Expert Systems with Applications 40 (2013) 1696–1707

Contents lists available at SciVerse ScienceDirect

Expert Systems with Applications

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

A Simulated Annealing-based parallel multi-objective approach to vehicle routing problems with time windows

Raúl Baños [a,*], Julio Ortega [a], Consolación Gil [b], Antonio Fernández [b], Francisco de Toro [c]

[a] Dpt. Computer Architecture and Technology, CITIC-UGR (Research Centre on Information and Communications Technology), University of Granada, C/ Periodista Daniel Saucedo s/n, 18071 Granada, Spain
[b] Dpt. Computer Architecture and Electronics, University of Almería, Carretera de Sacramento s/n, 04120 Almería, Spain
[c] Dpt. Signal Theory, Telematics and Communications, CITIC-UGR (Research Centre on Information and Communications Technology), University of Granada, C/ Periodista Daniel Saucedo s/n, 18071 Granada, Spain

ARTICLE INFO

Keywords:
Multi-objective meta-heuristics
Parallel processing
Simulated Annealing
Vehicle routing
Time windows
Route balancing

ABSTRACT

The Capacitated Vehicle Routing Problem with Time Windows (VRPTW) consists in determining the routes of a given number of vehicles with identical capacity stationed at a central depot which are used to supply the demands of a set of customers within certain time windows. This is a complex multi-constrained problem with industrial, economic, and environmental implications that has been widely analyzed in the past. This paper deals with a multi-objective variant of the VRPTW that simultaneously minimizes the travelled distance and the imbalance of the routes. This imbalance is analyzed from two perspectives: the imbalance in the distances travelled by the vehicles, and the imbalance in the loads delivered by them. A multi-objective procedure based on Simulated Annealing, the Multiple Temperature Pareto Simulated Annealing (MT-PSA), is proposed in this paper to cope with these multi-objective formulations of the VRPTW. The procedure MT-PSA and an island-based parallel version of MT-PSA have been evaluated and compared with, respectively, sequential and island-based parallel implementations of SPEA2. Computational results obtained on Solomon's benchmark problems show that the island-based parallelization produces Pareto-fronts of higher quality that those obtained by the sequential versions without increasing the computational cost, while also producing significant reduction in the runtimes while maintaining solution quality. More specifically, for the most part, our procedure MT-PSA outperforms SPEA2 in the benchmarks here considered, with respect to the solution quality and execution time.

1. Introduction

The vehicle routing problem is a type of combinatorial optimization problems that frequently occurs in real life applications, like transportation, logistics, and scheduling. Since the family of vehicle routing problems is included in the category of NP-hard problems (Lenstra & Rinnooy Kan, 1981), they are hard to solve, especially when the number of customers is large (Lee, Lee, Lin, & Ying, 2010). They have been addressed using many different techniques including both exact and heuristic methods (Eksioglu, Vural, & Reisman, 2009), but the high computational cost of exact methods and their poor performance in large problems has involved that the current research concentrates on stochastic algorithms that are capable of producing feasible but not necessarily optimal solutions in limited time. Real-world optimization problems encountered in economics, engineering, or industry require the design of efficient algorithms. When the problem complexity is high, such as NP-hard problems, it is generally useful to apply heuristic and meta-heuristic methods, thanks to the fact that they support managers in decision-making with approximate solutions to complex problems in a quick way (Glover & Kochenberger, 2003).

Although routing problems have been the subject of a large number of papers, few of them apply computational optimization algorithms for solving multi-objective vehicle routing problems. However, the majority of real life problems are not limited to minimizing the operational costs derived from travel distance, but also other objectives can be considered (Jozefowiez, Semet, & Talbi, 2008). Therefore, methods that provide a range of solutions representing the trade-off between objectives are crucial for many real-world applications. For this purpose, Pareto-based multi-objective optimization is often applied with the aim of obtaining a set of non-dominated solutions, which is later analyzed by the decision maker in order to choose a particular solution based on particular criteria. Meta-heuristics also produce quality solutions in the multi-objective context, but this is usually at the expense of longer computation times. Therefore, the use of parallel processing

* Corresponding author. Tel.: +34 958243228; fax: +34 958248993.
E-mail addresses: rbanos@atc.ugr.es (R. Baños), julio@atc.ugr.es (J. Ortega), cgilm@ual.es (C. Gil), afdezmolina@ual.es (A. Fernández), ftoro@ugr.es (F. de Toro).

0957-4174/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.eswa.2012.09.012

becomes an interesting, if not the best, tool to obtain competitive solutions in an acceptable runtime.

Parallel processing allows the researchers both to explore the solution space more extensively and to accelerate the search process. The traditional fields of improvement in parallelism have been orientated to experimentation on high-budget equipment, such as clusters of computers or shared memory machines thanks to their high-performance and scalability. The generalized presence of multi-core microprocessors in almost all the computing platforms makes it possible to take advantage of parallel processing even for the desktop computer user without the need for accessing a high performance server. Over time, researchers have developed more powerful algorithms that find better solutions to single-objective vehicle routing problems by taking advantage of faster computational resources (Groër, Golden, & Wasil, 2011). However, the application of parallel processing to solve multi-objective variants of vehicle routing problems is still an open research field.

This paper proposes the use of parallel processing to improve the performance of Pareto-based algorithms when solving multi-criteria vehicle routing problems with time windows. The paper is further organized as follows: Section 2 offers a brief overview of the state of the art of the vehicle routing problems and formally describes the Capacitated Vehicle Routing Problem with Time Windows (VRPTW). Furthermore, some previous parallel implementations for solving routing problems and some multi-objective variants are described. Section 3 presents the sequential Simulated Annealing-based meta-heuristic, and its parallelization using the Island model. Results of the empirical analysis carried out in a multi-core workstation are given in Section 4, while conclusions are drawn in Section 5.

2. Vehicle routing problems: An overview

The aim of this section is to provide a brief but comprehensive overview of the previous research works related with vehicle routing problems with time windows using parallel and multi-objective optimization techniques.

2.1. The Capacitated Vehicle Routing Problem with hard Time Windows (VRPTW)

The Traveling Salesman Problem (TSP) is an optimization problem where only one vehicle (salesman) has to visit all the customers. Unfortunately, most practical transportation applications involve that either the customer demands exceed the vehicle capacity or that it is possible to reduce the total distance travelled by using more than one vehicle. The vehicle routing problem (VRP) was first introduced by Dantzig and Ramser (1959) as a generalization of the TSP that overcomes this drawback by using multiple vehicles in the transportation process. Many extensions of the basic VRP have been proposed in last decades to include aspects such as characteristics of the network, the customers' demands, the fleet, the costs, or the number of objectives, which make the problem more difficult to solve.

The Capacitated Vehicle Routing Problem with (hard) Time Windows (CVRPTW or simply VRPTW) is an important variant of the VRP which consists of routing a homogeneous fleet of vehicles with identical capacity stationed at a central depot (logistic center) which operates within a certain time windows. The vehicles are used to visit and fully supply the demands of a set of geographically scattered customers so that each vehicle route starts and ends at a central depot, and that the total demand met by any route cannot exceed the vehicle capacity. The customers can be supplied only once by exactly one vehicle and have predefined requirements of goods and a service time. The distance between customers is measured by Euclidean distances, and the total distance travelled by all the vehicles defines the traveling times. Each customer has a certain time window, which involves that the vehicle must begin the service to the customer within the time window defined by the earliest time and the latest time when the customer allows the start of the service (El-Sherbeny, 2010). Therefore, the optimization problem is to determine the route followed by each vehicle to serve its assigned customers, while the distance travelled by the vehicles is minimized and the capacity and time windows constraints are satisfied.

The VRPTW has been the subject of intensive research because it is one of the most difficult problems in combinatorial optimization and has a considerable economic impact on all logistic systems (Alvarenga, Mateus, & de Tomic, 2007), especially due to the importance of just-in-time production systems and the increasingly tight coordination of supply chain operations. Logistics, and especially the distribution of goods, lies at the heart of scientific research and business activity because it is often coupled with inventory and production decisions, and the delivery cost has a significant portion in the total logistic costs (Alabas-Uslu & Dengiz, 2011). Logistic managers aim to improve the design of their logistic systems taking appropriate decisions concerning the strategies to provide customers with their services while satisfying the company's logistic priorities according to the available vehicle fleet (Derbel, Jarboui, Hanafi, & Chabchoub, 2010). Some exact methods have been proposed for the VRPTW, including Lagrange relaxation-based methods, column generation, and dynamic programming (El-Sherbeny, 2010). However, exact methods often perform poorly in intermediate and large problem instances, especially in some VRP variants (Kritikos & Ioannou, 2010). Due to the problem difficulty, heuristic and meta-heuristic algorithms are highly suitable for solving larger VRPTW instances. Some examples are: genetic algorithms (Alvarenga et al., 2007; Cheng & Wang, 2009), tabu search (Cordeau & Maischberger, 2012), memetic algorithms (Nagata, Bräysy, & Dullaert, 2010), particle swarm optimization (Marinakis, Marinaki, & Dounia, 2010), etc., and results demonstrate that these methods produce good solutions in reduced runtime (Bräysy & Gendreau, 2005).

2.2. Multi-objective VRPTW: Related work

Most real-world optimization problems, including the VRP and its variants, are multi-objective in nature, since they involve several objectives that must be optimized at the same time. However, so far, the majority of research papers in the field of VRPs is concentrated in the simplification of these problems using two main alternatives: to establish a hierarchy between the objectives to be optimized and to use aggregating approaches where all these objectives are included in a objective function using a combination of mathematical operations. The drawback in the former case is that it intrinsically involves the consideration of one objective as more important than others, while the main disadvantage of using an objective reduction approach is that the weights are difficult to determine precisely, particularly when there is often insufficient information or knowledge concerning the large real-world vehicle routing problems (Tan, Chew, & Lee, 2006).

