization of the time needed to complete the distribution. The former is referred to as TSP-rd(distance) and the latter as TSP-rd(time). The complexity of the two variants is analyzed for special topologies of the graph representing the distribution network, i.e., a line, modelling a distribution along a road, and a star, modelling the situation where the depot is the center of the distribution area. Further complexity analysis is carried out in Reyes et al. (2018), where, in addition to a distribution deadline, a service guarantee is considered, enforcing a maximum delay between the release date and the delivery to the customer. Shelbourne et al. (2017) consider the VRP with release and due dates, where the due date is the time by which the order should be delivered to the customer. The authors minimize a convex combination of operational costs and customer service level, measured by the total traveled distance and the total weighted delivery tardiness and no waiting time is considered, contrary to what happens in the TSP-rd(time). The authors present path relinking algorithm for the problem. The algorithm relies, among other things, on a parameter penalizing the solution that are infeasible with respect to the capacity constraint which makes it not suited for the TSP-rd(time), where such constraint is not considered.

As mentioned in Section 1, the vehicle routing problem with release dates finds applications in the context of the same-day delivery service. Recent contributions that study the same day delivery problem are Klapp et al. (2016), Klapp et al. (2018), and Voccia et al. (2017). In Klapp et al. (2016) the authors study the dynamic dispatch waves problem (DDWP) of a single vehicle on a line, where the problem is to decide whether to dispatch a vehicle to serve known customers or to wait for potential requests that may arrive later, with thhe objective to minimize expected vehicle operating costs and penalties for unserved requests. The study is extended to the case of a general network in Klapp et al. (2018). In Voccia et al. (2017) the same day delivery problem is investigated. A fleet of vehicles is used to serve requests characterized by time windows or a delivery deadline. Future requests are unknown but probabilistic information is available. The authors identify the circumstances that make waiting at the depot beneficial to maximize the number of requests that are served on time.

Finally, a problem which is strictly related to the one analyzed in this paper is the Multi-Trip Vehicle Routing Problem (MTVRP), where each vehicle may perform multiple trips and, thus, visit the depot multiple times. See Cattaruzza et al. (2016b) for a recent survey. The need of visiting the depot multiple times may come from the fact that routes need to have a short duration (see Azi et al., 2007 for potential applications) or because of capacity constraints. The problem with a single vehicle performing multiple routes within one workday is investigated in Azi et al. (2007), where customer requests with time windows are considered. The authors propose an exact algorithm for the problem. The work is extended in Azi et al. (2010, 2014) to consider multiple vehicles. An exact algorithm is designed in Azi et al. (2010) while an adaptive large neighborhood search is proposed in Azi et al. (2014).

In this paper we focus on the TSP-rd(time) which is introduced in Archetti et al. (2015). We propose the first mathematical formulation and solution approach for the problem. The TSP-rd(time) differs from the problem analyzed in Azi et al. (2007) as it considers release dates and does not consider duration constraints and time windows for the customers. It also differs from Klapp et al. (2016), Klapp et al. (2018), and Voccia et al. (2017) where a dynamic problem is studied and decision epochs are defined a priori, contrary to what happens in TSP-rd(time).

3. The Traveling Salesman Problem with release dates and completion time minimization

The TSP-rd(time) is defined as follows. Let $G = (V, A)$ be a complete graph. A traveling time and a traveling distance are associated with each arc $(i, j) \in A$. 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 can either arrive at time r_i, then $r_i > 0$, or be at the depot at the beginning of the distribution, e.g., because they arrived overnight, then $r_i = 0$. A single vehicle is allowed to perform a sequence of trips. Capacity constraints are not considered. The objective is to minimize the completion time, that is, the total traveling time plus the waiting time at the depot, and serve all customers. The waiting time is due to the fact that the vehicle has to wait at the depot until the latest release date of the customers it is going to serve in the next route. Without loss of generality, we assume that customers are ordered in non-decreasing order of their release dates, i.e., $r_i \leq r_j$ for $i < j, i, j = 1, \ldots, n$. Given a solution to the TSP-rd(time), we call

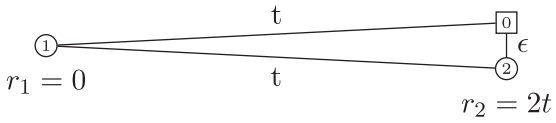

Fig. 1. Example.

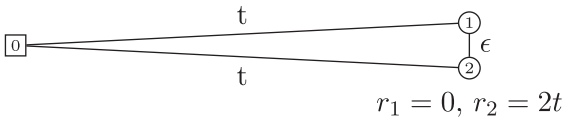

Fig. 2. Example.

route a trip starting and ending at the depot and not visiting the depot in between.

3.1. Properties

In this section some properties of the problem are presented which are used to enhance the mathematical formulation (Properties 1 and 2), to define the structure of a solution in the heuristic algorithm (Property 1) and to build an initial solution for the heuristic algorithm (Property 3). Property 4 shows a link between the optimal solution of the TSP-rd(time) and the optimal solution of the TSP. In addition, an approximation algorithm is presented in Section 3.2.

We adopt the following notation. We let $t(S)$ denote the completion time and $d(S)$ the traveled distance of solution S. S^* denotes the optimal solution of the TSP-rd(time) and d_{TSP} the value of the optimal solution of the Traveling Salesman Problem (TSP) defined on graph G with t_{ij} as the cost of traversing an arc.

The TSP-rd(time) is NP-hard as it has the TSP as special case when all release dates are zero. Furthermore, the problem always admits feasible solutions. In particular, the solution that waits until time r_n and executes an optimal TSP tour is feasible and gives an upper bound $r_n + d_{TSP}$ on the value of the solution.

Properties 1 and 2 give some simple characteristics satisfied by at least one optimal solution. Property 3 presents a lower bound on the value $t(S^*)$ of the optimal solution. Property 4 shows that release dates may have a huge impact on the total distance traveled.

Property 1. *There exists an optimal solution with no waiting time after the departure of the first route.*

This property is derived from Property 4.2 in Klapp et al. (2016) and thus we omit the proof.

Property 2. *There exists an optimal solution with exactly one route starting not earlier than time r_n.*

Proof. First, at least one route starts not earlier than r_n in every solution because customer n has to be served. Assume that several routes start not earlier than r_n in an optimal solution. Then, all the products are available when the first of these routes starts. Thus, they can be replaced by a single route restricted to the set of customers served in these routes without incrementing the traveling cost. □

Properties 1 and 2 will be used to reinforce the formulation in Section 3.3 and to define a solution in the heuristic algorithm.

Property 3. *Inequality $r_n + d_{TSP} \le 2 \times t(S^*)$ holds and the ratio is tight.*

Proof. $t(S^*) \ge r_n$ and $t(S^*) \ge d_{TSP}$, which proves the inequality. We now demonstrate that the ratio of 2 is tight. Consider the example shown in Fig. 1, where $n = 2$. On this example, $r_n = 2t$ and $d_{TSP} = 2t + \epsilon$. Solution S^* is the solution starting at time 0 with a route visiting customer 1 and continuing with a route visiting customer 2. Its completion time is $t(S^*) = 2t + 2\epsilon$. Setting ϵ to a small value, the ratio $\frac{r_n + d_{TSP}}{t(S^*)}$ can be as close as we want to 2, thus showing that the ratio is tight for $n = 2$. The example can be easily generalized to n customers. Set $t_{1i} = t_{i1} = t$ for $i \in V\backslash\{1\}$, $t_{ij} = t_{ji} = \epsilon$ for $i, j \in V\backslash\{1\}$, $r_1 = 0$, $r_i = 2t$ for $i \in V\backslash\{1\}$. Again $r_n = 2t$ and $d_{TSP} = 2t + (n-1)\epsilon$. Solution S^* serves customer 1 in a first route and all remaining customers in a second route: $t(S^*) = 2t + n\epsilon$. When ϵ tends to 0, $\frac{r_n + d_{TSP}}{t(S^*)}$ tends to 2. □

Property 3 provides a lower bound for the value $t(S^*)$ but also a performance guarantee for the initial solution of the proposed heuristic, introduced in Section 4.1.

Property 4. *Inequality $d(S^*) \le n \times d_{TSP}$ holds and the ratio is tight.*

Proof. Solution S^* has at most n routes and, because of the triangle inequality, the distance traversed by any route is not larger than d_{TSP}. Consequently, $d(S^*) \le n \times d_{TSP}$, which permits to conclude that the inequality holds. Let us now show that the ratio is tight, by considering the example shown in Fig. 2 where $n = 2$.

