Available online at www.sciencedirect.com
SCIENCE DIRECT®
Computers & Operations Research 32 (2005) 1685–1708
computers & operations research
www.elsevier.com/locate/dsw

A cooperative parallel meta-heuristic for the vehicle routing problem with time windows

Alexandre Le Bouthillier[a,*], Teodor Gabriel Crainic[b]

[a] *Département d'informatique et de recherché opérationnelle and Centre de recherche sur les transports, Université de Montréal, C.P. 6128, Succursale Centre-ville, Montréal, QC, Canada H3C 3J7*

[b] *Département de management et technologie, Université du Québec à Montréal, C.P. 8888, Succursale Centre-ville, Montréal, QC, H3C 3P8 and Centre de recherche sur les transports, Université de Montréal, C.P. 6128, Succursale Centre-ville, Montréal, QC, Canada H3C 3J7*

Abstract

This paper presents a parallel cooperative multi-search method for the vehicle routing problem with time windows. It is based on the solution warehouse strategy, in which several search threads cooperate by asynchronously exchanging information on the best solutions identified. The exchanges are performed through a mechanism, called solution warehouse, which holds and manages a pool of solutions. This enforces the asynchronous strategy of information exchanges and ensures the independence of the individual search processes. Each of these independent processes implements a different meta-heuristic, an evolutionary algorithm or a tabu search procedure. No attempt has been made to calibrate the individual procedures or the parallel cooperative method. The results obtained on an extended set of test problems show that the parallel procedure achieves linear accelerations and identifies solutions of comparable quality to those obtained by the best methods in the literature.

Keywords: Vehicle routing with time windows; Parallel meta-heuristics; Cooperative search; Solution warehouse strategy

1. Introduction

Parallel resolution methods offer the possibility of accelerating computations and, as such, these methods constitute an interesting field of research for combinatorial optimization. However, classical parallel approaches, based on functional or data decomposition, do not significantly modify the search trajectories of meta-heuristics. Thus, they cannot improve the quality of the solution, nor

* Corresponding author.
E-mail addresses: alexleb@crt.umontreal.ca (A.L. Bouthillier), theo@crt.umontreal.ca (T.G. Crainic).

0305-0548/$ - see front matter © 2003 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2003.11.023

do they enhance the robustness of the search when faced with different problem instances than those which were originally calibrated and applied. Consequently, in recent years, *multi-search* (or *multi-thread*) *meta-heuristics*, with varying degrees of *cooperation*, have increasingly been used for difficult combinatorial problems and have been shown to both speed up the search and dramatically improve the robustness and the quality of the solutions obtained (e.g., [1–15]).

The *vehicle routing problem with time windows* (*VRPTW*) is one of the central problems in operations research and combinatorial optimization with numerous practical applications [16]. The problem is NP-Hard and thus, not surprisingly, classical and modern heuristics have been extensively studied, with meta-heuristics generally offering the best performances (e.g., [16–30] and references therein).

The parallel cooperative multi-search method based on the *solution warehouse* strategy has been successfully applied to a number of difficult combinatorial problems (e.g., [3,6,31]) but not, at least at the authors' knowledge, to the VRPTW. Or, this parallel approach, based on several search threads that cooperate by asynchronously exchanging information about the best solutions identified so far, offers several advantages: simplicity of design, increased search robustness, the possibility to enhance performance through enhancement of some of the individual search threads. The objective of this paper is to explore the applicability of the solution warehouse-based cooperative multi-search method to VRPTW and determine its competitiveness compared to the current state of the art.

The contributions of this paper are as follows. We present an easy-to-build and efficient parallel solution method for the single-depot version of the VRPTW. The method makes use of the existing search methods without any particular calibration for the parallel cooperative method or the individual procedures (we use the parameter settings proposed for the sequential methods by the original authors). Experiments using both the standard benchmark test problems of Solomon [32] and the more recent one proposed by Homberger and Gehring [10] show that the parallel procedure achieves solutions of quality comparable to that obtained by the best methods of the literature. The proposed cooperative search identifies several new best solutions and finds new best cumulative numbers of vehicles for several problems ranging from 200 to 1000 customers. From a parallel computation point of view, the proposed method is efficient as it displays linear accelerations compared to the sequential methods. The paper also contributes to the general understanding of cooperative parallel methods by analyzing its behavior when applied to a new problem class and by exploring the impact of the quality of the individual search threads on the performance of the parallel search.

The paper is organized as follows. Section 2 briefly reviews the main sequential and parallel methods proposed to solve the VRPTW. Section 3 presents the cooperative search procedure, its general design and the methods selected for cooperation. Section 4 presents the computational results and analyzes them both from the point of view of solving the VRPTW and from that of the performance of the parallel strategy. Conclusions and perspectives are the subject of the last section.

2. The VRPTW and main solution methods

In this paper, we address the single depot VRPTW. One is given a set of customers with known positive demands and specific time intervals when service can be provided. A fleet of homogeneous vehicles of known capacity is available at a given depot to perform this service. The objective is to find a set of closed routes (or tours) that start and end at the depot within its opening hours, such

that the total cost of performing the service is minimized, customers are visited and served during the specified time windows, and vehicles are not overloaded. In the problem version we address, cost is a combination of two factors: the number of vehicles (routes) used and the total distance travelled. A high cost is associated with vehicle utilization to enforce the search towards solutions with reduced number of vehicles. Each customer is visited only once. A vehicle cannot arrive later than the customer's closing time, but is allowed to arrive before the associated opening time, in which case it waits, without explicit penalty, until the customer is ready. Once the service starts, it is carried on until completion, even if the service ending time might be later than the expiration of the time window. The recent review by Cordeau et al. [16], as well as the other reviews indicated in the Introduction, present mathematical formulations, variants, and solution methods for the problem.

Presenting a comprehensive review of solution methods for the VRPTW is beyond the scope of this paper, especially since several reviews have appeared recently. In the following, we briefly review a number of "successful" sequential and parallel solution approaches that obtained best-known results for the classical Solomon problem set and on the 200–1000-customer problems. Methods that are incorporated in the proposed cooperative search are presented in somewhat more detail in Section 3.2.

Efforts are being dedicated to the development of exact algorithms and most 100-customer VRPTW problems with a distance-minimization objective have been solved [33–37]. For most problem instances, heuristics and meta-heuristics are required, however. Most such efforts have addressed the combined objective of minimizing first the number of vehicles and then the total distance.

Tabu search [38–40] has often been used successfully to address vehicle routing problems including the VRPTW. *Taburoute* [41] and the *Unified Tabu Search* [42] stand out by the quality of the results. These two methods are used in the proposed framework and are reviewed in Section 3.2.

Hybrids that combine elements of different methodologies appear very promising. Rousseau et al. (2002) combined constraint programming and variable neighborhood search (VNS) [44–46]. The pruning operators and propagation techniques of constraint programming permit the search of large neighborhoods. This increases the probability of finding a good solution at each iteration and avoids the need for specialized neighborhood descent procedures. The utilization of constraint programming also allows additional constraints to be added easily without fundamentally modifying the methodology.

Bräysy [47] proposed the *reactive variable neighborhood search*, a variation on the VNS based on a four-phase method: initialization, route elimination, total distance minimization by using four new local search heuristics, and a modification of the objective function to allow escaping from local minima. Bräysy et al. [48] proposed a multi-start local search method that proceeds in two phases: (1) production of initial solutions using a fast construction heuristic and an ejection chain-based heuristic to reduce the number of total routes; (2) two improvement heuristics based on CROSS-Exchanges [49] to reduce the total distance. Bräysy and Dullaert [50] proposed the *fast evolutionary meta-heuristic*, an hybrid procedure based on evolutionary principles.

Bent and Van Hentenryck [51] presented a two phase method that combines simulated annealing (SA) and large neighborhood search (LNS). The first phase (SA) utilizes a lexicographic evaluation function to minimize the number of routes and the delay of the entire solution. The SA explores the neighborhood with traditional move operators: 2-exchange, or-exchange, and crossover. It also includes some components often seen in tabu search, such as an aspiration criteria and a bias towards

good solution in the random selection process. The second phase LNS [52] aims to minimize the total travel cost by using a branch and bound algorithm.

Many of the most successful meta-heuristics for the large VRPTW instances are based on some form of parallel computation. Two methodological approaches stand out: the *adaptive memory* strategy [13,49,53,54] and the combination of tabu search and evolutionary algorithms proposed by Gehring and Homberger [11], Homberger and Gehring [10].

Rochat and Taillard [53] proposed what may be considered as the first fully developed adaptive memory-based approach for the VRPTW. It is a cooperative multi-thread method, where the adaptive memory contains tours of good solutions identified by the tabu search process. The tours are ranked according to attribute values, including the objective values of their respective solutions. Each tabu search process then probabilistically selects tours in the memory, constructs an initial solution, improves it, and returns the corresponding tours to the adaptive memory. Despite the fact that it used a rather simple tabu search, this method produced many new best results at publication time. Taillard et al. [49] adapted this method for the vehicle routing problem with soft time windows. Late arrival at a customer is allowed at a penalty. By adjusting the penalty, hard time window cases may also be addressed. The authors significantly refined the tabu search by enriching the neighborhood and the intensification phase and by adding a post-optimization procedure. Badeau et al. [54] report the first "true" parallel implementation of this approach for the VRPTW with soft time windows. Another variant on the adaptive-memory idea may be found in the work of Schulze and Fahle [12]. Here, the pool of partial solutions is distributed among processes to eliminate the need for a "master". The elements of the best solutions found by each thread are broadcast to ensure that each search has still access to all the information when building new solutions. Implemented on a limited number of processors (eight), the method performed well.