Therefore, methods that provide a range of solutions representing the trade-off between objectives is crucial for many real-world applications and Pareto-based strategies (Goldberg, 1989) are well positioned for this purpose. Pareto-optimization methods establish relationships between solutions according to dominance relations in the following manner: it is said that a solution s_1 dominates (or is preferred) to another $s_2 (s_1 \succeq s_2)$ if and only if s_1 is better than s_2 in at least one objective, and not worse in the others. Two solutions are called indifferent if neither of them dominates the other one. Since all the objectives are considered as equally important, the aim of Pareto-based multi-objective optimization is to find a set of non-dominated solutions as representative of the true

Pareto-optimal front, which in complex problems is unknown. The practical advantage of Pareto-based multi-objective algorithms is that they provide a set of trade-off solutions that can be used by the decision maker to select one of the solutions according to specific criteria.

Multi-objective vehicle routing problems extend and generalize classic academic problems in order to improve their practical application and to study real-life cases in which the objectives have been clearly identified by the decision-maker (Jozefowiez et al., 2008). Probably, the most studied multi-objective formulation is the use of a hierarchical approach to minimize the number of vehicles used and the traveling distance, where the optimization algorithm finds first the solution that minimizes the number of tours (primary objective), and for the same number of tours then minimizes the total travelled distance (secondary objective) (Ombuki, Ross, & Hanshar, 2006; Nagata et al., 2010). However, several practical aspects suggest a direct consideration of the total traveling distance as the objective to be optimized instead of using the number of tours as primary objective. Firstly, the vehicles often belong to the company and therefore their cost is almost independent of their use, and the drivers often work either for the owner of the company of the goods or for the owner of the company of the vehicles, which is why they are remunerated in some way even in the case of not driving any vehicle. On the other hand, the traveling distance has a higher economical impact since the petrol consumption is the most expensive variable cost of the transportation process. Besides, since the traveling time depends on the traveling distance, this objective is more accurate when transporting perishable goods. Finally, these two objectives may be positively correlated with each other. The last reason is especially important from the viewpoint of multi-objective optimization, since some authors have demonstrated that both objectives (number of vehicles and traveling distance) may be positively correlated, or they may be conflicting, i.e. fewer vehicles employed in service do not necessarily increase the traveling distance (Tan et al., 2006).

Recently, some other multi-objective formulations have been proposed. Jozefowiez et al. (2008) classified the different objectives according to the component of the problem with which they are associated, i.e. the tour, the resources, and the node/arc activity. In particular, the most common objective related to the tour is to minimize the costs of the solutions, in terms of the distance travelled or the time required, while other approaches aim to minimize the length of the longest tour (makespan), to reduce the disparity (imbalance), etc. Workload imbalance can be expressed as the quantity of goods delivered, the time required or the tour length, number of customers visited, etc. Borgulya (2008) proposed a heuristic procedure for the CVRP that aims to find the route of each vehicle so as to accomplish in the shortest time period the visiting of all the demands nodes of the network. Jozefowiez, Semet, and Talbi (2009) applied a meta-heuristic method based on an evolutionary algorithm involving classical multi-objective operators to address a bi-objective CVRP, in which the objectives are to minimize the total length of routes and to balance the route lengths, i.e. minimizing the difference between the longest and the shortest routes. Recently, Kritikos and Ioannou (2010) proposed a variant, named balanced cargo VRPTW, that targets the balancing of the load carried by each active vehicle.

It is typical to require balanced load distribution among vehicles that could guarantee equivalent travel plus service times for each of them, and would not cause problems such as driver dissatisfaction (Kritikos & Ioannou, 2010). However, so far, there are few papers dealing with multi-objective formulations of the VRPTW that consider the workload imbalance. Therefore, a major research effort is necessary in this field, including the use of parallel approaches that take advantage of the nowadays common multi-core microprocessor architectures to improve the quality of the solutions and/or to reduce the runtime to get them. Here, the vehicle routing problem with hard time windows and workload balancing (here called WB-VRPTW) where in addition to the traveling distance (TD), the routes imbalance is also minimized is analysed. The imbalance is analyzed from two viewpoints: the imbalance in the distances travelled by the vehicles used (distance imbalance, DI) and the imbalance in the loads of these vehicles (load imbalance, LI). This formulation allows us to obtain solutions with different travel distances and route imbalances, so that the decision maker will decide, based on specific solutions, which of them is the most suitable one according to certain criteria.

2.3. VRPTW: Model formulation

The mathematical definition of the objectives and constraints of the VRPTW, including the frequently used notations such as route, depot, customer, and vehicles can be modeled as a graph theoretic problem (El-Sherbeny, 2010). Let $G = (V,E)$ be a non-directed complete graph, where the vertices $V = \{1,\ldots,N\}$ correspond to the depot and the customers, and the edges $e \in E\{(i,j):i,j \in V\}$ to the links between them.

Decision variable

$$X_{ij}^k = \begin{cases} 1 & \text{if vehicle } k \text{ travels from node } i \text{ to node } j \\ 0 & \text{otherwise} \end{cases}$$

Parameters

- a_j is the earliest time to allow the service by customer j,
- b_j is the latest time to allow the service by customer j,
- C_{ij} is the cost for traveling from node i to node j (here, C_{ij} is considered as the distance or time required for traveling from node i to node j),
- d_j is the demand at customer j,
- K is the maximum number of vehicles that can be used,
- N is the number of customers plus the depot (the depot is noted with number 1, and the customers are noted as 2, ..., N),
- Q^k is the loading capacity of vehicle k (the same capacity, Q, will be considered for all vehicles).

The VRPTW can be stated as follows:

$$\text{minimize}: \quad TD = \sum_{k=1}^{K}\sum_{i=1}^{N}\sum_{j=1}^{N} X_{ij}^k C_{ij} \tag{1}$$

$$\text{subject to}: X_{ii}^k = 0 \qquad (\forall i \in \{1,..,N\}, \forall k \in \{1,..,K\}) \tag{2}$$

$$X_{ij}^k \in \{0,1\} \qquad (\forall i,j \in \{1,..,N\}, \forall k \in \{1,..,K\}) \tag{3}$$

$$\sum_{k=1}^{K}\sum_{i=1}^{N} X_{ij}^k = 1 \qquad (\forall j \in \{2,..,N\}) \tag{4}$$

$$\sum_{i=1}^{N}\sum_{j=2}^{N} X_{ij}^k d_j \leqslant Q^k \qquad (\forall k \in \{1,..,K\}) \tag{5}$$

$$\sum_{k=1}^{K}\sum_{j=2}^{N} X_{1j}^k \leqslant K \tag{6}$$

$$\sum_{j=2}^{N} X_{1j}^k - \sum_{j=2}^{N} X_{j1}^k = 0 \qquad (\forall k \in \{l,...,K\}) \tag{7}$$

$$a_j \leqslant s_{kj} \leqslant b_j \qquad (\forall i,j \in \{1,..,N\}, \forall k \in \{1,..,K\}) \tag{8}$$

$$s_{ki} + C_{ij} - L\left(1 - X_{ij}^k\right) \leqslant s_{kj} \quad (\forall i,j \in \{1,..,N\}, \forall k \in \{1,..,K\}) \tag{9}$$

Eq. (1) is the objective function of the problem. Eq. (2) denotes that a vehicle must travel from one node to another different. Eq. (3) indicates that X_{ij}^k is equal to 1 if vehicle k goes from node i to node j, and it is equal to 0 otherwise, i.e. a route between

two cus tomers can or cannot be covered by a vehicle. Eq. (4) states that a customer is visited once by exactly one vehicle. By specifying the constraint of Eq. (5), it is taken into account that for a given vehicle k, the load that has to be transported to complete the routes assigned to such vehicle cannot exceed its capacity Q^k (it is considered that all vehicles have the same capacity, $Q^k = Q$). Eq. (6) specifies that there are up to K routes going out of the delivery depot. Eq. (7) guarantees that the vehicles depart from and return to the depot. Let s_{kj} be the sum of the distances travelled by vehicle k before arriving customer j. Eq. (8) ensures that time windows are observed. Given a large scalar, L, the inequality represented in Eq. (9) specifies that, if vehicle k is traveling from customer i to customer j, the vehicle cannot arrive at customer j before $s_{ki} + C_{ij}$. As specified by El-Sherbeny (2010), the variable s_{kj} corresponds to the time vehicle k starts to service customer j. If the vehicle k does not service j, s_{kj} is not calculated.

2.4. Parallel algorithms for vehicle routing problems: Related work

In computational optimization, the use of parallel computing involves that several processes work simultaneously on different processors with the common goal of solving a given problem instance. The use of parallel processing systems allows the improvement of the quality of the results of the sequential versions, to reduce the time required to solve the sequential algorithms, or to solve larger problem instances without increasing the computational effort. Scalable computing platforms, ranging from clusters of computers to shared memory machines, are platforms that provide high-performance and large-scale computing depending on their size. Nevertheless, there exist physical barriers that limit both the design of processors doubling their clock frequencies periodically and the simultaneous improvement in clock frequencies and instructions per cycle in the superscalar processors. Fortunately, the introduction of multi-core processors, that increase the computational power by adding more processing cores and a shared memory cache in a single microchip, allows parallel processing on low-cost multi-core computers using standard software components (Márquez, Gil, Baños, & Gómez, 2011).

