Contents lists available at GrowingScience

International Journal of Industrial Engineering Computations

homepage: www.GrowingScience.com/ijiec

A hybrid metaheuristic for the time-dependent vehicle routing problem with hard time windows

N. Rincon-Garcia[a,b*], B.J. Waterson[a] and T.J. Cherrett[a]

[a] *Transportation Research Group, University of Southampton, Southampton, UK*
[b]*Department of Industrial Engineering, School of Engineering , Pontificia Universidad Javeriana, Bogota, Colombia*

C H R O N I C L E

Article history:
Received April 4 2016
Received in Revised Format June 16 2016
Accepted June 16 2016
Available online June 16 2016

Keywords:
Vehicle routing problem
Time-dependent travel time
Hybrid metaheuristic algorithm

A B S T R A C T

This article paper presents a hybrid metaheuristic algorithm to solve the time-dependent vehicle routing problem with hard time windows. Time-dependent travel times are influenced by different congestion levels experienced throughout the day. Vehicle scheduling without consideration of congestion might lead to underestimation of travel times and consequently missed deliveries. The algorithm presented in this paper makes use of Large Neighbourhood Search approaches and Variable Neighbourhood Search techniques to guide the search. A first stage is specifically designed to reduce the number of vehicles required in a search space by the reduction of penalties generated by time-window violations with Large Neighbourhood Search procedures. A second stage minimises the travel distance and travel time in an 'always feasible' search space. Comparison of results with available test instances shows that the proposed algorithm is capable of obtaining a reduction in the number of vehicles (4.15%), travel distance (10.88%) and travel time (12.00%) compared to previous implementations in reasonable time.

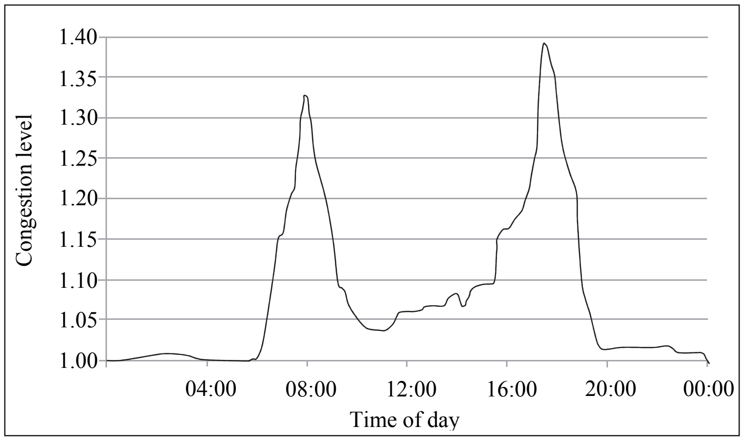
1. Introduction

In logistics operations, the routing and scheduling of collections and deliveries is vital to control costs and meet customer service levels (Chopra & Meindl, 2007). Effective planning improves productivity (DFT, 2010) and an important component of commercial routing and scheduling software commonly used by the industry is the model that supports the decision making process (Drexl, 2012). The impact of congestion has increased over the last 30 years, with the 101 largest US cities reporting that travel delay had increased from 1.1 billion hours in 1982 to 4.8 billion hours in 2011 (Chang et al., 2015). Fig. 1 shows the variation in the level of congestion by time-of-day that affects the freight transport in the US. Rising levels of traffic congestion mean that logistics providers face the challenge of maintaining time critical service levels whilst at the same time minimising the extra costs that congestion and delays impose.

* Corresponding author. Tel.: +57- 310-340-6449
E-mail: nicolas.rincon@javeriana.edu.co (N. Rincon-Garcia)

doi: 10.5267/j.ijiec.2016.6.002

The key impact congestion has on vehicle planning is that travel times between locations vary as a function of the changing traffic patterns. Failure to account for this in routing decisions leads to drivers running out of hours, additional overtime payments and missed deliveries (Haghani & Jung, 2005; Kok et al., 2012) with underestimation of travel times being reported as the most common problem by transport managers (Eglese et al., 2006; Ehmke et al., 2012).



US-Department-Of-Transport (2003)
Fig. 1. Variation in Congestion by time-of-day

The time-dependent vehicle routing problem with hard time windows (TDVRPTW) is the variant where travel time between locations depends on the time of the day with a strict (non-negotiable) time window for the delivery being initially established by the customer (Figliozzi, 2012; Malandraki & Daskin, 1992). The primary objective is to reduce the number of vehicles required to complete the schedule whilst minimising travel distance and travel time (Figliozzi, 2012).

Variants of the vehicle routing problem (VRP) are NP-Hard and metaheuristic algorithms have been developed to solve the problem with trials suggesting significant improvement in performance over current schedules. Some implementations have achieved high accuracy (difference between best-known values and results of the particular algorithm for available test instances) with execution times that allow logistics planners to realistically use the approach as part of their everyday operations (Cordeau et al., 2002; Drexl, 2012).

In an industry where the profit margin is 3% (FTA, 2015), optimization procedures that account for congestion can mitigate its impact. Although managerial solutions are available to account for congestion such as planning vehicle schedules with average travel times, it might lead to poor solutions with missed deliveries and extra costs (e.g. more vehicles, duty time and distance) when compared to solutions based on the use of time-dependent VRP variants (Kok et al., 2012).

In a recent survey conducted with 19 companies in the UK that showed the identification of the most important improvements required by the freight industry for vehicle routing software, companies ranked at one of the two most important capabilities the optimisation of routes minimising the impact of congestion (Rincon-Garcia et al., 2015).

2. Literature review

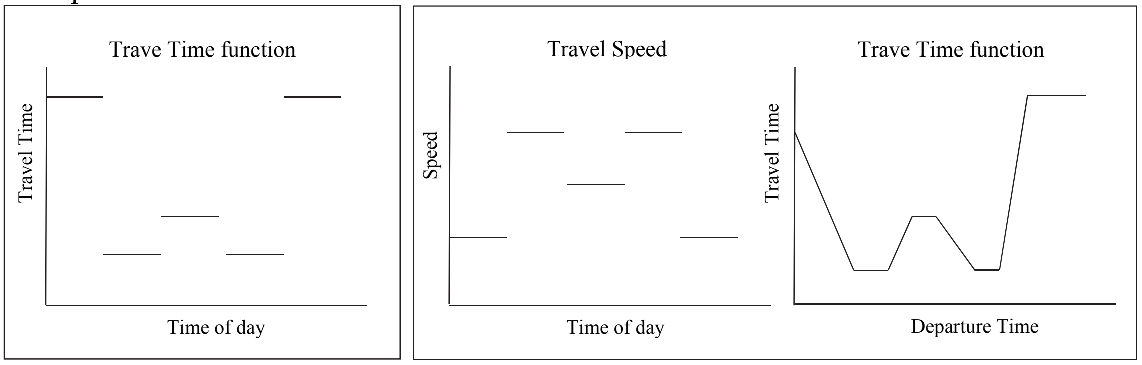
The importance of providing effective and efficient solutions for city logistics is imperative bearing in mind the low profit margin in the sector (FTA, 2015) and the growing expectations of customers. Current retail trends show that on-line sales represent 14% of all UK brick-and-mortar stores and e-commerce and this is expected to rise up to 35% by 2020 (Javelin-Group, 2011; Visser et al., 2014). Currently most home delivery services do not provide a time window for the delivery, something that customers highly value. Moreover, for deliveries that require customer presence, there is a 12% chance of failure of delivery (IMRG, 2012). With current information and communication technologies is feasible to provide customers with more accurate delivery information (Visser et al., 2014), but the challenge of creating more accurate schedules is sizeable.

Congestion is an ever present problem affecting all road users and delays have significant negative impacts on the freight transport industry with models capable of mitigating the negative impacts being given greater credence (Kok et al., 2012). Analysis of schedules supported by VRP models with time-dependent travel times against models using constant speed illustrate the impact of not considering congestion on routing decisions when speed varies during deliveries (Kok et al., 2012). The solutions presented by constant-speed models have been shown to underestimate the actual travel time, provide unrealistic solutions and fail to honour customer delivery times (Donati et al., 2008; Fleischmann et al., 2004; Ichoua et al., 2003).

The first formal formulation of the time-dependent VRP (TDVRP) was presented by Malandraki (1989) and Malandraki and Daskin (1992). In the TDVRP, a fixed number of vehicles with limited capacity serve a number of customers with fixed demands with the overall objective of minimising total route time. Travel time between customers depends on distance and time of day and a single depot exists from where vehicles must depart and return to after completing the delivery tour. No split deliveries (multiple visits to a customer by the same of different vehicles) are allowed and service time windows can be present. Two different solution approaches were proposed (a set of 'greedy' heuristic algorithms and a cutting-plane algorithm) using up to 25 customers, randomly generated without time windows. Although the cutting-plane algorithm was more computationally expensive and was able to solve to optimality, it returned incumbent solutions with greater accuracy in 66% of the tests compared to the heuristic approach.