Gehring and Homberger [11] introduced a cooperative parallel strategy where concurrent searches are performed with differently configured two-phase meta-heuristics. The first phase tries to minimize the number of vehicles by using an evolutionary meta-heuristic, while the second phase aims to minimize the total travelled distance by means of a tabu search. The evolution strategies are based on the previous work by Homberger and Gehring [10] and Homberger [55] with much emphasis on the minimization of the number of vehicles in the evolution strategy. The parallel meta-heuristic is initiated on different threads with different starting points and values for the time allocated to each search phase. Threads cooperate by exchanging solutions asynchronously through a master process. For the first time, results were also presented on new and larger problem instances, generated similarly to the original Solomon instances, but varying in size from 200 to 1000 customers.

Berger et al. [56] introduced a parallel genetic method where two populations co-evolve, each with a distinct objective. The first population aims to minimize the total distance, while the second focuses on minimizing the temporal constraints violation to generate feasible solutions. New feasible solutions are exchanged between populations. New genetic operators inspired by an insertion heuristic, a large neighborhood search method and an ant colony system are also used.

To conclude, it is worth noticing that while several solution methods have successfully addressed several classes of VRPTW problem instances, none can claim to dominate all the others. On the other hand, in general, the most successful methods combine several methodologies. The method we propose is also based on combining the search efforts of different methods, but it follows a principle not found among the methods reviewed that offers a general framework for algorithmic development for the VRPTW.

3. Cooperative meta-heuristics for the VRPTW

The multi-thread cooperative parallel search procedure we propose for the VRPTW is based on the solution warehouse approach. This section describes the general principles and main components of this solution strategy.

3.1. General design

A number of independent processes (threads) are defined and each executes a complete search through the solution space of the VRPTW (one of them may, as in the present implementation construct and improve solutions based on classical heuristics). When the same meta-heuristic is used by several search threads, the initial solution and particular setting of a number of important search parameters differentiates each search thread from the others. Thus, each thread follows a different *search strategy*. Full implementation details for the strategies used in the present method are given in the following sub-sections. Both an *independent search* method and a *cooperative search* algorithm are implemented. In the former, all searches proceed independently and the best solution is collected at the end. In the latter, it is hoped that by exchanging information among the search threads the efficiency of the global search will increase to yield a higher-quality final solution.

The cooperation aspect of the parallelization scheme is achieved through asynchronous exchanges of information. Information is shared through a *solution warehouse* or *pool of solutions*. In this scheme, whenever a thread desires to send out information, it sends it to the pool. Similarly, when a thread accesses outside information, it reaches out and takes it from the pool. Communications are initiated exclusively by the individual threads, irrespective of their role as senders or receivers of information. No broadcasting is taking place and there is no need for complex mechanisms to select the threads that will receive or send information and to control the cooperation. The solution warehouse is thus an efficient implementation device that allows for a strict asynchronous mode of exchange, with no pre-determined connection pattern, where no process is interrupted by another for communication purposes, but where any thread may access at all times the data previously sent out by any other search thread. The solution warehouse keeps the information in an order appropriate for the exchange mechanism considered.

To fully characterize the cooperation process, one has to specify (i) the information which is to be shared; (ii) the particular methods that make up the cooperative search; (iii) the time when communications occur; (iv) the utilization each thread makes of the imported information [1,4,5,57].

The information exchanged among cooperating procedures has to be meaningful, in the sense that it has to be useful for the decision process of the receiving threads. Information that gives the current status of the global search or, at least, of some other searches is, in this sense, meaningful. In the implementations described in this paper, threads share information about their respective good solutions identified so far. When a thread improves the imported solution or when it identifies a new best solution, it sends out the value of that solution. This scheme is intuitive and simple, and it satisfies the meaningfulness requirement.

The selection of the methods involved in cooperation was mainly oriented toward obtaining: (i) Good quality solutions; (ii) A broad diversity of solutions to facilitate the discovery of promising regions; (iii) The rapid production of intermediate solutions to feed the information exchange mechanisms. Tabu search methods and, recently, evolutionary algorithms have provided some of the best

solutions found so far in the literature. Moreover, evolutionary algorithms may contribute toward increasing the diversity of solutions exchanged among cooperating methods. It was thus decided to include tabu search and evolutionary algorithms in the cooperative framework.

Taburoute [41] and unified tabu search [42] were the two tabu search-based methods considered. Two evolutionary algorithms were also included with simple but different, crossover mechanisms. Construction and improvement heuristics were also included to generate an initial population (pool of solutions) and to perform post-optimization. These methods are described in the following sub-sections, together with the corresponding mechanisms for exchanging information with the solution warehouse and for processing this data.

3.2. The cooperating methods

3.2.1. Taburoute

The Taburoute tabu search method [41] was originally proposed for the vehicle routing problem with capacities and route length restrictions. In this algorithm, the neighborhood of a solution is defined by considering a sequence of adjacent solutions obtained by repeatedly removing a node from its current route and reinserting it into another route that contains one of its p nearest neighbors. Re-insertion is done by means of the *generalized insertion* (*GENI*) procedure. The post-optimization procedure *unstringing–stringing* (*US*) attempts to improve the solution by trying to remove and reinsert each node of a route. The two procedures make up the *GENIUS* heuristic [58,59]. When a node is moved, it is tagged as "tabu", to prevent it from moving back to its original route, for a number of iterations uniformly drawn from a [*inf,sup*] interval. Moving to infeasible solutions is allowed during the course of the algorithm, in order to decrease the likelihood of getting trapped in local optima, but a penalty term in the objective function is increased when solutions obtained in the last 10 iterations are infeasible. The diversification strategy penalizes vertices that have been moved frequently in order to avoid considering similar solutions. A false start is performed at the beginning by performing a local search on a (usually random) solution to find a good initialization.

In the time window versions of Taburoute and GENIUS developed for the cooperative parallel method, insertion is only allowed when no time window constraints are violated. The method will send improving solutions to the central warehouse and require solutions from the warehouse before diversification. This imported solution will be accepted if it is better than the current one. Otherwise, the algorithm will proceed from its own solution.

3.2.2. Unified tabu

The unified tabu search heuristic proposed by Cordeau et al. is one of the best tabu search-based methods for the VRPTW. The unified tabu also solves two important generalizations of the VRPTW: the periodic and the multi-depot vehicle routing problems with time windows. The major benefits of the approach are its speed, simplicity, flexibility, as well as the quality of the produced results.

Unified tabu uses several of the features of Taburoute, namely the same neighborhood structure, GENI insertions, long-term frequency-based penalties. It also differs from Taburoute in a number of ways. The search is applied to a single initial solution, fixed tabu durations are used and no intensification phase is included. The method allows intermediate infeasible solutions (similarly to Taburoute), but modifies its search parameters at each iteration according to whether the previous solution was feasible or not with respect to capacity or route duration. The tabu mechanism operates

on an attribute set $B(x)=(i,k)$: customer v_i is visited by vehicle k in solution x. Neighbor solutions are obtained by removing (i,k) from $B(x)$ and replacing it with (i,k), where $k' \neq k$. Attribute (i,k) is then declared tabu for a number of iterations, and the aspiration criterion is defined relative to attribute (i,k). At the end of n iterations, a GENIUS-based [59] post-optimization is applied to each route of the best feasible solution found.

To integrate this method into a cooperative mechanism, one has to create moments when communications are possible (this step is required since neither intensification nor diversification are in the original design). We decided to stop the method after w iterations and import a solution from the central warehouse. This solution is then used to re-start the search. On the other hand, improving feasible solutions are sent to the warehouse.

3.2.3. Evolutionary algorithms

Two evolutionary algorithms also participate in the cooperative global search. Both methods use the solution warehouse as population. Two parent solutions are chosen randomly and a probabilistic mutation is performed on a copy of each parent (for each arc, a 1% probability to replace it with a randomly chosen one). The algorithms differ by the crossover operator used, either an *order* crossover (OX) or an *edge recombination* (ER) crossover. When required, a repair procedure restores the feasibility of the solution by re-ordering or re-routing nodes reached after their closing time windows. The offspring is sent to the solution warehouse.

The OX operator is based on a path representation on each route and attempts to preserve the relative order of the customers in the second parent. It copies a sub-chain of the first parent to the offspring; the remaining positions are filled by customers who are not yet in the offspring, in the order that they appear in the second parent. The ER crossover operator aims to preserve the maximum number of arcs from the parent and to introduce a minimum number of new arcs. An adjacency table is constructed to represent, for each node, the nodes that are adjacent to it in each parent. The construction of the offspring is performed by selecting paths with a minimum number of arcs between two adjacent nodes that are not yet in the offspring. A backtrack scheme is performed when nodes are left with no active arcs.

