Computers & Operations Research 39 (2012) 2033–2050

Contents lists available at SciVerse ScienceDirect

Computers & Operations Research

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

A parallel iterated tabu search heuristic for vehicle routing problems

Jean-François Cordeau [a,*], Mirko Maischberger [b]

[a] Canada Research Chair in Logistics and Transportation and CIRRELT, HEC Montréal, 3000 chemin de la Côte-Sainte-Catherine, Montréal, Canada H3T 2A7
[b] Global Optimization Laboratory and DSI, Università degli Studi di Firenze, Via di S. Marta 3, 50139 Firenze, Italy

ARTICLE INFO

Available online 17 October 2011

Keywords:
Vehicle routing problem
Multi-depot
Periodic
Site-dependent
Time windows
Tabu search
Iterated local search
Parallel computing

ABSTRACT

This paper introduces a parallel iterated tabu search heuristic for solving four different routing problems: the classical vehicle routing problem (VRP), the periodic VRP, the multi-depot VRP, and the site-dependent VRP. In addition, it is applicable to the time-window constrained variant of these problems. Using the iterated local search framework, the heuristic combines tabu search with a simple perturbation mechanism to ensure a broad exploration of the search space. We also describe a parallel implementation of the heuristic to take advantage of multiple-core processors. Extensive computational results show that the proposed heuristic outperforms tabu search alone and is competitive with recent heuristics designed for each particular problem.

1. Introduction

The purpose of this paper is to present a simple parallel iterated tabu search heuristic for the vehicle routing problem (VRP) and several variants: the periodic VRP (PVRP), the multi-depot VRP (MDVRP), and the site-dependent VRP (SDVRP). The heuristic is also applicable to the corresponding problems with time windows: the VRPTW, the PVRPTW, the MDVRPTW, and the SDVRPTW. The heuristic builds upon the previous work of Cordeau et al. [13–15] on tabu search heuristics for VRP variants. It embeds tabu search within iterated local search and uses a simple parallel computing framework to take advantage of the multiple cores available on modern computers.

The VRP can be defined on a complete directed graph $G=(V,A)$, where $V=\{v_0,v_1,\ldots,v_n\}$ is the set of vertices and $A=\{(v_i,v_j): v_i,v_j\in V, i\neq j\}$ is the set of arcs. Vertex v_0 represents the depot while vertices $v_1,\ldots,v_n$ represent customers. Each customer vertex v_i has a non-negative demand q_i as well as a non-negative service duration d_i. A fleet of m vehicles is available to visit the customers. Vehicle k has a capacity of Q_k and the duration of its route cannot exceed D_k. With each arc (v_i,v_j) is associated a non-negative cost c_{ij} proportional to the travel time from vertex i to vertex j. The VRP consists in designing m vehicle routes of minimum total cost such that (i) each route starts and ends at the depot; (ii) each customer is visited by exactly one route; and (iii) the total demand of the customers visited by route k does not exceed Q_k and its duration does not exceed D_k. The VRP is a hard combinatorial optimization problem for which a large number of exact and heuristic algorithms have been proposed in the last decades (see, e.g., [47,25]). Among the best performing recent heuristics are the memetic algorithm of Nagata and Bräysy [36] and the iterated local search algorithm of Prins [42]. Another effective algorithm is the general adaptive large neighbourhood search (ALNS) heuristic of Pisinger and Ropke [39] which has been applied not only to the VRP but also to the MDVRP, the SDVRP, the VRPTW, and the Open VRP. Significant progress has also recently been made in terms of exact solution methods. In particular, Baldacci and Mingozzi [2] have introduced a unified approach capable of solving multiple variants of the VRP including the PVRP, MDVRP and SDVRP (see also [3,4]). We refer to Laporte [29] for a recent survey on the VRP.

In the PVRP, one considers a planning horizon of t days during which customer i requires f_i visits. These visits must follow an allowable combination of visit days as specified by a set C_i. On a five-day planning horizon, for example, a customer i with $f_i=2$ and $C_i=\{\{1,3\},\{2,4\},\{3,5\}\}$ should be visited either on days 1 and 3, on days 2 and 4, or on days 3 and 5. The PVRP consists in assigning an allowable visit combination to each customer and in designing vehicle routes for each day as in the VRP. It can be modelled by considering a multigraph in which an arc $(v_i,v_j)^{k,\ell}$ with cost $c_{ijk\ell}$ is associated with vehicle k going from vertex i to vertex j on day ℓ. The PVRP was first studied by Beltrami and Bodin [5] in the context of waste collection. More recent heuristics have been introduced by Chao et al. [9], Cordeau et al. [13], Alegre et al. [1], Hemmelmayr et al. [27], and Vidal et al. [48]. Among these, the best one is the hybrid genetic algorithm of Vidal et al. [48], which combines evolutionary search, local search, and sophisticated solution population management schemes to guide the search. For a review of the PVRP, we refer the reader to Francis et al. [20].

* Corresponding author. Tel.: +1 514 340 6278; fax: +1 514 340 6834.
E-mail address: jean-francois.cordeau@hec.ca (J.-F. Cordeau).

0305-0548/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2011.09.021

The MDVRP works with a one-day planning horizon but assumes that vehicles are based at more than one depot. The problem is to assign each customer to a depot and to construct routes for each depot according to the rules of the VRP, with the extra constraint that each route must start and end at the same depot. As explained by Cordeau et al. [13], the MDVRP can be seen as a PVRP where each depot is associated to a different day of the planning horizon. If there are t depots, then the set of visit combinations for each customer i should be $C_i = \{\{1\},\{2\},\ldots,\{t\}\}$. In addition, the values $c_{0ik\ell}$ and $c_{i0k\ell}$ should be set equal to the travel cost between depot ℓ and customer i using vehicle k. Effective heuristics for the MDVRP were designed by Chao et al. [8], Renaud et al. [44], Cordeau et al. [13], Pisinger and Ropke [39], but better results were recently obtained by Vidal et al. [48].

Finally, the SDVRP assumes that there are several types of vehicles in the fleet and that some customers can be visited only by vehicles of certain types. The problem thus consists in assigning a compatible vehicle type to each customer and in designing vehicle routes for the vehicles of each type as in the VRP. As pointed out by Cordeau and Laporte [12], the SDVRP can be modelled as a PVRP in which each day corresponds to a different vehicle type and the set C_i for customer i contains only the days associated with compatible vehicle types. The SDVRP was first introduced by Nag et al. [35] and improved heuristics were later described by Chao et al. [10] and by Cordeau and Laporte [12]. Again, better results were recently obtained by the ALNS heuristic of Pisinger and Ropke [39].

In the VRPTW, PVRPTW, MDVRPTW and SDVRPTW, one associates with each customer i a time window $[e_i,l_i]$ during which service at customer i must begin. The VRPTW is the most studied variant of the VRP and a very large number of heuristics have been proposed to solve the problem. Among the best recent ones are the active guided evolution strategies of Mester and Bräysy [34] and the memetic algorithm of Nagata et al. [37]. For exhaustive surveys on the VRPTW, we refer to Bräysy and Gendreau [7] and to Gendreau and Tarantilis [22]. The PVRPTW, MDVRPTW and SDVRPTW have received less attention than their counterparts without time windows. The PVRPTW and MDVRPTW were first addressed with a tabu search heuristic by Cordeau et al. [14,15]. An improved VNS heuristic for the PVRPTW was recently introduced by Pirkwieser and Raidl [38], while Polacek et al. [40,41] introduced VNS algorithms for the MDVRPTW. To the best of our knowledge, only Cordeau et al. [14,15] addressed the SDVRPTW.

In the last decade, parallel metaheuristics for the VRP and some variants have become increasingly popular. For example, Rego [43] introduced a parallel tabu search heuristic based on ejection chains for the VRP, while Doerner et al. [18] proposed parallel cooperative ant colony optimization methods. More recently, Groër et al. [26] developed a parallel algorithm that combines local search and integer programming, and Jin et al. [28] introduced a multi-neighbourhood cooperative tabu search. For the VRPTW, Schulze and Fahle [45] introduced a parallel tabu search and set covering heuristic, Gehring and Homberger [21] described a parallel two-phase heuristic combining an evolution strategy with tabu search, Berger and Barkaoui [6] developed a parallel hybrid genetic algorithm, and Le Bouthillier and Crainic [30] designed a cooperative parallel meta-heuristic combining different solution construction and improvement methods including tabu search and genetic algorithms. Parallel metaheuristics have also been developed for other variants of the VRP. In particular, Drummond et al. [19] introduced a parallel genetic algorithm for the PVRP while Polacek et al. [41] introduced a parallel implementation of their VNS for the MDVRPTW. For a recent survey of parallel solution methods for vehicle routing problems, we refer the interested reader to Crainic [16].

The purpose of this paper is to introduce a common heuristic to address the eight problem variants described above. Our aim is not to obtain the best heuristic for each variant but to propose a simple methodology that adapts easily to the different problems and achieves a good balance between solution quality and computing time. The tabu search heuristic and the iterated local search that serve as the core of this methodology are first described in Section 2. This is followed by the parallel iterated local search framework in Section 3, and by computational results in Section 4.

2. The iterated tabu search heuristic

This section describes a common iterated tabu search heuristic that can be used to solve all eight problem variants considered in the paper. We first present the general framework of iterated local search and we then explain how this framework is applied to solve vehicle routing problems by focusing on the tabu search heuristic that is used as a local search improvement procedure.

Iterated local search (ILS) was introduced by Lourenço et al. [31]. Its main idea consists in alternating between phases of local search around the current solution and perturbations that aim to diversify the search and escape from local optima. Starting from an initial solution s_0, the algorithm first applies local search to this solution to obtain an improved solution $\hat{s}$. Iteratively, solution $\hat{s}$ is then perturbed to obtain a new solution s' which is itself improved by local search to obtain a solution $\tilde{s}$. If $\tilde{s}$ satisfies an acceptance criterion, it replaces $\hat{s}$ and the next perturbation is applied to this solution. Otherwise, the search returns to the previous solution $\hat{s}$. This process is summarized in Algorithm 1, where $f(s)$ denotes the cost of solution s and s^* is the best solution found during the search.

Algorithm 1. Iterated local search.

$s_0 \leftarrow$ *initial solution*()
$s^* \leftarrow \hat{s} \leftarrow$ *improve*(s_0)
while ¬ *termination*() **do**
  $s' \leftarrow$ *perturb*($\hat{s}$)
  $\tilde{s} \leftarrow$ *improve*(s')
  if *accept*($\tilde{s}$) **then**
    $\hat{s} \leftarrow \tilde{s}$
  end if
  if $f(\tilde{s}) < f(s^*)$ **then**
    $s^* \leftarrow \tilde{s}$
  end if
end while
return s^*

In our implementation of ILS, we use a tabu search heuristic to perform the local search improvement step. Because this heuristic allows infeasible solutions during the search, the classical ILS framework is slightly adjusted to check whether solution $\tilde{s}$ is feasible before it can replace s^*. In addition, when the improved solution $\tilde{s}$ fails to satisfy the acceptance criterion, the search returns to s^* (i.e., $\hat{s} \leftarrow s^*$) before applying the next perturbation.

In the following sections we describe the procedure used to construct an initial solution, the tabu search heuristic used for the improvement step, the termination and acceptance criteria, and the perturbation strategy. We start by describing the basic route manipulation heuristics which are common to the other procedures.

2.1. Route manipulation heuristics

The construction of an initial solution, the tabu search, and the perturbation phase of the iterated local search all rely on a common, lower level heuristic for manipulating routes. This heuristic is responsible for computing the cost of removing and inserting a customer in a route and for evaluating the impact of these actions on the violation of capacity, duration and time

window constraints, if any. We have implemented two different route manipulation heuristics.

For problems without time windows, we use the generalized insertion (GENI) algorithm of Gendreau et al. [23] which performs customer insertions and removals by rearranging the vertices that belong to a small neighbourhood (of size q) around the customer. For problems with time windows, we use a simpler mechanism as in Cordeau et al. [14,15]: customers are inserted by considering the best insertion between two consecutive customers and removals are performed by reconnecting the predecessor and successor vertices. No rearrangement of the customers is performed. However, the total duration of the route is then minimized by using the forward time slack notion as explained by Cordeau et al. [15].

2.2. Construction of an initial solution

The construction of the initial solution uses the route manipulation heuristic and differs slightly depending on the problem type. For the PVRP, SDVRP, PVRPTW and SDVRPTW, a visit combination is assigned to each customer i by randomly selecting an element from the set C_i. For the MDVRP and MDVRPTW, we instead assign each customer to its nearest depot. (For the VRP and VRPTW, obviously, no such assignment is required.) In all cases, routes are first initialized with the depot vertex. Next, customers are sorted in increasing order of the angle they make with the depot and an arbitrary radius. Starting with the first vehicle and using this ordering, customers are finally inserted one by one into vehicle routes belonging to the days (or depots or vehicle types) to which they are assigned. Whenever the insertion of a customer in the current route would lead to a violation of the capacity or route duration constraint, a new route is initialized unless there are no more vehicles left. For the VRP, PVRP, MDVRP and SDVRP, this procedure thus ensures that the first $m-1$ routes are feasible, where m is the number of vehicles available on each day (in the PVRP), at each depot (in the MDVRP) or of each vehicle type (in the SDVRP). For problems with time windows, the insertion of a customer i is only allowed between two successive customers j_1 and j_2 such that $e_{j_1} \le e_i \le e_{j_2}$, or at the end of the route if no such pair of customers exist. For these problems, capacity and route duration constraints are guaranteed to be satisfied by the first $m-1$ routes but time window constraints may be violated by all routes.

2.3. Tabu search improvement heuristic

To perform the local search improvement step, we use a variant of the tabu search heuristic which was first introduced by Cordeau et al. [13] for the PVRP and MDVRP and was later adapted to the SDVRP by Cordeau and Laporte [12] and to problems with time windows by Cordeau et al. [14,15]. Because all problem variants are special cases of the PVRPTW, we explain the functioning of the heuristic with respect to this problem. Problems without time windows are handled in the same way unless otherwise specified.

The tabu search heuristic starts from the current ILS solution s' and moves at each iteration to the best non-tabu solution in the neighbourhood. To ensure a broad exploration of the search space, constraint relaxation and diversification mechanisms are used. Intensification is provided through a route refinement step that is applied periodically.

2.3.1. Objective function and constraint relaxation mechanism

An important feature of the tabu search heuristic is that it allows infeasible solutions during the course of the search. To this end, capacity, route duration, and time window constraints are relaxed and their violations are penalized in the objective function. This augmented objective function is computed as $f(s)=c(s)+\alpha q(s)+\beta d(s)+\gamma t(s)$, where $c(s)$ is the total routing cost, $q(s)$, $d(s)$ and $t(s)$ denote the total violations of capacity, duration and time window constraints, respectively, and α, β and γ are self-adjusting non-negative multipliers. These multipliers are initially set to 1 and their value is increased (resp. decreased) by a factor $1+\delta$ when the corresponding constraints are violated (resp. satisfied) at the end of a tabu search iteration. At the beginning of each improvement phase the value of parameter δ is chosen randomly from the uniform distribution on the interval [0,1].