Parallelism can be exploited in several ways. Three main strategies usually applied to parallelize meta-heuristics are: master-worker, island, and diffusion paradigms (Alba, 2005; Coello, Lamont, & Van Veldhuizen, 2007). The master-worker paradigm is the simplest approach, where the master processor distributes the current solution among the worker processors so that each one explores its part of the neighborhood and evaluates the objective function before sending its result to the master. The master processor is responsible for other functions such as collecting and distributing sub-populations. Therefore, the search space exploration is conceptually identical to that of the sequential executions. The main disadvantage of this approach is that it requires a high degree of synchronization and can involve important overheads. In particular, in those implementations in which the worker processors are only responsible of evaluating the objective functions, it is only possible to obtain advantages in terms of runtime if these objective functions are complex and time consuming. Otherwise, the communication time could become larger than the computation time thus increasing the runtime.

The island-based algorithms, also termed distributed or coarse-grained, consist in dividing the entire population of the sequential algorithm into several sub-populations distributed among different processors. These islands or sub-populations, whose number often coincides with the number of processors/cores, evolve in isolation executing all the steps of the meta-heuristic, while it is also allowed to migrate solutions between islands. Two advantages of the island-based paradigm are: synchronization only occurs in these communication iterations, which reduces the overheads in comparison with the master-worker paradigm, and using different initial solutions or different parameter settings in each island involves intrinsically searching many different regions of the search space, thus improving the quality of the solutions.

The diffusion paradigm deals with one conceptual population like the master-worker paradigm, but this population contains only a few individuals, which is why diffusion parallelism is also termed as fine-grained parallelism. Additionally, there is a fourth category named hierarchical hybrid, which consists in the combination of the previous ones (Coello et al., 2007).

In the field of vehicle routing problems, parallelization provides benefits in practical settings where routes must be produced within a short time interval. Since there is a clear trade-off between computational times and solution quality, parallel processing can increase service quality by allowing a larger number of requests to be routed within a reasonable runtime. Bräysy and Gendreau (2005) noticed the importance of designing faster meta-heuristics for solving VRPs incorporating speed-up techniques such as parallel implementations. Several parallel heuristic algorithms have been implemented for solving VRPs, including local search (Badeau, Guertin, Gendreau, Potvin, & Taillard, 1997; Rego, 2001; Groër et al., 2011; Cordeau & Maischberger, 2012), genetic algorithms (Berger & Barkaoui, 2004; Le Bouthillier & Crainic, 2005), problem decomposition (Taillard, 1993), etc. Nevertheless, almost all of them have been applied to solve single-objective formulations and only some authors have applied parallel processing to solve the two-phase VRPTW described above (Gehring & Homberger, 2002). One of the few parallel multi-objective approaches was developed by Jozefowiez et al. (2009), who solved the Capacitated Vehicle Routing Problem (without time windows) applying a co-operative parallel model based on the general island model focused to search a larger part of the solution space in a given time. To our knowledge, the application of parallel processing to multi-objective formulations of the VRPTW considering traveling distance and route balancing (distance imbalance and load imbalance) has not been reported in the literature.

3. Multi-temperature Pareto simulated annealing: Sequential and parallel implementations

This section describes the main characteristics of Simulated Annealing-based sequential multi-objective algorithm (MT-PSA), and how it have been parallelized using the island-based paradigm (pMT-PSA).

3.1. Sequential algorithm

The main components of the sequential algorithm here proposed are: the representation of the solutions, the method to construct initial solutions, the search operators, the strategy to manage the external archives and keep diversity in the populations, the neighborhood structures, the selection of a neighboring solution as the new current solution, and the termination conditions. These topics are now described:

3.1.1. Representation of the solutions

Most algorithms for solving vehicle routing problems use integer representation. This way, multi-objective approaches use a population P of p individuals (solutions), $P = \{P_1, P_2, \ldots, P_p\}$, where each individual represents the routes travelled by K vehicles used to deliver to all the demand nodes. Therefore, each individual, P_i, is represented by a set of chromosomes, C_{ik}, which consists of a variable number of genes, $C_{ik} = \left\{1, G^1_{ik}, G^2_{ik}, \ldots, G^l_{ik}, 1\right\}$ representing the route of the kth vehicle in the ith individual, with $2 \leqslant G^j_{ik} \leqslant N$. For example, chromosome $C_{6,3}$ = {1,4,19,41,33,85,1} indicates that

the third vehicle of the sixth individual departs from the depot, and visits customers 4, 19, 41, 33 and 85 before returning to the depot, which is represented by the identificator 1.

3.1.2. Construction of the solutions

The construction procedure used in our implementations is called time window based insertion heuristic (TWIH). This strategy starts by initializing the route under consideration with that customer that first opens its time windows. Other customers are then included in the current route according to an ascending order in their opening time windows, such that those ones which are available earlier are first visited by the vehicles whenever the time windows and capacity constraints are fulfilled. This procedure is very fast and intrinsically reduces the number of vehicles initially used.

3.1.3. Search operators

The feasible initial solutions obtained using the time window based insertion heuristic are improved by using search operators. The literature provides a large number of evolutionary search operators for VRP (Tan et al., 2006; Alvarenga et al., 2007; El-Sherbeny, 2010; Müller, 2010; Garcia-Najera & Bullinaria, 2011). In order to increase the diversification in the search, ten of these operators have been implemented. Some of them are based on modifying the vehicle assigned to the customers (Customer random migration, Customer best migration, Customer random exchange, Customer best exchange, Customer exchange with similar time-window), reallocating them in a different visiting order of the same vehicle (Customer random reallocation, Customer best reallocation), while other operators divide (Route partition), create (New route), or remove (Route elimination) a given route. These ten mutation operators are described in Fig. 1.

On the other hand, a typical crossover operator applied by some multi-objective evolutionary algorithms is graphically described in Fig. 2. It consists of taking a given number of routes from the first parent, which are copied in the child individual. After that, all the routes of the second parent that are not in conflict with the previously inserted ones are also included in the child. Finally, if there are customers not visited in the child individual, they are inserted in the existing routes starting from the shorter routes. If this insertion is not possible in one of the existing routes without violating the constraints, a new route is created, which is also used for insertion of other not assigned customers.

3.1.4. Improvement of solutions using Simulated Annealing

The initial solutions built by the construction methods must be improved by applying the search strategies. In our study, the solution improvement is controlled by a Simulated Annealing-based approach. Simulated Annealing (SA) (Kirkpatrick, Gelatt, & Vecchi, 1983) is a local search meta-heuristic that simulates the process undergone by misplaced atoms in a metal when heated and then slowly cooled. SA optimizes a solution by exposing it to high initial temperature, Ti, cooling it by means of a cooling rate, $T_{cooling}$, until the temperature falls below a given threshold, T_{stop}. Therefore, better neighboring solutions are always accepted, whereas worse solutions are accepted with a certain probability, which is dependent on the current temperature, t. Some authors have applied SA in the multi-objective context, but only few of them incorporate the concept of Pareto-dominance (Bandyopadhyay, Saha, Maulik, & Deb, 2008). In this paper, it is proposed the Multiple Temperature Pareto Simulated Annealing (MT-PSA). MT-PSA is a local search-based strategy that also uses a population of solutions P which are improved using mutation operators. Since the problem here solved is multi-objective, mutated solutions are accepted or rejected using a variation of the typical acceptance criterion used by SA: if the new solution obtained after the mutation is not dominated by its parent, it is directly accepted, while if the parent dominates the child solution, the latter is accepted according to the criterion of Metropolis, Rosenbluth, Rosenbluth, Teller, and Teller (1953), where the sum of the fitness values variations in the objectives considered and the current temperature are included as parameters. MT-PSA also uses an external archive of non-dominated solutions (ND). Thus, after applying a mutation, the new solution is inserted in the external archive only if it is not dominated by any of the solutions located in ND, then removing those solutions which were previously stored in the external archive but being now dominated by the new solution. When the number of non-dominated individuals exceeds the size of this archive, a truncation procedure that guarantees the preservation of boundary solutions is applied. Additionally, this method uses a multiple temperature strategy in which each individual is assigned a particular initial temperature that determines its annealing scheme with the aim of reducing the effect derived from selecting fixed annealing schemes. Therefore, an interval of initial temperatures, $[Ti_{min}, Ti_{max}]$ is established so that solution P_1 starts in Ti_{min}, solution P_p starts in Ti_{max}, and the others are equally distributed along this interval. This multiple temperature strategy was successfully applied when solving the single-objective graph partitioning problem (Baños, Gil, Ortega, & Montoya, 2004), and here it is extended to the multi-objective context to solve the WB-VRPTW problem.

3.2. Parallelization using the Island model

The aim of the parallel MT-PSA (pMT-PSA) is to obtain Pareto-fronts of better quality than the sequential algorithms without increasing the runtime. Of course, it is also possible to reduce the time required to obtain or even to improve the solution qualities. Thus, our parallel algorithms use the island paradigm so that each island works autonomously to improve the solutions, but also cooperate through the exchange of solutions. Some studies have shown that parallel search strategies based on using different parameter settings in each process allow to explore the solution space more extensively and to find better solutions (Rego, 2001). With this aim, the pMT-PSA uses different annealing scheduling in the individuals of the population, but it is also assumed that each island is capable of generating its own annealing process using different initial temperatures, thus providing more diversity to the search process. Moreover, when the migration is performed using pMT-PSA, the received solutions are improved using the same annealing parameters of the previously replaced solution.