On this example, $d_{TSP} = 2t + \epsilon$. Solution S^* is formed by a first route starting at time 0 and visiting customer 1 and a second route starting at time $2t$ and visiting customer 2, with value $t(S^*) = 4t$. The traveled distance in solution S^* is $d(S^*) = 4t$. Setting ϵ to a small value, the ratio $\frac{d(S^*)}{d_{TSP}}$ can be as close as we want to 2, thus showing that the ratio is tight for $n = 2$. The example can be easily generalized to n customers. Set $t_{0i} = t_{i0} = t$ for $i \in V\backslash\{0\}$, $t_{ij} = t_{ji} = \epsilon$ for $i, j \in V\backslash\{0\}$ and $r_i = 2(i-1)t$ for $i \in V\backslash\{0\}$. Now, $d_{TSP} = 2t + (n-1)\epsilon$. Solution S^* is given by n successive single-customer routes starting at time 0, with completion time $t(S^*) = 2nt$ and traveled distance $d(S^*) = 2nt$. When ϵ tends to 0, $\frac{d(S^*)}{d_{TSP}}$ tends to n. □

The proof of Property 4 is helpful in understanding the impact of the release dates on the total traveled distance $d(S^*)$. In fact, release dates may have a big impact on the value of the optimal solution of the problem, as it is shown in the analysis of the computational results (see Section 5.2.3).

3.2. Approximation algorithm

The following approximation algorithm for the TSP-rd(time) relies on the well-known Christofides $\frac{3}{2}$-approximation algorithm for the TSP.

Theorem 1. *Algorithm 1 has a performance guarantee of 2.5 for the TSP-rd(time) and the ratio is tight. Its running time is $O(n^3)$.*

Proof. $t(S^*) \ge r_n$ and $t(S^*) \ge d_{TSP}$. Thus, $r_n + \frac{3}{2} d_{TSP} \le 2.5\, t(S^*)$. Furthermore, we know that the completion time of the solution given by Algorithm 1 is not larger than $r_n + \frac{3}{2} d_{TSP}$, which proves the theorem. We now show that the ratio is tight. Consider a graph with $2k + 2$ vertices. The first $2k + 1$ are arranged as shown in Fig. 3, with k vertices at the upper level and $k + 1$ vertices at the bottom level (where one vertex, vertex 0, is the depot). The final vertex, vertex n, overlaid to vertex 0, i.e., $t_{0n} = t_{n0} = 0$. The solid edges have distance one and the dashed ones have distance $1 + \epsilon$. Let the release dates be $r_i = 0$, $i = 1, \ldots, 2k + 1$ and $r_n = d_{TSP}$. In addition,

Algorithm 1 Approximation algorithm for TSP-rd(time).

1: Relax release dates and apply Christofides $\frac{3}{2}$-approximation algorithm
2: Start the TSP tour obtained at time r_n

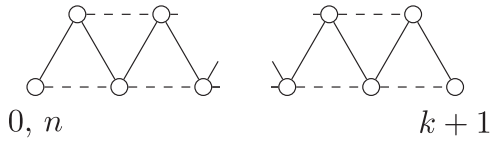

Fig. 3. Example.

let there be an edge connecting vertex 0 and vertex $k+1$ at cost $k(1+\epsilon)$. The optimal tour for the TSP on this graph is composed by the dashed edges and the rightmost and leftmost solid edges. It has cost $d_{TSP} = (2k-1)(1+\epsilon)+2$. The solution of the Christofides algorithm consists of the solid edges, i.e., the minimum spanning tree of the graph, and the edge connecting vertices 0 and $k+1$. It has cost $d_{CH} = 3k+\epsilon$.

The optimal solution of the TSP-rd(time) is composed of one route starting at time 0 and serving all customers except customer n followed by a second route serving customer n. This solution has $t(S^*) = (2k-1)(1+\epsilon)+2$. The solution of Algorithm 1 has $t(S_{Alg.1}) = (2k-1)(1+\epsilon)+2+3k+\epsilon$. Setting ϵ to a small value, as k increases the ratio $\frac{t(S^*)}{t(S_{Alg.1})} \approx \frac{5}{2}$. The running time of Algorithm 1 corresponds to the one of Christofides algorithm which is $O(n^3)$. □

3.3. Mathematical formulation

We propose a 3-index formulation for the TSP-rd(time), with flow variables indexed by the route in which the edge is traversed. We thus introduce the set of routes $K = \{1, \ldots, |K|\}$, where $|K|$ is an upper bound on the optimal number of routes. A simple upper bound is n, but it leads to a weak formulation with a large number of variables. In the following section, we present a way to determine a heuristic value for $|K|$. Also, empty routes are allowed, meaning that there may be a subset of the routes in K visiting the depot only. Thus, the number of routes effectively used to serve customers is given by $|K|$ minus the number of empty routes.

The formulation relies on the following decision variables:

- $x_{ij}^k = \begin{cases} 1 \text{ if route } k \in K \text{ travels through edge } (i,j) \in A, \\ 0 \text{ otherwise,} \end{cases}$
- $y_i^k = \begin{cases} 1 \text{ if route } k \in K \text{ visits vertex } i \in N, \\ 0 \text{ otherwise,} \end{cases}$
- τ_{start}^k = the starting time of route $k \in K$,
- τ_{end}^k = the ending time of route $k \in K$.

In addition, flow variables u_{ij}^k are added to enforce subtour elimination. Note that $x_{00}^k = 1$ means that the route k is an empty route as it visits no customers.

The resulting model is:

$$\text{minimize } \tau_{end}^K \tag{1}$$

subject to

$$\sum_{k \in K} y_i^k = 1 \quad i \in N, \tag{2}$$

$$\sum_{j \in V} x_{ij}^k = \sum_{j \in V} x_{ji}^k = y_i^k \quad i \in V, k \in K, \tag{3}$$

$$\sum_{j \in V} u_{ji}^k - \sum_{j \in V} u_{ij}^k = y_i^k \quad i \in N, k \in K, \tag{4}$$

$$u_{ij}^k \le (n-1) x_{ij}^k \quad (i,j) \in A, k \in K, \tag{5}$$

$$\tau_{end}^k = \tau_{start}^k + \sum_{(i,j) \in A} t_{ij} x_{ij}^k \quad k \in K, \tag{6}$$

$$\tau_{end}^k \le \tau_{start}^{k+1} \quad k \in K \setminus \{|K|\}, \tag{7}$$

$$\tau_{start}^k \ge r_i y_i^k \quad k \in K, i \in N, \tag{8}$$

$$\tau_{end}^k = \tau_{start}^{k+1} \quad k \in K \setminus \{|K|\}, \tag{9}$$

$$\tau_{start}^k \le r_n \quad k \in K \setminus \{|K|\}, \tag{10}$$

$$x_{ij}^k \le 1 - x_{00}^k \quad (i,j) \in A, k \in K \setminus \{|K|\}, \tag{11}$$

$$x_{00}^k \ge x_{00}^{k+1} \quad k \in K \setminus \{|K|\}, \tag{12}$$

$$x_{ij}^k \in \{0,1\} \quad (i,j) \in A, k \in K, \tag{13}$$

$$y_i^k \in \{0,1\} \quad i \in V, k \in K, \tag{14}$$

$$\tau_{start}^k, \tau_{end}^k \ge 0 \quad k \in K, \tag{15}$$

$$u_{ij}^k \ge 0 \quad (i,j) \in A, k \in K. \tag{16}$$

The objective function (1) minimizes the ending time of the last route. This is equivalent to the minimization of the total completion time computed as the total traveling time plus the waiting time at the depot. Constraint (2) ensure the visit of all customers. Constraints (3)–(5) impose that each route is a circuit connected to the depot. In particular, constraint (5) generate a flow that decreases while the vehicle visits customers, which prevents subtours. This set of constraints has been first proposed in Gavish and Graves (1978) and its performance has been assessed in Öncan et al. (2009). Constraints (6) and (7) set the relationship between time variables τ_{start}^k and τ_{end}^k. Constraint (8) establish that a route cannot start before the last release date of the customers served in it. Constraints (9) and (10) are added to reinforce the formulation. Constraint (9) follow from Property 1. Constraints (10) are deduced from Property 2. Constraints (11) and (12) are symmetry breaking constraints. In particular, constraint (11) set $x_{00}^k = 1$ if route k is empty, $x_{00}^k = 0$ otherwise, and constraint (12) impose that all the empty routes, if any, must precede all the non-empty routes, i.e., if route k is empty then all routes k' must be empty, with $k' = 1, \ldots, k-1$.

4. An iterated local search algorithm for the TSP-rd(time)

In this section we describe the heuristic algorithm we devised for the solution of the TSP-rd(time). It is an Iterated Local Search (ILS) algorithm which combines a local search phase with a perturbation phase (see Lourenço et al., 2010 for a detailed description of the ILS scheme). In particular, the perturbation phase is performed through a destroy-and-repair procedure.

The general scheme of the approach is presented in Algorithm 2. The idea is to build an initial solution through a construction phase and to iteratively find new solutions through destroy-and-repair (DR) followed by local search (LS). DR takes as input the current solution S as well as a parameter α which determines the size of the perturbation. The solution S' found by DR is given as input to LS. Finally, S_{best} represents the best solution found by the algorithm.