2.3.2. Neighbourhood structure

Solutions handled by the tabu search heuristic can be (partly) characterized by their set of attributes. If customer i is visited by vehicle k on day l, we say that attribute (i,k,ℓ) belongs to the solution and we denote by $B(s)$ the attribute set of solution s. A move from one solution to another can be expressed as a pair (R,R'), where R and R' are two different sets of attributes. Two types of moves are considered during the search: (i) moving a customer from one route to another one on a given day, i.e., replacing an attribute (i,k,ℓ) with an attribute (i,k',ℓ) for given i and ℓ, where $k \neq k'$, and (ii) changing the visit combination assigned to a customer. In the latter case, the customer is removed from all days that are not present in the new visit combination and is inserted in a least-cost route on each day that is present only in the new combination. This insertion is performed so as to minimize the increase in the value of the function $g(s)$ which is defined in Section 2.3.4, while taking the tabu tenure and aspiration criterion into account. In each iteration of the tabu search heuristic, both neighbourhoods are explored in full and we select the move leading to the best non-tabu solution. After applying a move (R,R') to a solution s we obtain a new solution s' with the attribute set $B(s')=(B(s)\backslash R)\cup R'$.

2.3.3. Tabu list and aspiration criterion

At the beginning of each improvement phase within the ILS, the tabu list tenure τ is chosen randomly from the discrete uniform distribution on the interval $[0,\lceil\sqrt{nmt}\rceil]$. When a move (R,R') is performed, the attributes in the set $R'\backslash R$ are declared tabu for τ iterations and any move adding at least one of these attributes cannot be performed unless it satisfies the aspiration criterion. The aspiration criterion overrides the tabu status of a move (R,R') if this move yields a solution s' such that $f(s')$ is smaller than the cost of the best solution found so far for at least one attribute $(i,k,\ell) \in B(s')$. If this condition is satisfied, then clearly the algorithm is not cycling.

2.3.4. Diversification

The objective function described in Section 2.3.1 is augmented with a weighted penalty term whose purpose is to help diversification. Whenever a local minimum is reached, non-improving moves (R,R') are penalized by a term proportional to the number of times that the attributes in $R'\backslash R$ have been added to the solution during the search. To take the number of iterations performed into account, these values are divided by the current iteration number λ. These penalties are further multiplied by a control parameter ζ. The cost of solution s' obtained by applying move (R,R') to solution s is thus computed as

$$g(s')=f(s')\left(1+\zeta\sum_{(i,k,\ell)\in R'\backslash R}\frac{r_{ik\ell}}{\lambda}\right), \tag{1}$$

where $r_{ik\ell}$ is the number of times attribute (i,k,ℓ) has been added to the solution since the beginning of the search. The value of the parameter ζ is set randomly from the uniform distribution on

[0,1] at the beginning of each improvement phase so as to vary the aggressiveness of the diversification.

2.3.5. Route refinement

The most important difference between the tabu search heuristic used here and the one introduced by Cordeau et al. [13] is the use of an intra-route optimization mechanism. During the course of the search, an intra-route refinement step is applied to each route after every few hundred iterations. This step consists in sequentially removing and reinserting each customer that belongs to a route by using the same route manipulation heuristics used to construct the initial solution and to explore the neighbourhood. Note that for problems without time windows, the parameter q controlling the neighbourhood size within the GENI algorithm is doubled during this intra-route optimization step. In the case of problems with time windows, a small improvement in terms of solution quality is achieved by randomizing the customer ordering before applying the refinement procedure. We have observed that a good balance between solution quality and computing time is obtained by performing this step every 200 iterations.

2.3.6. Tabu search stopping criterion

The tabu search heuristic stops after the best solution s^* has failed to improve for $\mu = \sqrt{(\eta-\lambda)\pi}$ consecutive iterations, where η is the total number of tabu search iterations to be performed during the complete execution of the ILS algorithm, λ is the current iteration number and π is the current perturbation size (see Section 2.4). The idea behind this rule is that the improvement phase should be allowed to run for a long time at the beginning of the solution process when it is called on a wide variety of solutions, but only for a short time at the end of the process when it is repeatedly called to improve similar solutions. The number of tabu search iterations to perform should also depend on the magnitude of the perturbation, which is represented by π.

2.4. Solution perturbation

The perturbation mechanism used in the ILS is inspired from the cluster removal heuristic used in the ALNS of Pisinger and Ropke [39]. To apply a perturbation, a cluster is constructed by choosing a seed customer at random and by identifying the π closest customers to the seed, where π is randomly selected in the interval $[0,\lceil\sqrt{n}\rceil]$. Next, these $\pi+1$ customers are removed from their routes and each customer i is assigned a new visit combination randomly chosen in the set C_i. Finally, customers are reinserted in the solution one by one, in a random order, by minimizing the increase in the cost $c(s)$.

2.5. Acceptance criterion

Whenever an improving solution is found during a call to the tabu search heuristic, it replaces the current best solution s^*. However, the working solution $\tilde{s}$ returned by the tabu search heuristic at the end of an improvement phase corresponds to the last solution visited by the search and not necessarily to the best one seen during this improvement phase. This solution $\tilde{s}$ is accepted with probability $1-(\lambda/\eta)^2$. This rule ensures that most solutions $\tilde{s}$ will be accepted in the early iterations of the ILS whereas the algorithm will increasingly revert to the best solution s^* as the search progresses.

2.6. Iterated local search stopping criterion

Because the most time-consuming step in the ILS heuristic is the improvement step in which tabu search is called to improve the perturbed solution s', we use as a stopping criterion for the ILS a total number of tabu search iterations. When the total number of iterations performed during all calls to the tabu search heuristic reaches η, the ILS stops regardless of the number of ILS iterations that have been performed.

3. Parallel iterated local search framework

To take advantage of the multiple cores available on modern CPUs, we have embedded the iterated tabu search heuristic described in the previous section within a parallel computing framework. Following the taxonomy introduced by Crainic et al. [17], the resulting algorithm is a pC/KS/MPDS heuristic: the control of the algorithm is shared between the various parallel processes (pC), knowledge about the solutions is shared at predetermined times (Knowledge Synchronization), and the search starts from different points using different parameters (multiple initial points, different search strategies).

In the parallel algorithm each process generates a different starting solution (using the same algorithm presented in Section 2.2 but with a different random seed). It then applies the improvement procedure, thus providing a natural multi-start environment. To further improve differentiation, each process chooses values for the δ, ζ, and τ parameters independently. However, to equally distribute the computation, the size of the perturbation π, and, by consequence, the maximum number of iterations without improvement μ are synchronized during the search. At the end of an improvement phase, each process $p \in \{1,2,\ldots,N\}$ decides whether to accept its working solution $\tilde{s}_p$ or to revert to the j-th best solution, with $j = \lfloor\sqrt{p}\rfloor$. We tested different strategies for this rule, including $p-1$, $p/2$, and $\sqrt{N}-\sqrt{p}$, but $\sqrt{p}$ seems to achieve the best balance between breadth and depth of search. Each process accepts its working solution with probability $1-(\lambda/\eta)^2$ as explained in Section 2.5.

This randomized procedure will spread the most promising solutions, during the execution, at an increasing rate: at the beginning the algorithm can broadly explore different parts of the solution space while near the end all processes will try to improve the best ones. This solution spreading mechanism is illustrated in Fig. 1.

Finally, to avoid losing useful information from any of the solutions explored by the different processes, a crossover step is performed before applying a perturbation. Each process applies a simple crossover with probability 0.1 using information from another solution chosen using a discrete uniform distribution. This crossover mechanism is similar to the perturbation described in Section 2.4 except that it uses the visit combinations obtained from the corresponding customers in the randomly chosen solution. Also, since it uses current information from another solution, the size of the cluster is sampled in the larger range $[0, 4\lceil\sqrt{n}\rceil]$.

4. Computational results

The algorithm was coded in C++ using the GNU Compiler Collection version 4 with some TR1 extensions, the Boost C++ Libraries, and the COIN-OR METSlib C++ framework for the development of single-point metaheuristics [32]. The parallel version was implemented using Boost::MPI and was run using OpenMPI 1.3. The sequential algorithm was run on a 2.93 GHz Intel Xeon CPU X 7 350, while the parallel runs were made on the Cottos cluster of RQCHP (Réseau québécois de calcul de haute performance), a Linux cluster with 128 nodes and Infiniband interconnections, each node being equipped with a 3 GHz dual Intel Xeon CPU E5 472. Preliminary results were reported by Maischberger and Cordeau [33].

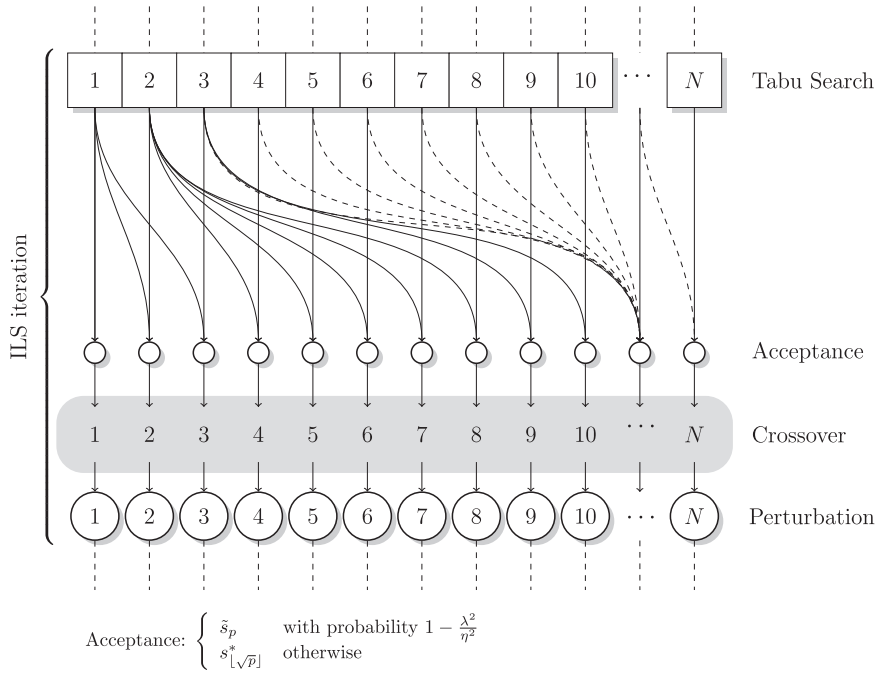

Fig. 1. A depiction of the spreading of the most promising solutions during the parallel algorithm execution. Each square represents an improvement phase, and a row contains the improvements that are executed in parallel. When going from one parallel ILS iteration to the next, each process takes its previous working solution or one of the most promising ones depending on randomness and the number of iterations remaining. Processes are ordered based on their solution quality. The gray box represents the crossover operation and the numbered circles the perturbations.

Table 1
Gaps with respect to best known solutions for the VRP.

Inst.	NB09		PR07 50K		ITS		
	Avg.	Best	Avg.	Best	Avg. 10^5	Avg. 10^6	Best 10^6
CMT	0.03	0.00	0.31	0.11	0.60	0.27	0.07
GWKC	0.12	0.02	1.02	0.49	1.30	0.79	0.41
t (s)	1291		423		188	1877	

Although our heuristic was designed with periodic, multi-depot or site-dependent problems in mind, it also performs reasonably well on the classical VRP and VRPTW. To support this statement, we compared our sequential algorithm with some of the best known heuristics. For these two problems, however, we do not report results for the parallel algorithm since it does not significantly improve over the sequential one.

Results obtained on the VRP instances of Christofides et al. [11] (CMT) and of Golden et al. [24] (GWKC) are reported in Table 1. In this table, columns NB09 and PR07 refer to the algorithms of Nagata and Bräysy [36] and of Pisinger and Ropke [39], respectively. Each entry indicates the percentage gap with respect to the best known solutions. For our algorithm, we report the average results obtained over 10 runs with $\eta = 10^5$ iterations and with $\eta = 10^6$ iterations, respectively. We also report the results for the best run with $\eta = 10^6$ iterations. For Nagata and Bräysy [36], the results correspond to the average over 10 runs and to the best run, respectively. For Pisinger and Ropke [39] the heuristic was applied to each instance five or 10 times, depending on the instance size, and 50,000 iterations were performed each time. In the last row of the table we report the average computing time in seconds required by the different algorithms. Nagata and Bräysy [36] used an Opteron 2.4 GHz computer whereas Pisinger and Ropke [39] ran their experiments on a 3 GHz Pentium 4. For Nagata and Bräysy [36] the reported average CPU time only takes into account the GWKC instances. Detailed results are reported in the Appendix. The results show that, although our algorithm is dominated by that of Nagata and Bräysy [36], it is competitive with that of Pisinger and Ropke [39]: the average results obtained with 10^6 iterations are similar to the average results obtained with 50,000 iterations of the ALNS. In addition, the best solutions found by the two algorithms are also similar. Our algorithm is, however, somewhat slower than the ALNS.

For the VRPTW, we report results on the classical Solomon [46] instances. For these instances, the primary objective is to minimize the number of vehicles while routing cost minimization is the secondary objective. Our algorithm does not have a direct mechanism to minimize the number of vehicles. Instead, we have set the fleet size m equal to the number m^* of vehicles used in the best known solution augmented by one, i.e., $m = m^* + 1$, and we have added a large cost per customer served by the vehicle with index m. As a result, the algorithm will try to empty the last route and find a feasible solution with just m^* routes. In Table 2 we report the results obtained by Nagata et al. [37] and by Pisinger and Ropke [39] (with 50,000 iterations). In each case, we indicate the average number of vehicles on the upper line and the average routing cost on the lower line. For our algorithm, we again report the average results obtained with $\eta = 10^5$ and $\eta = 10^6$ iterations as well as the best of the 10 runs with $\eta = 10^6$ iterations. In the last row we report the average computing time of the different algorithms. NBD10 used an Opteron 2.4 GHz, and PR07 a Pentium 4 running at 3 GHz. Detailed results are presented in the Appendix. Again, these results indicate that our algorithm is dominated by the specialized algorithm of Nagata et al. [37] but competitive with that of Pisinger and Ropke [39].

In Tables 3–8 we report the results obtained with the sequential and parallel implementations of our algorithm for the six other problem variants. Average gaps are taken over 10 runs and are computed with respect to previous best known solutions. Our results were obtained using the sequential algorithm with $\eta = 10^5$ and $\eta = 10^6$ iterations, and the parallel algorithm with $\eta = 125{,}000$ iterations on eight parallel processes (indicated by $P8$) for a total of 10^6 iterations and on 64 processes (indicated by P64) for a total

Table 2
Gaps with respect to best known solutions for the VRPTW.