Algorithm 1 describes the operation of this parallel framework when applying MT-PSA (pMT-PSA), where the input parameters are the number of processes, np, the population size p, the termination criteria TC, the probabilities of applying crossover, $Pr_{crossover}$, and mutation, $Pr_{mutation}$, the migration rate MR (number of iterations or runtime between migrations), and the Simulated Annealing parameters: T_{min}, T_{max}, $T_{cooling}$, T_{stop}. The high-level Message Passing Interface (MPI) is then initialized (line 2), after which the same code runs in each process (lines 3–26). In order to avoid confusions with the notation used to denote the individuals of the population, P_j, each process is denoted by the acronym C_i as it runs in one processing core, while $C_i[P_j]$ makes reference to the jth individual of the sub-population managed by the process C_i. Therefore, each process initializes its sub-population of solutions according to the time window based insertion heuristic described above (line 4), and each solution is assigned its initial temperature according to its identification number and the pre-established interval of initial temperatures (line 5). The main loop (lines 6–25) is repeated until the termination criterion is fulfilled. The inner loop (lines 7–14) consists in evolving the population of solutions so that mutation operators are applied to modify the current solution (line 8). Let us notice that, if mutation is applied, each of the ten mutation variants have the same probability of being applied (10%). The new solution obtained after applying one of the mutation operators

is accepted or rejected according to the multi-objective Metropolis criterion described above (line 9). The new solution is then included in the external archive (*ND*) if it is not dominated by any of those solutions inserted in *ND* (lines 10–11), taking into account that in case that the number of solutions inserted in *ND* exceeds its maximum allowed size (|*ND*|), a procedure is applied to remove some of these solutions, while preserving the boundary solutions (i.e. those non-dominated solutions having the best fitness values in one of the objectives to optimize). The temperature is updated according to the current temperature and the cooling rate (line 12), and if this temperature falls below T_{stop} it is re-initialized (line 13–14). Before continuing the search process, it is analyzed whether the cores perform communications in this iteration (lines 15–25) or not. If there are communications, a central process receives the non-dominated solutions from the remaining cores, composes the global non-dominated front, and sends it to the other cores, which continue the search process with this composed front.

Algorithm 1: parallel MT-PSA

1. Input: np, p, TC, $Pr_{mutation}$, MR, T_{min}, T_{max}, $T_{cooling}$, T_{stop};
2. Initialize MPI in each process C_i, $i \in [1 \ldots np]$
3. For each individual $P_j \in C_i[P]$
4. $C_i[P_j] \leftarrow$ TWIH ();
5. $C_i[P_j] \cdot t \leftarrow T_{min} + j*((T_{max} - T_{min})/p)$;
6. While (not TC)
7. For each $C_i[P_j] \in C_i[P]$
8. $(C_i[P_j])' \leftarrow$ Mutation ($C_i[P_j]$,$Pr_{mutation}$,random (1,10));
9. $C_i[P_j] \leftarrow$ Metropolis ($C_i[P_j]$, $(C_i[P_j])'$,$C_i[P_j] \cdot t$);
10. If ($\nexists C_i[ND_k] \succeq C_i[P_j]$, $\forall C_i[ND_j] \in C_i[ND]$) then
11. $C_i[ND] \leftarrow C_i[ND] \cup C_i[P_j]$;
12. $C_i[P_j] \cdot t \leftarrow C_i[P_j] \cdot t^* T_{cooling}$;
13. If ($C_i[P_j] \cdot t < T_{stop}$) then
14. $C_i[P_j] \cdot t \leftarrow T_{min} + j^*((T_{max} - T_{min})/p)$;
15. If (Migration (*current_iteration*/*current_runtime*,*MR*)) then
16. If ($i = 1$) then
17. For each process $C_r, \forall r \in [2 \ldots .np]$
18. Receive ($C_r[ND]$,C_r);
19. $C_i[P] \leftarrow CC[ND] \leftarrow$ Compose_fronts ($C_1[ND], \ldots, C_{np}[ND]$);
20. For each process $C_r, \forall r \in [2 \ldots np]$
21. Send ($CC[ND]$,C_r);
22. Else if ($i \geqslant 2$) then
23. Send ($C_i[ND]$,C_1);
24. Receive ($CC[ND]$,C_1);
25. $C_i[P] \leftarrow CC[ND]$;
26. Return (ND);

Fig. 3 shows graphically the migration process carried out by pMT-PSA. Each thread is started with initial solutions which are improved by applying the steps of MT-PSA. Each sub-population maintains the main population of solutions and an external archive of the same size as the main population (Fig. 3(a)). Moreover, each sub-population periodically sends its external archive to a central process (Fig. 3(b)) which evaluates all these non-dominated sets, including its external archive, then composing all of them in a global front (Fig. 3(c)). The central process sends this composed non-dominated front to all the processes, whose solutions are now considered as the individuals of the main population, then continuing the search process using the previous parameter settings (Fig. 3(d)). This parallel framework is very flexible in the sense that it can be applied to other multi-objective algorithms using the same methodology.

4. Empirical analysis

This section introduces the benchmark instances and the parameter settings of the empirical analysis, then discussing the experimental results. The performance of the newly proposed pMT-PSA for the WB-VRPTW is compared with the Strength Pareto Evolutionary Algorithm 2 (SPEA2) (Zitzler, Laumanns, & Thiele, 2001), a well-known multi-objective evolutionary algorithm often used in multi-criteria decision making. SPEA2 uses a fine-grained fitness assignment strategy, which takes into account, for each individual, the number of individuals that dominate it, and the number of individuals dominated by it. This multi-objective evolutionary algorithm applies a nearest neighbor density estimation technique that guides the search more efficiently by avoiding crowding on the front, while also uses an external archive of non-dominated solutions, *ND*, to store promising solutions found in the search, always preserving the boundary solutions. SPEA2 is parallelized (pSPEA2) using the same framework than the described in the Algorithm 1, but using an evolutionary selection strategy instead of the annealing scheme and the Metropolis function, while also including the crossover operator described above.

4.1. Test problems

The performance of both parallel multi-objective algorithms (pSPEA2 and pMT-PSA) is assessed using the well-known Solomon's benchmark instances (Solomon, 1987). The 56 Solomon's benchmark instances are based on six groups of problem instances with 100 customers. Each benchmark includes the characteristics of the customers such as their geometric coordinates, demands (d_i), time-windows (a_i, b_i), etc. Furthermore, the fleet characteristics such as the maximum number of vehicles to be used (K), and their capacities (Q^k) are also provided. These six categories are called C1, C2, R1, R2, RC1 and RC2. In problem sets R1 and R2, the customers are located randomly, in problem sets C1 and C2, the customers are clustered in some groups, and problem sets RC1 and RC2 have a mixture of random and clustered customers. Problem sets R1, C1, and RC1 have a shorter scheduling horizon, tighter time windows, and fewer customers per route than problem sets R2, C2, and RC2, respectively. Tan et al. (2006) offer a detailed description of these benchmarks.

4.2. Performance measures

It is well-known that, at least, two main criteria have to be taken into account to evaluate the performance of optimization algorithms: computational cost and the quality of the result. While the computational cost is often measured in terms of runtime or number of evaluations of the fitness function, the quality is more difficult to determine, especially in the multi-objective context. Several authors have proposed metrics to compute the performances of multi-objective algorithms (Deb, 2002; Coello et al., 2007). Most of them are convergence metrics which aim to determine the closeness of a non-dominated set of solutions to the Pareto-optimal front. In some optimization problems, the solutions are known beforehand and, therefore, it is possible to compute their true Pareto-front but, in other cases, including those NP-hard problems such as VRP and its variants, the optimal Pareto frontiers are unknown. With the aim of overcoming this drawback, a quasi-optimal Pareto-front has been constructed by executing both algorithms during a large number of iterations (three hours) in each Solomon's problem during 3 h, so that the returned fronts have been composed to be later used as reference to evaluate performance of the algorithms.

The quality of the non-dominated fronts is evaluated using a well-known convergence metric named hyper-volume, which is