The TDVRP formulation by Malandraki and Daskin (1992) was based on a travel time step function where each period of time has a specific travel time between nodes associated with it. This has been criticised as being unrepresentative of reality as a vehicle with a later departure time might arrive earlier than a vehicle with an earlier departure time across the same link, as it is point out by Ichoua et al. (2003). Later work on the TDVRP has seen the implementation of travel time functions that respect the "first-in-first-out" principal (FIFO) based on continuous functions (Donati et al., 2008; Figliozzi, 2012; Fleischmann et al., 2004). Fig. 2 shows the difference between Malandraki and Daskin (1992) step travel time function and Ichoua et al. (2003) continuous travel time function for 5 periods of time in a day.

a. Step function b. Continuous function



a. Malandraki and Daskin (1992)
b. Ichoua et al. (2003)

Fig. 2. Different Travel Time functions for constant distance with variable speed

Ichoua et al. (2003) studied the TDVRP with soft time windows, where violation of the required service time was permitted but with a resulting penalty in the objective function. The number of routes was fixed and vehicle capacity was not taken into consideration. The objective function was to minimise travel time and penalties incurred due to service time window violations. The solution approach was based on a Tabu Search metaheuristic algorithm with the resulting analysis against the constant-speed model suggesting that a time-dependent speed scenario lead to unrealistic solutions and greater overall travel times. Fleischmann et al. (2004) presented the TDVRP with and without time windows using scenarios based on real congestion patterns from a traffic information system in Berlin. The solution approach used different heuristic algorithms and local search techniques with the results suggesting that using constant-speed models might generate underestimates of travel time of 10%.

Donati et al. (2008) implemented an ant colony metaheuristic algorithm for TDVRP variants based on scenarios involving constant speed and where speeds were randomly assigned using Solomon (1987) instances in order to undertand the accuracy of the proposed algorithm by solving the vehicle routing problem with time windows and constant speed (VRPTW) with an algorithm capable to deal with time-dependent travel times. Solomon dataset provides the representation of deliveries for up to 100 customers with time windows and it is one of the most well-known set of instances to evaluate VRP algorithms, not only for the vehicle routing problem with time windows and constant speed (VRPTW) but for different VRP variants that consider additional restrictions to time windows (Figliozzi, 2012; Jiang et al., 2014; Liu & Shen, 1999; Toth & Vigo, 2001). A second scenario for the TDVRP without time windows was proposed based on the road network of Padua, Italy with congestion data taken from a traffic management system. Results from the constant-speed models when tested in a scenario with congested roads suggested that travel time was underestimated between 5% and 12%.

An exact algorithm for the TDVRPTW was presented by Dabia, Ropke, Van Woensel, and De Kok (2013) using a modified set of the well-known instances of Solomon (1987) for the VRPTW. Speed patterns in links between nodes were allocated randomly and the solution approach was based on a pricing algorithm utilising a column generation and a labelling algorithm. In trials, 63% of the 25 customer instances were solved, 38% of the 50 customer and 15% of the 100 customer. However, details about the categorization of links are not provided. Although Donati et al. (2008) and Fleischmann et al. (2004) used variations of Solomon set of instances to evaluate algorithms for the TDVRPTW, only Figliozzi (2012) provides the complete information to fully reproduce instances when speed variation is present. Figliozzi (2012) proposed an insertion heuristic (*IRCI*) to construct an initial feasible solution and a 'ruin-and-recreate' heuristic algorithm to improve the results (*IRCI-R&R*). Benchmarking for accuracy was performed with available best-known values for the VRPTW and executing the proposed algorithm for the TDVRPTW with constant-speed profiles. The VRPTW commonly has 2 objective functions, to minimise the number of vehicles and their travel distance. Results of Figliozzi (2012) approach suggested a 4.2% increase in vehicles required leading to an 8.6% increase in overall travel distance compared to the best-known values returned by the constant-speed case. Additional speed profiles are provided in order to analyse the impact of congestion patterns in results.

In some of the time-dependent variants the number of vehicles are fixed and optimisation is only based on travel time reduction across the fleet. However, finding the minimum number of vehicles required in the presence of hard time windows is in itself a complex problem, more computationally expensive with time-dependent travel times. Additionally, most current efficient methods for the VRP variants are intricate and difficult to reproduce (Vidal et al., 2013). Furthermore, some metaheuristic implementations have been tailored to work well in specific test instances by tuning parameters so specifically as considering the best random seed that provides high accuracy (Sörensen, 2015). There is clearly therefore a need for general and simple methods applicable to practical applications required by the industry (Vidal et al., 2013), such as effective algorithms that consider congestion and provide reliable schedules. However, the capability of current algorithms to provide high accurate solutions for time-dependent VRP variants is still not well understood due to the lack of adequate algorithm evaluation with previously proposed instances. Gendreau et al. (2015) conducted the most recent literature review in time-dependent routing problems. Some of the mentioned problems are the "*time-dependent point-to-point route planning*" (which is obtaining the optimal path between two locations in a road network) and the "*time-depend vehicle routing*" (time-dependent VRP variants). The challenge in the point-to-point route planning relies on providing efficient algorithms on-line for the next-generation web-based travel information systems that require results in milliseconds or microseconds. Although, it is required to use this problem to establish time-dependent travel times in the TDVRPTW, shortest paths might be determined in a pre-processing phase (Kok et al., 2012) prior to the execution of solving the actual schedule due to the fact that forecasted travel-times are used and there is no need of on-line applications when designing the routes for the following planning period. Additional VRP variants that consider fuel consumption are also mentioned. Gendreau et al. (2015) highlight the requirement of additional contributions of the Operational Research community in time-dependent routing problems, where techniques for constant-speed classic network optimization problems exist but it is required research for their time-dependent counterparts.

This research therefore presents the introduction of state-of-the-art techniques in VRP variants to provide efficient solutions to the industry by comparing quality of algorithm results. In the following subsections the set of restrictions that account for the TDVRPTW is presented along with the concepts of LNS and Variable Neighbourhood Search (VNS) used to guide the search in the algorithm that is proposed in this research. In order to solve VRP variants, exact methods (algorithms that can guarantee optimality) are only viable for small instances, and it is common to use metaheuristic algorithms capable to solve large instances in reasonable time (Drexl, 2012). However, it is important to understand the capability of algorithms to provide high accurate results. A test carried out by an academic group using different providers of software for vehicle routing showed significance difference in quality of solutions, up to 10% between the best vs. the worst schedule in instances of only 100 customers, where higher difference was found in larger problems (Bräysy & Hasle, 2014; Hallamaki et al., 2007). Therefore the importance to further investigate the capabilities of current time-dependent algorithms (Gendreau et al., 2015) to deliver improved performance across a variety of operational and traffic settings. Time-dependent models create new complexities for algorithm design related to tailoring existing search strategies designed specifically for constant-speed models. Common local search procedures require significant modification as alterations within a route as part of the search process could potentially affect the feasibility of the rest of the route. This might alter the departure times of subsequent visits to customers and consequently modify travel times. Route evaluation is considerably more computationally expensive with time-dependent travel times and accommodating hard time constrains for time windows also requires more computational resources than soft constraints where solutions with violations of time windows are allowed (Figliozzi, 2012). Large Neighborhood Search (LNS) is a search strategy that stands out in the range of concepts among state-of-the-art algorithms to solve vehicle routing models due to its simplicity and wider applicability and it has been extended to successfully address various variants (Vidal et al., 2013). In this research, a LNS algorithm is tailored to solve the complexities involved in the TDVRPTW with results compared against available test instances in order to understand its capabilities.

2.1. Problem definition

- Deliveries are requested by n customers
- let G= (V, A) be a graph where vertex V = $(v_0, v_1, \dots, v_n)$, v_0 is the depot and $v_1, \dots, v_n$ the set of customers. Each element of V has an associate demand $q_i \geq 0$ (which must be fulfilled), a service time $g_i \geq 0$ and a service time window $[e_i, l_i]$. Note that $0 = q_0 = g_0$
- An undetermined number of identical vehicles each with maximum capacity q_{max} are available and stationed at v_0. Vehicles must depart from and return to the depot v_0 at the end of the delivery tour and their maximum capacities cannot be exceeded

- The departure time of any given vehicle from v_i is denoted b_i, its arrival time a_i
- Arrival time to customer v_i must be before l_i. If arrival time is before e_i, the vehicle have to wait until e_i. Each customer can only be visited once
- Let A be the set of arcs between elements of V, having constant distance d_{ij} between v_i and v_j
- For each arc (i,j) ∈ A there exists a travel time $t_{ij}(b_i) \geq 0$ a function of departure time b_i, (e.g. of a form as proposed by Ichoua, Gendreau and Potvin (Ichoua et al., 2003) see appendix A).

The primary objective is to minimise the number of vehicles and the second objective is to minimise the sum of travel distance, travel time or duty time

2.2. Large Neighbourhood Search

LNS is an algorithm for neighbourhood exploration introduced by Shaw (1997) utilising a very similar concept to the 'ruin-and-recreate' algorithm introduced by Schrimpf, Schneider, Stamm-Wilbrandt, and Dueck (2000) (Ropke & Pisinger, 2006; Shaw, 1998). A number of partial-destruction procedures are used to remove customers from the solution and a different set of reconstruction procedures are used to create a new solution by inserting removed customers in a smart way, see Fig. 3.