Algorithm 2 The ILS algorithm for the TSP-rd(time).

```
S ← InitialSolution
S_best ← S
α ← InitializeDRParameter
repeat
    S' ← DR(S, α)
    if a solution is found then
        S ← LS(S')
        S_best ←UpdateBest(S_best, S)
        α ←UpdateDRParameter(α, S_best, S)
    end if
until theending conditions are satisfied
return S_best
```

Function UpdateBest(S_{best}, S) updates the best solution found in case $t(S) < t(S_{best})$ while the other functions are described in Sections 4.1–4.3.

The ILS makes use of an estimation of the unknown cardinality of the set of routes K. We call C_K this value:

$$C_K = 1 + \left\lfloor \frac{r_n \times (n-1)}{\sum_{i \in N}(t_{0i} + t_{i0})} \right\rfloor. \tag{17}$$

The rationale behind this formula is that the number of routes is 1 when $r_n = 0$ while the number of routes could be n when the maximal release date is equal to $\sum_{i \in N}(t_{0i} + t_{i0})$. Indeed, in the latter case, the optimal solution would have n routes if the values of release dates are $r_j = \sum_{i=1}^{j}(t_{0i} + t_{i0})$. Note that C_K is not a valid upper bound on the number of routes, as shown by the following example. Let $\sum_{i \in N}(t_{0i} + t_{i0}) > r_n \times (n-1)$, $r_1 = 0$, $t_{01} = t_{10} = 1$, and $r_2 = 2$; therefore $r_i \geq 2$ for $i > 2$. It follows that $C_K = 1$ while the upper bound on the number of routes is at least 2. Despite this and considering the heuristic nature of the algorithms, the C_K value has been kept instead of the upper bound equal to n mentioned in the previous section which proved to be extremely weak.

A solution S is composed by a vector of C_K routes. Because of Property 1, there is no waiting time between populated routes and all empty routes occupy the first positions of the vector, while populated routes are all at the end.

4.1. Initial solution

The initial solution is built by solving the TSP on the complete graph G with the Lin-Kernighan (LK) heuristic, described in Lin and Kernighan (1973), in the implementation provided by Helsgaun (2000). The starting time of this single route is then set to r_n. This solution is feasible and has a completion time greater than or equal to $r_n + d_{TSP}$.

4.2. Destroy-and-repair

The destroy-and-repair (DR) procedure works as follows. Given a solution S and an integer α, the DR first applies a destroy operator that removes a set $\tilde{N}$ of α customers from S. A repair operator is used to reassign customers from $\tilde{N}$ to a route, either empty or populated, in S. A final step optimizes each route in S.

The parameter α is initialized to a lower bound $\alpha_{\min}$ and has upper bound equal to $\alpha_{\max}$. The initial value of α is set to $\alpha_{\min}$ and is updated after each successful destroy-and-repair call. If an improving solution is found by the local search, the value is reset to $\alpha_{\min}$, otherwise α is increased by one. If the new value exceeds the upper bound $\alpha_{\max}$, then it is set to $\alpha_{\min}$. The updating of α is described in Algorithm 3.

In order to choose the customers that are removed, we first rank all customers on the basis of one of the following two criteria:

Algorithm 3 UpdateDRParameter(α, S_{best}, S).

```
if t(S) = t(S_best) then
    α = α_min
else
    α = α + 1
    if α > α_max then
        α = α_min
    end if
end if
```

- the starting time saving obtained by removing a customer from the corresponding route, i.e., how much the starting time of the route decreases if the vertex is removed from it;
- the classical detour saving, without taking into account changes in the starting time of the route.

Each time the DR is called, one of these two criteria is chosen at random. Then, α customers are removed as follows: customers are considered sequentially in the order of the ranking and each customer is removed with a probability equal to 50% until α customers are removed.

The assignment of the vertices from $\tilde{N}$ to the routes of S is accomplished by a repair operator. Two different operators are used which give rise to two variants of the heuristic: the first one is based on the mathematical formulation proposed in Section 3.3, called MIP hereafter, while the second one is a random repair approach.

4.2.1. MIP repair

Given $\tilde{N}$, a mixed integer linear program is obtained from MIP with the following two changes. First, we force the clustering of customers as in S, except for the removed customers. A cluster is given by the customers served in the same route. In order to do that, the following constraints are added to MIP:

$$\sum_{i \in N_k \setminus \tilde{N}} y_i^k = |N_k \setminus \tilde{N}| y_t^k \qquad k \in K, t \in N_k \setminus \tilde{N}, \tag{18}$$

where N_k is the set of customers visited in route k in S. Constraint (18) guarantee to preserve the clustering of customers of the current solution (except for the removed customers) without fixing the index k of the routes. In this way we can anticipate or postpone clusters with respect to S. Second, we relax the integrality condition on the edge variables. We call this problem *modified MIP*.

A maximum time limit is given to the solution of the modified MIP. Thus, the MIP repair operator is successful if a feasible solution is found within the time limit. The solution of the modified MIP provides a clustering for the customers. The routes are computed by applying the LK heuristic on each cluster and a solution is then obtained by computing the earliest possible starting time for each route and by scheduling the routes in the increasing order of these values.

4.2.2. Random repair

This repair operator randomly assigns each vertex in $\tilde{N}$ to a cluster N_k, $k \in K$, different from the cluster of the vertex in S. As in the MIP repair, a solution is then built by applying the LK heuristic on each cluster, sorting the obtained routes by the largest release date and computing the earliest possible starting time for each route. The aim of the random repair operator is to quickly find a feasible solution. Note that the solution found by the random repair operator is always feasible, contrary to what may happen with the MIP repair where the modified MIP may run out of time without finding any feasible solution.

We call MathTSPrd the algorithm originating from the scheme described in Algorithm 2 when the MIP repair operator is used, HeuTSPrd the one where the random repair operator is used.

4.3. Local search

After each DR move or, in the case of the MIP repair, each time a feasible solution is found, the local search (LS) is applied. The proposed LS is defined by seeking the first improvement in a neighborhood of four possible moves:

- *Vertex relocation*: a vertex is removed from its current route and relocated in a different route, either empty or populated. The new vertex position in the route is found through cheapest insertion.
- *Depot insertion*: a depot visit is added to a route, splitting it in two different routes. This move is not applied in case the number of populated routes is equal to C_K.
- *Depot shift*: a visit to the depot between two routes is either moved forward or backward in the solution. Let r and r' be two consecutive routes and let us consider the shift of the depot visit between r and r'. When the depot visit is moved forward, the set of vertices visited at the beginning of r' and before the new position of the depot will be shifted to r. On the contrary, when the depot visit is moved backward, the set of vertices visited at the end of r and before the new position of the depot will be shifted to r'. Backward shifting is evaluated up to the case where route r visits one customer only. Similarly, forward shifting is evaluated up to the case where r' visits one customer only.
- *Depot removal*: a depot visit between two routes is removed, thus merging the two routes.

All moves are evaluated by the corresponding value of the objective function, i.e., the ending time of the last route, therefore potentially accounting for savings in both the travelled distance and the starting time of each route. A scheme of the LS is presented in Algorithm 4:

Algorithm 4 LS algorithm.

```
repeat
    Randomly sort all vertices (including the depot)
    for i = 0 to n do
        if the ith vertex is the depot then
            Evaluate the depot insertion, shift, and removal moves
        else
            Evaluate vertex relocation move
        end if
        if an improving move is found then
            Apply the improving move
        end if
    end for
until no improving solution is found
Apply LK to each populated route
```

Note that, while the vertex relocation move is quite standard in heuristics for routing problems, moves involving the depot are instead specific for the TSP-rd(time) as they have an impact on the starting time of each route.

5. Computational experiments

In this section computational results are presented for instances derived from benchmark TSP instances. In Section 5.1 we describe how the instances have been generated while computational results are presented in Section 5.2.

Table 1
The symmetric instances selected from the TSPLIB, by instance size.

50–100 nodes	101–150 nodes	151–250 nodes	251–500 nodes
eil51	eil101	pr152	gil262
berlin52	lin105	u159	pr264
st70	pr107	rat195	a280
eil76	pr124	d198	pr299
pr76	bier127	kroA200	lin318
rat99	ch130	kroB200	rd400
kroA100	pr136	ts225	fl417
kroB100	pr144	tsp225	pr439
kroC100	ch150	pr226	pcb442
kroD100	kroA150		d493
kroE100	kroB150		
rd100			

Table 2
Number of instances solved to optimality.

n	# optimal solutions
10	24
15	24
20	24
25	13
30	7

5.1. Instances

The instances have been derived from Solomon instances (Solomon, 1987) and from the "TSPLIB" library (Reinelt, 1991), and are characterized by three values:

- n: the cardinality of the vertex set (not including the depot);
- d_{TSP}: the optimal TSP value on the resulting graph;
- β: the width of the interval in which release dates are defined.