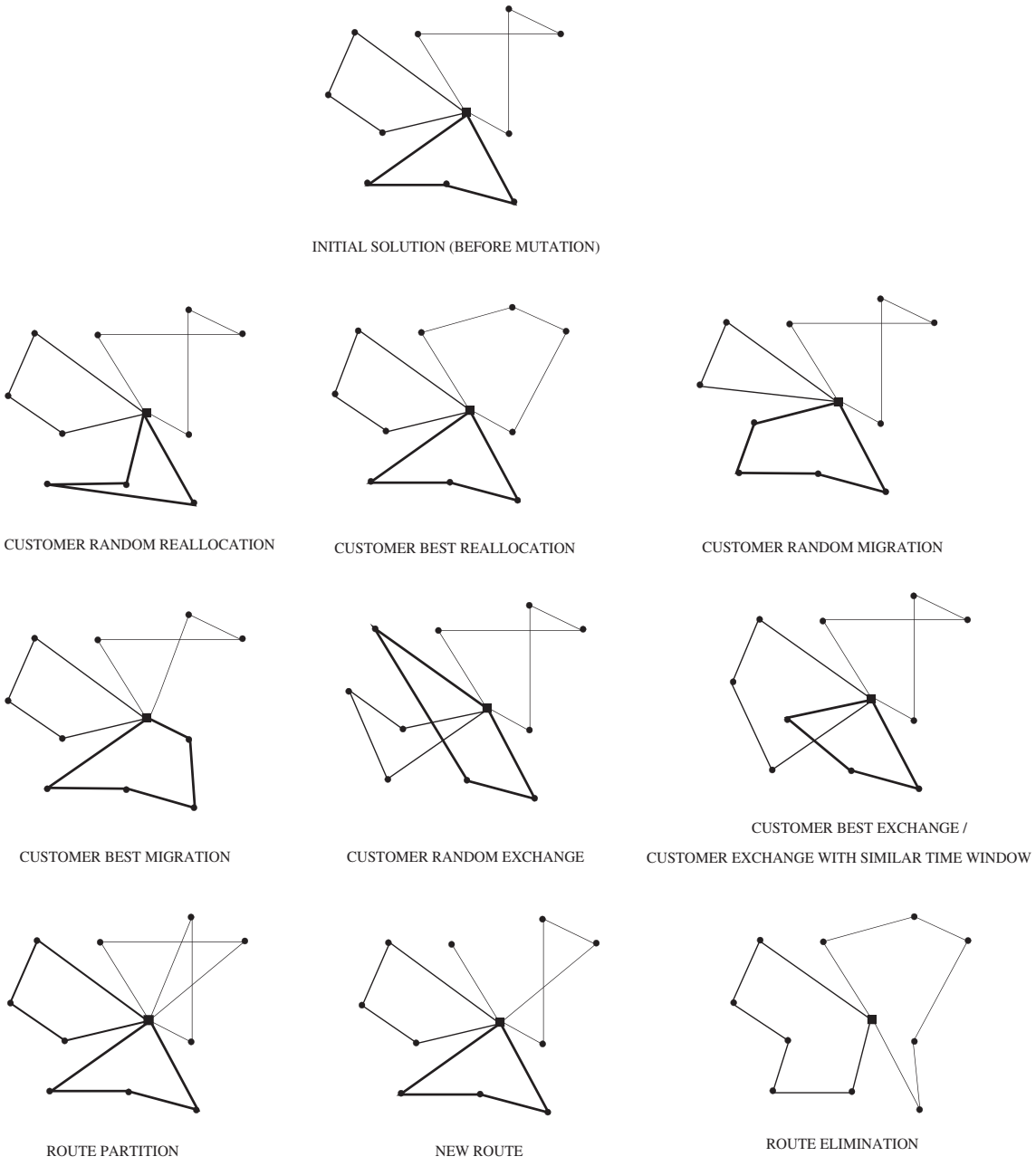

Fig. 1. Mutation operators.

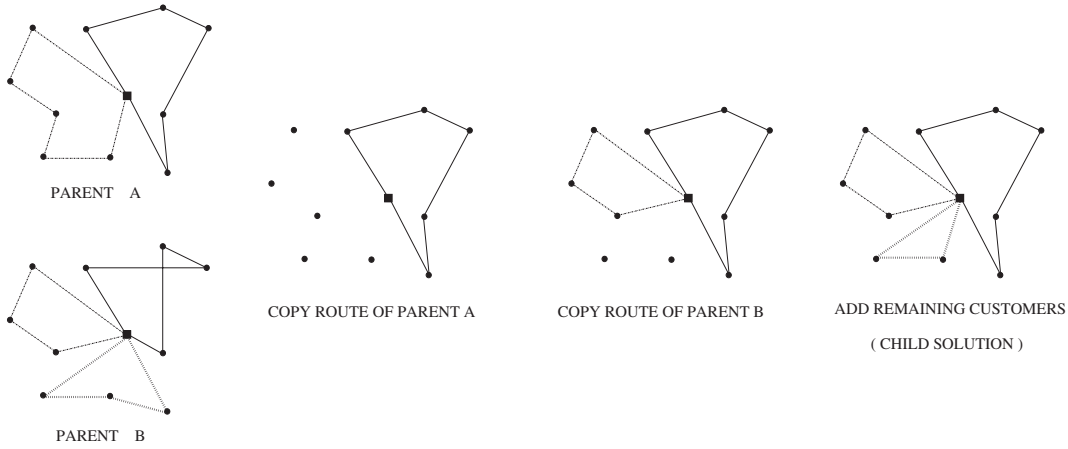

Fig. 2. Crossover operator.

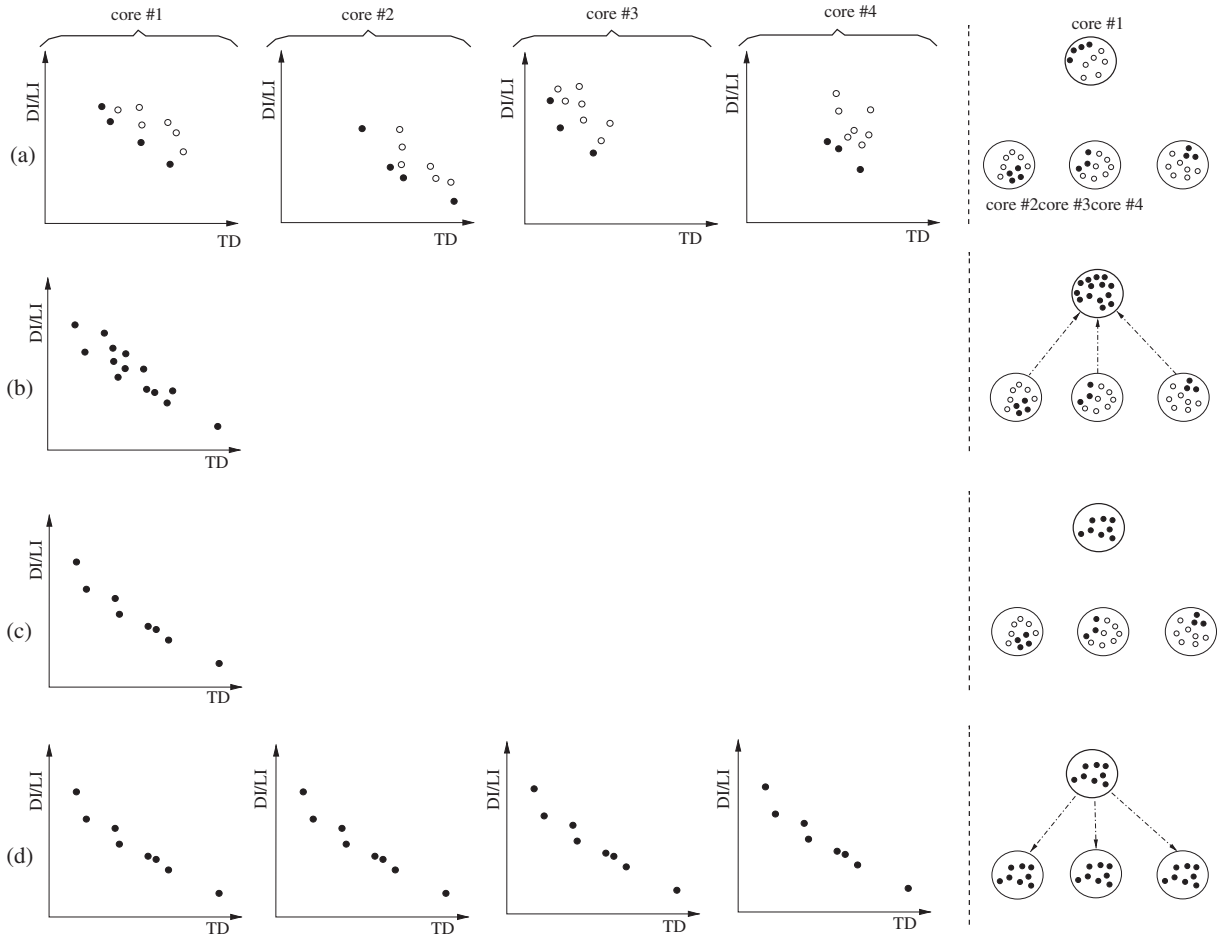

Fig. 3. The island model using four processes: (a) solutions before the migration; (b) each process migrates its *ND* (filled points) to a central process; (c) the central process composes the composed *ND*; (d) the central process migrates the composed *ND* to the remaining processes, which continue the search process.

based on ideas taken from the metric proposed by Zitzler and Thiele (1999) and has been recently used for solving multi-objective VRPTW (Garcia-Najera & Bullinaria, 2011). Since the objectives to optimize have different scales, the values returned by that metric have been normalized in the interval [0,1]. Let $X = (x_1, x_2, \ldots, x_n)$ be a set of non-dominated solutions. The hyper-volume (HV) is defined by the function *HV*(X), that returns the area of the objective space (bounded by a reference point) dominated by at least one of the members of *X*. Since some of the non-dominated solutions could have good fitness values in one objective, but worse in the other, the reference point considered here is the double of the fitness in each objective corresponding to the initial solution x_{ref} = (2*initial ($TD(P_i)$), 2*initial ($TI/Li(P_i)$)). Fig. 4(b) and (c) shows that HV (X) > HV (X′), i.e. the non-dominated front *X* is better than *X*′.

4.3. Parameter settings

The sequential multi-objective algorithms use a population of 200 individuals ($|P| = 200$), while the archive size also has this size ($|ND| = 200$). The individuals of the population are initialized using the time window based insertion heuristic (TWIH). The probability of applying crossover operator in SPEA2 is 25%, while the same probability is considered when applying mutation in both algorithms. If mutation is applied, each of the ten mutation variants is applied with the same probability (10%). The individuals of the sequential algorithm are distributed among the islands (processes/cores) so that if two processes (one per core) are used, each has a population and archive size of 100 individuals ($|C_i[P]| = |C_i[ND]| = 100$), while in case of using four processes, each island works with a population and an archive size of 50 individuals ($|C_i[P]| = |C_i[ND]| = 50$).