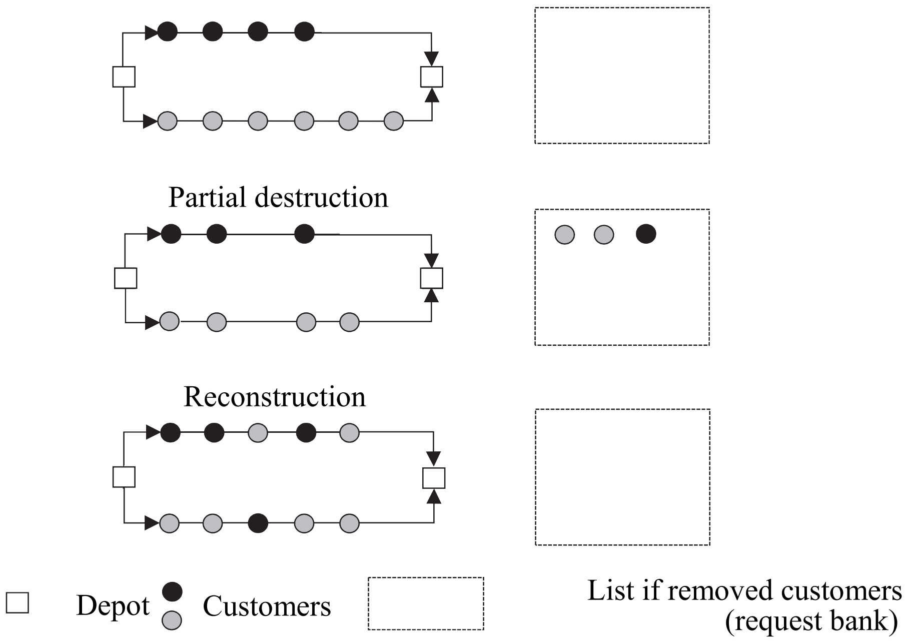


Fig. 3. Large Neighbourhood Search movement

Schrimpf et al. (2000) proposed some basic procedures for the VRPTW that were extended by Ropke and Pisinger (2006), Pisinger and Ropke (2007), and Mattos Ribeiro and Laporte (2012) for VRP variants, some are presented below:
Destruction procedures

- *Random-Ruin*: Randomly select and remove w customers from all customers in the solution.
- *Radial-Ruin*: Randomly select a customer v_i. Remove v_i and w -1 closest customers to v_i.
- *Sequential-Ruin*: Select a random route k (vehicle k) and select a random customer v_i in k. Remove a chain of consecutive customers of length w in k starting with v_i.
- *Worst-removal*: Remove the customers that have the most negative impact according to a function *removal-cost*(v_i).

Recreation procedure

- *Basic-greedy heuristic*: Given the list of removed customers U, calculate an *insertion-cost*(v_i, k, p) for all $v_i \in U$ in all possible routes and positions when v_i is inserted in route k in position p, and insert v_i with the lowest *insertion-cost*(v_i, k, p) in the solution. Repeat the procedure until all $v_i \in U$ are inserted or no feasible insertion exists.

- One characteristic of the LNS for VRP variants is that the request bank is an entity that allows the search process for the exploration of infeasible solutions (Ropke & Pisinger, 2006) without directly calculating the violation of restrictions. In the case of the TDVRPTW, any solution with unscheduled customers is infeasible. Additionally, insertion procedures are quite myopic, in order to avoid stagnating search processes, where destruction and recreation procedures keep performing the same modification to a solution, providing diversification in different levels of the search process might improve accuracy of solutions (Mattos Ribeiro & Laporte, 2012).

- Previous LNS implementations have made use of other metaheuristic algorithms at the master level to guide the search to new regions and accept improved solutions such as Simulated Annealing (Mattos Ribeiro & Laporte, 2012; Ropke & Pisinger, 2006). In the neighbourhood exploration, applying noise in recreation procedures also avoids stagnation e.g., by using randomisation in the insertion evaluation function in recreation procedures (Ropke & Pisinger, 2006), or tailoring recreation procedures to ensure diversification (Mattos Ribeiro & Laporte, 2012).

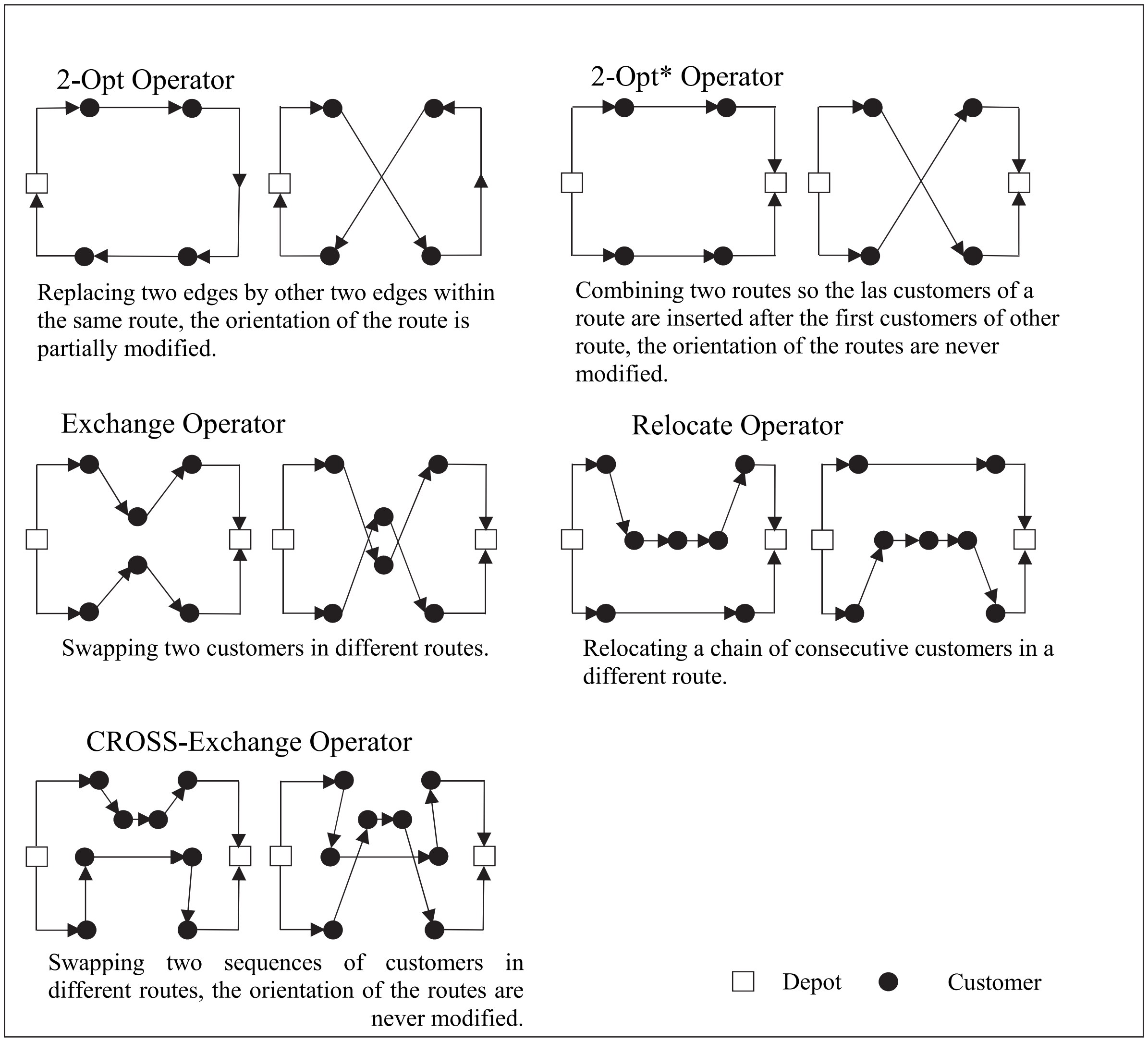
2.3. Variable Neighbourhood Search

VNS is a metaheuristic algorithm introduced by Mladenović and Hansen (1997) and Hansen and Mladenović (2001). VNS has been previously implemented in a range of combinatorial problems (Hansen et al., 2010) including VRP models (Bräysy, 2003; Kytöjoki, Nuortio et al., 2007) and the TDVRP with soft time windows (Kritzinger, Tricoire, Doerner, & Hartl, 2011). VNS uses local search neighbourhoods and avoids local optima with specially designed procedures called "*Shaking*" which usually have random elements (Hansen & Mladenović, 2001; Hansen et al., 2010). An additional element of VNS is the concept that a local optima in a neighbourhood is not necessarily a local optima for other neighbourhoods, therefore changing neighbourhoods might avoid local optima. The pseudo code that illustrates the basic concept of VNS is presented as follows:

Algorithm 1: Basic concepts of Variable Neighbourhood Search

```
Start
  1.  Initialization by selecting $H$ neighbourhood structures $H = \{h_1,…, h_{max}\}$
  2.  Initialize Incumbent solution
  3.  Current solution ← Incumbent solution
  4.  $h \leftarrow 1$
  5.  Repeat
  6.      Current solution ← Shaking with $h^{th}$ neighbourhood (Incumbent solution)
  7.      Current solution ← Local search (Current solution)
  8.          If (Current solution < Incumbent solution) then
  9.              Incumbent solution ← Current solution
  10.             $h \leftarrow 1$
  11.         Else
  12.             $h \leftarrow h+1$
  13.         EndIf
  14. Until $h = h_{max}$
End
```

A characteristic of the presented basic VNS concept is that it works on an ‘only-descendent’ approach. It changes the search space region when an improvement has been found, (lines 8 to 10). However, it may be easily transformed to a ‘descendent–ascent’ approach by introducing some selection criteria to allow exploration of regions with a deteriorated solution, e.g., randomness (Hansen & Mladenović, 2001) or a threshold acceptance value (Kritzinger et al., 2011). An additional characteristic of VNS is that it does not follow a trajectory, but it explores increasingly distant regions, the set of procedures for “*Shaking”* is at the core of VNS and provides a balance between obtaining a sufficiently large perturbation of the incumbent solution while still making sure desired attributes of “good” solutions are maintained. In order to guide the search, a metric in the “*Shaking”* procedure is introduced (Hansen & Mladenović, 2001), (lines 1 and 6). Local search, line 7, is a set of procedures that allow the exploration of the local search space. An example of VNS for the multi-depot VRPTW is presented by Polacek, Hartl, Doerner, and Reimann (2004), as initialization of an incumbent solution with a cheap heuristic and fast running times was proposed. The set of procedures for “*Shaking”* is based on the CROSS-exchange operator where orientation of routes is preserved and the iCROSS-exchange operator where orientation of routes is reversed.