Solomon's instances are composed by 6 sets of instances: $C1$, $C2$, $R1$, $R2$, $RC1$ and $RC2$. All the instances in the same set have the same coordinates of the vertices and differ in time windows only. As we do not consider time windows, we kept one instance from each set. In addition, we discarded $R2$ and $RC2$ as they have the same coordinates as $R1$ and $RC1$, respectively. The four remaining problem sets allow us to assess the performance of the heuristic when vertices are spread according to different schemes.

Given an instance $\mathcal{I}$ from Solomon (1987), the instance $\mathcal{I}\_n\_\beta$ for the TSP-rd(time) is generated as follows:

- data from the Solomon's instance are truncated after $n+1$ vertices;
- the first vertex of the instance is set to be the depot;
- let i be the index of the vertices in the instance (the depot receives index $i=0$, the first vertex index $i=1$ and so on). Its release date is defined as $r_i = \lfloor \beta \times d_{TSP} \times Y \rceil$, where $Y \sim U(0, 1)$, therefore assigning to each customer a uniformly random integral release date in $[0, \beta \times d_{TSP}]$.

Instances have been generated with $\beta \in \{0.5, 1, \ldots, 3\}$ resulting in 24 instances for each value of n.

From the TSPLIB we have chosen symmetric instances with less than 500 vertices, explicit vertex coordinates, edge weights computed in the two dimensional Euclidean space, and with no constraints on the TSP solution. The resulting selection comprises a total of 42 instances, reported in Table 1 according to the instance size.

We also tested the ILS algorithm on asymmetric instances in order to check whether the behavior of the two variants of the algorithm is affected by the asymmetry of the distance matrix. The

Table 3
Results on the $n = 20$ instances for different values of the MILP time limit.

	MILP: n/50 = 0.4 s		MILP: n/10 = 2 s		MILP: n/5 = 4 s		MILP: n/2 = 10 s	
β	Avg. % Best	Avg. # MILP	Avg. % Best	Avg. # MILP	Avg. % Best	Avg. # MILP	Avg. % Best	Avg. # MILP
0.5	0	1,017.25	0	974.00	0	974.75	0	974.25
1	0	1,053.00	0	560.00	0	517.25	0	505.75
1.5	0.10	927.75	0	378.25	0.10	275.50	0.10	228.50
2	0.09	879.75	0	322.25	0.18	215.75	0.18	148.75
2.5	0	843.00	0	295.50	0	193.25	0.04	124.00
3	0	796.25	0.03	285.75	0	178.00	0	111.25
Avg.	0.03		0.01		0.04		0.05	

Table 4
Results on the $n = 50$ instances for different values of the MILP time limit.

	MILP: n/50 = 1 s		MILP: n/10 = 5 s		MILP: n/5 = 10 s		MILP: n/2 = 25 s	
β	Avg. %Best	Avg. # MILP	Avg. % Best	Avg. # MILP	Avg. % Best	Avg. # MILP	Avg. % Best	Avg. # MILP
0.5	0.08	365.00	0.13	126.00	0.24	69.25	0.20	54.00
1	0.03	254.25	0.92	89.75	0.73	58.50	0.87	30.00
1.5	0.17	266.00	0.53	90.50	0.50	51.00	1.13	25.50
2	0.47	245.00	0.34	95.25	1.26	50.50	1.55	22.50
2.5	1.97	153.25	0.18	94.50	0.20	53.00	1.02	22.25
3	3.35	12.75	0.26	94.50	0.08	52.50	0.30	23.25
Avg.	1.01		0.39		0.50		0.85	

Table 5
Results on the $n = 100$ instances for different values of the MILP time limit.

	MILP: n/50 = 2 s		MILP: n/10 = 10 s		MILP: n/5 = 20 s		MILP: n/2 = 50 s	
β	Avg. % Best	Avg. # MILP	Avg. % Best	Avg. # MILP	Avg. % Best	Avg. # MILP	Avg. % Best	Avg. # MILP
0.5	1.33	19.50	0.79	30.00	0.48	26.00	1.59	15.50
1	3.23	1.75	1.03	7.00	1.28	10.00	0.14	9.00
1.5	4.72	1.00	1.37	13.25	0.02	12.00	0.23	8.25
2	2.35	1.00	0.35	11.75	1.22	12.50	1.18	8.25
2.5	4.47	1.00	1.19	8.50	0	18.00	0.93	10.75
3	7.00	0.75	1.21	4.00	0	15.75	0.94	10.25
Avg.	3.85		0.99		0.50		0.84	

Table 6
Results on the $n = 50$ instances for different values of ρ. MILP time limit is set to 5 s for all runs.

	$\rho = n/2$		$\rho = n/4$		$\rho = n/8$	
β	Avg. % Best	Avg. # MILP	Avg. % Best	Avg. # MILP	Avg. % Best	Avg. # MILP
0.5	0	127.75	0	158.00	0.25	175.00
1	0.87	93.25	0.19	102.25	0.40	109.00
1.5	0.45	89.00	0.76	96.50	0.21	97.25
2	0.16	95.25	0.17	96.50	0.47	99.25
2.5	0.50	94.50	0.50	96.75	0.11	97.25
3	0.55	96.00	0.30	96.00	0.02	100.75
Avg.	0.42		0.32		0.25	

Table 7
Results on the $n = 100$ instances for different values of ρ. MILP time limit is set to 20 s for all runs.

	$\rho = n/2$		$\rho = n/4$		$\rho = n/8$	
β	Avg. % Best	Avg. # MILP	Avg. % Best	Avg. # MILP	Avg. % Best	Avg. # MILP
0.5	1.06	23.50	1.20	23.50	0.13	26.50
1	0	9.50	0	9.75	0	10.50
1.5	0	12.00	0	12.00	0	12.75
2	0	12.75	0	12.50	0	12.00
2.5	0	18.00	0.33	17.75	0	17.00
3	0.28	15.575	0.20	15.75	0.58	14.50
Avg.	0.22		0.29		0.12	

Table 8
Comparison with optimal solutions for instances with $n = 10$.

		Optimal	Initial solution		MathTSPrd		HeuTSPrd	
β	Instance	Value	Value	% Exact	Value	% Exact	Value	% Exact
0.5	C101	79	79	0	79	0	79	0
	C201	186	189	1.61	186	0	186	0
	R101	217	244	12.44	217	0	217	0
	RC101	195	195	0	195	0	195	0
1	C101	101	102	0.99	101	0	101	0
	C201	221	245	10.86	221	0	221	0
	R101	261	316	21.07	261	0	261	0
	RC101	241	252	4.56	241	0	241	0
1.5	C101	113	125	10.62	113	0	113	0
	C201	250	300	20.00	250	0	250	0
	R101	323	388	20.12	323	0	323	0
	RC101	289	309	6.82	289	0	289	0
2	C101	134	149	11.19	134	0	134	0
	C201	297	356	19.87	297	0	297	0
	R101	373	460	23.32	373	0	373	0
	RC101	347	367	5.76	347	0	347	0
2.5	C101	157	172	9.55	157	0	157	0
	C201	338	412	21.89	338	0	338	0
	R101	432	532	23.15	432	0	432	0
	RC101	395	424	7.34	395	0	395	0
3	C101	180	195	8.33	180	0	180	0
	C201	386	467	20.89	386	0	386	0
	R101	495	604	22.02	495	0	495	0
	RC101	444	481	8.33	444	0	444	0
Avg.				12.12		0		0

Table 9
Comparison with optimal solutions for instances with $n = 15$.

		Optimal	Initial solution		MathTSPrd		HeuTSPrd	
β	Instance	Value	Value	% Exact	Value	% Exact	Value	% Exact
0.5	C101	145	145	0	145	0	145	0
	C201	228	241	5.70	228	0	228	0
	R101	288	327	13.54	288	0	288	0
	RC101	220	220	0	220	0	220	0
1	C101	176	189	7.39	176	0	176	0
	C201	271	314	15.39	271	0	271	0
	R101	361	426	18.01	364	0.83	361	0
	RC101	266	286	7.52	266	0	266	0
1.5	C101	217	7.37	9.23	217	0	217	0
	C201	327	387	18.35	327	0	327	0
	R101	427	524	22.72	427	0	427	0
	RC101	330	353	6.97	330	0	330	0
2	C101	261	277	6.13	261	0	261	0
	C201	375	460	22.67	375	0	375	0
	R101	509	623	22.40	510	0.20	510	0.20
	RC101	387	419	8.27	387	0	387	0
2.5	C101	304	321	5.59	304	0	304	0
	C201	446	533	19.51	446	0	446	0
	R101	572	722	26.22	572	0	572	0
	RC101	439	485	10.48	439	0	439	0
3	C101	347	365	5.19	347	0	347	0
	C201	517	605	17.02	517	0	517	0
	R101	663	820	23.68	663	0	663	0
	RC101	494	552	11.74	494	0	494	0
Avg.				12.60		0.04		0.01

following five asymmetric instances have been chosen: `ftv33`, `ft53`, `ftv70`, `kro124p`, and `rbg403`.