When comparing different algorithms, it is possible to determine that one technique is better than another one if it obtains a better performance (e.g. higher hyper-volume), given the same amount of computational cost. For computational optimization practitioners, such cost is typically measured considering a maximum number of fitness evaluations for all the methods, but this criterion assumes that the cost of other operations is either the same or almost the same in both algorithms. However, the multi-objective algorithms for solving the VRPTW require not only to evaluate the fitness of the solutions in terms of travelled distance and workload imbalance, but also to perform other expensive operations such as the selection mechanism, the external archive management, etc. For instance, pMT-PSA directly accepts/rejects solutions according to the multi-objective Metropolis criterion, while pSPEA2 uses a complex fitness assignment strategy within the selection mechanism. Therefore, the termination criterion used here is the runtime, since it seems to be the best way to measure the computational cost and it has been recently used in the context of routing problems (Derbel et al., 2010). Concretely, both sequential and parallel algorithms are executed during a fixed amount of time of 30 s in order to compare their performance.

To provide reliable statistics, each version of the algorithm was run 25 times for each benchmark instance, thus obtaining 25 non-dominated fronts for each algorithm and benchmark. Due to space restrictions, the parallel algorithms are tested on two of the

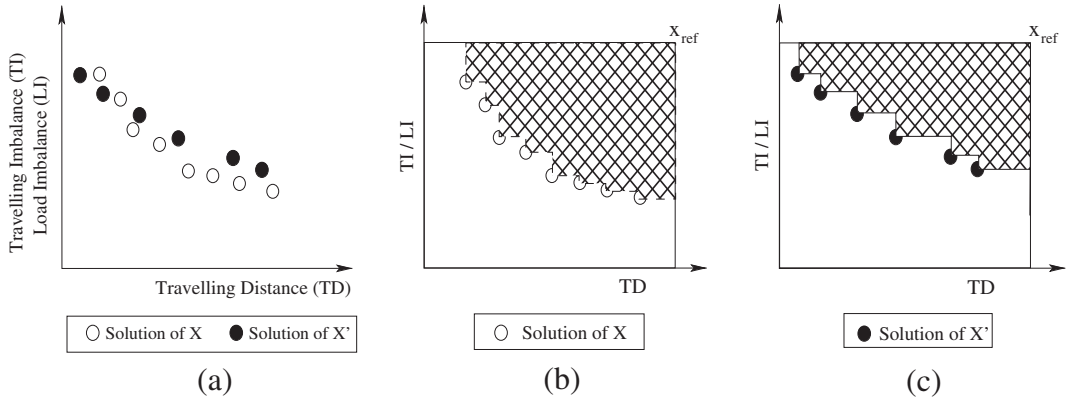

Fig. 4. Graphical explanation of the HV metric: (a) two non-dominated fronts (X and X'); (b) hyper-volume of front X; (c) hyper-volume of front X'.

problem instances of each of the six groups of Solomon's benchmarks: C103, C108, C203, C208, R103, R108, R203, R208, RC103, RC108, RC203, and RC208 which are representative for all the types of problems (Type R includes R103, R108, R203, R208; Type C includes C103, C108, C203, C208; Type RC includes RC103, RC108, RC203, RC208; Type 1 includes R103, R108, C103, C108, RC103, RC108; Type 2 includes R203, R208, C203, C208, RC203, RC208). The sequential algorithms, coded using C++, have been parallelized using the high-level Message Passing Interface (MPI), MPICH2 version 1.2.1p1, and experiments were performed on an Intel Core 2 Quad Processor Q6600 (4 cores, 2.40 GHz, 1066 MHz front-side bus, 8 MB Cache, 4 GB RAM).

4.4. Results and discussion

To justify the effectiveness of the parallel approach here presented, the computational experiments are conducted from two perspectives. First, both parallel multi-objective algorithms are executed using two and four cores (allowing ten communications per core in each run), then comparing their performance with the sequential versions in terms of hyper-volume (after 30 s, as it has been said before). Finally, these methods are again executed but considering as termination criterion to obtain a non-dominated front with a pre-established hyper-volume (92 and 95% of the hyper-volume of the composed fronts described above), which will provide information about the gain, in terms of runtime, of the parallel approaches.

4.4.1. Quality improvement using a fixed runtime as termination criterion

The first aspect analyzed here is how the quality of the solutions obtained by pSPEA2 and pMT-PSA improve using additional processes. Table 1 shows the solution values obtained by sequential (np = 1 process/core) and parallel (np = 2 and np = 4 processes/cores) algorithms in terms of best and median hyper-volume of the 25 independent runs obtained by each method when solving the multi-objective formulation that minimizes the total traveling distance and the distance imbalance among routes (TD,DI), considering as termination criterion a runtime of 30 s. The median front is that front with the 13th greater hyper-volume of the 25 independent runs (the best of the median fronts for each benchmark is displayed using bold font). On average (AVG), the median fronts obtained by pSPEA2 and pMT-PSA obtain a hyper-volume of 78.10, 85.76, and 88.53% using 1, 2, and 4 processes, respectively. From these results, it can be concluded that the quality of the solutions tends to be improved when the number of core processors

Table 1
HV when minimizing the traveling distance (TD) and distance imbalance (DI) using 1, 2 and 4 processes with ten communications and fixed runtime (30 s).

		1 Process		2 Processes		4 Processes	
		HV_{best}	HV_{median}	HV_{best}	HV_{median}	HV_{best}	HV_{median}
R103	pSPEA2	73.07	68.28	87.16	84.35	90.75	**89.17**
	pMT-PSA	88.33	84.37	88.97	87.15	90.94	88.09
R108	pSPEA2	70.17	66.44	85.62	82.87	93.89	**88.22**
	pMT-PSA	88.50	85.02	89.99	86.80	93.81	87.52
R203	pSPEA2	71.90	65.85	87.35	83.24	92.33	89.30
	pMT-PSA	90.69	89.10	92.05	90.37	93.49	**90.78**
R208	pSPEA2	80.92	75.51	88.83	85.76	91.63	**90.49**
	pMT-PSA	92.96	88.67	93.76	89.21	93.79	90.05
C103	pSPEA2	80.82	77.00	91.11	88.33	95.18	**92.72**
	pMT-PSA	91.38	88.04	92.63	88.09	92.87	88.37
C108	pSPEA2	68.17	64.72	81.85	79.70	89.62	85.48
	pMT-PSA	87.06	82.42	86.19	84.45	89.15	**85.49**
C203	pSPEA2	76.54	73.71	87.21	84.40	89.45	**87.89**
	pMT-PSA	88.43	87.22	88.78	87.54	89.23	87.59
C208	pSPEA2	82.40	76.66	88.19	85.46	93.34	90.16
	pMT-PSA	92.37	88.27	93.34	90.16	93.02	**90.35**
RC103	pSPEA2	71.63	62.36	86.89	79.37	89.16	**85.57**
	pMT-PSA	84.05	77.12	85.19	81.51	88.07	82.03
RC108	pSPEA2	74.20	69.32	88.25	84.08	92.88	**89.85**
	pMT-PSA	88.81	85.70	91.49	87.44	92.38	89.61
RC203	pSPEA2	75.22	65.13	87.53	82.87	89.42	86.21
	pMT-PSA	90.99	88.41	91.34	88.62	91.48	**88.86**
RC208	pSPEA2	78.92	75.75	89.95	87.17	91.99	**90.86**
	pMT-PSA	91.47	89.44	91.40	89.34	91.91	90.01

Table 2
Median hyper-volume obtained by pSPEA2 and PMT-PSA according to the formulation and the Solomon's type of problem, using a fixed runtime (30 s).

			Type R	Type C	Type RC	Type 1	Type 2	AVG
TD,DI	np = 1	pSPEA2	69.02	73.02	68.14	68.02	72.10	70.06
		pMT-PSA	87.19	86.49	85.17	83.78	88.78	**86.28**
	np = 2	pSPEA2	84.06	84.47	83.37	83.12	88.18	84.64
		pMT-PSA	88.38	87.56	86.73	85.91	91.78	**88.07**
	np = 4	pSPEA2	89.30	89.06	88.12	88.50	89.15	**88.83**
		pMT-PSA	89.11	87.95	87.63	86.85	89.61	88.23
TD,LI	np = 1	pSPEA2	65.83	72.80	66.66	66.34	70.52	68.43
		pMT-PSA	85.09	86.75	84.42	81.75	89.09	**85.42**
	np = 2	pSPEA2	78.83	83.11	79.32	77.99	82.85	80.42
		pMT-PSA	85.50	87.72	84.83	82.16	89.87	**86.01**
	np = 4	pSPEA2	83.07	85.87	82.58	81.65	86.03	83.84
		pMT-PSA	85.89	87.88	85.30	82.58	90.13	**86.36**