	NBD10		PR07 50K		ITS		
	Avg.	Best	Avg.	Best	Avg. 10^5	Avg. 10^6	Best 10^6
R1	11.92	11.92	12.03	11.92	12.16	12.02	12.00
	1210.34	1210.34	1215.16	1212.39	1224.76	1213.57	1209.19
R2	2.73	2.73	2.75	2.73	2.73	2.73	2.73
	951.71	951.03	965.94	957.72	974.23	959.62	951.17
C1	10.00	10.00	10.00	10.00	10.00	10.00	10.00
	828.38	828.38	828.38	828.38	828.38	828.38	828.38
C2	3.00	3.00	3.00	3.00	3.00	3.00	3.00
	589.86	589.86	589.86	589.86	591.27	590.85	589.86
RC1	11.50	11.50	11.60	11.50	11.91	11.55	11.50
	1384.30	1384.17	1385.56	1385.78	1383.39	1386.39	1385.90
RC2	3.25	3.25	3.25	3.25	3.25	3.25	3.25
	1119.43	1119.24	1135.46	1123.49	1152.77	1130.27	1120.53
t (s)	300		146		236	2357	

Table 3
Gaps with respect to best known solutions for the PVRP.

	Average gaps (%)					Minimum gaps (%)					
	VCGLR11	ITS				CGL97	VCGLR11	ITS			
		10^5	10^6	$P8$	$P64$			10^5	10^6	$P8$	Our best
a	0.31	0.75	0.44	0.44	0.13	0.71	0.00	0.36	0.19	0.09	0.00
b	0.71	1.73	0.92	0.92	0.51	1.87	0.00	1.14	0.53	0.54	0.16
ab	0.41	0.99	0.55	0.55	0.22	0.98	0.00	0.55	0.27	0.20	0.04
t (s)	334	91	913	144	213						

Table 4
Gaps with respect to best known solutions for the MDVRP.

	Average gaps (%)					Minimum gaps (%)					
	VCGLR11	ITS				CGL97	VCGLR11	ITS			
		10^5	10^6	$P8$	$P64$			10^5	10^6	$P8$	Our best
a	0.06	0.44	0.19	0.16	0.05	0.19	0.00	0.18	0.07	0.05	0.02
b	0.13	1.03	0.51	0.43	0.16	0.79	0.00	0.47	0.19	0.15	0.04
ab	0.34	0.62	0.29	0.24	0.09	0.37	0.00	0.27	0.11	0.08	0.03
t (s)	254	110	1101	149	197						

Table 5
Gaps with respect to best known solutions for the SDVRP.

	Average gaps (%)						Minimum gaps (%)					
	PR07		ITS				CL01	PR07	ITS			
	25K	50K	10^5	10^6	$P8$	$P64$			10^5	10^6	$P8$	Our best
a	0.69	0.52	0.88	0.20	0.19	−0.05	1.17	0.00	0.13	−0.06	−0.09	−0.16
b	1.01	0.80	1.99	0.79	0.72	0.19	2.19	0.00	0.92	0.25	0.24	−0.13
ab	0.80	0.62	1.26	0.40	0.37	0.03	1.52	0.00	0.40	0.05	0.02	−0.15
t (s)	81	162	155	1548	238	348						

of 8×10^6 iterations. A direct comparison can be made between the three configurations using the same number of iterations: the average gap obtained using 10^6 iterations, the average gap obtained using 125,000 iterations and eight processes, and the best gap obtained by 10 independent runs with 10^5 iterations each. The "minimum gaps" section reports the percentage deviations relative to the previous best known solution obtained by the various algorithms, while the "our best" column reports the best solution found over all runs with the addition of two independent runs using 32 and 64 processes and 10^6 iterations per process

Table 6
Gaps with respect to best known solutions for the PVRPTW.

	Average gaps (%)					Minimum gaps (%)					
	PR08	ITS				CLM04	PR08	ITS			
	RVNS/2	10^5	10^6	*P*8	*P*64			10^5	10^6	*P*8	Our best
a	–	1.02	0.03	0.08	−0.45	0.82	0.00	0.32	−0.49	−0.44	−1.01
b	–	1.25	−0.10	−0.01	−0.80	1.14	0.00	0.23	−1.05	−0.67	−1.50
ab	–	1.13	−0.04	0.03	−0.62	0.98	0.00	0.28	−0.75	−0.55	−1.25
t (s)		293	2932	456	679						

Table 7
Gaps with respect to best known solutions for the MDVRPTW.

	Average gaps (%)					Minimum gaps (%)					
	PBDH08	ITS				CLM04	PBDH08	ITS			
	10^8	10^5	10^6	*P*8	*P*64			10^5	10^6	*P*8	Our best
a	0.82	1.45	0.66	0.58	0.14	0.52	0.00	0.60	0.28	0.12	−0.10
b	1.40	1.99	0.62	0.56	0.06	0.78	0.00	0.82	−0.03	0.08	−0.20
ab	1.11	1.72	0.64	0.57	0.10	0.65	0.00	0.71	0.12	0.10	−0.15
t (s)	8765	180	1799	249	394						

Table 8
Gaps with respect to best known solutions for the SDVRPTW.

	Average gaps (%)						Minimum gaps (%)				
	CLM04		ITS				CLM04	ITS			
	10^5	10^6	10^5	10^6	*P*8	*P*64		10^5	10^6	*P*8	Our best
a	1.44	0.49	0.63	−0.25	−0.23	−0.72	0.00	−0.28	−0.72	−0.76	−1.05
b	1.62	0.52	0.57	−0.40	−0.54	−0.90	0.00	−0.48	−1.00	−1.01	−1.33
ab	1.53	0.50	0.60	−0.33	−0.39	−0.81	0.00	−0.38	−0.86	−0.89	−1.19
t (s)	798	~8000	160	1596	272	336					

(3.2×10^7 and 6.4×10^7 iterations overall). For each problem, detailed results obtained with the sequential and the parallel versions of our algorithm are presented in the Appendix.

Results for the PVRP and MDVRP are reported in Tables 3 and 4 respectively, and are compared with those obtained by the hybrid genetic algorithm of Vidal et al. [48]. For the latter (VCGLR11), we report the average results over 10 runs. We also report the results obtained by the original tabu search of Cordeau et al. [13] (CGL97). The last line of the tables indicates the average computing time in seconds ([48] used a 2.4 GHz AMD Opteron 250 computer). For the 42 PVRP instances (described in [13]), we have obtained three new best solutions and 24 ties. The average values obtained with just 10^5 iterations are also very close to the best solutions identified by the original tabu search heuristic. In addition, the parallel version yields much better results, both with eight and 64 cores. When performing 10^6 iterations, our sequential heuristic is, however, slower than the heuristic of Vidal et al. [48] and produces slightly worse solutions on average. On the 33 MDVRP instances we have obtained 27 ties. Again, the ITS heuristic clearly outperforms the previous tabu search heuristic but is slower than that of Vidal et al. [48].

SDVRP results are presented in Table 5 and are compared with those of Pisinger and Ropke [39]. The comparison with

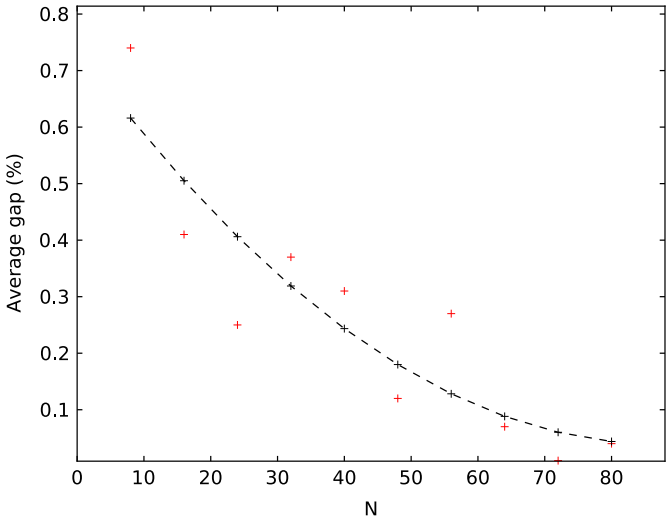

Fig. 2. Using instance pr05 of the PVRP benchmark set, the graph compares the solution quality against the number of processors used (with a fixed number of iterations per process). We report the quality of single solutions and a dashed interpolating line.

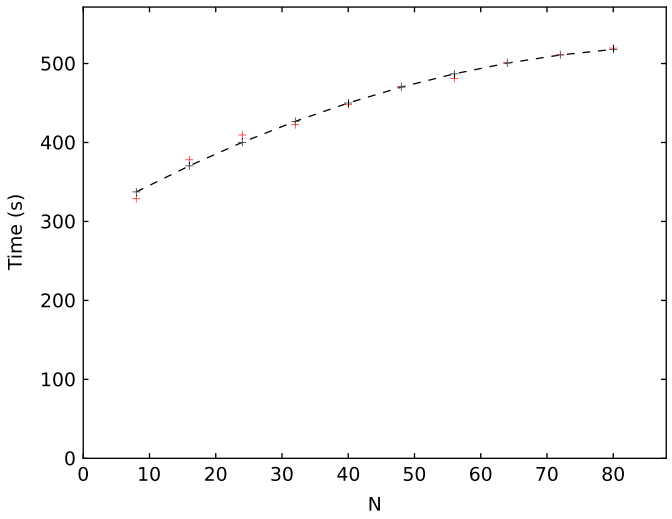

Fig. 3. Using instance pr05 of the PVRP benchmark set, the graph compares the time to completion against the number of processors used (with a fixed number of iterations per process). We report the time to completion of single solutions and a dashed interpolating line.

the original tabu search algorithm is made from the results of Cordeau and Laporte [12] (CL01). On the 35 instances, we have obtained 16 new best solutions and 17 ties. Our sequential algorithm appears to be competitive with that of Pisinger and Ropke [39] while the parallel one clearly achieves better results. Again, the comparison with the results reported by Cordeau and Laporte [12] shows that the sequential algorithm highly benefits from the iterated local search mechanism.

For each of the PVRPTW, MDVRPTW and SDVRPTW we have tested our algorithm on 20 instances divided into two sets with narrow and wide time windows, respectively (see [14,15]). Here, results are compared with those of Pirkwieser and Raidl [38] (PR08) for the PVRPTW and of Polacek et al. [41] (PBDH08) for the MDVRPTW. We should point out that for their variant using the forward time slack, Pirkwieser and Raidl [38] report only the best solutions found, not the average solution cost. As before, the last line reports the average computing time ([38] used a 2.2 GHz AMD Opteron 2214 PC, [41] a Pentium running at 3.6 GHz, and [15] a 2 GHz Pentium 4). For each problem, several new best solutions were found as can be seen from the detailed results reported in the Appendix. For the PVRPTW, one can observe that the sequential algorithm with 10^6 iterations and the parallel

Table 9
Comparison of results on the VRP for the sequential algorithm. All gaps are reported as % values w.r.t. the previous best known solutions. We provide the gaps for the average and the best values reported by Nagata and Bräysy [36]. The following columns report the gaps for the average and the best values obtained by our algorithm with 10^5 and 10^6 iterations, the cost of the best solution found during these runs, and the corresponding gap. All runs were repeated 10 times.

Inst.					Prev. best	NB09		ITS/1 10^5		ITS/1 10^6		ITS/1 best	
Name	n	m	D	Q	Cost	Avg.	Best	Avg.	Best	Avg.	Best	Cost	Gap
p01	50	5		160	524.61	–	–	–	–	–	–	524.61	–
p02	75	12		140	835.26	–	–	0.11	–	0.00	–	835.26	–
p03	100	10		200	826.14	–	–	0.17	–	0.06	–	826.14	–
p04	150	14		200	1028.42	–	–	0.62	0.13	0.30	–	1028.42	–
p05	199	20		200	1291.29	0.04	0.01	1.95	1.35	0.91	0.35	1295.78	0.35
p06	50	7	200	160	555.43	–	–	–	–	–	–	555.43	–
p07	75	13	160	140	909.68	0.08	–	–	–	–	–	909.68	–
p08	100	10	230	200	865.94	–	–	0.00	0.00	0.00	0.00	865.95	0.00
p09	150	16	200	200	1162.55	0.00	–	0.62	0.22	0.18	0.03	1162.89	0.03
p10	199	20	200	200	1395.85	0.18	–	1.22	0.49	0.83	0.43	1401.91	0.43
p11	120	8		200	1042.11	–	–	2.06	0.03	0.58	0.00	1042.12	0.00
p12	100	12		200	819.56	–	–	–	–	–	–	819.56	–
p13	100	13	720	200	1541.14	0.12	–	1.60	0.16	0.98	0.14	1543.27	0.14
p14	100	11	1040	200	866.37	–	–	–	–	–	–	866.37	–
Avg.					976.03	0.03	0.00	0.60	0.17	0.27	0.07	976.96	0.07
prn01	240	10	650	550	5626.81	0.09	–	0.74	0.47	0.64	0.30	5643.97	0.30
prn02	320	11	900	700	8431.66	0.10	–	0.41	0.32	0.27	0.17	8445.70	0.17
prn03	400	11	1200	900	11 036.22	–	–	0.24	0.02	0.11	−0.00	11 036.20	−0.00
prn04	480	11	1600	1000	13 592.88	0.19	–	0.76	0.33	0.78	0.27	13 629.30	0.27
prn05	200	6	1800	900	6460.98	–	–	1.02	–	0.27	–	6460.98	–
prn06	280	8	1500	900	8404.26	0.11	–	1.35	0.42	0.63	0.10	8412.90	0.10
prn07	360	10	1300	900	10 156.58	0.30	–	0.97	0.38	0.72	0.27	10 183.50	0.27
prn08	440	12	1200	900	11 643.90	0.41	0.41	1.72	0.59	1.20	0.26	11 673.70	0.26
prn09	255	15		1000	580.42	0.18	–	1.05	0.67	0.71	0.44	582.95	0.44
prn10	323	17		1000	738.49	0.14	–	1.27	0.81	0.64	0.20	739.95	0.20
prn11	399	19		1000	914.72	0.17	–	1.49	1.00	0.85	0.64	920.54	0.64
prn12	483	20		1000	1106.76	0.13	–	1.34	0.71	0.74	0.45	1111.76	0.45
prn13	252	27		1000	857.19	0.14	–	1.79	1.34	1.30	0.94	865.27	0.94
prn14	320	31		1000	1080.55	0.03	–	1.87	1.32	1.07	0.48	1085.76	0.48
prn15	396	34		1000	1342.53	0.13	–	2.17	1.70	1.30	0.99	1355.86	0.99
prn16	480	38		1000	1620.85	0.09	–	1.98	1.55	1.12	0.80	1633.84	0.80
prn17	240	23		200	707.76	0.00	–	0.53	0.13	0.27	0.06	708.17	0.06
prn18	300	28		200	995.13	0.08	–	1.71	1.27	1.17	0.58	1000.86	0.58
prn19	360	34		200	1365.97	0.05	–	1.30	0.92	0.74	0.52	1373.07	0.52
prn20	420	39		200	1820.02	0.09	–	2.32	1.85	1.20	0.69	1832.65	0.69
Avg.					4424.18	0.12	0.02	1.30	0.79	0.79	0.41	4434.85	0.41
Avg.					3004.35	0.08	0.01	1.01	0.53	0.58	0.27	3011.01	0.27

algorithm with 10^6 iterations subdivided between eight processes yield results similar to those of Pirkwieser and Raidl [38] while the sequential algorithm with just 10^5 iterations is dominated by the latter. The best solutions found by the parallel algorithm significantly improve upon the previous best known solutions. The situation for the MDVRPTW is similar in terms of average performance with respect to the algorithm of Polacek et al. [41]. However, the improvements in the best known solutions are much smaller. Finally, the results for the SDVRPTW indicate, once more, that the ITS algorithm is far superior to the original tabu search heuristic of Cordeau et al. [15].