As for Solomon instances, the first vertex is set for the depot. Then, the same procedure is used to generate the release dates, with $\beta \in \{0.5, 1, \ldots, 3\}$. Thus, we obtain 252 symmetric and 30 asymmetric instances in total.

All the instances can be downloaded at http://or-brescia.unibs.it/instances.

5.2. Results

This section is organized as follows. In the first part we identify the largest instances that can be solved to optimality. Secondly, we focus on the tuning of the parameters of the MathTSPrd, investigating how the time limit allowed to solve each modified MIP and the range of α affect the quality of the solution. Finally, the performance of the ILS is analyzed: firstly on small instances, in comparison with the optimal solution found by solving the formulation in

Table 10
Comparison with optimal solutions for instances with $n = 20$.

		Optimal		Initial solution		MathTSPrd			HeuTSPrd		
β	Instance	Value	TI	Value	% Exact	Value	% Exact	TI	Value	% Exact	TI
0.5	C101	177	0	177	0	177	0	0	177	0	0
	C201	247	5	258	4.45	247	0	13	247	0	115
	R101	326	3	376	15.34	326	0	7	326	0	118
	RC101	305	3	313	2.62	305	0	1	305	0	129
1	C101	212	6	230	8.49	212	0	1	212	0	160
	C201	285	11	336	17.89	285	0	1	285	0	153
	R101	409	42	484	18.34	409	0	110	409	0	139
	RC101	374	15	407	8.82	374	0	62	374	0	160
1.5	C101	260	39	284	9.23	260	0	2	260	0	172
	C201	349	74	414	18.62	349	0	6	349	0	169
	R101	475	129	596	25.47	479	0.84	536	475	0	216
	RC101	422	25	502	18.96	422	0	12	422	0	177
2	C101	306	67	337	10.13	306	0	83	306	0	191
	C201	402	180	492	22.39	402	0	9	402	0	174
	R101	568	871	708	24.65	571	0.53	214	571	0.53	174
	RC101	508	77	596	17.32	508	0	17	508	0	192
2.5	C101	356	271	391	9.83	356	0	21	356	0	201
	C201	478	403	570	19.25	478	0	57	478	0	193
	R101	648	333	820	26.54	648	0	141	648	0	180
	RC101	586	97	690	17.75	586	0	46	586	0	199
3	C101	406	1,426	444	9.36	406	0	5	406	0	199
	C201	551	1,801	648	17.60	551	0	5	551	0	195
	R101	750	185	932	24.27	750	0	67	750	0	166
	RC101	654	110	785	20.03	654	0	2	654	0	184
Avg.			257.21		15.31		0.06	59.08		0.02	164.83

Section 3.3 and secondly by comparing the two variants of the ILS on larger instances, generated both from Solomon's and the TSPLIB instances.

Note that, throughout this section, the ending conditions for Algorithm 2 are the maximum time allowed for each run and the maximum number of iterations without improvement. Unless otherwise specified, the maximum time is set to ten minutes and the maximum number of iterations is set to one thousand. For all the tested instances, the time required to build the initial solution for the ILS is below one second. Thus, it does not have an impact on the performance of the ILS.

The ILS was implemented in C++ and run using a 3.5 GHz Intel Xeon E5-1650v2 processor with 64GB of RAM, using CPLEX 12.6 as MILP solver, when necessary.

5.2.1. Largest instances solved to optimality

To determine the largest instance size for which CPLEX finds an optimal solution to formulation (1)–(16) within one hour, instances have been generated with $n \in \{10, 15, 20, 25, 30\}$ starting from Solomon instances. A total of 24 instances are generated for each size, given by the combination between the values of β, and the 4 Solomon instances (C101, C201, R101, RC101). As Table 2 shows, the maximum size for which the optimal solution is obtained within one hour is $n = 20$ customers. For $n = 25$ only 13 instances out of 24 are solved to optimality. For $n = 30$ the number of optimal solutions further decreases to 7.

5.2.2. Tuning of the parameters

The parameters to be tuned in the MathTSPrd are the time limit for each MIP repair and the range for the number α of customers to be removed by the DR. Note that the parameter α has also an impact on the HeuTSPrd. Thus, it was fine tuned on the MathTSPrd and the best value was kept for HeuTSPrd as well. The tuning has been carried out on the instances derived from Solomon with $n \in \{20, 50, 100\}$.

The impact of changes in the time limit parameter for each MIP repair solution in the MathTSPrd algorithm are reported in Table 3 for $n = 20$ instances, Table 4 for $n = 50$ instances and Table 5 for $n = 100$ instances. The results have been obtained with the bounds for α set to $\alpha_{\min} = \min\{5, \lfloor n/10 \rfloor\}$ and $\alpha_{\max} = \max\{\min\{10, n-1\}, \lfloor n/2 \rfloor\}$ and are reported for instances grouped by the value of β. The column "Avg. % Best" reports the average percentage gap with respect to the best solution found for each instance group. "Avg. # MILP" indicates the average number of modified MIPs successfully solved, i.e., the average number of modified MIPs for which a feasible solution has been found within the time limit. The best results have been found for the time limit equal to $n/10$ seconds when $n = 20$ and $n = 50$, corresponding to 2 and 5 s, respectively, and to $n/5 = 20$ s when $n = 100$. Note also that, while on instances with $n = 20$ the difference in the performance of MathTSPrd for different values of the MIP repair time limit is negligible, the value shows an increase when $n = 50$ and becomes remarkable when $n = 100$. Thus, the main message we may keep from these results is that the MIP repair needs more time to be effective on large instances.

We also investigated how the range of α, the number of vertices removed and inserted by the DR, affects the performance of the MathTSPrd algorithm. The value of $\alpha_{\min}$ has been fixed to $\alpha_{\min} = \min\{5, \lfloor n/10 \rfloor\}$ while $\alpha_{\max}$ has been set to $\alpha_{\max} = \max\{\min\{10, n-1\}, \lfloor \rho \rfloor\}$ and different values have been tested for ρ, i.e., $\rho \in \{n/2, n/4, n/8\}$. Table 6 presents the results for the $n = 50$ instances and Table 7 for the $n = 100$ instances. In both tables the MILP time limit chosen is the one that has shown the best results in the previous experiments, i.e., 5 s and 20 s, respectively. The results show an improving trend as ρ decreases hence the default value for the different runs reported hereafter has been fixed to $\rho = n/8$.

5.2.3. Comparison with optimal solutions

Tables 8–10 report the results of the comparison between the ILS and the optimal solution. In particular, we compare the optimal solution with the value of the initial solution (computed as a TSP tour starting at the latest release date), the value of the solution provided by the MathTSPrd and the value of the solution provided by HeuTSPrd. The comparison is made on instances with up to 20 customers. The column "% Exact" reports the percent-

Table 11
Growth of the average optimal value with respect to the value of β.

β	$n = 10$	$n = 15$	$n = 20$
0.5	100.00	100.00	100.00
1	121.71	121.91	121.95
1.5	144.02	147.67	143.61
2	170.01	173.89	170.23
2.5	195.27	199.89	197.33
3	222.30	229.40	225.29

age gap of the corresponding "Value" column to the exact solution value. The time limit for the modified MIP in the MIP repair operator is set to $n/10 = 2$ s. The tables shows that on these instances the two algorithms have a good performance with all the instances of size $n = 10$ solved to optimality and with no percentage gap above 1% in the larger instances. The HeuTSPrd shows a slight advantage, finding the optimal solution in all but one of the 24 instances in both the instances of size $n = 15$ and $n = 20$. The MathTSPrd finds the optimal solution in 22 instances of size $n = 15$ and in 22 instances of size $n = 20$. Table 10 also reports the running time, in seconds, required to find the incumbent solution by MathTSPrd and HeuTSPrd, respectively. The computing time, in seconds, is reported in the "TI" column. It can be observed that, while the MathTSPrd is shown to be the fastest on average, the HeuTSPrd has the least maximum time.

By looking at the results presented in Tables 8–10, it is also interesting to note the impact of release dates on the value of the optimal solution of the TSP-rd(time). To better highlight this impact, in Table 11 we report the average percentage increase in the value of the optimal solution of the TSP-rd(time) for different values of β with respect to the case with $\beta = 0.5$. Each cell reports the average increase over the four instances with the value of β of the corresponding row and the value of n of the corresponding column. We can notice that the more the release dates are spread out, the more the value of the optimal solution increases with a maximum increase of nearly 130% when $\beta = 3$. This is due to the fact that more routes are needed when release dates are more dispersed. In addition, the increase in the objective function seems to be not affected (or only slightly affected) by the size of the instance.