3.2.4. Post-optimization

The *ejection chain* principle is mainly used to explore large and complex neighborhoods in tabu search and correspond in general to chained-movements of solution elements among several solutions. Two types of ejection chains are used in the VRP literature [43,60–64]. The first type is called a multi-node exchange process and involves only one route at a time. All node exchanges are performed within the same route and the last node of the chain takes the position of the first node. The second type is called a multi-node insert process and involves several routes with exchanges being performed on two routes at a time. A node is ejected from the first route and inserted into a second one, from which another node is eventually ejected to be inserted into another route, and so on and so forth until a certain number of exchanges is attained.

The proposed ejection search procedure is based on the multi-node process and tries to empty a route, called the origin route, by sequentially ejecting its nodes and inserting them into other routes. An ejected node does not, however, replace an existing node. Rather, it is simply inserted in a destination route between two existing nodes. If the insertion of this node causes the receiving route

to become infeasible, other nodes of this route are ejected until feasibility is regained. This process is repeated for the subsequent destination routes until the maximum number of exchanges is attained, at which point the process backtracks to choose alternative destination routes if the route-feasibility criterion is still not met.

The origin route is chosen according to the shortest tour in distance, the tour with the smallest number of customer, the tour with the highest residual capacity, or the tour with the smallest sum of distance to all the distance of each customer to its closest neighbor. The nodes in the chosen origin tour are ejected one at a time to maximize the total saving in distance (the saving in the origin tour minus the added detour in the destination tour) and the slack in the available capacity of the destination route.

The proposed ejection chain search and classical 2-opt, 3-opt, or-opt heuristics [65,66] make up the post-optimization procedure. They are executed one after the other (ejection search followed by 2-opt, 3-opt and finally or-opt) to improve the solutions received by the solution warehouse as described in Section 3.3.

3.2.5. Initial solutions

A number of classical heuristics have also been included to generate initial solutions and help diversify the pool. Four simple construction heuristics from Bentley [67] were adapted for the VRPTW: (i) least successor; (ii) double-ended nearest neighbor, which starts from a seed node for each tour and sequentially adds the nearest feasible node on each side of the current tour until the tour is full; (iii) multiple fragments, which sequentially adds the shortest arcs to form feasible tours; and (iv) shortest arcs hybridizing, which is similar to the previous heuristic but adds arcs probabilistically according to their length (shorter length arcs have a greater probability of being chosen). When required, the repair procedure is applied to the resulting solutions.

3.3. The cooperation mechanism

The solution warehouse is the core of the cooperation mechanism. It keeps good, feasible solutions and is dynamically updated by the independent search processes. The pool of solutions forms therefore an elite population from which the independent procedures will require solutions at various stages of execution. Path representation is used to keep solutions in solution warehouse. These solutions are ordered first by the number of vehicles (in increasing order) and then according to the weighted sum of the corresponding travel time, total distance, waiting time and residual time for all customers (see Section 4).

The solution warehouse is divided into two sub-populations: *in-training* and *adult*. All solutions received from the independent processes are placed in the in-training part. The post-optimization procedure is then applied and the resulting solution is moved to the adult sub-population. Duplicate solutions are eliminated. All requests for solutions initiated by the independent processes are satisfied by the adult sub-population. Solutions are selected randomly according to probabilities biased toward the best based on the same function used to order solutions in the solution warehouse.

In the collaboration diagram depicted in Fig. 1, each independent method (the construction heuristics are grouped together) is encapsulated in an MPI process. An MPI process contains the particular algorithm, a communication mechanism for exchanging solutions, and a control mechanism for

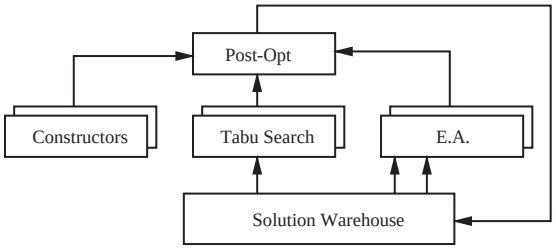

Fig. 1. Collaboration diagram.

parameter setting, initialization, termination, and solution acceptance. The cooperative method works through the solution warehouse, which is the central point of communication. It provides starting and diversification solutions to tabu search procedures and parents to the evolutionary algorithms (EA). The population size in the solution warehouse is fixed to a value related to the problem size (10*size) and the worst results are eliminated as needed. No direct communications take place between processes thus enforcing their independence and the asynchronous mode of exchange. This scheme makes the cooperation design simpler and, eventually, allows easy modification of the parallel system by adding new methods or dropping inefficient ones.

4. Computational experiments

The experimental study had a dual objective: (i) to compare the proposed warehouse cooperative parallel meta-heuristic to the best performing methods proposed in the literature for the VRPTW and, thus, to validate our claim that the proposed method offers comparable performance in terms of both solution quality and computation effort; (ii) to evaluate the role of cooperation and the contribution of the individual methods involved in the cooperation, in particular the impact of the quality of individual methods on the performance of the global search. The ultimate objective of the cooperative search is to show that by combining off-the-shelf methods, robustness can be attained with linear speedup and no calibration.

We start by indicating the problem data sets used, the settings of the individual and global searches, and the computing environment. Results are then presented, compared, and analyzed. Finally, we present a brief analysis of the behavior of the parallel cooperative search.

4.1. Experimental setting

A different search method is run on each of the four processors (two for tabu search and two for evolutionary algorithm). The solution warehouse, the post-optimization, and the construction methods are run on another processor for a total of five processors in this study.

Experimentations were performed on two versions of the cooperative search to study the influence of the individual methods on the global performance and the interaction between the methods when used in cooperation. The first experimentation (identified **LC02** in the following) consists of two Taburoute methods (**TS1** and **TS2**) and the two evolutionary algorithms (**OX** and **ER**). In the second

experiment (**LC03**), the unified tabu (**UT1**) replaces the second Taburoute method. The evolutionary algorithms do not change.

Two sets of parameters were defined for the Taburoute search threads based on those indicated in the original paper for the VRP [41]. Both sets used a tabu tag length of [5,10] iterations. Each set used a different seed as well as different p-neighborhood dimensions (15% and 20% of the nodes are evaluated in the first and second tabu search, respectively), initial solutions (the best solution out of 15 and 20 false starts), and penalty factors for frequently moved vertices (1 and 0.5). The parameters for the value function presented in the initial article by Cordeau, Laporte, and Mercier ([42]: $\alpha = 1,\ \beta = 1,\ \gamma = 1$) were used for the unified tabu search thread. Finally, an arc mutation probability of 1/100 was used on temporary copies of the parents for the crossovers used by the evolutionary algorithms.

Tests have been carried out on the standard set of test problems proposed by Solomon [32] and used by all authors. The set contains 56 problems of 100 customers each. We also used the extended set produced by Homberger and Gehring [10] with 300 problem instances that vary from 200 to 1000 customers.

The Solomon standard and extended problems are divided into six categories, named C1, C2, R1, R2, RC1, and RC2. For all problem instances, customers are distributed in a [0,100] square unit. The customers in sets C are clustered together, while those in sets R are distributed randomly. Problems in sets RC combine the two characteristics. Time windows at the depot are relatively small for problems of type 1, to allow less customers to be served by each route; time windows are larger for problems of type 2. The service time is of 10 units by customer for problems of type R and RC, and of 90 units for class C.

Solutions in the adult sub-population of the solution warehouse are sorted, first by the number of vehicles, second by a weighted sum, $C(p)$, of attributes: the total time required to serve all customers, the associated total distance and total waiting time at customers, and the sum of the slack left in each time window:

$$C(p) = W_1 * totalTime + W_2 * totalDistance + W_3 * totalWait + W_4 * totalSlack,$$

W_1–W_4 were set to 1 in all the reported experiments. This measure combined with the number of vehicles gives us an overall idea of the solution quality (*totalTime* and *totalDistance*) and flexibility (*totalWait* and *totalSlack*). The two last measures indicate how much slack there exists in the solution and how easily feasible neighboring solutions may be explored.

Runs of 12 min wall clock time are performed by the cooperative meta-heuristic for each of the 100 city problems, while 50 min are given to the sequential runs of the independent algorithms. Longer running times, equal to those reported by Homberger and Gehring [10] were allowed for the larger problem instances. These times go up to 50 min wall clock time for the 1000 city problem.

Clusters of five Pentium III computers were used for both versions of the cooperative search. The first version (**LC02**) was run on 500 MHz CPUs with 128 MB of RAM under Windows. The second version (**LC03**) was run on 850 MHz CPUs with 512 MB of RAM under Linux. Communications were managed by Voyager and MPICH for version **LC02** and **LC03**, respectively. Computations of distances have been carried out in double precision. The second implementation (**LC03**) is machine independent and can be run with MPICH on Unix, Windows, or Linux.