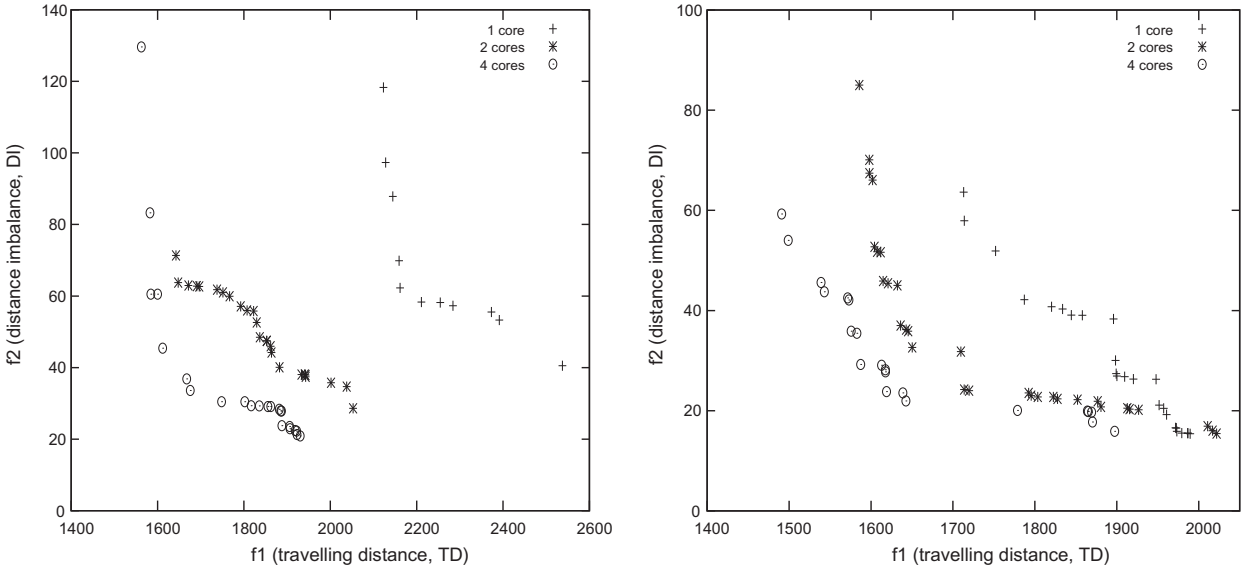

Fig. 5. Median fronts obtained after 30 s when solving the multi-objective formulation TD,DI using 1, 2 and 4 processes: (a) pSPEA2 in R103; (b) pMT-PSA in RC108.

Table 3
Runtimes (in seconds) required by pSPEA2 and pMT-PSA to reach an hyper-volume of 92%.

			Type R	Type C	Type RC	Type 1	Type 2	AVG
TD,DI	np = 1	pSPEA2	196.92	215.79	219.94	200.88	220.89	210.88
		pMT-PSA	74.03	86.01	73.46	93.48	62.18	**77.83**
	np = 2	pSPEA2	95.82	120.64	118.14	110.62	112.44	111.53
		pMT-PSA	46.49	70.40	61.27	71.97	46.80	**59.38**
	np = 4	pSPEA2	51.51	59.18	59.01	54.98	58.15	**56.56**
		pMT-PSA	63.34	41.09	71.19	80.43	36.65	58.54
TD,LI	np = 1	pSPEA2	413.73	191.49	407.43	365.61	309.49	337.55
		pMT-PSA	72.82	46.70	76.93	89.25	41.72	**65.48**
	np = 2	pSPEA2	199.23	100.71	192.07	207.10	120.91	164.01
		pMT-PSA	45.72	55.24	47.74	58.28	40.85	**49.57**
	np = 4	pSPEA2	118.77	64.33	119.75	121.64	80.26	100.95
		pMT-PSA	54.36	35.62	56.50	66.71	30.94	**48.82**

grows. It is also observed that pMT-PSA often outperforms pSPEA2, especially with 1 and 2 processes, while a similar performance is obtained when using np = 4 processes/cores.

Table 2 displays the median values obtained by pMT-PSA and pSPEA2 categorized according to the multi-objective formulation, the number of proceses, and the Solomon's type of problem. It is seen that, in both formulations, the quality of the non-dominated fronts obtained by the island-based paradigm improves when using additional processes. It is noteworthy that the hyper-volume obtained by pMT-PSA is often higher than that obtained by pSPEA2 in most test problems using both formulations, but the difference in their performance decreases when the number of processes grows.

Fig. 5(a) displays the median fronts in terms of hyper-volume obtained by pSPEA2 when minimizing the traveling distance and distance imbalance of benchmark R103, while Fig. 5(b) shows the median fronts obtained by pMT-PSA in the optimization of the problem instance RC108. From both figures, it can be concluded that, independently of the algorithm applied, the parallel implementation allows us to obtain better non-dominated fronts in the same execution time.

Table 4
Runtimes (in seconds) required by pSPEA2 and pMT-PSA to reach an hyper-volume of 95%.

			Type R	Type C	Type RC	Type 1	Type 2	AVG
TD,DI	np = 1	pSPEA2	324.79	381.98	504.65	391.26	416.36	403.81
		pMT-PSA	83.16	251.58	125.58	137.59	169.29	**153.44**
	np = 2	pSPEA2	176.88	213.42	246.53	209.23	215.32	212.28
		pMT-PSA	73.75	193.58	101.49	115.62	130.26	**122.94**
	np = 4	pSPEA2	87.73	127.77	117.34	99.54	122.35	110.94
		pMT-PSA	66.91	165.32	90.49	99.37	115.78	**107.57**
TD,LI	np = 2	pSPEA2	764.67	500.25	935.78	812.06	655.07	733.57
		pMT-PSA	190.42	133.84	187.19	237.71	103.25	**170.48**
	np = 2	pSPEA2	361.83	202.69	427.59	389.90	271.51	330.70
		pMT-PSA	158.92	107.29	162.49	199.89	85.91	**142.90**
	np = 4	pSPEA2	217.48	135.11	218.50	218.41	162.32	190.36
		pMT-PSA	129.04	75.63	142.89	164.86	66.84	**115.85**

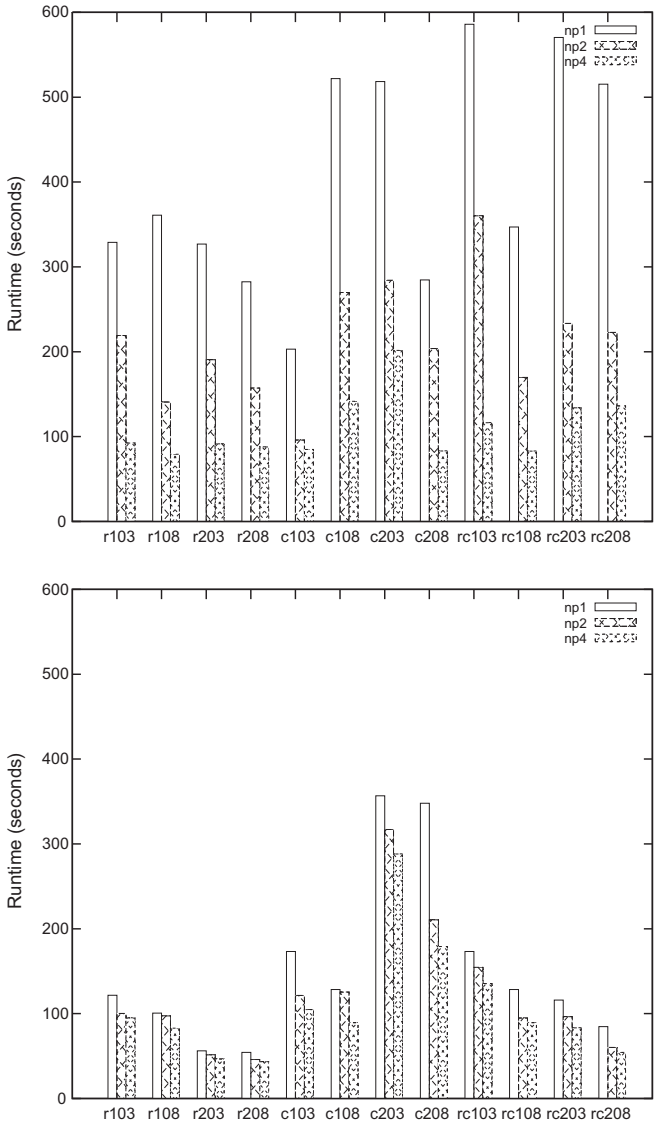

Fig. 6. Runtime required by pSPEA2 and pMT-PSA to reach an hyper-volume of 95% (TD,DI).

4.4.2. Runtime improvement using a fixed hyper-volume as termination criterion

Once the way in which the quality of the non-dominated fronts improves when the number of processes increases using a fixed runtime has been analyzed, the second studied aspect is the gain, in terms of runtime, provided by the parallel algorithms to obtain non-dominated fronts of a similar quality. Table 3 shows the runtimes required by pSPEA2 and pMT-PSA to obtain a hyper-volume of 92%, taking as reference the hyper-volume of the composed quasi-optimal front. From these results, which are classified according to the Solomon's type of problem and number of processes, it can be seen that the parallel implementation of MT-PSA (pMT-PSA) is faster than pSPEA2. Nevertheless, the impact of increasing the number of processes is higher when parallelizing SPEA2 (pSPEA2). The reason is the higher algorithmic complexity of pSPEA2, which uses a density estimator and a truncation operator with $\mathcal{O}(M^2 logM)$ (Zitzler et al., 2001), which involves that dividing the sequential population into two or four islands reduces the number of individuals per island (process), thus increasing the number of iterations within the same runtime. Algorithm pMT-PSA also reduces the runtime required to obtain a non-dominated front having the pre-established hyper-volume. Similar conclusions can be obtained from Table 4, which shows the runtimes required by pSPEA2 and pMT-PSA to obtain a hyper-volume of 95%. Obviously, the runtime in the second comparison is higher than in the first one. Thus, the average runtime required by pSPEA2 to reach a hyper-volume of 92% oscillates between one and three minutes according to the number of processes used and the problem formulation, while pMT-PSA requires less than one minute. Nevertheless, the higher difficulty derived from establishing a hyper-volume of 95% as termination condition involves that the average runtime required by pSPEA2 oscillates between two and six minutes, while pMT-PSA requires between one and three minutes. Fig. 6 shows graphically the runtimes required by pSPEA2 and pMT-PSA to obtain a hyper-volume of 95%, these runtimes categorized according to the number of processes, and the Solomon's problem, which reinforces the previous conclusions.