To evaluate the scaling behaviour of the parallel algorithm, we have fixed the number of iterations for each process to 125,000. When fixing the number of iterations per process, the total number of iterations increases with the number of cores used, but the computing time, excluding inter-process communications, remains stable. To illustrate this, we have tested a medium size PVRP instance, pr05, using between eight and 80 processors. In Fig. 2 we compare the solution quality (% gap) reached when using different numbers of processors. In Fig. 3 we report information on performance degradation when increasing the number of processors: ideally the time should remain constant. We see in fact that it increases by about 25% every time the number of cores involved in the computation triples. The reason for this performance degradation (given the same number of iterations per processor) is twofold: on the one hand there is some overhead due to inter-process communication, while on the other hand, on current hardware, the real performance of an N-core CPU with ccNUMA architecture, when using all cores at the same time, is far less than that of N equivalent and independent CPUs.

5. Conclusions

This paper has introduced a common iterated tabu search algorithm to address the vehicle routing problem and several of its variants. Computational experiments show that embedding tabu search within the framework of iterated local search yields very significant improvements over the use of tabu search alone. In addition, the use of parallel computing brings further improvements and has allowed the identification of new best known solutions to a larger number of test problems. A major strength of the approach is its simplicity. The algorithm relies on a simple tabu search heuristic and a single type of perturbation to escape from local optima. The method is also very flexible in the sense that it can address many problem variants with a unified methodology and common parameter settings. Finally, the algorithm is reasonably fast and effective, being competitive with most recent heuristics for each particular problem variant. Future work will focus on extending the methodology to a wider array of problems including VRPs with pickup and delivery, location-routing problems and multi-echelon VRPs.

Table 10
Comparison of results on the VRPTW for the sequential algorithm (series 1). We report values for the number of vehicles and route duration and we compare first on the number of vehicles m and then on the total routing cost. From left to right we report the instance name, the best known solution, the best values reported by Nagata et al. [37], and the average and best results from 10 independent runs of our sequential algorithm with either 10^5 or 10^6 iterations.

Inst.	Best known		NBD10 best		ITS/1 10^5				ITS/1 10^6			
	m	Cost	m	Cost	Best m	Best cost	Avg. m	Avg. cost	Best m	Best cost	Avg. m	Avg. cost
R101	19	1645.79	19	1650.8	19	1654.37	19	1657.51	19	1650.80	19	1652.13
R102	17	1486.12	17	1486.12	17	1487.28	17	1491.75	17	1486.12	17	1487.49
R103	13	1292.68	13	1292.68	13	1297.89	13.9	1240.50	13	1294.23	13.2	1282.91
R104	9	1007.24	9	1007.31	10	983.00	10	1009.93	10	981.20	10	987.58
R105	14	1377.11	14	1377.11	14	1379.80	14	1390.30	14	1377.11	14	1377.99
R106	12	1251.98	12	1252.03	12	1257.96	12	1259.56	12	1252.62	12	1257.95
R107	10	1104.66	10	1104.66	10	1109.78	10	1116.27	10	1104.66	10	1111.53
R108	9	960.88	9	960.88	9	971.11	9	988.77	9	963.99	9	971.10
R109	11	1194.73	11	1194.73	11	1211.34	11.2	1236.63	11	1194.73	11	1201.62
R110	10	1118.59	10	1118.84	10	1125.75	10.1	1153.28	10	1118.84	10	1128.98
R111	10	1096.72	10	1096.73	10	1096.74	10	1127.66	10	1096.73	10	1099.83
R112	9	982.14	9	982.14	9	1000.63	9.7	1024.98	9	989.27	9	1003.69
R1 Avg.	11.92	1209.89	11.92	1210.34	12	1214.64	12.16	1224.76	12	1209.19	12.02	1213.57
C101	10	828.94	10	828.94	10	828.94	10	828.94	10	828.94	10	828.94
C102	10	828.94	10	828.94	10	828.94	10	828.94	10	828.94	10	828.94
C103	10	828.06	10	828.06	10	828.07	10	828.07	10	828.07	10	828.07
C104	10	824.78	10	824.78	10	824.78	10	824.78	10	824.78	10	824.78
C105	10	828.94	10	828.94	10	828.94	10	828.94	10	828.94	10	828.94
C106	10	828.94	10	828.94	10	828.94	10	828.94	10	828.94	10	828.94
C107	10	828.94	10	828.94	10	828.94	10	828.94	10	828.94	10	828.94
C108	10	828.94	10	828.94	10	828.94	10	828.94	10	828.94	10	828.94
C109	10	828.94	10	828.94	10	828.94	10	828.94	10	828.94	10	828.94
C1 Avg.	10	828.38	10	828.38	10	828.38	10	828.38	10	828.38	10	828.38
RC101	14	1696.94	14	1696.95	14	1696.95	14.8	1651.08	14	1696.95	14.1	1690.19
RC102	12	1554.75	12	1554.75	12	1558.24	12.8	1565.83	12	1554.75	12	1560.87
RC103	11	1261.67	11	1261.67	11	1262.98	11	1270.18	11	1261.67	11	1263.06
RC104	10	1135.48	10	1135.48	10	1135.52	10	1136.70	10	1135.48	10	1135.73
RC105	13	1629.44	13	1629.44	13	1636.94	13.9	1588.98	13	1633.72	13.3	1623.37
RC106	11	1424.73	11	1424.73	11	1432.12	11.8	1443.62	11	1424.73	11	1426.95
RC107	11	1230.48	11	1230.48	11	1232.26	11	1237.97	11	1232.20	11	1233.11
RC108	10	1139.82	10	1139.82	10	1151.22	10	1172.78	10	1147.69	10	1157.82
RC1 Avg.	11.5	1384.16	11.5	1384.17	11.5	1388.28	11.91	1383.39	11.5	1385.90	11.55	1386.39
Avg.	11.21	1139.56	11.21	1139.75	11.24	1142.67	11.42	1145.51	11.24	1139.76	11.26	1141.70

Table 11
Comparison of results on the VRPTW for the sequential algorithm (series 2). We report values for the number of vehicles and route duration and we compare first on the number of vehicles m and then on the total routing cost. From left to right we report the instance name, the best known solution, the best values reported by Nagata et al. [37], and the average and best results from 10 independent runs of our sequential algorithm with either 10^5 or 10^6 iterations.

Inst.	Best known		NBD10 best		ITS/1 10^5				ITS/1 10^6			
	m	Cost	m	Cost	Best m	Best cost	Avg. m	Avg. cost	Best m	Best cost	Avg. m	Avg. cost
R201	4	1252.37	4	1252.37	4	1253.02	4	1255.75	4	1252.37	4	1253.50
R202	3	1191.70	3	1191.7	3	1197.67	3	1219.56	3	1191.70	3	1201.00
R203	3	939.50	3	939.5	3	944.45	3	965.16	3	941.08	3	950.34
R204	2	825.52	2	825.52	2	829.87	2	856.03	2	825.52	2	833.61
R205	3	994.42	3	994.43	3	994.67	3	1011.62	3	994.43	3	995.18
R206	3	906.14	3	906.14	3	913.18	3	932.09	3	906.14	3	910.93
R207	2	890.61	2	890.61	2	905.28	2	919.78	2	890.61	2	901.70
R208	2	726.75	2	726.82	2	734.42	2	738.74	2	726.82	2	733.58
R209	3	909.16	3	909.16	3	915.16	3	926.43	3	909.16	3	917.60
R210	3	939.34	3	939.37	3	942.27	3	960.53	3	939.37	3	945.81
R211	2	885.71	2	885.71	2	899.32	2	930.87	2	885.71	2	912.53
R2 Avg.	2.73	951.02	2.73	951.03	2.73	957.21	2.73	974.23	2.73	951.17	2.73	959.62
C201	3	591.56	3	591.56	3	591.56	3	591.56	3	591.56	3	591.56
C202	3	591.56	3	591.56	3	591.56	3	602.80	3	591.56	3	591.56
C203	3	591.17	3	591.17	3	591.17	3	591.17	3	591.17	3	599.14
C204	3	590.60	3	590.6	3	590.60	3	590.66	3	590.60	3	590.60
C205	3	588.88	3	588.88	3	588.88	3	588.88	3	588.88	3	588.88
C206	3	588.49	3	588.49	3	588.49	3	588.49	3	588.49	3	588.49
C207	3	588.29	3	588.29	3	588.29	3	588.29	3	588.29	3	588.29
C208	3	588.32	3	588.32	3	588.32	3	588.32	3	588.32	3	588.32
C2 Avg.	3	589.86	3	589.86	3	589.86	3	591.27	3	589.86	3	590.85
RC201	4	1406.91	4	1406.94	4	1406.94	4	1432.55	4	1406.94	4	1410.15
RC202	3	1365.65	3	1365.65	3	1387.01	3	1436.34	3	1367.09	3	1400.67
RC203	3	1049.62	3	1049.62	3	1050.64	3	1087.29	3	1050.64	3	1061.31
RC204	3	798.41	3	798.46	3	808.98	3	819.08	3	798.46	3	800.06
RC205	4	1297.19	4	1297.65	4	1308.44	4	1322.79	4	1297.65	4	1305.28
RC206	3	1146.32	3	1146.32	3	1156.21	3	1179.93	3	1153.61	3	1160.88
RC207	3	1061.14	3	1061.14	3	1061.84	3	1096.55	3	1061.14	3	1067.40
RC208	3	828.14	3	828.14	3	828.71	3	847.61	3	828.71	3	836.45
RC2 Avg.	3.25	1119.17	3.25	1119.24	3.25	1126.10	3.25	1152.77	3.25	1120.53	3.25	1130.27
Avg.	2.96	893.83	2.96	893.86	2.96	898.41	2.96	913.66	2.96	894.30	2.96	900.92

Table 12
Comparison of results on the PVRP for the sequential algorithm. All gaps are reported as % values w.r.t. the previous best known solutions. For the latter solutions we report the value and the reference (CGW95 refers to [9], CGL97 to [13], HDH09 to [27], and VCGLR11 to [48]). We then provide the gaps for the values reported by Cordeau et al. [13] and by Vidal et al. [48]. The following columns report the gaps for the average values obtained by our algorithm with 10^5 and 10^6 iterations, the cost of the best solution found during these runs, and the corresponding gap. All runs were repeated 10 times.

Instance						Previous best		CGL97	VCGLR11		ITS/1		ITS/1 best	
Name	n	m	t	D	Q	Cost	Ref.	Best	Avg.	Best	Avg. 10^5	Avg. 10^6	Cost	Gap
p01	50	3	2		160	524.61	CGW95	–	–	–	–	–	524.61	–
p02	50	3	5		160	1322.87	CGW95	–	–	–	0.18	0.44	1322.87	–
p03	50	1	5		160	524.61	CGW95	–	–	–	–	–	524.61	–
p04	75	2	5		140	835.26	HDH09	0.02	0.16	–	0.69	0.09	835.26	–
p05	75	6	5		140	2024.96	VCGLR11	0.15	0.43	–	0.93	0.48	2030.54	0.28
p06	75	1	10		140	835.26	HDH09	0.13	0.86	–	0.72	0.13	835.26	–
p07	100	4	2		200	826.14	CGL97	–	0.11	–	0.30	0.08	826.14	–
p08	100	5	5		200	2022.47	VCGLR11	0.58	0.02	–	1.41	1.08	2034.15	0.58
p09	100	1	8		200	826.14	CGL97	–	0.10	–	0.17	0.06	826.14	–
p10	100	4	5		200	1593.43	VCGLR11	0.15	0.74	–	0.68	0.47	1594.11	0.04
p11	139	4	5		235	770.89	VCGLR11	1.09	0.64	–	2.48	1.26	774.85	0.51
p12	163	3	5		140	1186.47	VCGLR11	0.79	0.74	–	1.58	0.87	1187.78	0.11
p13	417	9	7		2000	3492.89	VCGLR11	0.54	3.06	–	2.38	0.74	3493.10	0.01
p14	20	2	4		20	954.81	CGW95	–	–	–	–	–	954.81	–
p15	38	2	4		30	1862.63	CGW95	–	–	–	–	–	1862.63	–
p16	56	2	4		40	2875.24	CGW95	–	–	–	–	–	2875.24	–
p17	40	4	4		20	1597.75	CGL97	–	–	–	–	–	1597.75	–
p18	76	4	4		30	3131.09	VCGLR11	0.52	–	–	0.66	0.77	3142.80	0.37
p19	112	4	4		40	4834.34	CGL97	–	0.00	–	0.34	0.18	4834.34	–
p20	184	4	4		60	8367.40	CGW95	–	–	–	–	–	8367.40	–
p21	60	6	4		20	2170.61	HDH09	0.62	–	–	0.62	0.62	2184.03	0.62

Table 12 (*continued*)

Instance						Previous best		CGL97	VCGLR11		ITS/1		ITS/1 best	
Name	*n*	*m*	*t*	*D*	*Q*	Cost	Ref.	Best	Avg.	Best	Avg. 10^5	Avg. 10^6	Cost	Gap
p22	114	6	4		30	4193.95	HDH09	1.84	0.01	–	1.76	0.47	4193.95	–
p23	168	6	4		40	6420.71	HDH09	2.83	0.21	–	2.18	2.41	6516.71	1.50
p24	51	3	6		20	3687.46	CGL97	–	–	–	0.15	0.21	3687.46	–
p25	51	3	6		20	3777.15	CGL97	–	–	–	0.10	0.08	3777.15	–
p26	51	3	6		20	3795.32	HDH09	0.00	–	–	–	–	3795.32	–
p27	102	6	6		20	21 833.87	VCGLR11	0.56	0.24	–	0.54	0.20	21 846.40	0.06
p28	102	6	6		20	22 242.51	VCGLR11	3.11	0.14	–	0.46	0.13	22 244.60	0.01
p29	102	6	6		20	22 543.75		1.62	0.09	–	0.26	0.19	22 551.10	0.03
p30	153	9	6		20	73 875.19	VCGLR11	1.55	0.89	–	1.77	0.91	74 233.20	0.48
p31	153	9	6		20	76 001.57	VCGLR11	2.87	0.90	–	1.92	1.16	76 636.80	0.84
p32	153	9	6		20	77 598.00	VCGLR11	3.71	0.74	–	1.81	0.99	78 208.40	0.79
Avg.						11 204.67		0.71	0.31	–	0.75	0.44	11 259.98	0.19
pr01	48	2	4	500	200	2209.02	CGL97	–	0.00	–	0.34	0.04	2209.02	–
pr02	96	4	4	480	195	3767.50	VCGLR11	0.84	0.04	–	0.87	0.31	3773.90	0.17
pr03	144	6	4	460	190	5153.54	VCGLR11	1.25	0.41	–	1.37	1.03	5178.97	0.49
pr04	192	8	4	440	185	5877.37	VCGLR11	2.30	1.00	–	2.27	1.20	5915.87	0.66
pr05	240	10	4	420	180	6581.86	VCGLR11	2.86	1.06	–	2.24	1.26	6644.25	0.95
pr06	288	12	4	400	175	8207.21	VCGLR11	2.62	0.95	–	2.36	1.33	8286.56	0.97
pr07	72	3	6	500	200	4996.14	HDH09	0.03	–	–	0.05	–	4996.14	–
pr08	144	6	6	475	190	6970.68	VCGLR11	1.78	0.93	–	2.05	1.01	6998.30	0.40
pr09	216	9	6	450	180	10 038.43	VCGLR11	3.31	1.23	–	2.85	1.35	10 103.90	0.65
pr10	288	12	6	425	170	12 897.01	VCGLR11	3.67	1.50	–	2.95	1.62	13 028.10	1.02
Avg.						6669.88		1.87	0.71	–	1.73	0.92	6713.50	0.53
Avg.						10 124.96		0.98	0.41	–	0.99	0.55	10 177.49	0.27

Table 13
Comparison of results on the PVRP for the parallel algorithm. All gaps are reported as % values w.r.t. the previous best known solutions. For the latter solutions we report the value and the reference (CGW95 refers to [9], CGL97 to [13], HDH09 to [27], and VCGLR11 to [48]). We then provide the gaps for the values reported by Cordeau et al. [13] and by Vidal et al. [48], and the gaps for the average values obtained by our sequential algorithm with 10^6 iterations. The following columns report the gaps for the average values with 8, 16, 32, and 64 cores. Finally, we report the cost of the best solution found during all runs, and the gap with respect to the previous best known solution.