Replacing two edges by other two edges within the same route, the orientation of the route is partially modified.

Combining two routes so the las customers of a route are inserted after the first customers of other route, the orientation of the routes are never modified.

Swapping two customers in different routes.

Relocating a chain of consecutive customers in a different route.

Swapping two sequences of customers in different routes, the orientation of the routes are never modified.

□ Depot ● Customer

Adapted from: Bräysy and Gendreau (2005a)

Fig. 4. Well-known neighbourhood structures

Fig. 4 shows some well-known neighbourhood exploration procedures. The "*Shaking*" metric between solutions was established as the number of depots used to generate the new solution and the maximum length of the removed sequence in the neighbourhood construction. Local search was the 3-opt operator neighbourhood with reverse of route orientation not allowed and the length of the sequences to be exchanged bounded by an upper limit of three. Decision about moving the search to a new region follows the descendent–ascent approach with a threshold acceptance value. Analysis of results in terms of accuracy and speed showed that the proposed VNS was competitive to other metaheuristic algorithms and it was capable to improve some of the best-known results at that moment. A very similar VNS approach for the TDVRP with soft time windows was proposed by Kritzinger et al. (2011).

3. A hybrid metaheuristic for the TDVRPTW

Search procedures for the TDVRPTW are computationally expensive, with the proposed algorithm designed to guide the search to high accurate solutions in reasonable time. The search process is divided into two stages, the first is where an initial incumbent solution is created using a fast construction heuristic to undertake a reduction of vehicles. In the second stage the objective is either the reduction of travel distance or travel time.

3.1. Minimising the number of vehicles

In previous implementations of the LNS for constant-speed models, minimising the required number of vehicles relied on removing routes from an incumbent solution and placing customers in the request bank until a solution was found without unscheduled customers (some customers still in the request bank) (Pisinger & Ropke, 2007; Ropke & Pisinger, 2006). In the initial test of this approach for the TDVRPTW, lengthy computational time was required in order to get high accuracy due to the complexity in movement evaluation originated by time-dependent travel times.

In order to speed up the search process, a strategy to quickly guide the search towards higher accuracy was designed. Solutions which violated time windows were allowed, and the objective function was minimising the sum of violations of time windows (penalty). Due to the myopic behaviour of LNS, the search quickly reached stagnation. Small time window violations were generated by frequent pairs of customers. In order to avoid stagnation, a tabu list of forbidden pairs of customers was introduced to force the recreation procedure to unexplored trajectories.

At the extend of the knowledge of the authors, this is the first implementation of LNS with allowed time window violations that exploits the destruction and recreation procedures and introduces a tabu list of forbidden pairs of customers for VRP variants with time windows. This strategy allows sequences of customers to be identified that are particularly difficult to accommodate in a solution with a reduced number of vehicles, and allows the algorithm to focus on scheduling these customers without incurring in-time window violations while avoiding stagnation. The local search consists of a modified *Worst-removal* procedure to remove customers that generate penalties along with other destruction procedures before executing a modified *Basic-greedy heuristic* for minimisation of time window penalties (*LNS-Penalty Procedure*). The pseudo code of *Number of vehicles minimisation procedure* is presented as Algorithm 2.

Route reduction procedure follows a descendent–ascent approach, (lines 24 to 26). Initially, a feasible incumbent solution is created with the *IRCI* heuristic proposed by Figliozzi (2012). The vehicle with the minimum number of customers is removed and its customers are allocated to the request bank, (line 2). Each time that the search process reaches a solution with the penalty equal to 0, a feasible schedule, the solution is stored, line 19, and a new vehicle is removed, line 20.

Algorithm 2: *Number of vehicles minimisation procedure*

```
Start
 1.  Incumbent solution ← Construction of Initial Solution with IRCI
 2.  Incumbent-penalty solution ← Remove one route (Incumbent solution)
 3.  Tabu List = { Ø }
 4.  $h$ ← 1
 5.  Repeat
 6.      Repeat
 7.          Current solution ← Shaking-Route Reduction Procedure ($h$ , Tabu List, Incumbent-penalty solution)
 8.          Current solution ← LNS Penalty Reduction Procedure (Current solution, Tabu List)
 9.          If penalty(Current solution) > 0
10.              Tabu List ← Tabu List U Elements-Generate-Penalty(Current solution)
11.              If penalty(Current solution) < penalty (Incumbent-penalty solution)
12.                  $h$ ← 1
13.                  Incumbent-penalty solution ← Current solution
14.              else
15.                  $h$ ← $h$ + 1
16.          EndIf
17.      Until $h = h_{max}$ or penalty (Current solution) = 0
18.      If penalty (Current solution) = 0
19.          Incumbent solution ← Current solution
20.          Incumbent-penalty solution ← Remove one route (Incumbent solution)
21.          Current solution ← Incumbent-penalty solution
22.          Tabu List = { Ø }
23.          $h$ ← 1
24.      Else
25.          Incumbent-penalty solution ← Current solution
26.          $h$ ← 1
27.      End if
28.  Until stop criteria met
Return Incumbent solution
End
```

As previously mentioned the local search consists of minimising penalties with LNS procedures described in subsection 2.2, line 8 (*LNS Penalty Reduction Procedure*). Pairs of customers that are in the tabu list are not allowed. In order to prevent stagnating search processes within the LNS neighbourhood movement, *IRCI* is used to diversify the search. All customers in a number of vehicles are removed from the solution and assigned to these vehicles with *IRCI* without violation of time windows. Customers that could not be inserted in these vehicles are inserted in other vehicles using the *Basic-greedy heuristic.*

The proposed *Shaking-Route Reduction Procedure*, line 7, exchanges h customers between routes in order to create new solutions using the exchange operator. Customers to be exchanged are preferably those that generate penalty.

Each time the search reaches a local optima, the pairs of customers that generate penalties in Incumbent-penalty solution are identified and recorded, if they appear in different occasions they are included into the tabu list, line 10.

3.2. Travel time and travel distance minimisation

This procedure relies on the exploration of promising distant regions. In each iteration, a distance region is visited and explored with a fast algorithm. It is established if the region is promising for intensification with a threshold value. Intensification is based on LNS and VNS. The proposed *Travel timed and travel distance minimisation procedure* is presented below as follows:

Algorithm 3: *Travel time and travel distance minimisation procedure*

```
Start
  1.  Incumbent solution ← Number of vehicles minimisation procedure
  2.  Current solution ← Incumbent solution
  3.  Repeat
  4.     Current solution ← Shaking procedure (Current solution)
  5.     Current solution ← Educate procedure  (Current solution)
  6.     If objective (Current solution) < Threshold value
  7.          Current solution ← LNS-VNS Intensification procedure (current solution)
  8.          If objective (Current solution) < objective (Incumbent solution)
  9.                 Incumbent solution ← Current solution
  10.         EndIf
  11.    EndIf
  12. Until stop criteria met
End
```

This procedure starts with the solution provided by *Number of vehicles minimisation procedure*, line 1. The objective function value in the search can be travel time or travel distance. *Shaking procedure*, line 4, creates a solution in a distant region of the search space, only feasible solutions are allowed. Firstly, a random $vehicle_k$ is selected and all v_i in $vehicle_k$ are inserted in the remaining vehicles when insertions are feasible (no violation of time windows are allowed). Secondly, vehicles are randomly sorted, $S' = \{ vehicle_1, \ldots , vehicle_k, \ldots , vehicle_m\}$. Thirdly, each v_i in $vehicle_1$ is exchanged, with the exchange operator, in the first feasible insertion in the subsequent vehicles. The third part of the procedure is repeated in all $vehicle_k \in S', k \neq m$.

Promising solutions are evaluated by executing the *Educate procedure* that consist of 2-opt operator, 2-opt* operator and relocate operator with the length of the sequences to be exchanged bounded by an upper limit of three, line 6. These operators conduct a systematic search by modifying the position of customers within the same route and between routes. Although computationally expensive for a large number of iterations, they are used to identify search regions where a fast reduction in the objective function can be achieved. *Educate procedure* consist of a limited number of iterations.

Identified regions that achieve certain objective function value are selected for intensification, line 7. *LNS-VNS Intensification procedure* is guided by VNS with LNS movements and a "*Shaking*" procedure based on the exchange operator and relocate operator.

