Published in IEEE Intelligent Systems 20, issue 4, 36-42, 2005
which should be used for any reference to this work

A Guided Cooperative Search for the Vehicle Routing Problem with Time Windows

Alexandre Le Bouthillier, *Université de Montréal*

Teodor G. Crainic, *Université du Québec à Montréal*

Peter Kropf, *Université de Neuchâtel*

An enhanced cooperative-search mechanism creates new information from exchanged solutions and guides the global search with a pattern-identification mechanism.

Parallel computing can significantly enhance efficiency and robustness, the primary qualities of good metaheuristics. Three forms of parallelism are applicable to metaheuristics:

- division of compute-intensive tasks at a low algorithmic level,
- explicit domain decomposition of the solution or search space, and
- multithread search.

An independent multithread search produces the best solution among the solutions found by each independent search. In multithread cooperative search, a mechanism lets the search threads exchange information (such as solutions). Researchers have developed several such mechanisms, and in many cases, cooperative multisearch produces qualitatively better solutions and shorter resolution times than independent search algorithms.[1]

Although all parallelization strategies speed resolution, cooperative search methods also increase global-search robustness.[1] Cooperative search uses a *solution warehouse* to combine the efforts of several independent metaheuristics. The search threads send to the warehouse "good" solutions. The warehouse, on demand and according to the internal logic of individual search methods, provides them with solutions to diversify the search, act as parents in population-based metaheuristics, or provide initial solutions. This simple mechanism allows asynchronous communication and an exchange of solutions that influences each method's search trajectory. Researchers have successfully enhanced cooperation with simple extraction rules to address several combinatorial problems: network design,[2] multicommodity location-allocation,[3] circuit partitioning,[4] and the *vehicle routing problem with time windows* (VRPTW).[5,6]

From a conceptual viewpoint, the solution warehouse contains useful indirect information for the global search. We can therefore use the solution discovery history and the frequency with which certain attributes appear in solutions of particular quality to create new information about the search space. Such information creation transforms the solution warehouse into an intelligent warehouse holding complex information and guiding independent methods toward promising or unexplored regions.

We have developed a pattern-identification mechanism that endows cooperative search with capabilities to create new information and guide the global search. The proposed mechanism sends information to independent metaheuristics about promising and unpromising patterns in the solution space. By fixing or prohibiting specific solution attribute values in certain search metaheuristics, we can focus the search on desired regions. The mechanism thus enforces better coordination between individual methods and controls the global search's diversification and intensification. Because our approach does not assume a specific problem structure, we can apply it to a wide range of combinatorial problems.

A cooperative-search framework

Figure 1 illustrates the cooperative-search framework, consisting of independent processes (threads) of possibly different types communicating through a solution warehouse. Search threads perform such actions as

- heuristically constructing new solutions,
- executing neighborhood-based improving meta-

heuristics through the search space,
- implementing population-based metaheuristics (such as evolutionary algorithms, scatter search, and path relinking), and
- performing postoptimization procedures (such as intensive local search) on solutions in the solution warehouse.

Neighborhood-based improving metaheuristics, such as tabu search,[7] aggressively explore the search space, whereas population-based metaheuristics, such as evolutionary algorithms,[8] help increase the diversity of solutions exchanged among cooperating methods. When several search threads use the same metaheuristic, the initial solution and the particular setting of search parameters differentiate them.

When a thread wants to send out information (for example, when it identifies a new local optimum), it sends the information to the solution warehouse. Similarly, threads access outside information (to diversify the search, for example) through the warehouse. Individual threads, irrespective of their roles as information senders or receivers, initiate communication. No broadcasting occurs, and no complex mechanisms exist for selecting the threads that will receive or send information or for controlling the cooperation.

The solution warehouse is thus an efficient implementation paradigm that allows for strict asynchronous exchange with no predetermined connection pattern. No thread interrupts another for communication purposes, but any thread can access at any time the data any other search thread has sent.

We characterize the cooperation process by specifying

- the information to be shared,
- the search methods constituting the cooperative search,
- when and how communications occur, and
- how the threads use the imported information.[1]

The information exchanged among cooperating procedures must be *meaningful*—that is, it must be useful in the receiving threads' decision processes. In this sense, information indicating the current status of the global search or, at least, of some independent metaheuristic, is meaningful.

Two main classes of cooperation mechanisms exist. *Adaptive-memory* approaches store partial elements of good solutions and combine them to create new complete solutions, which the cooperating threads then improve.[9] *Central-memory* approaches exchange complete elite solutions (within the top 10 percent of solutions, for example) among neighborhood and population-based metaheuristics.[1,3,5,6]

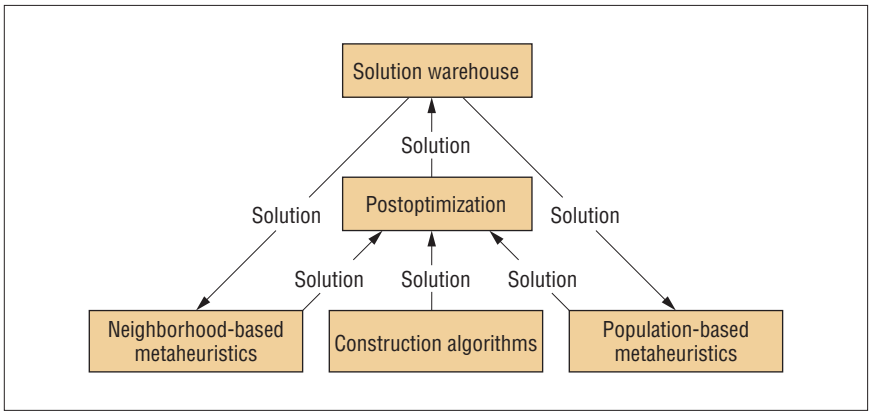

Figure 1. An abstract framework for cooperative search.