5.2.4. Comparison of the two variants of the heuristic

A comparison of the two variants has also been carried out on larger instances. Tables 12 and 13 presents the results for the Solomon instances with $n = 50$ and $n = 100$, respectively. The column "TI" identifies the time in seconds after which the incumbent solution has been found and the column "TI MathTSPrd" reports the time at which HeuTSPrd has found the first solution at least as good as the one found by MathTSPrd. In both the instances with $n = 50$ and $n = 100$ it can be observed that the MathTSPrd is outperformed by the HeuTSPrd in all but one instance. In the instances of size $n = 50$ the average percentage gap of the MathTSPrd from the HeuTSPrd is 0.86%. The best solutions of the MathTSPrd are found in an average time of 265 s, while the HeuTSPrd takes an average time of 43 s to find the first solution at least as good, if any. For the instances of size $n = 100$ the average percentage gap of the MathTSPrd from the HeuTSPrd is 3.33%. The MathTSPrd finds the best solution in an average time of 246 s, while the HeuTSPrd takes an average time of 47 s to match the result of the MathTSPrd, when this is achieved.

The number of routes of each solution is reported in the column "# Routes". The value appears to follow a positive trend with respect to the value of the parameter β. The trend is shared by both the solutions found by the MathTSPrd and the HeuTSPrd version of the heuristic. This leads to the conclusion that the more the release dates get spread out in time compared to the cost of visiting the vertices, measured as the cost of the TSP solution on the instance, the more it is convenient to visit customers as soon as they are available. To further confirm these results, the two variants of the ILS have been compared with a "myopic" heuristic visiting customers as they become available, on instances of $n = 50$ vertices. This heuristic works as follows: every time the vehicle reaches the depot, it either departs to serve the customers whose goods arrived while the vehicle was traveling or waits for the next customer that can be served. The results show the benefit of the ILS compared to the intuitive rule of serving customers as soon as they become available. The "myopic" algorithm performs on average 16% worse than the best solution, with the worst case taking 33.25% more time to complete the distribution than the solution found by the HeuTSPrd version. The percentage gap from the best solution has a negative trend with respect to the parameter β, confirming the conclusion drawn above.

An indicator of the operational implications of the solution is given in the column "% Wait" where the percentage of time spent waiting at the depot over the total time required to complete the distribution is reported. We observe that in most of the tested instances the waiting time is less than 10% of the total service time and only one solution is found with more than 15% of waiting time for the instances with size $n = 50$ and $n = 100$. The average percentage waiting time appears to decrease as the instance size increases. In particular, focusing on the solution found by the HeuTSPrd, the value goes from 6.51% to 4.29% when the instance size goes from $n = 50$ to $n = 100$. We also observe that the average waiting time of the HeuTSPrd is greater than the one of the MathTSPrd for both the instances of size $n = 50$ and $n = 100$, indicating that, on average, the best solution known for each instance has a higher waiting time to total service time ratio. A further comparison with the "myopic" heuristic can be carried out on instances with $n = 50$. The "myopic" heuristic has a very little waiting time component but the solutions found with this method are poor compared to those found by the HeuTSPrd, as discussed above. This validates the conclusion that the choice of serving customers versus waiting for more goods to arrive at the depot is non-trivial and deserves a thorough analysis.

The results obtained indicate that the time required by the operator based on the MIP, the one adopted in the MathTSPrd, limits the number of iterations performed in the given computational time to a value that is much lower than the number of iterations performed when the random operator is adopted in the HeuTSPrd. This implies that exploring a broader portion of the solution space is more effective than spending more time to find better solutions to feed the LS.

Given the better results of HeuTSPrd, we investigated the impact of the ending conditions on its performance, by extending the time limit to two hours and removing the limit on the number of iterations without improvement. The results of such runs are plotted in Fig. 4 for instances with $\beta = 3$. The x-axis reports the time in seconds in logarithmic scale and the y-axis the percentage gap over the best solution found after two hours. The figure shows that it is not beneficial to give a longer maximum time limit to HeuTSPrd. In fact, in all instances the solution obtained after 10 min is less than 2.5% worse than the one obtained in two hours and the time required to improve the solution obtained after such time limit is often very large (we recall that time is on a logarithmic scale).

Further comparisons of the performance of the two variants have been carried out on the instances originated from the TSPLIB. The selected subset of instances spans from a minimum size of 50 vertices (i.e. `eil51`) to a maximum of 492 vertices (i.e. `d493`). In the MathTSPrd the time limit for the MILP is set to 20 s as

Table 12
Comparison of MathTSPrd and HeuTSPrd on instances with $n = 50$.

		MathTSPrd					HeuTSPrd						Myopic		
β	Instance	Value	% Best	TI	# Routes	% Wait	Value	% Best	TI	# Routes	% Wait	TI MathTSPrd	Value	% Best	% Wait
0.5	C101	333	0	284	3	11.71	333	0	190	3	11.71	190	412	23.72	0
	C201	415	0	7	3	6.47	415	0	3	3	6.27	3	553	33.25	0
	R101	594	1.02	408	5	2.02	588	0	59	5	3.91	3	691	17.52	0
	RC101	530	0	88	3	2.64	530	0	22	3	2.64	22	730	37.74	0
1	C101	422	0	126	5	3.55	423	0.24	34	4	4.73	–	547	29.62	0
	C201	534	1.14	74	5	2.43	528	0	60	5	1.14	15	675	27.84	0
	R101	745	2.34	270	6	4.83	728	0	407	7	1.10	1	851	16.90	0
	RC101	661	0	144	4	7.41	661	0	14	4	7.41	14	737	11.50	0
1.5	C101	523	0.58	29	7	1.53	520	0	262	6	4.62	101	623	19.81	0
	C201	668	1.21	349	6	2.25	660	0	2	7	3.48	2	787	19.24	0
	R101	893	1.48	245	9	4.70	880	0	188	9	1.02	105	973	10.57	0
	RC101	824	0.61	579	6	3.16	819	0	119	6	5.37	6	916	11.84	0
2	C101	625	1.46	141	7	9.28	616	0	200	8	4.55	3	713	15.75	0
	C201	793	1.67	410	8	5.04	780	0	418	8	10.77	133	875	12.18	0
	R101	1,087	2.94	461	10	6.99	1,056	0	519	10	6.82	3	1,195	13.16	0.17
	RC101	978	1.14	21	7	7.98	967	0	40	8	6.93	13	1,148	18.72	0.09
2.5	C101	719	0.56	279	10	1.25	715	0	80	8	15.10	16	797	11.47	0
	C201	921	0.22	503	11	3.37	919	0	148	10	2.07	148	997	8.49	0.10
	R101	1,265	0.64	452	14	6.64	1,257	0	27	13	9.39	26	1,349	7.32	0.74
	RC101	1,132	1.34	482	10	6.89	1,117	0	175	8	6.27	6	1,286	15.13	0.08
3	C101	823	1.11	387	13	6.32	814	0	178	12	11.67	5	853	4.79	0.11
	C201	1,073	0.28	318	11	2.98	1,070	0	182	11	12.71	33	1,158	8.22	0.09
	R101	1,462	0.62	172	18	5.75	1,453	0	418	21	7.85	18	1,513	4.13	1.19
	RC101	1,284	0.23	128	10	5.84	1,281	0	490	11	8.59	122	1,384	8.04	0.07
Avg.			0.86	264.88		5.04		0.01			6.51	42.96		16.12	0.11

Table 13
Comparison of MathTSPrd and HeuTSPrd on instances with $n = 100$.

		MathTSPrd					HeuTSPrd					
β	Instance	Value	% Best	TI	# Routes	% Wait	Value	% Best	TI	# Routes	% Wait	TI MathTSPrd
0.5	C101	677	0	414	5	2.81	677	0	338	5	2.07	338
	C201	749	1.90	272	4	3.20	735	0	537	5	4.90	2
	R101	840	2.07	292	5	2.26	823	0	68	4	1.58	47
	RC101	860	2.50	600	5	0	839	0	112	5	5.60	10
1	C101	906	2.95	124	8	3.09	880	0	51	7	7.50	10
	C201	975	3.17	365	6	4.62	945	0	577	5	6.77	1
	R101	1,078	7.37	208	10	3.53	1,004	0	524	9	0.10	6
	RC101	1,090	5.31	284	8	3.67	1,035	0	78	7	0.10	1
1.5	C101	1,111	3.25	209	12	7.20	1,076	0	61	9	7.53	9
	C201	1,192	4.29	484	10	4.70	1,143	0	589	8	4.37	12
	R101	1,286	4.64	9	12	3.89	1,229	0	409	11	2.60	5
	RC101	1,295	2.70	202	9	2.08	1,261	0	406	10	0.87	4
2	C101	1,351	4.97	10	16	2.37	1,287	0	148	13	1.17	7
	C201	1,412	3.44	229	10	4.18	1,365	0	537	12	6.45	1
	R101	1,478	0	8	16	3.38	1,483	0.34	106	15	0.34	-
	RC101	1,629	8.10	170	13	1.10	1,507	0	155	11	1.66	2
2.5	C101	1,519	0.73	56	17	3.75	1,508	0	120	15	6.23	120
	C201	1,619	2.86	235	16	2.53	1,574	0	360	15	7.37	71
	R101	1,746	1.45	174	20	6.30	1,721	0	396	22	3.49	348
	RC101	1,844	3.71	291	15	6.72	1,778	0	12	13	4.05	12
3	C101	1,757	3.35	286	22	4.67	1,700	0	354	16	8.18	17
	C201	1,858	2.77	494	20	5.33	1,808	0	64	20	4.15	33
	R101	2,056	3.26	10	19	10.02	1,991	0	143	22	13.51	8
	RC101	2,165	5.15	458	18	7.21	2,059	0	529	18	2.28	2
Avg.			3.33	245.17		4.11		0.01			4.29	46.35

this is the best value for large instances and a shorter time provides almost the same results on small instances, as shown in Section 5.2.2. Table 14 reports the results of the two algorithms on 252 instances for the different values of β. The column "HeuTSPrd Value" gives the value of the solution found by the HeuTSPrd algorithm and the column "% MathTSPrd" the percentage gap of the value of the solution found by the MathTSPrd over the corresponding value in the "HeuTSPrd Value" column to the left. In this set of instances the MathTSPrd is systematically outperformed by the HeuTSPrd. The average percentage gaps of the MathTSPrd from the HeuTSPrd does not show a specific trend.