Instance						Previous best		CGL97	VCGLR11		ITS/1	ITS/N average				ITS best	
Name	*n*	*m*	*t*	*D*	*Q*	Cost	Ref.	Best	Avg.	Best	Avg. 10^6	8	16	32	64	Cost	Gap
p01	50	3	2		160	524.61	CGW95	–	–	–	–	–	–	–	–	524.61	–
p02	50	3	5		160	1322.87	CGW95	–	–	–	0.44	–	–	–	–	1322.87	–
p03	50	1	5		160	524.61	CGW95	–	–	–	–	–	–	–	–	524.61	–
p04	75	2	5		140	835.26	HDH09	0.02	0.16	–	0.09	0.11	0.00	0.00	0.02	835.26	–
p05	75	6	5		140	2024.96	VCGLR11	0.15	0.43	–	0.48	0.36	0.32	3.31	0.20	2025.45	0.02
p06	75	1	10		140	835.26	VCGLR11	0.13	0.86	–	0.13	0.10	0.18	0.00	0.05	835.26	–
p07	100	4	2		200	826.14	CGL97	–	0.11	–	0.08	0.09	0.06	0.03	–	826.14	–
p08	100	5	5		200	2022.47	VCGLR11	0.58	0.02	–	1.08	0.64	0.45	0.55	0.43	2022.47	–
p09	100	1	8		200	826.14	CGL97	–	0.10	–	0.17	0.06	0.09	–	0.03	826.14	–
p10	100	4	5		200	1593.43	VCGLR11	0.15	0.74	–	0.47	0.28	0.12	0.11	0.10	1593.43	–
p11	139	4	5		235	770.89	VCGLR11	1.09	0.64	–	1.26	1.02	0.88	0.71	0.58	773.72	0.37
p12	163	3	5		140	1186.47	VCGLR11	0.79	0.74	–	0.87	0.77	0.52	0.14	0.19	1186.47	–
p13	417	9	7		2000	3492.89	VCGLR11	0.54	3.06	–	0.74	0.86	0.41	0.19	−0.30	3462.73	−0.86
p14	20	2	4		20	954.81	CGW95	–	–	–	–	–	–	–	–	954.81	–
p15	38	2	4		30	1862.63	CGW95	–	–	–	–	–	–	–	–	1862.63	–
p16	56	2	4		40	2875.24	CGW95	–	–	–	–	–	–	–	–	2875.24	–
p17	40	4	4		20	1597.75	CGL97	–	–	–	–	–	–	–	–	1597.75	–
p18	76	4	4		30	3131.09	VCGLR11	0.52	–	–	0.77	0.24	0.09	0.07	–	3131.09	–
p19	112	4	4		40	4834.34	CGL97	–	0.00	–	0.18	–	–	–	–	4834.34	–
p20	184	4	4		60	8367.40	CGW95	–	–	–	–	–	–	–	–	8367.40	–
p21	60	6	4		20	2170.61	HDH09	0.62	–	–	0.62	0.46	0.33	–	–	2170.61	–
p22	114	6	4		30	4193.95	HDH09	1.84	0.01	–	0.47	0.55	0.25	0.29	0.10	4193.95	–
p23	168	6	4		40	6420.71	HDH09	2.83	0.21	–	2.41	1.47	0.90	0.90	0.60	6420.71	–
p24	51	3	6		20	3687.46	CGL97	–	–	–	0.21	–	–	–	–	3687.46	–
p25	51	3	6		20	3777.15	CGL97	–	–	–	0.08	0.03	0.08	0.06	–	3777.15	–
p26	51	3	6		20	3795.32	HDH09	0.00	–	–	–	–	–	–	–	3795.32	–
p27	102	6	6		20	21 833.87	VCGLR11	0.56	0.24	–	0.20	0.23	0.19	0.16	0.14	21 838.30	0.02
p28	102	6	6		20	22 242.51	VCGLR11	3.11	0.14	–	0.13	0.16	0.12	0.03	0.03	22 242.50	−0.00
p29	102	6	6		20	22 543.75		1.62	0.09	–	0.19	0.13	0.14	0.09	0.08	22 545.70	0.01
p30	153	9	6		20	73 875.19	VCGLR11	1.55	0.89	–	0.91	0.86	0.69	0.71	0.63	74 062.90	0.25
p31	153	9	6		20	76 001.57	VCGLR11	2.87	0.90	–	1.16	0.99	0.87	0.75	0.54	76 098.10	0.13
p32	153	9	6		20	77 598.00	VCGLR11	3.71	0.74	–	0.99	0.95	0.87	0.62	0.77	77 711.20	0.15

Table 13 (*continued*)

Instance						Previous best		CGL97	VCGLR11		ITS/1	ITS/N average				ITS best	
Name	*n*	*m*	*t*	*D*	*Q*	Cost	Ref.	Best	Avg.	Best	Avg. 10^6	8	16	32	64	Cost	Gap
Avg.						11 204.67		0.71	0.31	–	0.44	0.32	0.24	0.18	0.13	11 216.45	0.00
pr01	48	2	4	500	200	2209.02	CGL97	–	0.00	–	0.04	0.02	–	–	–	2209.02	–
pr02	96	4	4	480	195	3767.50	VCGLR11	0.84	0.04	–	0.31	0.17	0.26	0.10	0.09	3767.54	0.00
pr03	144	6	4	460	190	5153.54	VCGLR11	1.25	0.41	–	1.03	0.67	0.69	0.43	0.33	5159.37	0.11
pr04	192	8	4	440	185	5877.37	VCGLR11	2.30	1.00	–	1.20	1.04	0.93	0.76	0.66	5892.41	0.26
pr05	240	10	4	420	180	6581.86	VCGLR11	2.86	1.06	–	1.26	1.19	1.11	0.71	0.64	6586.24	0.07
pr06	288	12	4	400	175	8207.21	VCGLR11	2.62	0.95	–	1.33	1.49	1.10	0.96	0.92	8234.15	0.33
pr07	72	3	6	500	200	4996.14	HDH09	0.03	–	–	–	–	–	–	–	4996.14	–
pr08	144	6	6	475	190	6970.68	VCGLR11	1.78	0.93	–	1.01	1.11	0.99	0.79	0.75	6997.42	0.38
pr09	216	9	6	450	180	10 038.43	VCGLR11	3.31	1.23	–	1.35	1.56	1.06	0.98	0.77	10 052.10	0.14
pr10	288	12	6	425	170	12 897.01	VCGLR11	3.67	1.50	–	1.62	1.84	1.67	1.58	0.95	12 935.40	0.30
Avg.						6669.88		1.87	0.71	–	0.92	0.91	0.78	0.63	0.51	6 682.98	0.16
Avg.						10 124.96		0.98	0.41	–	0.55	0.46	0.37	0.29	0.22	10 137.05	0.04

Table 14
Comparison of results on the MDVRP for the sequential algorithm. All gaps are reported as % values w.r.t. the previous best known solutions. For the latter solutions we report the value and the reference (CGW95 refers to [9], RLB96 to [44], CGL97 to [13], PR07 to [39], and VCGLR11 to [48]). We then provide the gaps for the values reported by Cordeau et al. [13] and by Vidal et al. [48]. The following columns report the gaps for the average values obtained by our algorithm with 10^5 and 10^6 iterations, the cost of the best solution found during these runs, and the corresponding gap. All runs were repeated 10 times.

Instance						Previous best		CGL97	VCGLR11		ITS/1		ITS/1 best	
Name	*n*	*m*	*t*	*D*	*Q*	Cost	Ref.	Best	Avg.	Best	Avg. 10^5	Avg. 10^6	Cost	Gap
p01	50	5	4		80	576.87	CGW95	–	–	–	–	–	576.87	–
p02	50	5	4		160	473.53	RLB96	–	–	–	–	–	473.53	–
p03	75	7	2		140	641.19	CGW95	–	–	–	0.05	–	641.19	–
p04	100	6	2		100	1001.04	PR07	0.05	0.02	–	0.51	0.16	1001.04	–
p05	100	6	2		200	750.03	CGL97	–	–	–	0.16	0.05	750.03	–
p06	100	8	3		100	876.50	RLB96	–	–	–	0.40	0.06	876.50	–
p07	100	1	4		100	881.97	PR07	0.43	0.28	–	0.70	0.25	881.97	–
p08	249	2	2	310	500	4372.78	VCGLR11	1.48	0.56	–	2.16	1.48	4409.35	0.84
p09	249	3	3	310	500	3858.66	VCGLR11	1.08	0.26	–	1.74	0.91	3881.16	0.58
p10	249	4	4	310	500	3631.11	VCGLR11	0.88	0.14	–	1.68	0.80	3633.88	0.08
p11	249	5	5	310	500	3546.06	PR07	0.23	0.06	–	1.03	0.22	3548.09	0.06
p12	80	6	2		60	1318.95	RLB96	–	–	–	–	–	1318.95	–
p13	80	2	2	200	60	1318.95	RLB96	–	–	–	–	–	1318.95	–
p14	80	4	2	180	60	1360.12	CGL97	–	–	–	–	–	1360.12	–
p15	160	6	4		60	2505.42	CGL97	–	–	–	–	–	2505.42	–
p16	160	8	4	200	60	2572.23	RLB96	–	–	–	0.05	–	2572.23	–
p17	160	10	4	180	60	2709.09	CGL97	–	–	–	–	–	2709.09	–
p18	240	12	6		60	3702.85	CGL97	–	–	–	0.14	0.03	3702.85	–
p19	240	4	6	200	60	3827.06	RLB96	–	–	–	0.26	–	3827.06	–
p20	240	6	6	180	60	4058.07	CGL97	–	–	–	–	–	4058.07	–
p21	360	9	9		60	5474.84	CGL97	–	0.03	–	0.65	0.22	5474.84	–
p22	360	3	9	200	60	5702.16	CGL97	–	–	–	0.23	0.11	5702.16	–
p23	360	4	9	180	60	6078.75	PR07	0.27	–	–	0.46	0.12	6078.75	–
Avg.						2662.53		0.19	0.06	–	0.44	0.19	2665.31	0.07
pr01	48	1	4	500	200	861.32	CGL97	–	–	–	–	–	861.32	–
pr02	96	2	4	480	195	1307.34	PR07	0.02	–	–	0.09	0.03	1307.34	–
pr03	144	3	4	460	190	1803.80	VCGLR11	0.16	–	–	0.29	0.08	1804.36	0.03
pr04	192	4	4	440	185	2058.31	VCGLR11	0.69	0.05	–	1.24	0.64	2058.47	0.01
pr05	240	5	4	420	180	2331.20	VCGLR11	2.34	0.39	–	1.93	1.04	2340.31	0.39
pr06	288	6	4	400	175	2676.30	VCGLR11	1.76	0.21	–	1.64	0.87	2688.54	0.46
pr07	72	1	6	500	200	1089.56	CGL97	–	–	–	–	–	1089.56	-
pr08	144	2	6	475	190	1664.85	PR07	0.11	0.01	–	0.46	0.12	1664.85	–
pr09	216	3	6	450	180	2133.20	VCGLR11	0.93	0.05	–	1.48	0.81	2141.45	0.39
pr10	288	4	6	425	170	2868.26	VCGLR11	1.87	0.64	–	3.18	1.52	2887.05	0.66
Avg.						1879.41		0.79	0.13	–	1.03	0.51	1884.33	0.19
Avg.						2425.22		0.37	0.08	–	0.62	0.29	2428.65	0.11

Table 15
Comparison of results on the MDVRP for the parallel algorithm. All gaps are reported as % values w.r.t. the previous best known solutions. For latter solutions we report the value and the reference (CGW95 refers to [9], RLB to [44], CGL97 to [13], PR07 to [39], and VCGLR11 to [48])). We then provide the gaps for the values reported by Cordeau et al. [13] and by Vidal et al. [48], and the gaps for the average values obtained by our sequential algorithm with 10^6 iterations. The following columns report the gaps for the average values with 8, 16, 32, and 64 cores. Finally, we report the cost of the best solution found during all runs, and the gap with respect to the previous best known solution.