Table 1
Internal comparisons

Class	UT1	TS1	TS2	ER	OX
R1	**12.08**	12.17	12.17	12.25	12.49
	1210.14	1211.10	1215.45	1211.38	1218.79
C1	**10.00**	**10.00**	**10.00**	**10.00**	**10.00**
	828.38	828.38	828.38	828.38	828.38
RC1	11.50	**11.50**	11.67	11.88	11.74
	1389.78	1388.24	1384.65	1394.28	1380.29
R2	**2.73**	2.94	2.96	2.94	3.02
	969.57	955.18	958.95	966.78	954.32
C2	**3.00**	**3.00**	**3.00**	**3.00**	**3.00**
	589.86	**589.86**	**589.86**	**589.86**	**589.86**
RC2	**3.25**	3.25	3.25	3.38	3.31
	1134.52	1145.29	1147.00	1118.63	1144.13

4.2. Computational results

Average results for each set of problem instances are given in this section. Complete results are presented in the Appendix for **LC03** and in Le Bouthillier and Crainic [68] for **LC02**. Tables display average values for the total number of vehicles and the total distance (just below the number of vehicles) for each problem class and, globally, for each meta-heuristic. Best results are indicated in bold characters. Tables 3–9 also display the cumulative number of vehicles (CNV) and cumulative total distance (CTD) for each procedure. Unless otherwise indicated, results were obtained on the classical Solomon data set with 100 customers.

First, a comparison is made between the sequential methods involved in the cooperative search. Results are displayed in Table 1. The five meta-heuristics equally solve the problems of type C that have clustered customers. This is not surprising since it is easy to assign distant customers to distinct routes which belong to the optimal solution. Problems with randomly positioned customers and large time windows are more difficult to solve. There is no clear winner, although the unified tabu search slightly outperforms our version of Taburoute and the evolutionary algorithms.

Table 2 displays results for the cooperative (**LC02 and LC03**) and independent (**LCIND**) parallel methods. In the independent runs, there is no communication between the individual searches and the solution warehouse serves only to report the best solution of all the four methods (UT1, TS1, ER, OX). It is worth recalling that each independent method runs for 50 min and the parallel method runs for 12 min for each five processors (one for each method and one for the solution warehouse) and that consequently there is speedup superior to three when comparing the total clock time of the two parallel methods.

The results indicate that cooperation increases the robustness of the search, without an increase in computation cost. The cooperative search is able to produce better results than the best independent method. Furthermore, even accounting for a significant speedup, the cumulative number of vehicles is reduced by two for **LC02** when compared to the independent method and by an additional two vehicles for **LC03**. With only one fifth of the time available when running in sequential mode,

Table 2
Cooperative versus independent methods

Class	LCIND	LC02	LC03
R1	12.08	12.17	**12.08**
	1210.14	1209.27	1209.19
C1	**10.00**	**10.00**	**10.00**
	828.38	828.38	828.38
RC1	11.50	11.50	**11.50**
	1388.24	1389.22	1386.38
R2	273	2.82	**2.73**
	969.57	965.91	960.95
C2	**3.00**	**3.00**	**3.00**
	589.86	589.86	589.86
RC2	3.25	3.25	**3.25**
	1134.52	1143.70	1133.30

individual methods benefit from the cooperation and are thus able to obtain the same or better solution quality than in sequential (with only one exception out of 356 problems—R205—with 0.49% more distance for **LC03**). On average the cooperative method shows better total distance for each class of problems compared to the independent approach.

To assess the performance of the cooperative parallel meta-heuristic, we compared the results of the two variants (**LC02** and **LC03**) to those of the best methods (published or not) for the VRPTW, reviewed in Section 2: Rochat and Taillard ([53]—**RT**), Homberger and Gehring ([10]—**GH99**), Taillard et al. ([69]—**TB**), Rousseau et al. ([43]—**RGP**), Cordeau et al. ([42]—**CLM**), Bräysy, Hasle, and Dullaert ([48]—**BHD**), Homberger ([55]—**H99**), Bräysy and Dullaert ([50]—**EA2**), Gehring and Homberger ([11]—**GH01**), Bräysy ([47]—**B01**), and Bent and Henten-ryck ([51]—**BVH**). Table 3 displays the results on the standard 100-customer problem sets (the only set addressed by all). The table also presents the CNV, the CTD for each method, and the experiment settings (CPU time, number of runs and minutes for each run).

The performance of the two versions of the proposed cooperative are competitive even if no particular calibration has been performed. The solution quality is comparable to the best of the sequential and parallel methods (**LC03** is in 7th position with regard to the CNV), while computation times are reasonable, even when we take into account the five processors used.

We also compared our results to those in the literature on the extended data sets provided by Homberger and Gehring [10]. Table 9 shows the total CNV and CTD for all 300 problems of the extended set. Average results by problem size and class can be found in Tables 4–8. Detailed results by problem and class can be found in Tables 12–16 in the Appendix. For each problem size and class, each table displays the CNV, the CTD, and the CPU time for each method.

Results are again very satisfying. The quality of results and the computation effort are on a par with the best methods available. Five new best solutions are found:

R1-2-3 18 3164,41,
RC2-2-6 4 3086,76,
C1-4-7 39 7668,33,

Table 3
Comparison of average results on 100-customer problems

Author	R1	R2	C1	C2	RC1	RC2	CNV/CTD	Experiment
RT	12.25	2.91	10.00	3.00	11.88	3.38	415	SG. 100 MHz
	1208.50	961.72	828.38	589.86	1377.39	1117.44	57231	1 run, 92.2 min
GH99	12.42	2.82	10.00	3.00	11.88	3.25	415	4 P200 MHz,
	1198	947	829	590	1356	1144	56946	1 run, 5 min
TB	12.17	2.82	10.00	3.00	11.50	3.38	410	Sun 10
	1209.27	980.27	828.385	589.86	1388.24	1140.42	57521	1 run, 248 min
LC02	12.17	2.82	10.00	3.00	11.50	3.25	409	5 P500 MHz
	1209.27	965.91	828.38	589.86	1 389.22	1143.70	57573	1 run, 12 min
RGP	12.00	2.82	10.00	3.00	11.50	3.38	408	Sun U10
	1203.38	950.77	828.38	589.86	1 366.45	1093.46	56752	n/a
CLM	12.08	2.73	10.00	3.00	11.50	3.25	407	n/a
	1210.14	969.57	828.38	589.86	1389.78	1134.52	57555	Sun U2 300 MHz
LC03	12.08	2.73	10.00	3.00	**11.50**	3.25	407	5xP850 MHz,
	1209.19	963.62	828.38	589.86	1389.22	1143.70	57412	1 run, 12 min
BHD	12.00	2.73	10.00	3.00	11.50	3.25	407	AMD 700 MHz,
	1239.77	1016.86	832.06	590.68	1417.79	1199.95	59210	1 run, 2.6 min
H99	11.92	2.73	10.00	3.00	11.63	3.25	406	P200 MHz,
	1228.06	969.95	828.38	589.86	1 392.57	1144.43	57876	10 runs, 13 min
EA2	12.00	2.73	10.00	3.00	11.50	3.25	406	AMD 700 MHz
	1220.14	977.57	828.38	589.86	1 397.44	1140.06	57870	3 runs, 9.1 min
GH01	12.00	2.73	10.00	3.00	11.50	3.25	406	4 P400 MHz,
	1217.57	961.29	828.63	590.33	1395.13	1139.37	57641	5 runs, 13.5 min
B01	11.92	2.73	10.00	3.00	11.50	3.25	405	P200 MHz,
	1222.12	975.12	828.38	589.86	1389.58	1128.39	57710	1 run, 82.5 min
BVH	12.18	2.73	10.00	3.00	11.50	3.25	405	Sun U10 440 MHz
	1231.08	954.18	828.38	589.86	1 384.17	1124.47	57 272	5 run, 120 min

RC2-8-6 15 19744,73, and
RC2-10-5 18 29352,08.

New best-known CNV and CTD measures for the 200-customer problem set and six new best averages for RC1-100, R2-200, RC2-200, RC2-400, RC2-600, and RC2-1000 customers problem class are also reported in bold in Tables 4–8. These results are a clear indicator of the robustness of the proposed cooperative search, which also seems to perform better than the other on the RC2 problem class.

4.3. Solution warehouse evolution

The results presented in the preceding sub-section emphasize the interest of cooperation. We now turn to the issue of the behavior of the cooperation.

Table 10 displays the distribution of the number of solutions sent to the solution warehouse by each individual method for problem instance RC204 for methods *LC02* and *LC03*, respectively. For the first version, the Taburoute methods identify the largest number of good solutions, with the first

Table 4
Average results 200-customers

Class	GH99	GH01	BVH	GES	LC02	BHD	LC03
R1	18.20	18.20	18.20	**18.20**	18.40	18.20	18.20
	3705	3855.03	3677.96	3618.68	3716.36	3718.30	3676.95
R2	4.00	4.00	4.10	4.00	4.00	4.00	**4.00**
	3055	3032.49	3023.62	2942.92	2986.01	3014.28	2986.01
C1	18.90	18.90	18.90	**18.80**	19.30	18.90	18.90
	2782	2842.08	2726.63	2717.21	2757.36	2749.83	2743.66
C2	6.00	6.00	6.00	**6.00**	6.00	6.00	6.00
	1846	1856.99	1860.17	1833.57	1837.02	1842.65	1836.10
RC1	18.00	18.10	**18.00**	18.00	18.00	18.00	18.00
	3555	3674.91	3279.99	3221.34	3449.71	3329.62	3449.71
RC2	4.30	4.40	4.50	4.40	4.30	4.40	**4.30**
	2675	2671.34	2603.08	2519.79	2627.31	2585.89	2613.75
CNV	694	696	697	694	700	695	**694**
CTD	176 180	179 328	171 715	168 573	173 738	172 406	173 061
Computer	P200	P400	Sun U10	P2000	P500	AMD700	P850
CPU Min	4 × 10	4 × 2.1	n/a	1 × 8	5 × 10	2.4	5 × 10