To better highlight the comparison of the two variants of the ILS with respect to the instance size, Table 15 summarizes the results obtained on symmetric instances derived from TSPLIB, grouped by the following size ranges: 50–100, 101–150, 151–250 and 251–500 vertices. It can be observed that the average percentage gap of MathTSPrd has a positive trend with respect to the instance size, while the HeuTSPrd shows consistent results on the same metric, with a slight increase in the 251–500 instance group. For both algorithms, the spread with the initial solution decreases as the size increases, from 14.26% in the 50–100 range to 4.06% in the 251–500 range for MathTSPrd and from 17.40% in the 50–100

Table 14
Comparison of MathTSPrd and HeuTSPrd on the instances derived from TSPLIB.

	$\beta = 0.5$		$\beta = 1$		$\beta = 1.5$		$\beta = 2$		$\beta = 2.5$		$\beta = 3$	
Instance	HeuTSPrd Value	% MathTSPrd	HeuTSPrd Value	% MathTSPrd	HeuTSPrd Value	% MathTSPrd	HeuTSPrd Value	% MathTSPrd	HeuTSPrd Value	% MathTSPrd	HeuTSPrd Value	% MathTSPrd
eil51	545	0.18	664	4.97	850	0.94	1,027	1.46	1,212	1.82	1,386	1.08
berlin52	9,984	1.75	12,000	4.19	14,930	1.15	18,169	1.00	21,529	1.04	24,892	0.48
st70	911	0.44	1,112	5.76	1,364	4.40	1,626	3.63	1,917	2.66	2,217	0.45
eil76	703	2.42	868	4.26	1,027	4.87	1,236	4.85	1,456	2.20	1,687	3.26
pr76	141,425	1.99	172,651	8.88	211,749	3.56	255,415	4.49	301,773	0.69	345,979	1.86
rat99	1,664	1.44	2,063	1.21	2,523	1.55	3,010	2.76	3,527	1.90	4,080	1.35
kroA100	28,071	2.18	36,167	2.21	44,772	3.03	53,563	4.40	62,322	5.36	71,122	3.62
kroB100	29,868	1.13	37,906	3.27	46,076	1.60	54,261	3.28	63,010	4.35	72,596	4.84
kroC100	27,999	1.23	34,929	4.14	43,153	2.40	51,722	1.80	60,880	2.34	70,322	2.61
kroD100	28,715	1.02	36,501	3.08	44,956	3.34	53,455	3.01	61,877	7.33	71,573	5.37
kroE100	29,387	1.73	37,219	4.15	45,835	3.38	55,079	2.86	63,941	3.19	73,748	2.61
rd100	10,546	1.38	13,458	2.21	16,649	3.06	19,928	2.66	23,169	1.35	26,738	1.52
eil101	823	0.97	1,028	4.77	1,233	1.62	1,470	2.72	1,728	4.22	2,008	4.28
lin105	20,153	0.29	25,640	2.19	31,358	2.29	37,347	1.41	43,711	2.75	50,013	2.44
pr107	62,841	0.28	81,590	1.05	99,526	2.04	117,771	1.82	137,778	1.64	157,633	1.28
pr124	81,592	0.24	105,827	1.29	131,301	2.67	154,628	5.16	182,232	2.21	209,157	2.59
bier127	153,870	1.32	190,369	5.01	234,997	8.39	283,277	9.13	332,902	8.58	389,597	6.70
ch130	8,238	3.45	10,305	8.22	12,806	4.24	15,324	3.39	17,826	4.02	20,445	19.40
pr136	131,354	2.39	164,984	1.65	202,170	2.77	241,218	5.72	284,208	4.13	328,775	2.72
pr144	84,662	1.68	111,381	2.35	136,360	4.55	162,547	3.49	187,253	5.45	213,833	9.12
ch150	8,814	3.14	11,041	6.63	13,670	7.24	16,296	2.41	19,045	3.52	22,029	4.02
kroA150	35,462	1.58	45,060	4.40	55,676	2.59	66,775	3.50	79,010	3.34	90,724	2.01
kroB150	35,269	0.73	44,446	9.06	54,459	5.70	65,446	7.18	77,219	2.46	88,263	4.50
pr152	105,710	0.32	137,972	0.92	170,156	2.42	203,653	0.93	236,474	2.94	267,376	4.70
u159	57,578	1.68	74,511	2.19	90,709	4.40	108,865	3.59	126,932	4.84	145,806	5.43
rat195	3,179	2.49	4,042	5.24	4,810	5.18	5,861	6.69	6,863	3.82	7,867	1.91
d198	23,045	2.66	29,621	0.45	36,595	2.09	43,710	2.39	51,107	2.65	58,513	1.96
kroA200	40,123	0.75	50,710	6.21	61,891	5.21	74,533	2.93	86,712	2.23	99,878	3.78
kroB200	40,037	1.71	51,429	1.35	62,562	2.69	74,816	4.47	87,065	6.03	99,910	6.72
ts225	178,901	2.57	229,069	1.53	278,429	5.72	334,784	2.97	390,320	5.22	446,692	3.55
tsp225	5,431	3.94	6,839	4.87	8,396	5.32	9,819	10.41	11,374	7.97	13,348	5.60
pr226	111,997	7.58	148,556	2.85	184,889	2.29	221,846	1.24	259,471	2.34	297,197	1.36
gil262	3,337	0.87	4,200	5.33	5169	1.70	6,087	5.06	7,170	1,81	8,226	1.92
pr264	69,397	0.95	89,983	2.12	110,907	3.89	133,099	2.47	154,463	4.01	178,378	5.14
a280	3,626	5.38	4,602	2.28	5,503	5.91	6,659	4.23	7,740	5.63	8,889	15.91
pr299	67,752	0.55	86,133	1.23	105,509	2.54	125,318	2.86	146,635	1.90	167,359	1.17
lin318	59,008	1.52	75,732	3.78	93,531	1.52	111,255	3.70	129,062	13.84	147,677	13.70
rd400	20,649	10.94	26,602	14.79	32,719	16.64	38,882	17.77	45,556	17.26	52,452	16.39
fl417	17,519	1.49	22,036	7.55	27,321	8.42	33,125	7.30	38,324	8.19	43,698	8.43
pr439	150,834	6.57	194,527	10.14	237,929	12.54	282,937	13.56	330,485	13.41	375,922	13.94
pcb442	68,173	11.66	88,116	15.16	107,594	17.87	130,733	16.40	153,334	15.77	175,161	15.81
d493	50,548	3.81	64,404	8.61	79,553	9.89	94,619	10.86	107,956	13.35	125,521	11.40
Avg.		7.00		4.56		4.61		4.81		4.99		5.40

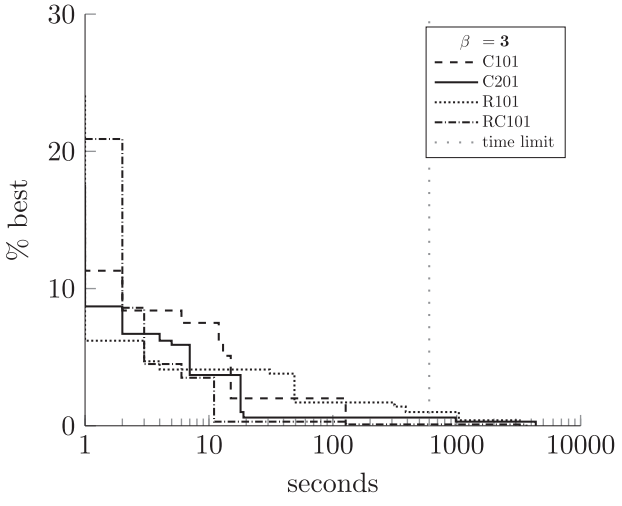

Fig. 4. Plots of the percentage gap of the current best solution over the final solution found in the two hours run of HeuTSPrd on the instances with $n = 100$ and $\beta = 3$.

range to 12.16% in the 251–500 for HeuTSPrd, indicating the increasing difficulty to find a solution that improves the initial one.