Instance						Previous best		CGL97	VCGLR11		ITS/1	ITS/N average				ITS best	
Name	n	m	t	D	Q	Cost	Ref.	Best	Avg.	Best	Avg. 10^6	8	16	32	64	Cost	Gap
p01	50	5	4		80	576.87	CGW95	–	–	–	–	–	–	–	–	576.87	–
p02	50	5	4		160	473.53	RLB96	–	–	–	–	–	–	–	–	473.53	–
p03	75	7	2		140	641.19	CGW95	–	–	–	–	–	–	–	–	641.19	–
p04	100	6	2		100	1001.04	PR07	0.05	0.02	–	0.16	0.15	0.10	0.12	0.03	1001.04	–
p05	100	6	2		200	750.03	CGL97	–	–	–	0.05	0.04	–	–	–	750.03	–
p06	100	8	3		100	876.50	RLB96	–	–	–	0.06	0.05	0.01	–	–	876.50	–
p07	100	1	4		100	881.97	PR07	0.43	0.28	–	0.25	0.35	–	–	–	881.97	–
p08	249	2	2	310	500	4372.78	VCGLR11	1.48	0.56	–	1.48	1.35	1.25	1.15	0.71	4388.31	0.36
p09	249	3	3	310	500	3858.66	VCGLR11	1.08	0.26	–	0.91	0.68	0.59	0.49	0.28	3861.00	0.06
p10	249	4	4	310	500	3631.11	VCGLR11	0.88	0.14	–	0.80	0.68	0.42	0.56	0.17	3631.11	–
p11	249	5	5	310	500	3546.06	PR07	0.23	0.06	–	0.22	0.18	0.20	0.06	0.06	3547.85	0.05
p12	80	6	2		60	1318.95	RLB96	–	–	–	–	–	–	–	–	1318.95	–
p13	80	2	2	200	60	1318.95	RLB96	–	–	–	–	–	–	–	–	1318.95	–
p14	80	4	2	180	60	1360.12	CGL97	–	–	–	–	–	–	–	–	1360.12	–
p15	160	6	4		60	2505.42	CGL97	–	–	–	–	–	–	–	–	2505.42	–
p16	160	8	4	200	60	2572.23	RLB96	–	–	–	–	–	–	–	–	2572.23	–
p17	160	10	4	180	60	2709.09	CGL97	–	–	–	–	–	–	–	–	2709.09	–
p18	240	12	6		60	3702.85	CGL97	–	–	–	0.03	–	–	–	–	3702.85	–
p19	240	4	6	200	60	3827.06	RLB96	–	–	–	–	–	–	–	–	3827.06	–
p20	240	6	6	180	60	4058.07	CGL97	–	–	–	–	–	–	–	–	4058.07	–
p21	360	9	9		60	5474.84	CGL97	–	0.03	–	0.22	0.14	0.21	0.08	–	5474.84	–
p22	360	3	9	200	60	5702.16	CGL97	–	–	–	0.11	0.09	0.02	–	–	5702.16	–
p23	360	4	9	180	60	6078.75	PR07	0.27	–	–	0.12	0.06	–	–	–	6078.75	–
Avg.						2662.53		0.19	0.06	–	0.19	0.16	0.12	0.11	0.05	2663.39	0.02
pr01	48	1	4	500	200	861.32	CGL97	–	–	–	–	–	–	–	–	861.32	–
pr02	96	2	4	480	195	1307.34	PR07	0.02	–	–	0.03	–	–	–	–	1307.34	–
pr03	144	3	4	460	190	1803.80	VCGLR11	0.16	–	–	0.08	0.07	0.06	0.07	0.02	1803.82	0.00
pr04	192	4	4	440	185	2058.31	VCGLR11	0.69	0.05	–	0.64	0.54	0.52	0.11	0.13	2058.31	–
pr05	240	5	4	420	180	2331.20	VCGLR11	2.34	0.39	–	1.04	0.77	0.76	0.19	0.26	2331.20	–
pr06	288	6	4	400	175	2676.3	VCGLR11	1.76	0.21	–	0.87	0.72	0.56	0.54	0.25	2677.64	0.05
pr07	70	1	6	500	200	1089.56	CGL97	–	–	–	–	–	–	–	–	1089.56	–
pr08	144	2	6	475	190	1664.85	PR07	0.11	0.01	–	0.12	0.17	0.19	0.12	0.04	1664.85	–
pr09	216	3	6	450	180	2133.20	VCGLR11	0.93	0.05	–	0.81	0.70	0.47	0.45	0.27	2133.20	–
pr10	288	4	6	425	170	2868.26	VCGLR11	1.87	0.64	–	1.52	1.35	1.26	1.04	0.64	2878.35	0.35
Avg.						1879.41		0.79	0.13	–	0.51	0.43	0.38	0.25	0.16	1880.56	0.04
Avg.						2425.22		0.37	0.08	–	0.29	0.24	0.20	0.15	0.09	2426.17	0.03

Table 16
Comparison of results on the SDVRP for the sequential algorithm. All gaps are reported as % values w.r.t. the previous best known solutions. For the latter solutions we report the value and the reference (CGW95 refers to [10], CL01 to [12] and PR07 to [39]). We then provide the gaps for the values reported by Cordeau et al. [12] and by Vidal et al. [39]. The following columns report the gaps for the average values obtained by our algorithm with 10^5 and 10^6 iterations, the cost of the best solution found during these runs, and the corresponding gap. All runs were repeated 10 times.

Instance						Previous best		CL01	PR07		ITS/1		ITS/1 best	
Name	n	m	t	D	Q	Cost	Ref.	Best	Avg. 25K	Avg. 50K	Avg. 10^5	Avg. 10^6	Cost	Gap
p01	55	5	3		60	640.32	PR07	0.37	0.74	0.41	0.27	0.22	640.32	–
p02	52	5	2		120	598.10	CL01	–	0.22	0.12	–	–	598.10	–
p03	80	7	3		60	957.04	PR07	0.24	0.56	0.64	0.85	0.19	955.68	−0.14
p04	76	6	2		120	854.43	CL01	–	0.42	0.21	0.47	–	854.43	–
p05	103	6	3		100	1003.57	PR07	1.66	0.89	0.55	1.18	0.43	1003.57	–
p06	104	8	2		160	1028.52	PR07	0.73	0.54	0.40	0.51	0.24	1028.52	–
p07	27	1	3		150	391.30	CGW95	–	–	–	–	—	391.30	–
p08	54	2	3		150	664.46	CGW95	–	–	–	–	–	664.46	–
p09	81	3	3		150	948.23	CGW95	–	1.10	1.38	0.45	–	948.23	–
p10	108	4	3		150	1218.75	PR07	0.42	0.88	0.54	0.66	–	1218.75	–
p11	135	5	3		150	1463.33	PR07	0.11	1.71	0.86	0.64	−0.37	1448.17	−1.04
p12	162	6	3		150	1678.40	PR07	1.03	1.17	0.67	1.14	−0.40	1665.55	−0.77
p13	54	2	3		150	1194.18	PR07	0.21	0.02	0.06	0.26	0.18	1194.18	–
p14	108	4	3		150	1960.62	PR07	0.10	0.02	0.01	−0.03	−0.03	1959.96	−0.03
p15	162	6	3		150	2685.09	PR07	2.47	1.01	0.62	0.74	0.03	2685.09	–
p16	216	8	3		150	3396.36	PR07	2.79	0.75	0.45	2.05	0.44	3396.49	0.00
p17	270	10	3		150	4084.92	PR07	3.58	0.60	0.72	2.86	0.75	4091.12	0.15
p18	324	12	3		150	4755.50	PR07	3.66	1.39	0.84	3.35	1.44	4788.59	0.70

Table 16 (*continued*)

Instance						Previous best		CL01	PR07		ITS/1		ITS/1 best	
Name	*n*	*m*	*t*	*D*	*Q*	Cost	Ref.	Best	Avg. 25K	Avg. 50K	Avg. 10^5	Avg. 10^6	Cost	Gap
p19	104	4	3		150	846.07	PR07	0.51	0.71	0.29	0.05	0.00	843.15	−0.35
p20	156	6	3		150	1030.78	PR07	1.49	1.74	1.10	1.24	0.74	1033.39	0.25
p21	209	9	3		150	1271.75	PR07	5.20	0.77	1.67	2.07	0.51	1267.47	−0.34
p22	122	3	3		150	1008.71	PR07	0.41	0.16	0.01	0.80	0.18	1008.71	–
p23	102	4	3		150	803.29	PR07	1.92	0.55	0.46	0.64	0.12	803.29	–
Avg.						1499.29		1.17	0.69	0.52	0.88	0.20	1499.50	−0.07
pr01	48	1	4	500	100	1380.77	PR07	0.24	0.48	0.95	–	–	1380.77	–
pr02	96	2	4	500	100	2311.54	PR07	0.41	–	0.82	0.03	0.02	2311.54	–
pr03	144	3	4	500	100	2590.01	PR07	1.29	0.71	0.68	0.46	−0.33	2575.78	−0.55
pr04	192	4	4	500	100	3474.01	PR07	0.77	1.04	0.45	1.23	−0.07	3456.96	−0.49
pr05	240	5	4	500	100	4382.65	PR07	2.21	1.09	1.11	2.03	1.59	4403.54	0.48
pr06	288	6	4	500	100	4444.52	PR07	2.30	0.70	0.46	2.04	0.83	4422.02	−0.51
pr07	72	1	6	500	125	1889.82	PR07	3.45	1.94	1.41	2.53	0.30	1889.82	–
pr08	144	2	6	500	125	2976.76	PR07	3.55	0.84	1.05	1.41	0.21	2977.50	0.02
pr09	216	3	6	500	125	3536.20	PR07	3.62	1.28	0.88	3.05	1.12	3547.22	0.31
pr10	288	4	6	500	125	4646.96	PR07	1.99	0.62	0.57	2.21	1.24	4666.03	0.41
pr11	1008	21	4	500	100	12 719.65	PR07	4.00	2.11	0.72	5.23	3.00	13 002.00	2.22
pr12	720	10	6	500	125	9388.07	PR07	2.49	1.30	0.53	3.67	1.58	9448.61	0.64
Avg.						4478.41		2.19	1.01	0.80	1.99	0.79	4506.82	0.21
Avg.						2520.71		1.52	0.80	0.62	1.26	0.40	2530.58	0.03

Table 17
Comparison of results on the SDVRP for the parallel algorithm. All gaps are reported as % values w.r.t. the previous best known solutions. For latter solutions we report the value and the reference (CGW95 refers to [10], CL01 to [12] and PR07 to [39]). We then provide the gaps for the values reported by Cordeau and Laporte [12] and by Pisinger and Ropke [39], and the gaps for the average values obtained by our sequential algorithm with 10^6 iterations. The following columns report the gaps for the average values with 8, 16, 32, and 64 cores. Finally, we report the cost of the best solution found during all runs, and the gap with respect to the previous best known solution.

Instance						Previous best		CL01	PR07		ITS/1	ITS/N average				ITS best	
Name	*n*	*m*	*t*	*D*	*Q*	Cost	Ref.	Best	Avg. 25K	Avg. 50K	Avg. 10^6	8	16	32	64	Cost	Gap
p01	55	5	3		60	640.32	PR07	0.37	0.74	0.41	0.22	0.15	0.07	–	–	640.32	–
p02	52	5	2		120	598.10	CL01	–	0.22	0.12	–	–	–	–	–	598.10	–
p03	80	7	3		60	957.04	PR07	0.24	0.56	0.64	0.19	0.13	−0.08	0.03	−0.20	954.32	−0.28
p04	76	6	2		120	854.43	CL01	–	0.42	0.21	–	–	–	–	–	854.43	–
p05	103	6	3		100	1003.57	PR07	1.66	0.89	0.55	0.43	1.01	0.05	–	–	1003.57	–
p06	104	8	2		160	1028.52	PR07	0.73	0.54	0.40	0.24	0.22	0.11	0.07	0.03	1028.52	–
p07	27	1	3		150	391.30	CGW95	–	–	–	–	–	–	–	–	391.30	–
p08	54	2	3		150	664.46	CGW95	–	–	–	–	–	–	–	–	664.46	–
p09	81	3	3		150	948.23	CGW95	–	1.10	1.38	–	–	–	–	–	948.23	–
p10	108	4	3		150	1218.75	PR07	0.42	0.88	0.54	–	–	–	–	–	1218.75	–
p11	135	5	3		150	1463.33	PR07	0.11	1.71	0.86	−0.37	−0.41	−0.85	−1.04	−1.04	1448.17	−1.04
p12	162	6	3		150	1678.40	PR07	1.03	1.17	0.67	−0.40	0.01	−0.19	−0.51	−0.51	1665.55	−0.77
p13	54	2	3		150	1194.18	PR07	0.21	0.02	0.06	0.18	0.06	0.02	–	–	1194.18	–
p14	108	4	3		150	1960.62	PR07	0.10	0.02	0.01	−0.03	−0.03	−0.03	−0.03	−0.03	1959.96	−0.03
p15	162	6	3		150	2685.09	PR07	2.47	1.01	0.62	0.03	0.08	–	–	–	2685.09	–
p16	216	8	3		150	3396.36	PR07	2.79	0.75	0.45	0.44	0.48	0.09	0.02	0.09	3393.55	−0.08
p17	270	10	3		150	4084.92	PR07	3.58	0.60	0.72	0.75	0.64	0.47	0.23	0.09	4066.15	−0.46
p18	324	12	3		150	4755.50	PR07	3.66	1.39	0.84	1.44	1.59	1.28	1.17	0.61	4751.27	−0.09
p19	104	4	3		150	846.07	PR07	0.51	0.71	0.29	0.00	−0.17	−0.07	−0.15	−0.26	843.15	−0.35
p20	156	6	3		150	1030.78	PR07	1.49	1.74	1.10	0.74	0.63	0.52	0.38	0.31	1030.78	–
p21	209	9	3		150	1271.75	PR07	5.20	0.77	1.67	0.51	0.06	−0.11	−0.09	−0.29	1263.71	−0.63
p22	122	3	3		150	1008.71	PR07	0.41	0.16	0.01	0.18	0.00	–	–	–	1008.71	–
p23	102	4	3		150	803.29	PR07	1.92	0.55	0.46	0.12	0.03	–	–	–	803.29	–
Avg.						1499.29		1.17	0.69	0.52	0.20	0.19	0.06	0.00	−0.05	1496.33	−0.16
pr01	48	1	4	500	100	1380.77	PR07	0.24	0.48	0.95	–	–	–	–	–	1380.77	–
pr02	96	2	4	500	100	2311.54	PR07	0.41	–	0.82	0.02	−0.03	−0.06	−0.07	−0.20	2303.89	−0.33
pr03	144	3	4	500	100	2590.01	PR07	1.29	0.71	0.68	−0.33	−0.30	−0.33	−0.41	−0.56	2575.36	−0.57
pr04	192	4	4	500	100	3474.01	PR07	0.77	1.04	0.45	−0.07	0.05	−0.13	−0.10	−0.55	3449.84	−0.70
pr05	240	5	4	500	100	4382.65	PR07	2.21	1.09	1.11	1.59	1.08	0.64	0.65	0.09	4377.35	−0.12
pr06	288	6	4	500	100	4444.52	PR07	2.30	0.70	0.46	0.83	0.80	0.08	0.09	−0.03	4422.02	−0.51
pr07	72	1	6	500	125	1889.82	PR07	3.45	1.94	1.41	0.30	–	–	–	–	1889.82	–
pr08	144	2	6	500	125	2976.76	PR07	3.55	0.84	1.05	0.21	0.25	−0.01	–	0.01	2971.01	−0.19
pr09	216	3	6	500	125	3536.20	PR07	3.62	1.28	0.88	1.12	0.92	0.85	0.68	0.62	3536.20	–
pr10	288	4	6	500	125	4646.96	PR07	1.99	0.62	0.57	1.24	0.61	0.52	0.24	0.41	4639.62	−0.16
pr11	1008	21	4	500	100	12 719.65	PR07	4.00	2.11	0.72	3.00	3.60	2.76	2.43	1.73	12 845.60	0.99
pr12	720	10	6	500	125	9388.07	PR07	2.49	1.30	0.53	1.58	1.66	1.37	0.91	0.77	9392.84	0.05
Avg.						4478.41		2.19	1.01	0.80	0.79	0.72	0.47	0.37	0.19	4482.03	−0.13
Avg.						2520.71		1.52	0.80	0.62	0.40	0.37	0.20	0.13	0.03	2520.00	−0.15

Table 18
Comparison of results on the PVRPTW for the sequential algorithm. All gaps are reported as % values w.r.t. the previous best known solutions reported in PR08 [38]. We provide the gaps for the values reported by Cordeau and Laporte [15]. The following columns report the gaps for the average values obtained by our algorithm with 10^5 and 10^6 iterations, the cost of the best solution found during these runs, and the corresponding gap. All runs were repeated 10 times.