Table 5
Average results 400-customers

Class	GH99	GH01	BVH	GES	LC02	BHD	LC03
R1	36.40	36.40	36.40	**36.30**	36.70	36.40	36.50
	8925	9478.22	8713.37	8530.03.05	8851.55	8692.17	8839.28
R2	8.00	8.00	8.00	**8.00**	8.20	8.00	8.00
	6502	6650.28	6959.75	6209.94	6249.05	6382.63	6437.68
C1	38.00	38.00	38.00	**37.90**	38.00	37.90	37.90
	7584	7855.82	7220.96	7148.27	7651.18	7230.48	7447.09
C2	12.00	12.00	12.00	**12.00**	12.30	12.00	12.00
	3935	3940.19	4154.40	3840.85	3890.24	3894.48	3940.87
RC1	36.10	36.10	36.10	**36.00**	36.00	36.00	36.00
	8763	9294.99	8330.98	8066.44	8704.82	8305.55	8652.01
RC2	8.60	8.80	8.90	8.80	8.70	8.90	**8.60**
	5518	5629.43	5631.70	5243.06	5447.28	5407.87	5511.22
CNV	1390	1392	1393	**1389**	1398	1391	1390
CTD	412270	428 489	410 112	390 386	409 741	399 132	408 281
Computer	P200	P400	Sun U10	P2000	P500	AMD700	P850
CPU Min	4 × 20	4 × 7.1	n/a	1 × 17	5 × 20	7.9	5 × 20

Table 6
Average results 600-customers

Class	GH99	GH01	BVH	GES	LC02	BHD	LC03
R1	54.50	54.50	55.00	**54.50**	54.80	54.50	54.80
	20854	21864.47	19308.62	18358.68	20624.85	19081.18	19869.82
R2	11.00	11.00	11.00	**11.00**	11.20	11.00	11.20
	13335	13656.15	14855.43	12703.52	13097.70	13054.83	13093.97
C1	57.90	**57.70**	57.80	57.80	58.00	57.80	57.90
	14792	14817.25	14357.11	14003.09	14523.65	14165.90	14205.58
C2	17.90	**17.80**	17.80	17.80	18.00	18.00	17.90
	7787	7889.96	8259.04	7455.83	7704.85	7528.73	7743.92
RC1	55.10	55.00	55.10	**55.00**	55.20	55.00	55.20
	18411	19114.02	17035.91	16418.63	18012.52	16994.22	17678.13
RC2	11.80	11.90	12.40	12.10	11.80	12.10	**11.80**
	11522	11670.29	11987.89	10677.46	11189.75	11212.36	11034.71
CNV	2082	**2079**	2091	2082	2090	2084	2088
CTD	867010	890121	808646	796172	851553	820372	836261
Computer	P200	P400	Sun U10	P2000	P500	AMD700	P850
CPU Min	4 × 30	4 × 12.9	n/a	1 × 40	5 × 30	16.2	5 × 30

Table 7
Average results 800-customers

Class	GH99	GH01	BVH	GES	LC02	BHD	LC03
R1	72.80	72.80	72.70	**72.80**	73.10	72.80	73.10
	34586	34653.88	33337.91	31918.47	33552.40	32748.06	33552.40
R2	15.00	15.00	15.00	**15.00**	15.20	15.00	15.00
	21697	21672.85	24554.63	20295.28	20158.94	21170.15	21157.56
C1	76.70	76.10	**76.10**	76.20	76.30	76.30	76.30
	26528	26936.68	25391.67	25132.27	26278.74	25170.88	25668.82
C2	24.00	23.70	24.40	**23.70**	24.20	24.20	24.10
	12451	11847.92	14253.83	11352.29	12056.74	11648.92	11985.11
RC1	72.40	72.30	73.00	73.00	**72.30**	73.00	72.30
	38509	40532.35	30500.15	30731.07	37722.62	30005.95	37722.62
RC2	16.10	16.10	16.60	15.80	**15.80**	16.30	15.80
	17741	17941.23	18940.84	16729.18	17464.30	17686.65	17441.60
CNV	2770	**2760**	2778	2768	2769	2776	2766
CTD	1515 120	1535 849	1469 790	1373 662	1472337	1384 306	1475 281
Computer	P200	P400	Sun U10	P1500	P500	AMD700	P850
CPU Min	4 × 40	4 × 23.2	n/a	367.1	5 × 40	26.2	5 × 40

Table 8
Average results 1000-customers

Class	GH99	GH01	BVH	GES	LC02	BHD	LC03
R1	**91.90**	91.90	92.80	92.10	92.20	92.10	92.20
	57186	58 069.61	51193.47	49281.48	55280.49	50025.64	55176.95
R2	19.00	19.00	19.00	**19.00**	19.20	19.00	19.20
	31930	31873.62	36736.91	29860.32	30951.07	31458.23	30919.77
C1	96.00	95.40	95.10	**95.10**	95.30	95.80	95.30
	43273	43392.59	42505.35	41569.67	44061.43	42086.77	43283.92
C2	30.20	29.70	30.30	**29.70**	30.10	30.60	29.90
	17570	17574.72	18546.13	16639.54	17365.26	17035.88	17443.50
RC1	90.00	90.10	90.20	**90.00**	90.00	90.00	90.00
	50668	50950.14	48634.15	45396.41	49711.36	46736.92	49711.36
RC2	19.00	18.50	19.40	18.70	18.50	19.00	**18.50**
	27012	27175.98	29079.78	25063.51	26309.10	25994.12	26001.11
CNV	3461	**3446**	3468	3446	3453	3465	3451
CTD	2276 390	2290 367	2266 959	2 078 110	2236 787	2133 376	2225 366
Computer	P200	P400	Sun U10	P2000	P500	AMD700	P850
CPU Min	4 × 50	4 × 30.1	n/a	1 × 600	5 × 50	39.6	5 × 50

Table 9
Cumulative results 200–1000-customers

	GH99	GH01	BVH	GES	LC02	BHD	LC03
CNV	10397	**10373**	10427	10383	10410	10411	10389
CTD	5246970	5324154	5176616	4851217	5144136	4909592	5 118250

Table 10
Number of best solutions found by methods LC02 and LC03 for RC204

Method	LC02	LC03
Construction	24	0
TB1	44	11
TB2	21	
UT		24
ER	9	4
OX	3	2

thread, which has the set of parameters recommended by the authors, contributing the largest number. The unified tabu search contributes the largest number of best solutions in the second version, the Taburoute thread taking second place. It is noteworthy that for both versions, all processes contribute solutions to the warehouse and, as the comparison to the independent search method shows

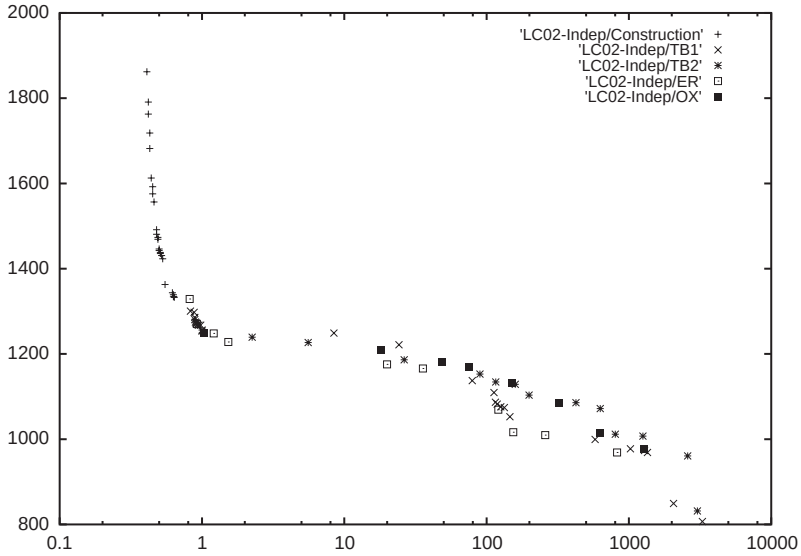

Fig. 2. Evolution of solutions for RC204 by independent search.

(Table 2), these solutions are important in order for the performance of the cooperative method in terms of solution quality.

This insight is also supported by the evolution of the cooperative meta-heuristic, as illustrated by the evolution of the best solution kept in the solution warehouse, and its comparison to the evolution of the independent search method. Figs. 2–4 display this behavior for a typical problem (RC204) for the Independent, **LC02**, and **LC03** search methods, respectively. In these figures, the best value of the objective function is plotted in time (in seconds) and the individual method that generated each solution is identified. The best sequential method value indicated in Figs. 3 and 4 corresponds to the last solution identified by thread *TB1* in Fig. 2. These figures clearly indicate that, after a warm-up period, all methods contribute to the improvement of the solution. One also notices that the cooperative search finds better solutions faster than the independent search, and that it does so in the early stages of the resolution. This indicates that cooperation impacts the search early on in the process. Even in the case were a method like the unified tabu search finds better solutions more often, the other methods have a positive influence and can lead to better solutions as shown in Fig. 4: while the unified tabu search was not able to find this best solution when run independently, it yielded better solutions at the beginning of the cooperative search, which significantly reduced the warm-up period. Figs. 3 and 4 also illustrate how cooperation may allow to get out of a local minimum and continue the search toward a significantly improved solution.