Furthermore, a set of experiments on asymmetric instances has been carried out to check whether the behavior of the two variants is affected by the asymmetry of the distance matrix. From the results, reported in Table 16, we observe that while the HeuTSPrd variant is able to greatly improve over the initial solution, the asymmetry of the distance matrix considerably worsens the performance of the MathTSPrd. In particular, in all but 4 instances, all having $\beta = 0.5$, the algorithm is not able to find an improvement of the initial solution.

6. Conclusions

In this paper the Traveling Salesman Problem (TSP) with release dates and completion time minimization, known as the TSPrd(time) problem, is studied. The derived properties contribute to a deeper understanding of a problem belonging to a relatively unexplored class of routing problems. The mathematical formulation has allowed us to solve to optimality instances with up to 20 customers and to assess the quality of the two proposed variants of a heuristic based on an iterated local search, where the perturbation

Table 15
Summary of results on the instances generated from TSPLIB.

		MathTSPrd		HeuTSPrd	
Size	β	Avg % of initial	Avg % Best	Avg % of initial	Avg % Best
50–100	0.5	9.69	1.41	11.23	0
	1	13.62	4.03	18.20	0
	1.5	16.32	2.77	19.54	0
	2	15.87	3.02	19.36	0
	2.5	15.37	2.85	18.61	0
	3	14.69	2.42	17.44	0
		14.26	2.75	17.40	0
101–150	0.5	8.15	1.46	9.72	0
	1	10.48	4.24	15.18	0
	1.5	12.61	4.01	17.11	0
	2	12.97	4.18	17.68	0
	2.5	12.71	3.85	17.05	0
	3	10.40	5.37	16.14	0
		11.22	3.85	15.48	0
151–250	0.5	5.45	2.63	8.19	0
	1	8.46	2.85	11.56	0
	1.5	9.78	3.92	14.11	0
	2	9.93	3.96	14.29	0
	2.5	9.76	4.23	14.42	0
	3	9.60	3.89	13.87	0
		8.83	3.58	12.74	0
250–500	0.5	2.35	4.37	6.75	0
	1	4.11	7.10	11.32	0
	1.5	5.18	8.09	13.43	0
	2	5.19	8.42	13.81	0
	2.5	4.38	9.52	14.03	0
	3	3.17	10.38	13.62	0
		4.06	7.98	12.16	0

Table 16
Comparison MathTSPrd and HeuTSPrd on the asymmetric instances derived from TSPLIB.

		Initial solution		MathTSPrd		HeuTSPrd	
β	Instance	Value	% Best	Value	% Best	Value	% Best
0.5	ftv33	2,424	31.38	1,845	0	1,854	0.49
	ft53	14,690	25.99	12,677	8.72	11,660	0
	ftv70	4,137	21.82	3,842	13.13	3,396	0
	kro124	59,583	17.00	53,247	4.56	50,925	0
	rbg403	5,606	14.99	5,606	14.99	4,875	0
1	ftv33	3,037	36.49	3,037	36.49	2,225	0
	ft53	17,980	28.66	17,980	28.66	13,975	0
	ftv70	5,067	27.86	5,067	27.86	3,963	0
	kro124	77,033	22.34	77,033	22.34	62,964	0
	rbg403	6,837	22.55	6,837	22.55	5,579	0
1.5	ftv33	3,650	42.58	3,650	42.58	2,560	0
	ft53	21,271	32.27	21,271	32.27	16,081	0
	ftv70	5,996	24.30	5,996	24.30	4,824	0
	kro124	94,484	27.54	94,484	27.54	74,083	0
	rbg403	8,067	29.80	8,067	29.80	6,215	0
2	ftv33	4,262	37.75	4,262	37.75	3,094	0
	ft53	24,562	30.87	24,562	30.87	18,768	0
	ftv70	6,925	26.65	6,925	26.65	5,468	0
	kro124	111,934	25.90	111,934	25.90	88,904	0
	rbg403	9,298	24.35	9,298	24.35	7,477	0
2.5	ftv33	4,875	35.12	4,875	35.12	3,608	0
	ft53	27,852	33.42	27,852	33.42	20,875	0
	ftv70	7,854	24.00	7,854	24.00	6,334	0
	kro124	129,385	26.90	129,385	26.90	101,961	0
	rbg403	10,528	22.09	10,528	22.09	8,623	0
3	ftv33	5488	34.41	5,488	34.41	4,083	0
	ft53	31,143	31.00	31,143	31.00	23,774	0
	ftv70	8,784	26.77	8,784	26.77	6,929	0
	kro124	146,835	24.85	146,835	24.85	117,612	0
	rbg403	11,759	20.14	11,759	20.14	9,788	0
Avg.			27.66		25.33		0.02

is performed by means of a destroy-and-repair procedure. The two variants differ in the repair operator, aimed at inserting customers in a partial solution. The variant named MathTSPrd uses a MILP model as repair operator while the one named HeuTSPrd inserts customers at random. The computational results show that the HeuTSPrd outperforms the MathTSPrd. Heuristics that make use of MILP models, the often so called matheuristics, have been shown to be effective in several cases when compared to other more traditional heuristics. However, in our study, using MILP models as a repair operators in an iterative framework has not turned out to be beneficial for the solution of the TSP-rd(time).

Several research directions remain open. An interesting direction is related to designing new heuristics for the problem and investigating the situations where the use of a MILP model is effective in a heuristic framework, and possibly identifying characteristics of the problems or the matheuristic for which this happens. It would be interesting also to study the extension of the TSP-rd(time) problem to the case where a fleet of vehicles is available. Finally, it would be interesting to study stochastic and dynamic versions of the problem, which are closer to the practical applications described in the introduction.

References

Archetti, C., Feillet, D., Speranza, M., 2015. Complexity of routing problems with release dates. Eur. J. Oper. Res. 247, 797–803.

Azi, N., Gendreau, M., Potvin, J.-Y., 2007. An exact algorithm for a single-vehicle routing problem with time windows and multiple routes. Eur. J. Oper. Res. 178 (3), 755–766.

Azi, N., Gendreau, M., Potvin, J.-Y., 2010. An exact algorithm for a vehicle routing problem with time windows and multiple use of vehicles. Eur. J. Oper. Res. 202 (3), 756–763.

Azi, N., Gendreau, M., Potvin, J.-Y., 2014. An adaptive large neighborhood search for a vehicle routing problem with multiple routes. Comput. Oper. Res. 41, 167–173.

Cattaruzza, D., Absi, N., Feillet, D., 2016. The multi-trip vehicle routing problem with time windows and release dates. Transp. Sci. 50, 676–693.

Cattaruzza, D., Absi, N., Feillet, D., 2016. Vehicle routing problems with multiple trips. 4OR 14 (3), 223–259.

Gavish, B., Graves, S., 1978. The Travelling Salesman Problem and Related Problems. Technical Report. Massachusetts Institute of Technology, Operations Research Center, OR 078-78.

Helsgaun, K., 2000. An effective implementation of the Lin–Kernighan traveling salesman heuristic. Eur. J. Oper. Res. 126 (1), 106–130.

Klapp, M.A., Erera, A.L., Toriello, A., 2018. The dynamic dispatch waves problem for same-day delivery. Eur. J. Oper. Res.

Klapp, M., Erera, A., Toriello, A., 2016. The one-dimensional dynamic dispatch waves problem. Transp. Sci. 10.1287/trsc.2016.0682.

Lin, S., Kernighan, B., 1973. An effective heuristic algorithm for the traveling-salesman problem. Oper. Res. 21, 498–516.

Lourenço, H.R., Martin, O.C., Stützle, T., 2010. Iterated local search: framework and applications. In: Gendreau, M., Potvin, J.-Y. (Eds.), Handbook of Metaheuristics. Springer US, Boston, MA, pp. 363–397.

Öncan, T., Altınel, I.K., Laporte, G., 2009. A comparative analysis of several asymmetric traveling salesman problem formulations. Comput. Oper. Res. 36, 637–654.

Reinelt, G., 1991. TSPLIB - a traveling salesman problem library. ORSA J.Comput. 3 (4), 376–384.

Reyes, D., Erera, A.L., Savelsbergh, M.W., 2018. Complexity of routing problems with release dates and deadlines. Eur. J. Oper. Res. 266 (1), 29–34.

Shelbourne, B., Battarra, M., Potts, C., 2017. The vehicle routing problem with release and due dates. INFORMS J. Comput. 29 (4), 705–723.

Solomon, M., 1987. Algorithms for the vehicle routing and scheduling problems with time window constraints. Oper. Res. 35, 254–265.

Voccia, S., Campbell, A., Thomas, B., 2017. The same-day delivery problem for online purchases. Transp. Sci. 10.1287/trsc.2016.0732.