Instance						PR08 best	CLM04	ITS/1		ITS/1 best	
Name	n	m	t	D	Q	Cost	Best	Avg. 10^5	Avg. 10^6	Cost	Gap
pr01	48	3	4	500	200	2909.02	0.07	0.09	0.00	2909.02	–
pr02	96	6	4	480	195	5036.27	0.37	1.04	0.27	5032.06	−0.08
pr03	144	9	4	460	190	7138.70	1.28	0.91	−0.08	7098.57	−0.56
pr04	192	12	4	440	185	7882.06	0.90	1.69	0.47	7859.81	−0.28
pr05	240	15	4	420	180	8492.45	1.18	1.30	−0.13	8425.26	−0.79
pr06	288	18	4	400	175	10 713.75	1.99	1.92	0.44	10 655.10	−0.55
pr07	72	5	6	500	200	6787.72	0.55	0.70	0.25	6788.95	0.02
pr08	144	10	6	475	190	9721.25	0.28	1.04	−0.26	9623.41	−1.01
pr09	216	15	6	450	180	13 463.96	1.12	1.08	0.12	13 406.70	−0.43
pr10	288	20	6	425	170	17 650.89	0.48	0.44	−0.84	17 443.70	−1.17
Avg.						8979.61	0.82	1.02	0.03	8924.26	−0.49
pr11	48	3	4	500	200	2277.44	0.73	1.85	1.01	2277.44	–
pr12	96	6	4	480	195	4137.45	2.90	3.00	1.77	4144.97	0.18
pr13	144	9	4	460	190	5575.27	1.32	1.60	0.15	5536.45	−0.70
pr14	192	12	4	440	185	6476.67	1.82	1.02	−0.14	6383.28	−1.44
pr15	240	15	4	420	180	6970.33	1.21	0.68	−1.28	6834.57	−1.95
pr16	288	18	4	400	175	8819.32	1.24	0.35	−0.85	8671.40	−1.68
pr17	72	4	6	500	200	5504.67	0.01	1.09	0.30	5482.51	−0.40
pr18	144	8	6	475	190	7729.32	1.89	1.96	0.50	7720.87	−0.11
pr19	216	12	6	450	180	10 885.93	0.04	1.40	−0.60	10 707.60	−1.64
pr20	288	16	6	425	170	13 943.61	0.26	−0.42	−1.91	13 560.90	−2.74
Avg.						7232.00	1.14	1.25	−0.10	7132.00	−1.05
Avg.						8105.80	0.98	1.13	−0.04	8070.80	−0.75

Table 19
Comparison of results on the PVRPTW for the parallel algorithm. All gaps are reported as % values w.r.t. the previous best known solutions reported in PR08 [38]. We provide the gaps for the values reported by Cordeau et al. [15] and for the average values obtained by our sequential algorithm with 106 iterations. The following columns report the gaps for the average values with 8, 16, 32, and 64 cores. Finally, we report the cost of the best solution found during all runs, and the gap with respect to the previous best known solution.

Instance						PR08 best	CLM04	ITS/1	ITS/N average				ITS best	
Name	n	m	t	D	Q	Cost	Best	Avg. 10^6	8	16	32	64	Cost	Gap
pr01	48	3	4	500	200	2909.02	0.07	0.00	–	–	–	–	2909.02	–
pr02	96	3	4	480	195	5036.27	0.37	0.27	0.21	0.10	−0.00	−0.03	5026.57	−0.19
pr03	144	4	4	460	190	7138.70	1.28	−0.08	−0.06	−0.27	−0.39	−0.58	7062.00	−107
pr04	192	5	4	440	185	7882.06	0.90	0.47	0.53	0.24	−0.11	0.32	7807.32	−0.95
pr05	240	6	4	420	180	8492.45	1.18	−0.13	0.30	−0.04	−0.61	−0.69	8358.96	−1.57
pr06	288	7	4	400	175	10 713.75	1.99	0.44	0.40	−0.05	−0.67	−0.46	10 542.10	−1.60
pr07	72	2	6	500	200	6787.72	0.55	0.25	0.17	0.22	0.18	0.18	6782.68	−0.07
pr08	144	3	6	475	190	9721.25	0.28	−0.26	0.08	−0.26	−0.54	−0.37	9603.92	−1.21
pr09	216	4	6	450	180	13 463.96	1.12	0.12	−0.03	−0.46	−0.21	−0.69	13 299.80	−1.22
pr10	288	5	6	425	170	17 650.89	0.48	−0.84	−0.83	−0.98	−1.51	−1.62	17 261.30	−2.21
Avg.						8979.61	0.82	0.03	0.08	−0.15	−0.39	−0.45	8865.37	−1.01
pr11	48	3	4	500	200	2277.44	0.73	1.01	0.04	–	–	–	2277.44	–
pr12	96	6	4	480	195	4137.45	2.90	1.77	1.53	1.40	0.68	0.04	4124.76	−0.31
pr13	144	9	4	460	190	5575.27	1.32	0.15	0.47	0.33	0.26	−0.06	5489.84	−1.53
pr14	192	12	4	440	185	6476.67	1.82	−0.14	0.09	−0.17	−0.65	−0.67	6383.28	−1.44
pr15	240	15	4	420	180	6970.33	1.21	−1.28	−0.97	−1.16	−1.46	−1.77	6800.45	−2.44
pr16	288	18	4	400	175	8819.32	1.24	−0.85	−0.67	−0.99	−1.24	−1.40	8659.44	−1.81
pr17	72	4	6	500	200	5504.67	0.01	0.30	0.18	−0.01	−0.06	−0.18	5481.61	−0.42
pr18	144	8	6	475	190	7729.32	1.89	0.50	0.56	0.24	0.06	−0.04	7656.13	−0.95
pr19	216	12	6	450	180	10 885.93	0.04	−0.60	−0.13	−0.40	−0.67	−1.24	10 579.50	−2.81
pr20	288	16	6	425	170	13 943.61	0.26	−1.91	−1.19	−1.76	−1.89	−2.63	13 490.80	−3.25
Avg.						7232.00	1.14	−0.10	−0.01	−0.25	−0.50	−0.80	7094.33	−1.50
Avg.						8105.80	0.98	−0.04	0.03	−0.20	−0.44	−0.62	7979.85	−1.25

Table 20
Comparison of results on the MDVRPTW for the sequential algorithm. All gaps are reported as % values w.r.t. the previous best known solutions. For the latter solutions we report the value and the reference (CLM04 refers to [15], PHDR04 refers to [40] and PBDH08 to [41]). We then provide the gaps for the values reported by Cordeau et al. [15] and for the average values obtained over 32 runs and 10^8 iterations by Polacek et al. [41]. The following columns report the gaps for the average values obtained by our algorithm with 10^5 and 10^6 iterations, the cost of the best solution found during these runs, and the corresponding gap. All runs were repeated 10 times.

Instance						Previous best		CLM04	PBDH08	ITS/1		ITS/1 best	
Name	n	m	t	D	Q	Cost	Ref.	Best	Avg.	Avg. 10^5	Avg. 10^6	Cost	Gap
pr01	48	2	4	500	200	1074.12	CLM04	–	–	0	–	1074.12	–
pr02	96	3	4	480	195	1762.21	PHDR04	–	0.08	0.17	0.04	1762.21	–
pr03	144	4	4	460	190	2373.65	PHDR04	–	0.64	1.58	0.79	2387.03	0.56
pr04	192	5	4	440	185	2815.48	PHDR04	1.31	1.14	2.02	0.80	2823.95	0.30
pr05	240	6	4	420	180	2965.18	PBDH08	2.17	1.69	2.84	1.84	3004.58	1.33
pr06	288	7	4	400	175	3612.72	PBDH08	0.40	1.71	2.00	1.03	3612.15	−0.02
pr07	72	2	6	500	200	1418.22	PHDR04	–	–	0.41	–	1418.22	–
pr08	144	3	6	475	190	2096.73	PHDR04	0.28	0.31	1.57	0.78	2100.58	0.18
pr09	216	4	6	450	180	2727.42	PBDH08	0.38	0.96	2.31	0.86	2731.13	0.14
pr10	288	5	6	425	170	3483.22	PBDH08	0.63	1.66	1.56	0.47	3486.55	0.10
Avg.						2432.90		0.52	0.82	1.45	0.66	2440.05	0.26
pr11	48	1	4	500	200	1005.73	PHDR04	–	0.59	0.98	0.86	1005.73	–
pr12	96	2	4	480	195	1467.72	PBDH08	0.74	1.40	1.89	0.63	1470.20	0.17
pr13	144	3	4	460	190	2001.83	PHDR04	0.47	0.53	0.96	0.33	2001.81	−0.00
pr14	192	4	4	440	185	2196.28	PBDH08	0.26	1.95	3.05	0.61	2195.33	−0.04
pr15	240	5	4	420	180	2456.52	PBDH08	1.55	1.72	2.46	0.58	2445.97	−0.43
pr16	288	6	4	400	175	2853.32	PBDH08	1.67	1.97	2.05	0.84	2855.67	0.08
pr17	72	1	6	500	200	1236.24	PHDR04	–	0.91	1.14	0.06	1236.24	–
pr18	144	2	6	475	190	1788.18	PBDH08	0.25	1.18	2.21	0.53	1788.18	–
pr19	216	3	6	450	180	2269.33	PBDH08	0.69	1.10	2.38	0.84	2270.14	0.04
pr20	288	4	6	425	170	3013.71	PBDH08	2.17	2.65	2.79	0.91	3010.16	−0.12
Avg.						2028.89		0.78	1.40	1.99	0.62	2027.94	−0.03
Avg.						2230.89		0.65	1.11	1.72	0.64	2234.00	0.11

Table 21
Comparison of results on the MDVRPTW for the parallel algorithm. All gaps are reported as % values w.r.t. the previous best known solutions. For the latter solutions we report the value and the reference (CLM04 refers to [15], PHDR04 refers to [40] and PBDH08 to [41]). We then provide the gaps for the values reported by Cordeau et al. [15], for the average values over 32 runs and 10^8 iterations by Polacek et al. [41], and for the average values obtained by our sequential algorithm with 10^6 iterations. The following columns report the gaps for the average values with 8, 16, 32, and 64 cores. Finally, we report the cost of the best solution found during all runs, and the gap with respect to the previous best known solution.

Instance						Previous best		CLM04	PBDH08	ITS/1	ITS/N average				ITS best	
Name	n	m	t	D	Q	Cost	Ref.	Best	Avg.	Avg. 10^6	8	16	32	64	Cost	Gap
pr01	48	2	4	500	200	1074.12	CLM04	–	–	–	–	–	–	–	1074.12	–
pr02	96	3	4	480	195	1762.21	PHDR04	–	0.08	0.04	0.03	0.01	0.01	–	1762.21	–
pr03	144	4	4	460	190	2373.65	PHDR04	–	0.64	0.79	0.89	0.50	0.72	0.28	2373.65	–
pr04	192	5	4	440	185	2815.48	PHDR04	1.31	1.14	0.80	0.92	0.65	0.58	0.26	2819.76	0.15
pr05	240	6	4	420	180	2965.18	PBDH08	2.17	1.69	1.84	1.44	1.22	0.97	0.74	2971.90	0.23
pr06	288	7	4	400	175	3612.72	PBDH08	0.40	1.71	1.03	0.71	0.63	0.53	0.11	3590.58	−0.61
pr07	72	2	6	500	200	1418.22	PHDR04	–	–	–	–	–	–	–	1418.22	–
pr08	144	3	6	475	190	2096.73	PHDR04	0.28	0.31	0.78	0.69	0.35	0.32	0.23	2096.73	–
pr09	216	4	6	450	180	2727.42	PBDH08	0.38	0.96	0.86	0.44	0.48	0.16	−0.13	2717.69	−0.36
pr10	288	5	6	425	170	3483.22	PBDH08	0.63	1.66	0.47	0.63	0.20	−0.06	−0.05	3469.29	−0.40
Avg.						2432.90		0.52	0.82	0.66	0.58	0.40	0.32	0.14	2429.42	−0.10
pr11	48	1	4	500	200	1005.73	PHDR04	–	0.59	0.86	–	0.37	–	–	1005.73	–
pr12	96	2	4	480	195	1467.72	PBDH08	0.74	1.40	0.63	0.40	0.22	−0.02	0.04	1464.50	−0.22
pr13	144	3	4	460	190	2001.83	PHDR04	0.47	0.53	0.33	0.14	0.08	−0.00	–	2001.81	−0.00
pr14	192	4	4	440	185	2196.28	PBDH08	0.26	1.95	0.61	0.73	0.49	0.32	0.16	2195.33	−0.04
pr15	240	5	4	420	180	2456.52	PBDH08	1.55	1.72	0.58	0.83	0.68	−0.10	−0.51	2434.94	−0.88
pr16	288	6	4	400	175	2853.32	PBDH08	1.67	1.97	0.84	0.78	0.54	0.38	0.32	2852.25	−0.04
pr17	72	1	6	500	200	1236.24	PHDR04	–	0.91	0.06	0.18	0.12	–	0.06	1236.24	–
pr18	144	2	6	475	190	1788.18	PBDH08	0.25	1.18	0.53	0.32	0.35	0.36	0.13	1788.18	–
pr19	216	3	6	450	180	2269.33	PBDH08	0.69	1.10	0.84	0.62	0.57	0.56	0.31	2263.74	−0.25
pr20	288	4	6	425	170	3013.71	PBDH08	2.17	2.65	0.91	1.55	0.94	0.34	0.10	2995.08	−0.62
Avg.						2028.89		0.78	1.40	0.62	0.56	0.44	0.18	0.06	2023.78	−0.20
Avg.						2230.89		0.65	1.11	0.64	0.57	0.42	0.25	0.10	2226.60	−0.15

Acknowledgements

This work was partly funded by the Natural Sciences and Engineering Research Council of Canada under Grant 227837-09. This support is gratefully acknowledged.

Appendix A. Detailed computational results

Tables 9–23 report the detailed computational results obtained with the sequential ITS heuristic on the VRP and VRPTW, and with both the sequential and parallel ITS heuristics

Table 22
Comparison of results on the SDVRPTW for the sequential algorithm. All gaps are reported as % values w.r.t. the previous best known solutions reported by CLM04 [15]. The following columns report the gaps for the average values obtained by our algorithm with 10^5 and 10^6 iterations, the cost of the best solution found during these runs, and the corresponding gap. All runs were repeated 10 times.