On a general note, one unified tabu search cooperating with one Taburoute were able to perform better than two Taburoute version, mainly due to the quality of solution produced by the unified tabu search. This indicates that the quality of the individual methods in cooperation has an impact on the global performance of the method. However, as illustrated by the performance of *LC02*, the cooperative search will produce good results for as long as the individual threads produce reasonably good solutions.

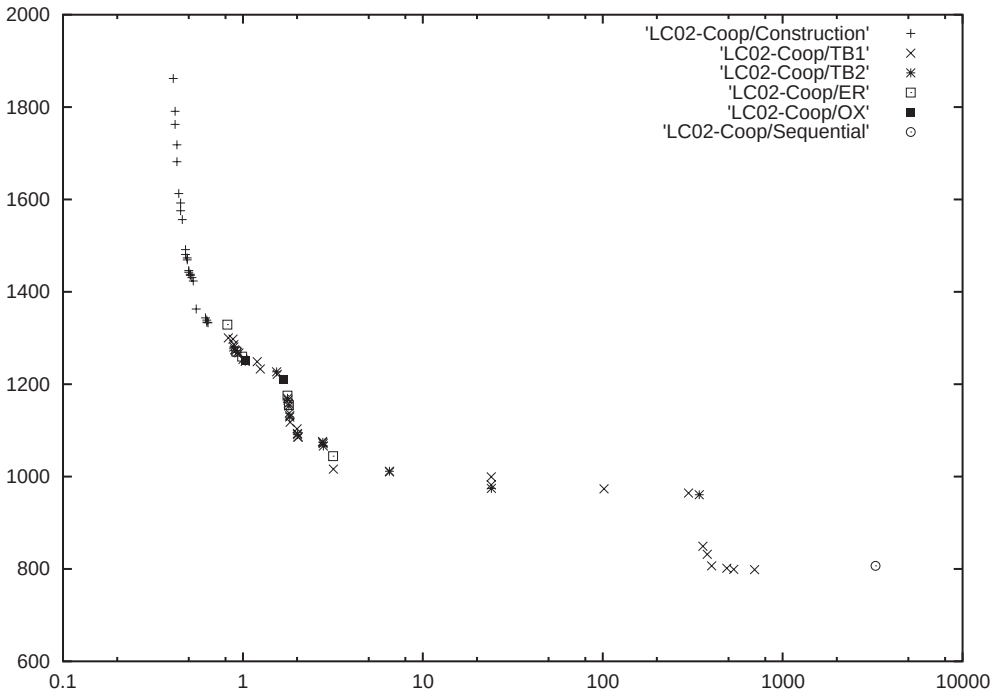

Fig. 3. Evolution of solutions for RC204 by LC02.

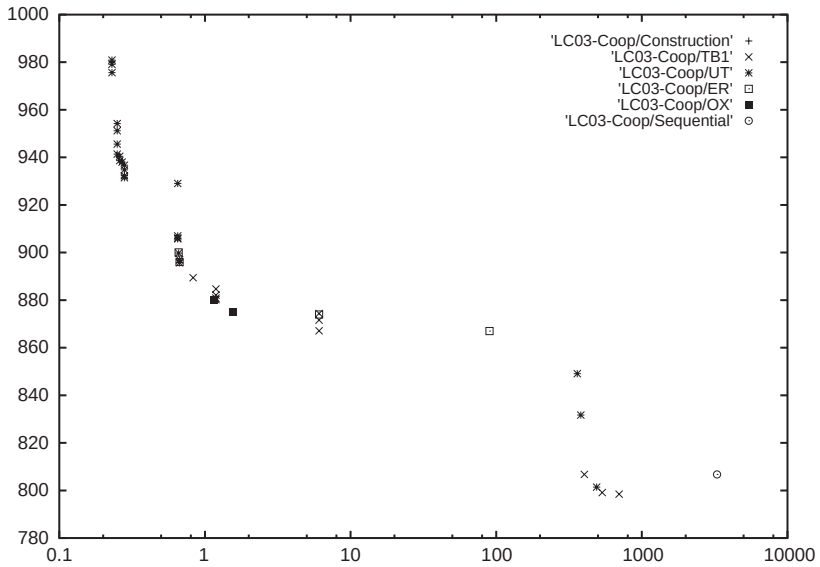

Fig. 4. Evolution of solutions for RC204 by LC03.

The results also indicate that cooperation changes the behavior of each method by inducing new information. This property strongly suggests that cooperative search should be considered a method by itself and be studied as such. With respect to the VRPTW in particular, the solution warehouse cooperation between the tabu search and the evolutionary mechanisms allows excellent quality solutions in very reasonable computing times.

5. Conclusions

We have presented a new parallel cooperative meta-heuristic for the VRPTW. The proposed meta-heuristic displays very good performance in terms of solution quality, computational effort, and robustness over the broad range of problem instances. Linear speedups have been attained and results are comparable to the best in the literature. The cooperative framework is simple to implement and expand, and shows great promise in addressing difficult combinatorial problems, the VRPTW in particular.

Acknowledgements

Many thanks are given to Olli Bräysy and Louis-Martin Rousseau for interesting discussions and comments. We are grateful to Gilbert Laporte, Jean-Francois Cordeau, and Anne Mercier (for the Unified Tabu Search) and to Michel Gendreau, Alain Hertz, and Gilbert Laporte (for Taburoute) for their very precious collaboration. Special thanks are transmitted to Prof. Dr. B. Monien and many of our colleagues at the Paderborn center for parallel computing for the use of their PC^2 cluster on which some key tests were performed in this study for the second version (LC03) of our cooperative search.

Funding for this project has been provided by the Natural Sciences and Engineering Council of Canada, through its Research Grant programs, and by the Fonds F.C.A.R. of the Province of Québec.

Table 11
Detail results LC03, 100 customers

#	R1	R2	C1	C2	RC1	RC2
01	19.00/1650.79	4.00/1253.26	10.00/828.94	3.00/591.56	14.00/1696.94	4.00/1406.94
02	17.00/1487.60	3.00/1197.66	10.00/828.94	3.00/591.56	12.00/1554.75	3.00/1407.52
03	13.00/1294.24	3.00/942.64	10.00/828.06	3.00/591.17	11.00/1262.98	3.00/1073.39
04	10.00/982.72	2.00/855.21	10.00/824.78	3.00/590.60	10.00/1135.48	3.00/798.46
05	14.00/1377.11	3.00/1013.47	10.00/828.94	3.00/588.88	13.00/1643.38	4.00/1326.83
06	12.00/1253.23	3.00/932.47	10.00/828.94	3.00/588.49	11.00/1427.13	3.00/1158.81
07	10.00/1113.69	2.00/837.20	10.00/828.94	3.00/588.29	11.00/1230.54	3.00/1062.05
08	9.00/964.38	2.00/734.85	10.00/828.94	3.00/588.32	10.00/1139.82	3.00/832.36
09	11.00/1199.63	3.00/916.47	10.00/828.94			
10	10.00/1125.04	3.00/963.37				
11	10.00/1104.83	2.00/923.80				
12	10.00/957.04					

Appendix A

Appendix A contains detailed results obtained by the **LC03** cooperative search meta-heuristic. Table 11 displays detailed results for the standard 100-customer problem instances. Tables 12–16 display detailed results for the extended set of problems, from 200 to 1000 customers with best-known results outlined in bold face (as of 2003/08/31 from the Sintef web site: http://www.sintef.no/static/am/opti/projects/top/).

Table 12
Detail results LC03, 200 customers

#	R1	R2	C1	C2	RC1	RC2
01	20/4 829.21	4/4 502.17	20/2 708.61	6/1 931.44	18/3 946.11	6/3 136.85
02	18/4 362.31	4/3 703.86	18/2 972.24	6/1 863.16	18/3 468.85	5/2 827.45
03	**18/3 164.41**	4/2 976.64	18/2 775.40	6/1 775.11	18/3 203.68	4/2 637.41
04	18/3 141.02	4/2 034.20	18/2 672.45	6/1 742.00	18/2 963.00	4/2 084.89
05	18/4 213.56	4/3 417.98	20/2 702.05	6/1 879.01	18/3 902.58	4/3 087.98
06	18/3 659.9	4/2 959.14	20/2 701.04	6/1 857.75	18/3 632.80	**4/3 086.76**
07	18/3 175.75	4/2 532.95	20/2 723.11	6/1 849.46	18/3 456.98	4/2 550.56
08	18/2 999.35	4/1 872.11	19/2 778.82	6/1 824.43	18/3 277.21	4/2341.34
09	18/3 889.96	4/3 115.60	18/2 739.33	6/1 832.04	18/3 385.56	4/2291.43
10	18/3 334.02	4/2 745.45	18/2 663.50	6/1 806.60	18/3 260.31	4/2 092.74