5. Conclusions

The design of efficient methods capable of solving vehicle routing problems has been a subject that has attracted many research efforts due to its influence in transportation, logistics, and supply chain management. Often, the number of customers combined with the complexity of real-life data does not allow an exact solution of the problem, being then necessary to apply approximated methods, such as heuristic approaches, that provide of approximated solutions in reasonable time horizons. Recently, some authors have extended the single-objective VRPTW to cope with other objectives than route costs, which are also important for transportation and logistic managers. In particular, the multi-objective variant analyzed here is the workload balancing VRPTW, that considers not only the total distance travelled by the vehicles used to service the customers, but also their workload imbalance. The imbalance is analyzed from two points of view: the imbalance in the distances travelled by the vehicles and the imbalance in the loads delivered

by each vehicle. Since these multi-objective variants are even more complex than the single-objective ones, parallelization could provide benefits in terms of the quality of solutions. This paper provides a fairly comprehensive background on vehicle routing problems and presents a new multi-objective procedure based on simulated annealing, the Multiple Temperature Pareto Simulated Annealing (MT-PSA), to cope with these multi-objective formulations of the VRPTW. The sequential MT-PSA is also parallelized using the well-known island paradigm parallel, and its results are compared with, respectively, sequential and island-based parallel implementations of SPEA2. The main insight in this work is that the parallelization of evolutionary or local search multi-objective algorithms on a single workstation with a multi-core processor using the island paradigm allows us to quickly achieve competitive non-dominated fronts. In fact, results obtained using a well-known set of benchmarks show that the parallel approach obtains high quality fronts in comparison with the sequential versions without increasing the computational cost, while also producing a significant gain, in terms of runtime, while maintaining solution quality. To the best of our knowledge, this is the first parallel multi-objective approach that analyzes these two formulations, and, therefore, this paper could be useful for investigators and distribution enterprisers interested in parallelizing other multi-objective formulations of the vehicle routing problem and its variants, thus increasing the application of this problem to real-life environment.

Acknowledgments

This work has been partially by the Spanish Ministry of Economy and Competitiveness under projects TIN2012-32039 and SAF2010-20558. R. Baños also acknowledges the support of a Juan de la Cierva postdoctoral fellowship funded by the Spanish Ministry of Economy and Competitiveness.

References

Alabas-Uslu, C., & Dengiz, B. (2011). A self-adaptive local search algorithm for the classical vehicle routing problem. *Expert Systems with Applications, 38*(7), 8990–8998.

Alba, E. (2005). *Parallel metaheuristics: A new class of algorithms*. Wiley.

Alvarenga, G. B., Mateus, G. R., & de Tomic, G. (2007). A genetic and set partitioning two-phase approach for the vehicle routing problem with time windows. *Computers & Operations Research, 34*(6), 1561–1584.

Badeau, F., Guertin, F., Gendreau, M., Potvin, J-Y., & Taillard, E. (1997). Parallel tabu search heuristic for the vehicle routing problem with time windows. *Transportation Research – Part C, 5*(2), 109–122.

Bandyopadhyay, S., Saha, S., Maulik, U., & Deb, K. (2008). A simulated annealing-based multiobjective optimization algorithm: AMOSA. *IEEE Transactions on Evolutionary Computation, 12*(3), 269–283.

Baños, R., Gil, C., Ortega, J., & Montoya, F. G. (2004). A parallel multilevel metaheuristic for graph partitioning. *Journal of Heuristics, 10*(3), 315–336.

Berger, J., & Barkaoui, M. (2004). A parallel hybrid genetic algorithm for the vehicle routing problem with time windows. *Computers & Operations Research, 31*(12), 2037–2053.

Borgulya, I. (2008). An algorithm for the capacitated vehicle routing problem with route balancing. *Central European Journal of Operations Research, 16*(4), 331–343.

Bräysy, O., & Gendreau, M. (2005). Vehicle routing problem with time windows, part II: Metaheuristics. *Transportation Science, 39*, 119–139.

Cheng, C-B., & Wang, K-P. (2009). Solving a vehicle routing problem with time windows by a decomposition technique and a genetic algorithm. *Expert Systems with Applications, 36*(4), 7758–7763.

Coello, C. A., Lamont, G. B., & Van Veldhuizen, D. A. (2007). *Evolutionary algorithms for solving multi-objective problems. Genetic and evolutionary computation series*. Springer.

Cordeau, J-F., & Maischberger, M. (2012). A parallel iterated tabu search heuristic for vehicle routing problems. *Computers & Operations Research, 39*(9), 2033–2205.

Dantzig, G. B., & Ramser, J. H. (1959). The truck dispatching problem. *Management Science, 6*(1), 80–91.

Deb, K. (2002). *Multi-objective optimization using evolutionary algorithms*. John Wiley & Sons.

Derbel, H., Jarboui, B., Hanafi, S., & Chabchoub, H. (2010). An iterated local search for solving a location-routing problem. *Electronic Notes in Discrete Mathematics, 36*, 875–882.

Eksioglu, B., Vural, A. V., & Reisman, A. (2009). The vehicle routing problem: A taxonomic review. *Computers & Industrial Engineering, 57*(4), 1472–1483.

El-Sherbeny, N. A. (2010). Vehicle routing with time windows: An overview of exact, heuristic and metaheuristic methods. *Journal of King Saud University (Science), 22*(3), 123–131.

Garcia-Najera, A., & Bullinaria, J. A. (2011). An improved multi-objective evolutionary algorithm for the vehicle routing problem with time windows. *Computers & Operations Research, 38*(1), 287–300.

Gehring, H., & Homberger, J. (2002). Parallelization of a two-phase metaheuristic for routing problems with time windows. *Journal of Heuristics, 8*(3), 251–276.

Glover, F., & Kochenberger, G. A. (Eds.). (2003). *Handbook of metaheuristics. International series in operations research and management science* (Vol. 57). Dordrecht, The Netherlands: Kluwer Academic Publishers.

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization and machine learning*. New York: Addison Wesley.

Groër, C., Golden, B., & Wasil, E. (2011). A parallel algorithm for the vehicle routing problem. *INFORMS Journal on Computing, 23*(2), 315–330.

Jozefowiez, N., Semet, F., & Talbi, E-G. (2008). Multi-objective vehicle routing problems. *European Journal of Operational Research, 189*(2), 293–309.

Jozefowiez, N., Semet, F., & Talbi, E-G. (2009). An evolutionary algorithm for the vehicle routing problem with route balancing. *European Journal of Operational Research, 195*(3), 761–769.

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science, 220*(4598), 671–680.

Kritikos, M. N., & Ioannou, G. (2010). The balanced cargo vehicle routing problem with time windows. *International Journal of Production Economics, 123*(1), 42–51.

Le Bouthillier, A., & Crainic, T. G. (2005). A cooperative parallel meta-heuristic for the vehicle routing problem with time windows. *Computers & Operations Research, 32*(7), 1685–1708.

Lee, C-Y., Lee, Z-J., Lin, S-W., & Ying, K-C. (2010). An enhanced ant colony optimization (EACO) applied to capacitated vehicle routing problem. *Applied Intelligence, 32*(1), 88–95.

Lenstra, J. K., & Rinnooy Kan, A. H. G. (1981). Complexity of vehicle routing and scheduling problems. *Networks, 11*(2), 221–227.

Marinakis, Y., Marinaki, M., & Dounia, G. (2010). A hybrid particle swarm optimization algorithm for the vehicle routing problem. *Engineering Applications of Artificial Intelligence, 23*(4), 463–472.

Márquez, A. L., Gil, C., Baños, R., & Gómez, J. (2011). Parallelism on multicore processors using Parallel.FX. *Advances in Engineering Software, 42*(5), 259–265.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A., & Teller, E. (1953). Equation of state calculations by fast computing machines. *Journal of Chemical Physics, 21*(6), 1087–1092.

Müller, J. (2010). Approximative solutions to the bicriterion vehicle routing problem with time windows. *European Journal of Operational Research, 202*(1), 223–231.

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.

Ombuki, B., Ross, B. J., & Hanshar, F. (2006). Multi-objective genetic algorithms for vehicle routing problem with time windows. *Applied Intelligence, 24*(1), 17–30.

Rego, C. (2001). Node-ejection chains for the vehicle routing problem: Sequential and parallel algorithms. *Parallel Computing, 27*(3), 201–222.

Solomon, M. M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research, 35*(2), 254–265.

Taillard, É. (1993). Parallel iterative search methods for vehicle routing problems. *Networks, 23*(8), 661–673.

Tan, K. C., Chew, Y. H., & Lee, L. H. (2006). A hybrid multiobjective evolutionary algorithm for solving vehicle routing problem with time windows. *Computational Optimization and Applications, 34*(1), 115–151.

Zitzler, E., & Thiele, L. (1999). Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation, 3*(4), 257–271.

Zitzler, E., Laumanns, M., & Thiele, L. (2001). SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization. In *Proceedings of Evolutionary Methods for Design, Optimisation, and Control* (pp. 95–100) Barcelona, Spain.