4. Benchmark instances

Due to the fact that the TDVRPTW is an extension of the VRPTW, Figliozzi (2012) proposed a modification to the well-known instances for the VRPTW of Solomon (1987) to account for congestion by adding speed profiles. Solomon instances consist of 56 problems with 100 customers and a single depot. Problems are divided in six classes namely: R1, R2, C1, C2, RC1 and RC2. R accounts for random locations, C for clustered locations and RC for a mix of random and clustered locations. Type 1 consist of schedules with tight time windows that allow fewer customers per vehicle than type 2.

Figliozzi (2012) proposed 4 types of speed profiles, with 3 cases for each type, for a total of 12 speed cases. The depot working time $[e_0,l_0]$ is divided into five periods of equal duration. An additional instance with constant speed of 1 is also introduced in order to compare the performance of the proposed solution for the TDVRPTW with available best-known values for the largely studied VRPTW. Speed profiles are as following:

CASES TYPE a (Fast periods between depot opening and closing times)

TD1a = [1.00, 1.60, 1.05, 1.60, 1.00]
TD2a = [1.00, 2.00, 1.50, 2.00, 1.00]
TD3a = [1.00, 2.50, 1.75, 2.50, 1.00]

CASES TYPE b (Higher travel times at the extremes of the working day)

TD1b = [1.60, 1.00, 1.05, 1.00, 1.60]
TD2b = [2.00, 1.00, 1.50, 1.00, 2.00]
TD3b = [2.50, 1.00, 1.75, 1.00, 2.50]

CASES TYPE c (Higher travel speeds are found at the beginning of the working day)

TD1c = [1.60, 1.60, 1.05, 1.00, 1.00]
TD2c = [2.00, 2.00, 1.50, 1.00, 1.00]
TD3c = [2.50, 2.50, 1.75, 1.00, 1.00]

CASES TYPE d (Higher travel speeds at the end of the working day)

TD1d = [1.00, 1.00, 1.05, 1.60, 1.60]
TD2d = [1.00, 1.00, 1.50, 2.00, 2.00]
TD3d = [1.00, 1.00, 1.75, 2.50, 2.50]

5. Implementation and experimental results

Algorithm benchmarking is commonly evaluated in terms of accuracy and speed (Bräysy & Gendreau, 2005a, 2005b; Toth & Vigo, 2001). Accuracy can be easily evaluated when data sets are available. However, different factors have an impact on speed such as hardware (processor, ram), code efficiency and compiler (Figliozzi, 2012). Additionally, it is mentioned in the literature that better results might be obtained by tailoring algorithms accordingly to the test instance. This practice is impractical in industry applications that require fast and reliable solution procedures capable to consistently provide high accurate results (Cordeau et al., 2002; Drexl, 2012; Figliozzi, 2012; Golden et al., 1998).

The proposed LNS approach was coded in Java Eclipse version Juno. It has random elements and results might vary in each run, where multi-core processors offer the possibility to execute multiple threads simultaneously. In this research a computer with processor Intel core i7-2600 3.40GHz and 16 GB of ram was used, three independent threads were run simultaneously and the best result of the three was chosen. The algorithm was run with two different sets of parameters according to termination criteria, which consist of maximum number of iterations, maximum running time, and allowed running time without improvement. The first set of parameters (named F) was set to produce a fast algorithm whereas the second one (named L) produces a more thorough search.

Donati et al. (2008) and Figliozzi (2012) have presented results for metaheuristic approaches for the TDVRPTW using Solomon instances with constant speed in order to compare results with best-known values for the VRPTW, table 1. Best-known values for the minimum number of vehicles for the 56 problems was 405. The result for the proposed algorithm, with set of parameter L, is 408 and best-known values can be achieved by extending running time. However, parameter tuning was set to deal with 672 problems (56 problems x 12 speed cases).

Running the proposed algorithm with sets of parameters F and L provide higher accuracy than previous implementations for the TDVRPTW in the primary objective (average number of vehicles) for instances

R1, RC1 and RC2. Analysis of the secondary objective (distance) shows that the proposed LNS obtained higher accuracy than *IRCI-R&R* (Figliozzi, 2012) in all instances, results within 1% of best-known values can be achieved by increasing running time. Ant colony approach (Donati et al., 2008) obtained higher accuracy in the secondary objective. However, reduced distances might be achieved easily when more vehicles are used, e.g.: problem type R1, reduction of 0.93% from best-known values is achieved with 5.88% more vehicles.

Table 1
VRPTW results for Solomon's 56 problems with 100 customers – Constant speed

Method	R1	Δ	R2	Δ	C1	Δ	C2	Δ	RC1	Δ	RC2	Δ
Average NV												
(1) Best known value	**11.9**		**2.7**		**10.0**		**3.0**		**11.5**		**3.3**	
(2) IRCI – R&R	12.6	5.88%	3.0	11.11%	10.0	0.00%	3.0	0.00%	12.1	5.22%	3.4	4.62%
(3) Ant Colony	12.6	5.88%	3.1	14.81%	10.0	0.00%	3.0	0.00%	12.1	5.22%	3.8	16.92%
(4) LNS F Best 3 runs	12.2	2.52%	3.0	11.11%	10.0	0.00%	3.0	0.00%	11.6	1.09%	3.3	0.00%
(5) LNS L Best 3 runs	12.0	0.84%	2.8	4.38%	10.0	0.00%	3.0	0.00%	11.6	1.09%	3.3	0.00%
Average Distance												
(1) Best known value	**1210.3**		**954.0**		**828.4**		**589.9**		**1384.8**		**1119.2**	
(2) IRCI – R&R	1248.0	3.11%	1124.0	17.82%	841.0	1.52%	626.0	6.12%	1466.0	5.86%	1308.0	16.87%
(3) Ant Colony	1199.0	-0.93%	967.0	1.36%	828.0	-0.05%	590.0	0.02%	1374.0	-0.78%	1156.0	3.29%
(4) LNS F Best 3 runs	1222.3	0.99%	961.7	0.81%	834.0	0.68%	590.3	0.07%	1405.9	1.52%	1170.0	4.54%
(5) LNS L Best 3 runs	1232.2	1.81%	969.6	1.64%	828.6	0.02%	590.3	0.07%	1404.1	1.39%	1160.0	3.65%

(**1**) Best-known values as reported in Nagata, Bräysy, and Dullaert (2010) **(2)** Figliozzi (2012) CPU Time 19 min, processor Intel Core Duo 1.2 GHz. **(3)** Donati et al. (2008) CPU Time 168 min, Pentium IV 2.66 GHz. **(4)** LNS F (3 threads of 26 min) Intel Core i7 3.4 GHz. **(5)** LNS L (3 threads of 62 min) Intel Core i7 3.4 GHz.

In the TDVRPTW the proposed primary objective is the minimisation of the number of vehicles, secondary objective might be minimisation of travel distance, travel time or both. Figliozzi (2012) proposed the sum of distance and travel time as secondary objective. Tables (2-5) show benchmarking of *IRCI-R&R* and the proposed LNS. The proposed algorithm is capable of obtaining a reduction in vehicles required of up to 12.96% (cases type b, set of parameter L, instance R2, Table 3) and secondary reduction objective up to 19.60% in travelled time and distance with fewer vehicles (cases type a, set of parameter L, instance RC2, Table 2) in reasonable computational time. In average each problem can be solved in 26.78 seconds using 3 threads with set of parameter S and 65.35 seconds with set L. Table 6 shows the overall sum of number of vehicles, total travelled distance and total travelled time required to solve the 56 problems in each speed profile.

Table 2
TDVRPTW Average results for 3 instances in Cases Type (a) 100 customers

Method	R1	Δ	R2	Δ	C1	Δ	C2	Δ	RC1	Δ	RC2	Δ
(1) Figliozzy IRCI-R&R												
NV	10.8		2.5		10.0		3.0		10.6		3.0	
Distance	1263.3		1243.0		874.3		669.3		1387.3		1444.0	
Travel Time	923.3		875.0		660.3		514.3		1004.3		1012.3	
Second objective	2197.4		2120.5		1544.7		1186.7		2402.3		2459.3	
(2) LNS F												
NV	10.4	-3.53%	2.5	-1.83%	10.0	0.00%	3.0	0.00%	10.2	-4.02%	2.9	-2.03%
Distance	1164.4	-7.83%	1007.4	-18.95%	841.6	-3.74%	589.9	-11.87%	1309.9	-5.58%	1168.6	-19.07%
Travel Time	833.2	-9.76%	674.2	-22.95%	625.9	-5.21%	447.6	-12.97%	924.8	-7.92%	789.2	-22.04%
Second objective	1997.6	-9.09%	1681.7	-20.69%	1467.5	-5.00%	1037.5	-12.57%	2234.7	-6.98%	1957.7	-20.40%
(3) LNS L												
NV	10.3	-4.66%	2.4	-7.19%	10.0	0.00%	3.0	0.00%	10.0	-5.90%	2.8	-7.09%
Distance	1165.6	-7.74%	1013.9	-18.43%	834.7	-4.53%	589.4	-11.94%	1313.0	-5.36%	1178.7	-18.37%
Travel Time	832.6	-9.82%	685.9	-21.61%	619.6	-6.16%	447.2	-13.05%	922.5	-8.14%	795.9	-21.38%
Second objective	2008.5	-8.60%	1702.2	-19.73%	1464.3	-5.20%	1039.7	-12.39%	2245.6	-6.52%	1977.3	-19.60%

(1) Figliozzi (2012) CPU Time 54.1 min, processor Intel Core Duo 1.2 GHz. **(2)** LNS F (3 threads of 78 min) processor Intel Core i7 3.4 GHz. **(3)** LNS L (3 threads of 183 min) processor Intel Core i7 3.4 GHz.