Table 13
Detail results LC03, 400 customers

#	R1	R2	C1	C2	RC1	RC2
01	40/10 709.18	8/9 403.89	40/7 152.06	12/4 116.14	36/9 438.51	11/7 019.89
02	36/9 266.18	8/7 811.05	37/7 517.45	12/4 048.07	36/8 587.76	10/6 579.41
03	36/8 551.10	8/6 474.06	36/8 035.08	12/4 179.01	36/8 032.76	8/6 579.41
04	37/7 421.58	8/4 338.61	36/7 340.59	12/3 929.24	36/7 460.19	8/3 882.74
05	36/10 139.75	8/7 633.26	40/7 152.06	12/3 941.05	36/9 372.21	9/6 292.63
06	36/8 881.72	8/6 182.17	40/7 153.46	12/3 877.65	36/9 285.81	8/6 054.46
07	36/8 023.67	8/5 378.68	**39/7 668.33**	12/3 894.13	36/8 867.02	8/5 727.01
08	36/7 464.09	8/4 349.54	38/7 211.43	12/3 803.17	36/8 686.80	8/5 093.18
09	36/9 343.73	8/6 711.32	37/8 198.38	12/3 924.39	36/8 536.53	8/4 745.35
10	36/8 585.78	8/6 094.22	36/7 042.09	12/3 695.85	36/8 252.50	8/4 418.64

Table 14
Detail results LC03, 600 customers

#	R1	R2	C1	C2	RC1	RC2
01	59/21875.04	11/18 833.86	60/14 095.64	18/7 776.16	56/18 295.89	15/13 275.93
02	55/20 131.27	11/15 412.26	56/14 332.05	18/8 106.79	55/17 413.64	12/12 071.40
03	55/18 517.45	12/12122.38	56/14 376.87	18/8 189.67	56/16 422.98	11/10 446.72
04	55/16 656.90	11/8 896.13	56/14 063.21	18/7 637.67	55/15 847.45	11/8 016.95
05	54/21 580.31	12/15 854.63	60/14 104.35	18/7 578.67	55/19 769.65	13/12324.05
06	54/20 291.10	11/13 164.53	60/14 093.83	18/7 500.99	55/18 482.75	12/10 700.42
07	54/18 570.23	11/10 697.64	60/14 295.01	18/7 761.43	55/18 286.55	11/11 852.97
08	54/17 030.55	11/8 473.53	59/14 259.39	18/7 483.18	55/17 598.42	11/11108.08
09	54/23 250.86	11/14 379.54	56/14 387.23	18/7 380.21	55/17 228.39	11/10 602.84
10	54/20 794.47	11/13 105.21	56/14 048.18	17/8 024.43	55/17 435.59	11/9 927.76

Table 15
Detail results LC03, 800 customers

#	R1	R2	C1	C2	RC1	RC2
01	80/37 666.58	15/293 83.35	80/25 030.36	24/11 677.72	73/33 213.55	20/20 869.21
02	73/35 472.29	15/23 942.54	75/25 518.17	24/13 724.04	72/39 696.20	17/18 672.43
03	73/31 690.97	15/19 050.31	72/27 341.54	24/12369.05	72/35 577.87	15/15 711.84
04	73/29 711.21	15/14 423.95	72/26 111.68	24/11 932.64	72/32 654.10	15/12 314.59
05	72/38 633.08	15/25 837.8	80/25 167.67	24/11 559.98	73/32 860.34	16/19 639.76
06	72/33 388.80	15/21 720.99	80/25 167.00	24/11 418.93	73/32 701.01	**15/19 744.73**
07	72/30 737.79	15/17 810.47	80/25 514.25	24/11 791.88	72/43 829.43	15/17 964.11
08	72/29 282.40	15/13 741.92	77/25 802.26	25/11 410.32	72/43 694.60	15/16 939.31
09	72/34 976.41	15/23 734.84	74/25 569.05	24/11 736.54	72/41 816.70	15/16 767.66
10	72/33 964.51	15/21 929.44	73/25 466.2	24/12230.00	72/41182.44	15/15 792.31

Table 16
Detail results LC03, 1000 customers

#	R1	R2	C1	C2	RC1	RC2
01	100/54 724.64	19/43 975.28	100/42 478.95	30/16889.95	90/51 529.12	22/31 457.37
02	92/56 362.29	19/35 901.41	93/44 459.70	30/18606.36	90/48 146.09	19/28 071.59
03	92/50 604.26	20/27 169.56	90/44 000.84	30/17801.03	90/46 201.21	18/22 680.56
04	92/46 017.14	19/20 002.45	90/42 373.79	29/19662.83	90/44 072.32	18/17 449.25
05	91/70 838.01	20/37 656.48	100/42 549.13	30/16586.46	90/52 780.36	**18/29 352.08**
06	91/54 952.03	19/32167.55	100/42 479.15	30/16473.14	90/52 842.71	18/28 797.14
07	91/50 040.50	19/25 261.34	100/42 689.28	31/17662.34	90/51 230.16	18/27 202.21
08	91/46 554.80	19/19 294.61	97/43 302.82	29/17219.59	90/50 648.85	18/26 037.65
09	91/61 862.40	19/35 146.70	92/45 372.21	30/16984.60	90/50 354.98	18/25 115.60
10	91/59 813.39	19/32 622.36	91/43 133.33	30/16548.74	90/49 307.75	18/23 847.69

References

[1] Crainic TG. Parallel computation, co-operation, tabu search. In: Rego C, Alidaee B, editors. Adaptive memory and evolution: tabu search and scatter search. Norwell, MA: Kluwer Academic Publishers; 2002.

[2] Crainic TG, Gendreau M. Towards an evolutionary method—cooperating multi-thread parallel tabu search hybrid. In: Voß S, Martello S, Roucairol C, Osman IH, editors. Meta-heuristics: Advances and Trends in Local Search Paradigms for Optimization. Norwell, MA: Kluwer Academic Publishers; 1998. p. 331–44.

[3] Crainic TG, Gendreau M. Cooperative parallel tabu search for capacitated network design. Journal of Heuristics 2002;8(6):601–27.

[4] Crainic TG, Toulouse M. Parallel metaheuristics. In: Crainic TG, Laporte G, editors. Fleet management and logistics. Norwell, MA: Kluwer Academic Publishers; 1998. p. 205–51.

[5] Crainic TG, Toulouse M. Parallel strategies for meta-heuristics. In: Glover F, Kochenberger G, editors. State-of-the-art handbook in metaheuristics. Norwell, MA: Kluwer Academic Publishers; 2002. p. 475–513.

[6] Crainic TG, Toulouse M, Gendreau M. Parallel asynchronous tabu search for multicommodity location-allocation with balancing requirements. Annals of Operations Research 1995;63:277–99.

[7] Cung V-D, Martins SL, Ribeiro CC, Roucairol C. Strategies for the parallel implementations of metaheuristics. In:

Ribeiro C, Hansen P, editors. Essays and surveys in metaheuristics. Norwell, MA: Kluwer Academic Publishers; 2001. p. 263–308.
[8] Garcia BL, Potvin J-Y, Rousseau JM. A parallel implementation of the tabu search heuristic for vehicle routing problems with time window constraints. Computers & Operations Research 1994;21(9):1025–33.
[9] Gambardella L, Taillard ÉD, Agazzi G. MACS-VRPTW: a multiple ant colony system for vehicle routing problems with time windows. Technical report IDSIA-9-99, IDSIA, Lugano, Switzerland, 1999.
[10] Homberger J, Gehring H. Two evolutionary metaheuristics for the vehicle routing problem with time windows. INFOR 1999;37:297–318.
[11] Gehring H, Homberger J. A parallel two-phase metaheuristic for routing problems with time windows. Asia-Pacific Journal of Operational Research 2001;18(1):35–47.
[12] Schulze J, Fahle T. A parallel algorithm for the vehicle routing problem with time window constraints. Annals of Operations Research 1999;86:585–607.
[13] Taillard ÉD, Gambardella LM, Gendreau M, Potvin J-Y. Programmation à mémoire adaptative. Calculateurs Parallèles, Réseaux et Systèmes répar tis 1998;10:117–40.
[14] Taillard ÉD. Recherches itératives dirigées parallèles. Ph.D. thesis, École Polytechnique Fédérale de Lausanne, 1993.
[15] Toulouse M, Crainic TG, Sansó B, Thulasiraman K. Self-organization in cooperative search algorithms. In: Proceedings of the 1998 IEEE International Conference on Systems, Man, and Cybernetics, Madisson, Wisconsin: Omnipress; 1998. p. 2379–85.
[16] Cordeau J-F, Desaulniers G, Desrosiers J, Solomon MM, Soumis F. The VRP with time windows. In: Toth P, Vigo D. editors. The vehicle routing problem, SIAM, Monographs on Discrete Mathematics and Applications, Philadelphia, PA: SIAM; 2002a. p. 157–93, [chapter 7].
[17] Bräysy O, Gendreau M. Tabu search heuristics for the vehicle routing problem with time windows. Top 2002;10(2):211–38. (see http://top.umh.es/issue10-2.html)
[18] Cordeau J-F, Gendreau M, Laporte G, Potvin J-Y, Semet F. A guide to vehicle routing heuristics. Journal of the Operational Research Society 2002b;53:512–22.
[19] Christofides N, Mingozzi A, Toth P. The vehicle routing problem. In: Christofides NAM, Toth P, Sandi C, editors. Combinatorial Optimization. New York: Wiley; 1979. p. 315–38.
[20] Desrosiers J, Dumas Y, Solomon MM, Soumis F. Time constrained routing and scheduling. In: Ball M, Magnanti TL, Monma CL, Nemhauser GL, editors. Network routing. Handbooks in operations research and management science, vol. 8. Amsterdam: North-Holland; 1995. p. 35–139.
[21] Desaulniers G, Desrosiers J, Ioachim I, Solomon MM, Soumis F, Villeneuve D. A unified framework for deterministic time constrained vehicle routing and crew scheduling problems. In: Crainic TG, Laporte G, editors. Fleet management and logistics. Norwell, MA: Kluwer Academic Publishers; 1998. p. 57–93.
[22] Fisher ML. Vehicle routing. In: Ball M, Magnanti TL, Monma CL, Nemhauser GL, editors. Network routing. Handbooks in operations research and management science, vol. 8. Amsterdam: North-Holland; 1995. p. 1–33.
[23] Gendreau M, Laporte G, Potvin J-Y. Metaheuristics for the vehicle routing problem. In: Toth P, Vigo D, editors. The vehicle routing problem. SIAM Monographs on Discrete Mathematics and Applications, vol. 9. Philadelphia, PA: SIAM; 2002. p. 129–54.
[24] Golden BL, Assad AA, editors. Vehicle routing: methods and studies. Amsterdam: North-Holland; 1988.
[25] Golden BL, Wasil EA, Kelly JP, Chao IM. Metaheuristics in vehicle routing. In: Crainic T, Laporte G, editors. Fleet management and logistics. Boston, MA: Kluwer Academic Publishers; 1998. p. 33–56.
[26] Laporte G. The vehicle routing problem: an overview of exact and approximate algorithms. European Journal of Operational Research 1992;59:345–58.
[27] Laporte G, Semet F. Classical heuristics for the vehicle routing problem. In: Toth P, Vigo D, editors. The vehicle routing problem. SIAM Monographs on Discrete Mathematics and Applications, vol. 9. Philadelphia, PA: SIAM; 2002. p. 109–28.
[28] Laporte G, Gendreau M, Potvin J-Y, Semet F. Classical and modern heuristics for the vehicle routing problem. International Transactions in Operational Research 2000;7(4/5):285–300.
[29] Toth P, Vigo D. Exact solution of the vehicle routing problem. In: Crainic TG, Laporte G, editors. Fleet management and logistics. Norwell, MA: Kluwer Academic Publishers; 1998. p. 1–31.
[30] Toth P, Vigo D, editors. The vehicle routing problem. SIAM Monographs on Discrete Mathematics and Applications, vol. 9. Philadelphia, PA: SIAM; 2002.