Instance						CLM04 best	ITS/1		ITS/1 best	
Name	n	m	t	D	Q	Cost	Avg. 10^5	Avg. 10^6	Cost	Gap
pr01	48	2	4	500	100	1655.42	–	–	1655.42	–
pr02	96	3	4	500	100	2904.13	0.28	0.02	2904.13	–
pr03	144	4	4	500	100	3321.44	0.87	0.03	3317.33	−0.12
pr04	192	5	4	500	100	4509.36	0.12	−0.56	4461.13	−1.07
pr05	240	6	4	500	100	5777.56	0.48	−0.79	5663.32	−1.98
pr06	288	7	4	500	100	5769.87	1.71	−0.26	5698.93	−1.23
pr07	72	2	6	500	125	2167.46	0.31	0.07	2166.88	−0.03
pr08	144	3	6	500	125	3904.84	1.07	−0.25	3880.58	−0.62
pr09	216	4	6	500	125	4875.62	0.75	−0.49	4818.32	−1.18
pr10	288	5	6	500	125	5969.50	0.66	−0.26	5908.53	−1.02
Avg.						4085.52	0.63	−0.25	4047.46	−0.72
pr11	48	2	4	500	100	1429.35	–	–	1429.35	–
pr12	96	3	4	500	100	2494.53	0.19	−0.07	2479.56	−0.60
pr13	144	4	4	500	100	2798.46	0.56	−0.14	2781.22	−0.62
pr14	192	5	4	500	100	3711.96	0.78	−0.49	3674.53	−1.01
pr15	240	6	4	500	100	4672.41	1.78	0.13	4613.58	−1.26
pr16	288	7	24	500	100	4873.08	0.76	−0.75	4788.39	−1.74
pr17	72	2	6	500	125	1837.94	0.62	0.06	1837.94	–
pr18	144	3	6	500	125	3170.46	0.27	−0.44	3149.77	−0.65
pr19	216	4	6	500	125	3985.52	1.28	−0.53	3937.53	−1.20
pr20	288	5	6	500	125	5147.68	−0.55	−1.77	4996.72	−2.93
Avg.						3412.14	0.57	−0.40	3368.86	−1.00
Avg.						3748.83	0.60	−0.33	3708.16	−0.86

Table 23
Comparison of results on the SDVRPTW for the parallel algorithm. All gaps are reported as % values w.r.t. the previous best known solutions reported by CLM04 [15]. We then provide the gaps for the average values obtained by our sequential algorithm with 10^6 iterations. The following columns report the gaps for the average values with 8, 16, 32, and 64 cores. Finally, we report the cost of the best solution found during all runs, and the gap with respect to the previous best known solution.

Instance						CLM04 best	ITS/1	ITS/N average				ITS/1 best	
Name	n	m	t	D	Q	Cost	Avg. 10^6	8	16	32	64	Cost	Gap
pr01	48	2	4	500	100	1655.42	–	–	–	–	–	1655.42	–
pr02	96	3	4	500	100	2904.13	0.02	–	–	–	–	2904.13	–
pr03	144	4	4	500	100	3321.44	0.03	0.03	−0.05	−0.22	−0.13	3304.13	−0.52
pr04	192	5	4	500	100	4509.36	−0.56	−0.52	−0.83	−0.88	−1.35	4438.97	−1.56
pr05	240	6	4	500	100	5777.56	−0.79	−0.95	−1.24	−1.83	−1.93	5620.56	−2.72
pr06	288	7	4	500	100	5769.87	−0.26	−0.03	−0.56	−0.97	−1.02	5670.66	−1.72
pr07	72	2	6	500	125	2167.46	0.07	0.07	−0.03	−0.03	−0.03	2166.88	−0.03
pr08	144	3	6	500	125	3904.84	−0.25	0.10	−0.14	−0.07	−0.50	3874.32	−0.78
pr09	216	4	6	500	125	4875.62	−0.49	−0.43	−0.75	−0.74	−0.75	4801.47	−1.52
pr10	288	5	6	500	125	5969.50	−0.26	−0.57	−0.75	−1.10	−1.47	5868.03	−1.70
Avg.						4085.52	−0.25	−0.23	−0.44	−0.58	−0.72	4040.46	−1.05
pr11	48	2	4	500	100	1429.35	–	–	–	–	–	1429.35	–
pr12	96	2	4	500	100	2494.53	−0.07	−0.19	−0.23	−0.14	−0.48	2479.56	−0.60
pr13	144	4	4	500	100	2798.46	−0.14	−0.45	−0.47	−0.47	−0.65	2775.61	−0.82
pr14	192	5	4	500	100	3711.96	−0.49	−0.45	−0.77	−0.94	−1.05	3655.48	−1.52
pr15	240	6	4	500	100	4672.41	0.13	0.13	−0.15	−0.49	−0.39	4613.09	−1.27
pr16	288	7	4	500	100	4873.08	−0.75	−1.01	−1.19	−1.54	−1.96	4752.04	−2.48
pr17	72	2	6	500	125	1837.94	0.06	0.06	–	–	–	1837.94	–
pr18	144	3	6	500	125	3170.46	−0.44	−0.30	−0.44	−0.63	−0.66	3144.91	−0.81
pr19	216	4	6	500	125	3985.52	−0.53	−0.65	−1.15	−1.51	−1.14	3894.64	−2.28
pr20	288	5	6	500	125	5147.68	−1.77	−2.59	−2.55	−3.01	−2.69	4967.59	−3.50
Avg.						3412.14	−0.40	−0.54	−0.70	−0.87	−0.90	3355.02	−1.33
Avg.						3748.83	−0.33	−0.39	−0.57	−0.73	−0.81	3692.74	−1.19

on the PVRP, MDVRP, SDVRP, PVRPTW, MDVRPTW and SDVRPTW.

For the parallel heuristic, experiments with up to 16 cores were run 10 times whereas those involving a larger number of cores were run five times. All parallel runs were configured for 125,000 iterations, so that the overall number of iterations when using eight cores is the same as for the sequential algorithm. For Tables 12–23, the last two columns report the cost of the best solution value found and the corresponding gap. These values correspond to the best solution found by either of the two algorithms for the runs described, plus two more runs using 32 and 64 cores and configured for 10^6 iterations per core.

References

[1] Alegre J, Laguna M, Pacheco J. Optimizing the periodic pick-up of raw materials for a manufacturer of auto parts. European Journal of Operational Research 2007;179:736–46.

[2] Baldacci R, Mingozzi A. A unified exact method for solving different classes of vehicle routing problems. Mathematical Programming 2009;120:347–80.

[3] Baldacci R, Bartolini E, Mingozzi A, Roberti R. An exact solution framework for a broad class of vehicle routing problems. Computational Management Science 2010;7:229–68.

[4] Baldacci R, Bartolini E, Mingozzi A, Valletta A. An exact algorithm for the period routing problem. Operations Research 2011;59:228–41.

[5] Beltrami EJ, Bodin LD. Networks and vehicle routing for municipal waste collection. Networks 1974;4:65–94.

[6] Berger J, Barkaoui M. A parallel hybrid genetic algorithm for the vehicle routing problem with time windows. Computers & Operations Research 2004;31:2037–53.

[7] Bräysy O, Gendreau M. Vehicle routing problem with time windows, Part II: metaheuristics. Transportation Science 2005;39:119–39.

[8] Chao I-M, Golden BL, Wasil EA. A new heuristic for the multi-depot vehicle routing problem that improves upon best-known solutions. American Journal of Mathematical and Management Sciences 1993;13:371–406.

[9] Chao I-M, Golden BL, Wasil EA. An improved heuristic for the period vehicle routing problem. Networks 1995;26:25–44.

[10] Chao I-M, Golden BL, Wasil EA. A computational study of a new heuristic for the site-dependent vehicle routing problem. INFOR 1999;37:319–36.

[11] Christofides N, Mingozzi A, Toth P. The vehicle routing problem. in: Christofides N, Mingozzi A, Toth P, Sandi L, editors. Combinatorial optimization. Chichester, UK: Wiley; 1979. p. 315–38.

[12] Cordeau J-F, Laporte G. A tabu search algorithm for the site dependent vehicle routing problem with time windows. INFOR 2001;39:292–8.

[13] Cordeau J-F, Gendreau M, Laporte G. A tabu search heuristic for periodic and multi-depot vehicle routing problems. Networks 1997;30:105–19.

[14] Cordeau J-F, Laporte G, Mercier A. A unified tabu search heuristic for vehicle routing problems with time windows. Journal of the Operations Research Society 2001;52:928–36.

[15] Cordeau J-F, Laporte G, Mercier A. Improved tabu search algorithm for the handling of route duration constraints in vehicle routing problems with time windows. Journal of the Operations Research Society 2004;55:542–6.

[16] Crainic TG. Parallel solution methods for vehicle routing problems. in: Golden BL, Raghavan S, Wasil WA, editors. The vehicle routing problem: latest advances and new challenges. Berlin: Springer; 2008. p. 171–98.

[17] Crainic TG, Gendreau M, Potvin J-Y. Parallel tabu search. in: Alba E, editor. Parallel metaheuristics: a new class of algorithms. Chichester, UK: Wiley; 2005. [Chapter 13].

[18] Doerner KF, Hartl RF, Benkner S, Lucka M. Parallel cooperative savings based ant colony optimization - multiple search and decomposition approaches. Parallel Processing Letters 2006;16:351–69.

[19] Drummond LMA, Ochi LS, Vianna DS. An asynchronous parallel metaheuristic for the period vehicle routing problem. Future Generation Computer Systems 2001;17:379–86.

[20] Francis PM, Smilowitz KR, Tzur M. The period vehicle routing problem and its extensions. in: Golden BL, Raghavan S, Wasil EA, editors. The vehicle routing problem: latest advances and new challenges. Berlin: Springer; 2008. p. 73–102.

[21] Gehring H, Homberger J. Parallelization of a two-phase metaheuristic for routing problems with time windows. Journal of Heuristics 2002;8:251–76.

[22] Gendreau M, Tarantilis CD. Solving large-scale vehicle routing problems with time windows: the state-of-the-art. Technical Report 2010-04. CIRRELT, University of Montreal; 2010.

[23] Gendreau M, Hertz A, Laporte G. New insertion and postoptimization procedures for the traveling salesman problem. Operations Research 1992;40:1086–94.

[24] Golden BL, Wasil EA, Kelly J, Chao I-M. The impact of metaheuristics on solving the vehicle routing problem: algorithms, problem sets, and computational results. in: Fleet management and logistics. Boston: Kluwer Academic Publishers; 1998. p. 33–56.

[25] Golden BL, Raghavan S, Wasil EA, editors. The vehicle routing problem: latest advances and new challenges. Operations research computer science interfaces series. Springer; 2008.

[26] Groër C, Golden BL, Wasil EA. A parallel algorithm for the vehicle routing problem, INFORMS Journal on Computing 2011;23:315–30.

[27] Hemmelmayr VC, Doerner KF, Hartl RF. A variable neighborhood search heuristic for periodic routing problems. European Journal of Operational Research 2009;195:791–802.

[28] Jin J, Crainic TG, Lokketangen A. A parallel multi-neighborhood cooperative tabu search for capacitated vehicle routing problems. Technical Report 2010-54. CIRRELT, University of Montreal; 2010.

[29] Laporte G. Fifty years of vehicle routing. Transportation Science 2009;43: 408–16.

[30] Le Bouthillier A, Crainic TG. A cooperative parallel meta-heuristic for the vehicle routing problem with time windows. Computers & Operations Research 2005;32:1685–708.

[31] Lourenço H, Martin O, Stützle T. Iterated local search. in: Handbook of metaheuristics of international series in operations research & management science, vol. 57. Kluwer Academic Publisher; 2003. p. 321–53. [Chapter 11].

[32] Maischberger M. Routing of vehicles. from point to point trips to fleet routing. PhD thesis. Università degli Studi di Firenze; 2010.

[33] Maischberger M, Cordeau J-F. Solving variants of the vehicle routing problem with a simple parallel iterated tabu search. in: Pahl J, Reiners T, Voss S, editors. Proceedings of INOC 2011, international network optimization conference. Hamburg, Germany: Springer-Verlag; 2011. p. 395–400.

[34] Mester D, Bräysy O. Active guided evolution strategies for large-scale vehicle routing problems with time windows. Computers & Operations Research 2005;32:1593–614.

[35] Nag B, Golden BL, Assad AA. Vehicle routing with site dependencies. in: Golden BL, Assad AA, editors. Vehicle routing: methods and studies. Amsterdam: North-Holland; 1988. p. 149–59.

[36] Nagata Y, Bräysy O. Edge assembly-based memetic algorithm for the capacitated vehicle routing problem. Networks 2009;54:205–15.

[37] Nagata Y, Bräysy O, Dullaert W. A penalty-based edge assembly memetic algorithm for the vehicle routing problem with time windows. Computers & Operations Research 2010;37:724–37.

[38] Pirkwieser S, Raidl GR. A variable neighborhood search for the periodic vehicle routing problem with time windows. in: Prodhon C, editor. Proceedings of the 9th EU/meeting on metaheuristics for logistics and vehicle routing; 2008.

[39] Pisinger D, Ropke S. A general heuristic for vehicle routing problems. Computers & Operations Research 2007;34:2403–35.

[40] Polacek M, Hartl RF, Doerner K, Reimann M. A variable neighborhood search for the multi depot vehicle routing problem with time windows. Journal of Heuristics 2004;10:613–27.

[41] Polacek M, Benkner S, Doerner K, Hartl RF. A cooperative and adaptive variable neighborhood search for the multi depot vehicle routing problem with time windows. BuR–Business Research Journal 2008;1:207–18.

[42] Prins C. A GRASP × evolutionary local search hybrid for the vehicle routing problem. in: Bio-inspired algorithms for the vehicle routing problem, vol. 161. Berlin: Springer; 2009. p. 35–53.

[43] Rego C. Node-ejection chains for the vehicle routing problem: sequential and parallel algorithms. Parallel Computing 2001;27:201–22.

[44] Renaud J, Laporte G, Boctor FF. A tabu search heuristic for the multi-depot vehicle routing problem. Computers & Operations Research 1996;23:229–35.

[45] Schulze J, Fahle T. A parallel algorithm for the vehicle routing problem with time window constraints. Annals of Operations Research 1999;86:585–607.

[46] Solomon MM. Algorithms for the vehicle routing and scheduling problems with time window constraints. Operations Research 1987;35:254–65.

[47] Toth P, Vigo D, editors. The vehicle routing problem. Discrete mathematics and applications. Philadelphia, PA: SIAM; 2002.

[48] Vidal T, Crainic TG, Gendreau M, Lahrichi N, Rei W. A hybrid genetic algorithm for multi-depot and periodic vehicle routing problems. Technical Report 2011-05. CIRRELT, University of Montreal; 2011.