Table 3
TDVRPTW Average results for 3 instances in Cases Type (b) 100 customers

Method	R1	Δ	R2	Δ	C1	Δ	C2	Δ	RC1	Δ	RC2	Δ
(1) Figliozzy IRCI-R&R												
NV	11.8		2.8		10.0		3.0		11.5		3.2	
Distance	1277.7		1225.0		880.3		683.7		1441.7		1439.7	
Travel Time	925.7		917.0		655.3		486.0		1035.3		1078.0	
Second objective	2215.1		2144.8		1545.7		1172.7		2488.5		2520.9	
(2) LNS F												
NV	11.2	-4.92%	2.7	-4.26%	10.0	0.00%	3.0	0.00%	10.9	-4.89%	3.0	-6.54%
Distance	1197.7	-6.26%	1004.7	-17.98%	847.3	-3.75%	590.0	-13.70%	1633.2	13.29%	1202.7	-16.46%
Travel Time	853.9	-7.75%	730.6	-20.33%	602.3	-8.09%	432.0	-11.11%	966.4	-6.66%	896.7	-16.82%
Second objective	2051.6	-7.38%	1735.3	-19.09%	1449.6	-6.22%	1022.0	-12.85%	2332.6	-6.26%	2099.4	-16.72%
(3) LNS L												
NV	11.1	-5.68%	2.5	-12.96%	10.0	0.00%	3.0	0.00%	10.8	-6.20%	2.9	-9.14%
Distance	1204.5	-5.72%	1027.8	-16.10%	837.8	-4.83%	589.9	-13.72%	1373.9	-4.70%	1213.1	-15.74%
Travel Time	859.5	-7.15%	752.6	-17.93%	599.3	-8.54%	432.0	-11.11%	968.1	-6.50%	903.7	-16.17%
Second objective	2075.2	-6.32%	1782.9	-16.88%	1447.2	-6.37%	1024.9	-12.60%	2352.7	-5.46%	2119.7	-15.91%

(1) Figliozzi (2012) CPU Time 57.4 min, processor Intel Core Duo 1.2 GHz. **(2)** LNS F (3 threads of 75 min) processor Intel Core i7 3.4 GHz. **(3)** LNS L (3 threads of 186 min) processor Intel Core i7 3.4 GHz.

Table 4
TDVRPTW Average results for 3 instances in Cases Type (c) 100 customers

Method	R1	Δ	R2	Δ	C1	Δ	C2	Δ	RC1	Δ	RC2	Δ
(1) Figliozzy IRCI-R&R												
NV	10.9		2.5		10.0		3.0		10.8		2.9	
Distance	1280.0		1242.0		863.3		668.7		1419.0		1439.0	
Travel Time	916.0		868.7		626.7		502.3		1034.0		1020.0	
Second objective	2206.9		2113.2		1500.0		1174.0		2463.8		2461.9	
(2) LNS F												
NV	10.4	-4.50%	2.5	-1.83%	10.0	0.00%	3.0	0.00%	10.3	-4.57%	2.8	-4.00%
Distance	1171.6	-8.47%	1142.0	-8.05%	836.8	-3.07%	589.3	-11.87%	1335.3	-5.90%	1197.2	-16.80%
Travel Time	826.2	-9.80%	800.0	-7.90%	605.1	-3.44%	445.3	-11.35%	960.9	-7.07%	857.2	-15.96%
Second objective	1997.9	-9.47%	1942.0	-8.10%	1441.9	-3.87%	1034.6	-11.87%	2296.2	-6.80%	2054.3	-16.56%
(3) LNS L												
NV	10.3	-5.37%	2.3	-9.57%	10.0	0.00%	3.0	0.00%	10.1	-6.19%	2.7	-7.14%
Distance	1172.7	-8.39%	1022.4	-17.68%	829.7	-3.90%	589.3	-11.86%	1322.6	-6.79%	1196.3	-16.87%
Travel Time	828.7	-9.53%	710.6	-18.19%	601.2	-4.06%	445.3	-11.35%	954.6	-7.68%	855.9	-16.09%
Second objective	2011.6	-8.85%	1735.3	-17.88%	1440.9	-3.94%	1037.6	-11.61%	2287.3	-7.16%	2054.9	-16.53%

(1) Figliozzi (2012) CPU Time 55.9 min, processor Intel Core Duo 1.2 GHz. **(2)** LNS F (3 threads of 72 min) processor Intel Core i7 3.4 GHz. **(3)** LNS L (3 threads of 174 min) processor Intel Core i7 3.4 GHz.

Table 5
TDVRPTW Average results for 3 instances in Cases Type (d) 100 customers

Method	R1	Δ	R2	Δ	C1	Δ	C2	Δ	RC1	Δ	RC2	Δ
(1) Figliozzy IRCI-R&R												
NV	11.6		2.8		10.0		3.0		11.3		3.3	
Distance	1292.0		1216.3		865.0		678.3		1421.7		1410.0	
Travel Time	976.0		935.3		665.0		502.3		1063.7		1073.3	
Second objective	2279.6		2154.5		1540.0		1183.7		2496.7		2486.6	
(2) LNS F												
NV	11.3	-3.32%	2.7	-3.23%	10.0	0.00%	3.0	0.00%	10.7	-5.51%	3.1	-5.87%
Distance	1195.1	-7.50%	1026.5	-15.61%	833.5	-3.64%	590.1	-13.01%	1373.0	-3.42%	1178.2	-16.44%
Travel Time	901.3	-7.65%	782.6	-16.33%	642.9	-3.32%	437.7	-12.87%	1020.8	-4.03%	890.4	-17.04%
Second objective	2096.3	-8.04%	1809.1	-16.03%	1476.4	-4.13%	1030.0	-12.98%	2393.8	-4.12%	2068.6	-16.81%
(3) LNS L												
NV	11.3	-3.32%	2.6	-6.59%	10.0	0.00%	3.0	0.00%	10.7	-5.51%	3.0	-7.64%
Distance	1184.2	-8.34%	1005.2	-17.36%	831.3	-3.90%	591.4	-12.82%	1367.3	-3.83%	1169.6	-17.05%
Travel Time	894.2	-8.38%	764.1	-18.31%	641.5	-3.53%	438.6	-12.68%	1013.0	-4.76%	883.9	-17.65%
Second objective	2089.6	-8.34%	1771.9	-17.76%	1482.8	-3.71%	1033.1	-12.72%	2391.0	-4.23%	2056.5	-17.30%

(1) Figliozzi (2012) CPU Time 56.8 min, processor Intel Core Duo 1.2 GHz. **(2)** LNS F (3 threads of 81 min) processor Intel Core i7 3.4 GHz. **(3)** LNS L (3 threads of 189 min) processor Intel Core i7 3.4 GHz.

Table 6
Total Number of vehicles, distance and travelled time in all 56 problems (100 customers) in each of the 12 speed profiles (case types)

Case Type		**(1) Figliozzy IRCI-R&R**			**(2) VNS L**	
	NV	Distance	Travel Time	NV	Distance	Travel Time
TD1a	402	64875.0	53643.0	387	57439.0	46703.4
TD2a	378	64580.0	45847.0	361	57105.5	39505.4
TD3a	360	64667.0	41198.0	348	57358.7	35105.3
TD1b	420	65044.0	54053.0	403	57950.2	47892.0
TD2b	398	64925.0	46773.0	378	59178.5	41878.0
TD3b	393	65781.0	42837.0	370	59018.2	37480.2
TD1c	402	65304.0	53346.0	387	57842.2	47051.2
TD2c	380	64921.0	45583.0	360	57794.1	40599.2
TD3c	365	64791.0	40985.0	350	57317.0	36004.6
TD1d	417	64858.0	54930.0	401	57639.0	48841.1
TD2d	399	64304.0	47905.0	387	57317.9	42465.6
TD3d	388	65084.0	44466.0	375	58368.9	39472.9
TOTAL	**4702**	**779134.0**	**571566.0**	**4507**	**694329.1**	**502998.9**
Δ				**-4.15%**	**-10.88%**	**-12.00%**

(1) Figliozzi (2012) **(2)** LNS L

5.1. Analysis of results

The proposed algorithm consistently provided improved results for the TDVRPTW. As previously mentioned, route evaluation in the search process is computationally expensive in TDVRP variants. Therefore, selection of neighborhood structures and its adequate tailoring is at the most importance in algorithm design.

Analysis of the computational complexity of different neighborhood structures and performance shows the capability of the proposed LNS tailoring to quickly achieve high accuracy over other procedures. Well-known neighborhood structures involve the deletion of (up to) x arcs of the current solution and the generation of x new arcs to create the subsequent solution, the complexity of neighborhood exploration is $O(n^x)$ (Zachariadis & Kiranoudis, 2010). 2-opt and 2-opt* operators are commonly used in VRP variants with time windows (Bräysy & Gendreau, 2005a), the first one relocates customers within the same vehicle and the second one relocates customers in different vehicles and their complexity of exhaustively examining all possible solutions "naive exploration" is $O(n^2)$, more complex operators with more arc removals are consequently more computationally complex (Zachariadis & Kiranoudis, 2010).