[31] Aiex RM, Martins SL, Ribeiro CC, Rodriguez NR. Cooperative multi-thread parallel tabu search with an application to circuit partitioning. In: Proceedings of IRREGULAR'98—Fifth International Symposium on Solving Irregularly Structured Problems in Parallel, Lecture Notes in Computer Science, vol. 1457. Berlin: Springer; 1998. p. 310–31.
[32] Solomon MM. Time window constrained routing and scheduling problems. Operations Research 1987;35:254–65.
[33] Chabrier A. Vehicle routing problem with elementary shortest path based column generation. Working paper, ILOG, Madrid: Spain; 2002.
[34] Cook W, Rich JL. A parallel cutting plane algorithm for the vehicle routing problem with time windows. Working paper, Computational and Applied Mathematics, Rice University, Houston, TX, 1999.
[35] Kallehauge B, Larsen J, Madsen OBG. Lagrangean duality and non-differentiable optimization applied on routing with time windows—experimental results. Internal report imm-rep-2000-8, Department of Mathematical Modelling, Technical University of Denmark, Lyngby, Denmark, 2000.
[36] Larsen J. Parellellization of the vehicle routing problem with time windows. Ph.D. thesis, Department of Mathematical Modelling, Technical University of Denmark, Lyngby, Denmark, 1999.
[37] Kohl N, Desrosiers J, Madsen OBG, Solomon MM, Soumis F. 2-path cuts for the vehicle routing problem with time windows. Transportation Science 1999;33(1):101–16.
[38] Glover F. Tabu search—Part I. ORSA Journal on Computing 1989;1(3):190–206.
[39] Glover F. Tabu search—Part II. ORSA Journal on Computing 1990;2(1):4–32.
[40] Glover F, Laguna M. Tabu search. Norwell, MA: Kluwer Academic Publishers; 1997.
[41] Gendreau M, Hertz A, Laporte G. A tabu search heuristic for the vehicle routing problem. Management Science 1994;40:1276–90.
[42] Cordeau J-F, Laporte G, Mercier A. A unified tabu search heuristic for vehicle routing problems with time windows. Journal of the Operational Research Society 2001;52:928–36.
[43] Rousseau L-M, Gendreau M, Pesant G. Using constraint-based operators with variable neighborhood search to solve the vehicle routing problem with time windows. Journal of Heuristics 2002;8(1):43–58.
[44] Mladenović N, Hansen P. Variable neighborhood search. Computers & Operations Research 1997;24:1097–100.
[45] Hansen P, Mladenović N. An introduction to variable neighborhood search. In: Voß S, Martello S, Roucairol C, Osman IH, editors. Meta-heuristics 98: theory and applications. Norwell, MA: Kluwer Academic Publishers; 1999. p. 433–58.
[46] Hansen P, Mladenović N. Developments of variable neighborhood search. In: Ribeiro C, Hansen P, editors. Essays and surveys in metaheuristics. Norwell, MA: Kluwer Academic Publishers; 2002. p. 415–39.
[47] Bräysy O. A reactive variable neighborhood search algorithm for the vehicle routing problem with time windows. INFORMS Journal on Computing, 2003;15(4):347–68.
[48] Bräysy O, Hasle G, Dullaert W. A multi-start local search algorithm for the vehicle routing problem with time windows. European Journal of Operational Research, In Press, corrected proof available online 8th October 2003.
[49] Taillard ÉD, Badeau P, Gendreau M, Guertin F, Potvin J-Y. A tabu search heuristic for the vehicle routing problem with soft time windows. Transportation Science 1997;31(2):170–86.
[50] Bräysy O, Dullaert W. A fast evolutionary metaheuristic for the vehicle routing problem with time windows. International Journal on Artificial Intelligence 2003;12(2):153–72.
[51] Bent R, Van Hentenryck P. A two-stage hybrid local search for the vehicle routing problem with time windows. Transportation Science, to appear.
[52] Shaw P. Using constraint programming and local search methods to solve vehicle routing problems. Lecture Notes in Computer Science 1998;1520:417–31.
[53] Rochat Y, Taillard ÉD. Probabilistic diversification and intensification in local search for vehicle routing. Journal of Heuristics 1995;1(1):147–67.
[54] Badeau P, Guertin F, Gendreau M, Potvin J-Y, Taillard ÉD. A parallel tabu search heuristic for the vehicle routing problem with time windows. Transportation Research C: Emerging Technologies 1997;5(2):109–22.
[55] Homberger J. Verteilt-parallele metaheuristiken zur tourenplanung. Technical report, Gaber, Wiesbaden, 2000.
[56] Berger J, Barkaoui M, Bräysy O. A parallel hybrid genetic algorithm for the vehicle routing problem with time windows. Working paper, Defense Research Establishment Valcartier, Canada, 2001.
[57] Toulouse M, Crainic TG, Gendreau M. Communication issues in designing cooperative multi thread parallel searches. In: Osman IH, Kelly JP, editors. Meta-heuristics: theory & applications. Norwell, MA: Kluwer Academic Publishers; 1996. p. 501–22.

[58] Gendreau M, Hertz A, Laporte G. New insertion and postoptimization procedures for the traveling salesman problem. Operations Research 1992;40(6):1086–94.
[59] Gendreau M, Hertz A, Laporte G, Stan M. A generalized insertion heuristic for the traveling salesman problem with time windows. Operations Research 1998;46(3):330–5.
[60] Glover F. Multilevel tabu search and embedded search neighborhoods for the traveling salesman problem. Working paper, College of Business & Administration, University of Colorado, USA, 1991.
[61] Glover F. Ejection chains, reference structures and alternating path methods for traveling salesman problems. Discrete Applied Mathematics 1992;49:231–55.
[62] Rego C. A subpath ejection method for the vehicle routing problem. Management Science 1998;44:1447–59.
[63] Rego C. Node ejection chains for the vehicle routing problem: sequential and parallel algorithms. Parallel Computing 2001;27:201–22.
[64] Caseau Y, Laburthe F. Heuristics for large constrained vehicle routing problems. Journal of Heuristics 1999;5: 281–303.
[65] Kindervater G, Savelsbergh M. Vehicle routing: handling edge exchanges. In: Aarts E, Lenstra JK, editors. Local search in combinatorial optimization, New York: Wiley; 1997. p. 337–60, [chapter 10].
[66] De Backer B, Furnon V, Shaw P, Kilby P, Prosser P. Solving vehicle routing problems using constraint programming and metaheuristics. Journal of Heuristics 2000;6:501–23.
[67] Bentley J. Fast algorithms for geometric traveling salesman problems. ORSA Journal on Computing 1992;4(4): 388–411.
[68] Le Bouthillier A, Crainic TG. Cooperative parallel method for vehicle routing problems with time windows. Publication CRT-00-55, Centre de recherche sur les transports, Université de Montréal, Montréal, QC, Canada, 2002.