The computational complexity of a LNS procedure that makes use of *Basic-greedy heuristic(U)* for recreation depends on the number of elements in the request bank, the number of routes (vehicles) and the number of customers in the modified route in current solution. After the first insertion, the subsequent customer insertions are only evaluated in the previously modified route with *insertion-cost*(v_i, k, p). Therefore, computational complexity largely varies according to the number of routes in current solution, being the worst case a solution with one route and quickly reducing computational complexity with more routes.

It is important to highlight that the concept of LNS relies on designing a neighborhood exploration using a group of LNS procedures, that might make use of random elements to diversify the search process, and effectively exploration of neighborhood is wider than well-known neighbourhood structures.

In order to understand the computational complexity of the proposed LNS tailoring and its benefits, a simplified algorithm for travel time minimisation is introduced where different neighborhood structures are used for local search, namely LNS procedures and 2-opt along with 2-opt* procedures. The pseudo code is presented as follows:

Algorithm 4: *Travel time minimisation procedure 2*

```
Start
  1. Incumbent solution ← Construction heuristic
  2. Current solution ← Incumbent solution
  3. Repeat
  4.    Current solution ← Local Search procedure (Current solution)
  5.    If objective (Current solution) < objective (Incumbent solution)
  6.        Incumbent solution ← Current solution
  7.    EndIf
  8.    Current solution ← Diversification procedure (Current solution)
  9. Until stop criteria met
End
```

The tested instances are R101 and R201 with speed case TD1a. The termination criterion is allowed execution time and it was executed 50 times in a single thread for different execution times in order to illustrate the impact of parameter variation. Note that 2-opt* is restricted, the "naive exploration" only performs a fraction of iterations obtaining deteriorated results in solutions with few routes, such as R201. The behaviour of different local search procedures in *Travel time minimisation procedure 2* is shown in Fig. 5. The numbers of routes as a result of Construction heuristic are respectively 21 in R101 and 5 in R201.

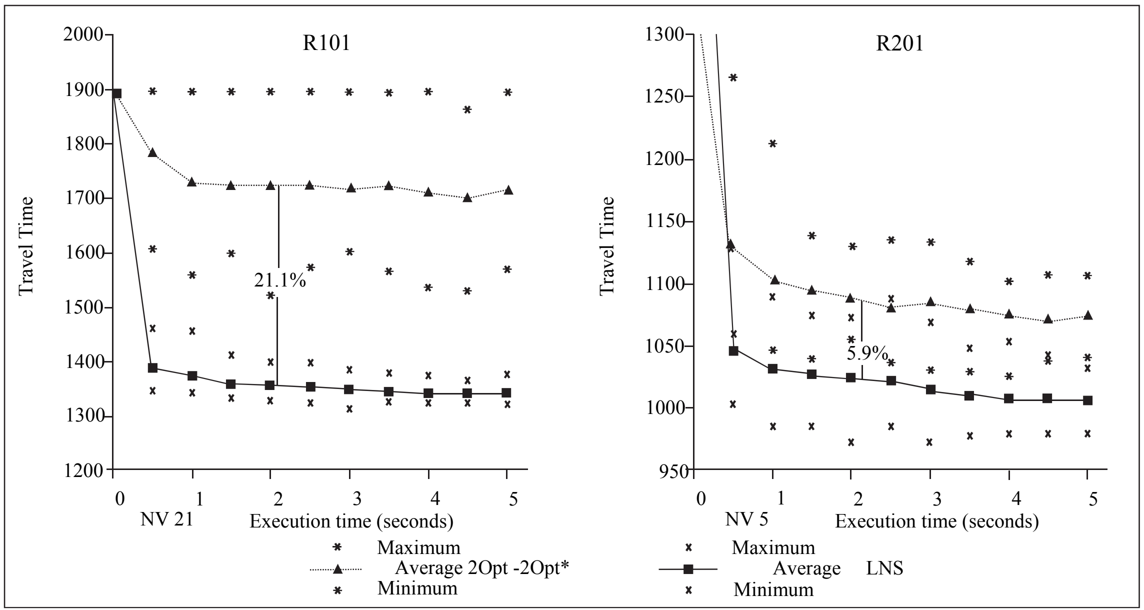


Fig. 5. Behaviour of LNS movements vs. 2-opt and 2-opt* in the presence of time-dependent travel times. Solomon instances R101 and R102 (100 customers) – Figliozzi (2012) speed case TD1a.

Results in execution time of 0.5 seconds illustrate the computational complexity and the accuracy of the proposed LNS, see table 7. 2-opt and 2-opt* were executed a few hundred times whereas LNS complete removals and insertions were executed in average 7,544 times in R101 and 4073 in R102. LNS in instance R101 obtained an average travel time reduction of 22.0% and minimum value reduction of 16.16% over 2-opt and 2opt* local search, in the case of R102 reductions respectively are 7.5% and 5.3%. LNS clearly provides a more accurate local search.

Table 7
Results of executing *travel time minimisation procedure* with different local search procedures at execution time 0.5 s

Instance			2-opt & 2-opt*	LNS	Δ
R101	Best average travel time		1,608.3	1,348.4	16.2%
	Average travel time		1,781.0	1,389.0	22.0%
	Worst travel time		1,893.5	1,460.0	22.9%
	Average Iterations	2-opt	178.9	7,544.2	
		2-opt*	192.7		
R201	Best average travel time		1,059.9	1,003.5	5.3%
	Average travel time		1,132.0	1,046.5	7.5%
	Worst travel time		1,265.3	1,131.2	10.6%
	Average Iterations	2-opt	239.9	4,073.1	
		2-opt*	229.1		

Solomon instances (100 customers) – Figliozzi (2012) speed case TD1a.

Although Figliozzi (2012) implementation for the TDVRPTW was based on the 'ruin-and-recreate' concept, alternative destruction and recreation procedures were proposed rather than LNS procedures. The route improvement procedure consisted of iteratively removing all the customers in selected vehicles in order to rearrange them with a fast heuristic introduced by the author. The criteria to select the vehicles were: a) geographical proximity (distance between any two routes' centre of gravity), b) number of customers in vehicles.

Donati et al. (2008) made use of well-known neighbourhood operators and algorithm tailoring was based on restricting movements taking under consideration customer proximity and the introduction of a variable called "slack time" for each delivery in order to evaluate how long the delivery could be delayed so subsequent visits in the route will not violate time windows in the search process in the presence of time-dependent travel times.

This research consistently provides improved results for the TDVRPTW when compared with previous implementations. It is based on the capabilities of LNS movements to provide a fast and wide exploration of the search space that can quickly reach high accurate results in the presence of time-dependent travel times and time windows. Furthermore, taking advantage of the capabilities of LNS, additionally tailoring of destruction and recreation procedures also achieved high accuracy in the minimisation of required number of vehicles.

6. Conclusions

It is clear from the literature that time-dependent algorithms are necessary to substantially improve vehicle planning and scheduling in congested environments, with existing approaches that do not take congestion into consideration leading to extra time and missed deliveries. However, the added complexities of including time-dependent functions in models requires increased computational process capacity to provide as near optimal results as constant-speed models.

Tailoring algorithms to effectively and efficiently solve VRP variants is proven to be a challenging task. In this research, it is shown how two different strategies are used to solve different elements of the time-dependent vehicle routing problem with hard time windows.

For the minimisation of vehicles it was necessary to have a very specific approach of minimisation of time window violations in order to focus the search in scheduling customers that are particularly difficult to accommodate. Large Neighbourhood Search procedures guided with Variable Neighbourhood Search achieved high accuracy with the proposed algorithm tailoring. It provided a reduction of 4.15% vehicles than previous implementations in the 672 test instances.

Travel time or travel distance minimisation strategy was based on the search of distant regions in order to obtain a robust exploration of the search space. When compared to previous implementations, the algorithm was capable to obtain reductions in some test problems up to 19.60% in travel time and distance with fewer vehicles. It consistently provided improved solutions in the 672 test instances. Although the proposed algorithm makes use of random elements to escape from local optima and can therefore be run on a single processor core if required, parallel computing is also demonstrated here to take advantage of current processor architecture to execute multiple threads and explore different regions of the search space simultaneously without increasing the overall time of the search. Large Neighborhood Search is a strategy that stands out in vehicle routing problem solution algorithms due to its simplicity and wider applicability to solve different variants. The proposed approach shows its capacity to provide planners and drivers with accurate and reliable schedules when congestion is present using current computer architecture in reasonable time with adequate algorithm tailoring.

References

Bräysy, O. (2003). A reactive variable neighborhood search for the vehicle-routing problem with time windows. *INFORMS Journal on Computing, 15*(4), 347-368.

Bräysy, O., & Gendreau, M. (2005a). Vehicle routing problem with time windows, Part I: Route construction and local search algorithms. *Transportation science, 39*(1), 104-118.

Bräysy, O., & Gendreau, M. (2005b). Vehicle routing problem with time windows, Part II: Metaheuristics. *Transportation science, 39*(1), 119-139.

Bräysy, O., & Hasle, G. (2014). Software Tools and Emerging Technologies for Vehicle Routing and Intermodal Transportation. *Vehicle Routing: Problems, Methods, and Applications, 18*, 351.

Chang, Y. S., Lee, Y. J., & Choi, S. B. (2015). More Traffic Congestion in Larger Cities?-Scaling Analysis of the Large 101 US Urban Centers.

Chopra, S., & Meindl, P. (2007). *Supply chain management. Strategy, planning & operation*: Springer.

Cordeau, J.-F., Gendreau, M., Laporte, G., Potvin, J.-Y., & Semet, F. (2002). A guide to vehicle routing heuristics. *Journal of the Operational Research society, 53*(5), 512-522.

Dabia, S., Ropke, S., Van Woensel, T., & De Kok, T. (2013). Branch and price for the time-dependent vehicle routing problem with time windows. *Transportation science, 47*(3), 380-396.

DFT. (2010). Integrated Research Study: HGV Satellite Navigation and Route Planning. Retrieved April 01, 2013, from http://www.freightbestpractice.org.uk/products/3705_7817_integrated-research-study--hgv-satellite-navigation-and-route-planning.aspx.

Donati, A. V., Montemanni, R., Casagrande, N., Rizzoli, A. E., & Gambardella, L. M. (2008). Time dependent vehicle routing problem with a multi ant colony system. *European journal of operational research, 185*(3), 1174-1191.

Drexl, M. (2012). Rich vehicle routing in theory and practice. *Logistics Research, 5*(1-2), 47-63.

Eglese, R., Maden, W., & Slater, A. (2006). A road timetableTM to aid vehicle routing and scheduling. *Computers & operations research, 33*(12), 3508-3519.

Ehmke, J. F., Steinert, A., & Mattfeld, D. C. (2012). Advanced routing for city logistics service providers based on time-dependent travel times. *Journal of Computational Science, 3*(4), 193-205.

Figliozzi, M. A. (2012). The time dependent vehicle routing problem with time windows: Benchmark problems, an efficient solution algorithm, and solution characteristics. *Transportation Research Part E-Logistics and Transportation Review, 48*(3), 616-636. doi: DOI 10.1016/j.tre.2011.11.006

Fleischmann, B., Gietz, M., & Gnutzmann, S. (2004). Time-varying travel times in vehicle routing. *Transportation science, 38*(2), 160-173.

FTA. (2015). Logistics Report 2015.Delivering Safe, Efficient, Sustainable Logistics. In FTA (Ed.), About. Tunbridge Wells: FTA.

Gendreau, M., Ghiani, G., & Guerriero, E. (2015). Time-dependent routing problems: A review. *Computers & operations research, 64*, 189-197.

Golden, B., Wasil, E., Kelly, J., & Chao, I. (1998). Fleet Management and Logistics, chapter The Impact of Metaheuristics on Solving the Vehicle Routing Problem: algorithms, problem sets, and computational results: Kluwer Academic Publishers, Boston.

Haghani, A., & Jung, S. (2005). A dynamic vehicle routing problem with time-dependent travel times. *Computers & operations research, 32*(11), 2959-2986.

Hallamaki, A., Hotokka, P., Brigatti, J., Nakari, P., Bräysy, O., & T, R. (2007). Vehicle Routing Software: A Survey and Case Studies with Finish Data. Technical Report. Finland: University of Jyväskylä.

Hansen, P., & Mladenović, N. (2001). Variable neighborhood search: Principles and applications. *European journal of operational research, 130*(3), 449-467.

Hansen, P., Mladenović, N., & Pérez, J. A. M. (2010). Variable neighbourhood search: methods and applications. *Annals of Operations Research, 175*(1), 367-407.

Ichoua, S., Gendreau, M., & Potvin, J.-Y. (2003). Vehicle dispatching with time-dependent travel times. *European journal of operational research, 144*(2), 379-396.

IMRG. (2012). UK valuing home delivery review 2012. In IMRG (Ed.).

Javelin-Group. (2011). How many stores will we really need? UK non-food retailing in 2020. In L. J. Group. (Ed.).

Jiang, J., Ng, K. M., Poh, K. L., & Teo, K. M. (2014). Vehicle routing problem with a heterogeneous fleet and time windows. *Expert Systems with Applications, 41*(8), 3748-3760.

Kok, A., Hans, E., & Schutten, J. (2012). Vehicle routing under time-dependent travel times: the impact of congestion avoidance. *Computers & operations research, 39*(5), 910-918.

Kritzinger, S., Tricoire, F., Doerner, K. F., & Hartl, R. F. (2011). Variable neighborhood search for the time-dependent vehicle routing problem with soft time windows *Learning and Intelligent Optimization* (pp. 61-75): Springer.

Kytöjoki, J., Nuortio, T., Bräysy, O., & Gendreau, M. (2007). An efficient variable neighborhood search heuristic for very large scale vehicle routing problems. *Computers & operations research, 34*(9), 2743-2757.

Liu, F.-H., & Shen, S.-Y. (1999). The fleet size and mix vehicle routing problem with time windows. *Journal of the Operational Research society*, 721-732.

Malandraki, C. (1989). Time dependent vehicle routing problems: Formulations, solution algorithms and computational experiments.

Malandraki, C., & Daskin, M. S. (1992). Time dependent vehicle routing problems: Formulations, properties and heuristic algorithms. *Transportation science, 26*(3), 185-200.

Mattos Ribeiro, G., & Laporte, G. (2012). An adaptive large neighborhood search heuristic for the cumulative capacitated vehicle routing problem. *Computers & operations research, 39*(3), 728-735.

Mladenović, N., & Hansen, P. (1997). Variable neighborhood search. *Computers & operations research, 24*(11), 1097-1100.

Nagata, Y., Bräysy, O., & Dullaert, W. (2010). A penalty-based edge assembly memetic algorithm for the vehicle routing problem with time windows. *Computers & operations research, 37*(4), 724-737.

Pisinger, D., & Ropke, S. (2007). A general heuristic for vehicle routing problems. *Computers & operations research, 34*(8), 2403-2435.

Polacek, M., Hartl, R. F., Doerner, K., & Reimann, M. (2004). A variable neighborhood search for the multi depot vehicle routing problem with time windows. *Journal of heuristics, 10*(6), 613-627.

Rincon-Garcia, N., Waterson , B. J., & Cherret, T. J. (2015). Requirements from Vehicle Routing Software: Perspectives from literature, developers and the freight industry. *Under Review*.

Ropke, S., & Pisinger, D. (2006). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation science, 40*(4), 455-472.

Schrimpf, G., Schneider, J., Stamm-Wilbrandt, H., & Dueck, G. (2000). Record breaking optimization results using the ruin and recreate principle. *Journal of Computational Physics, 159*(2), 139-171.

Shaw, P. (1997). A new local search algorithm providing high quality solutions to vehicle routing problems. *APES Group, Dept of Computer Science, University of Strathclyde, Glasgow, Scotland, UK*.

Shaw, P. (1998). Using constraint programming and local search methods to solve vehicle routing problems *Principles and Practice of Constraint Programming—CP98* (pp. 417-431): Springer.

Solomon, M. M. (1987). Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints *Operations Research, 35*(2), 254-265.

Sörensen, K. (2015). Metaheuristics—the metaphor exposed. *International Transactions in Operational Research, 22*(1), 3-18.

Toth, P., & Vigo, D. (2001). *The vehicle routing problem*: Siam.

US-Department-Of-Transport. (2003). Final Report -Traffic Congestion and Reliability: Linking Solutions to Problems.

Vidal, T., Crainic, T. G., Gendreau, M., & Prins, C. (2013). A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. *Computers & operations research, 40*(1), 475-489.

Visser, J., Nemoto, T., & Browne, M. (2014). Home delivery and the impacts on urban freight transport: A review. *Procedia-social and behavioral sciences, 125*, 15-27.

Zachariadis, E. E., & Kiranoudis, C. T. (2010). A strategy for reducing the computational complexity of local search-based methods for the vehicle routing problem. *Computers & operations research, 37*(12), 2089-2105.

Appendix A: Travel time function

Travel time between any two given customers i,j depends on the travel speed function, specific data format and departure time from customer i, the following algorithm proposed by Ichoua et al. (2003) (with notation of Figliozzi (2012)), returns arrival time a_j to customer j when departing from customer i at b_i:

```
Data:
  $T=T_1, T_2, \ldots, T_p$ where each period $T_k$ has an associated constant travel speed $s_k$, an initial time $t_{\underline{k}}$ and
    a final time $t_{\overline{k}}$
Start
  if $a_i < e_i$ then
    $b_i \leftarrow e_i + g_i$
  else
    $b_i \leftarrow a_i + g_i$
  endif
  find k where $t_{\underline{k}} \leq b_i \leq t_{\overline{k}}$
  $a_j \leftarrow b_i + d_{ij}/s_k$
  $d \leftarrow d_{ij},\ t \leftarrow b_i$
  while $a_j > t_{\overline{k}}$ do
    $d \leftarrow d - (t_{\overline{k}} - t)\, s_k$
    $t \leftarrow t_{\overline{k}}$
    $a_j \leftarrow t + d_{ij}/s_{k+1}$
    $k \leftarrow k+1$
  end while
  Return: $a_j$, arrival time at customer j
End
```