In a simple central-memory cooperation scheme,[5] threads share information about the respective good solutions they have identified. When a thread improves the imported solution or identifies a new best solution, it sends it to the solution warehouse. This simple, intuitive scheme satisfies the meaningfulness requirement.

The selection of search methods involved in cooperative search should aim to obtain

- good-quality solutions,
- diverse solutions to facilitate discovery of promising regions,
- rapid production of intermediate solutions to feed the information exchange mechanisms,
- a mechanism that combines various solutions to create diversity, and
- a mechanism that can escape local optima.

The solution warehouse is thus the cooperation mechanism's core. It keeps good solutions and it is dynamically updated by the independent search threads, which require solutions from this warehouse at various stages of execution. The warehouse orders solutions according to a predetermined utility measure that quantifies their quality. The solution utility can be the solution's objective value or a combined measure of its properties.

Independent search methods send their improved solutions to the postoptimization algorithms. These solutions are "in training" until postoptimized. Thereafter, they are sent as "adult" solutions to the solution warehouse. The solution warehouse eliminates any duplicate solutions it receives.

All requests for solutions initiated by the independent threads are sent to the solution warehouse, which responds with an adult solution. The warehouse selects solutions randomly according to probabilities biased toward the "best" solution based on the function used to order solutions in the solution warehouse.

We set the solution warehouse's population size relative to the problem size and eliminate the worst results. No direct communication occurs between processes, which enforces their independence and an asynchronous exchange mode. This scheme simplifies the parallelism in the cooperation design. We can easily modify the parallel system by adding new metaheuristics or dropping inefficient ones. Moreover, because the scheme does not assume a specific problem, it is equally relevant for problems with easily defined solution components (such as the routes in vehicle routing problems) and for problems in which such structures are much less apparent (for example, network design). We improve this simple cooperating scheme by extracting new knowledge from the information exchanged to yield a more efficient global search.

Pattern identification

Our mechanism for extracting knowledge from the information exchanged uses pattern identification on the solution warehouse. The mechanism then fixes or prohibits specific solution attributes (such as arcs in network-based problems) found in patterns for part of the search performed by independent metaheuristics. This lets us constrain the search space of particular cooperating metaheuristics and thus perform global intensification and diversification phases to guide exploration of

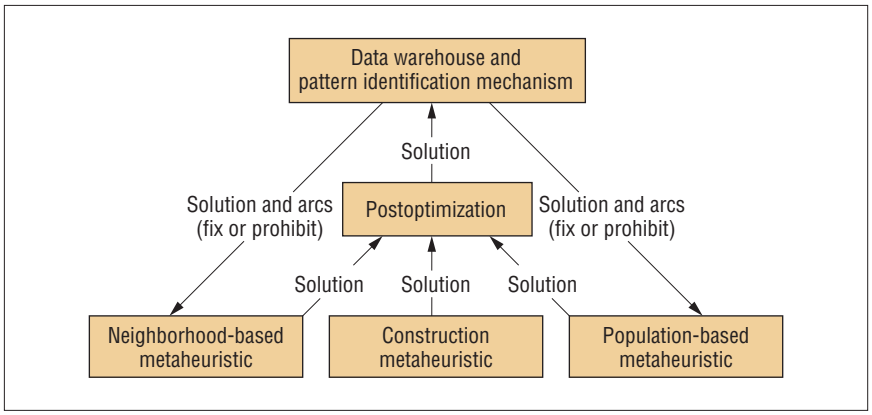

Figure 2. Guided cooperative search applied to the vehicle routing problem with time windows (VRPTW).

the solution space and control the solution warehouse population's quality and diversity.

To illustrate our mechanism's utility, we described it in terms of the inclusion or exclusion of arcs in a given network problem.

Pattern definition

For combinatorial problems such as network design, the traveling salesman problem, and vehicle routing, we use a simple, general pattern definition based on the inclusion of arcs in particular solutions.

Consider the frequency of arc inclusion in a given subset of the solution warehouse. This subset can be the entire population or an elite (top 10 percent), average (middle 10 to 90 percent), or worst (bottom 10 percent) group of solutions. An arc with a high frequency in a given group signals that the cooperating metaheuristics often produce solutions that include the arc. By tagging solutions to identify its providing algorithm, we can induce similar information about a subset of participating algorithms. When we consider several arcs' inclusion frequency, we see patterns emerge among solutions in the solution warehouse or in the subset examined.

We define a pattern of length n as a subset of arcs of cardinality $n = 1, 2, \ldots$, to the maximum number of arcs in the problem definition. A frequent pattern relative to a set of arcs consists of arcs appearing with high frequency in the set's solutions. We select high-frequency arcs sequentially in decreasing order, starting with the highest frequency value. Conversely, an infrequent pattern relative to a set of arcs consists of arcs appearing with low frequency in the set's solutions. We select low-frequency arcs sequentially in increasing order, starting with the lowest frequency value.

We select patterns from specific subpopulations (elite, average, and worst) and compare the patterns' rate of appearance among them. These comparisons form the basis of our pattern-identification mechanism.

Pattern-appearance frequencies

An *in-pattern* appears in at least one solution of a subset; a *statistical pattern* does not appear in any solution of the subset. A statistical pattern is thus only a consequence of the statistical process of accounting for the frequency of individual arcs. For example, the arcs x and y might be the two most frequent arcs in solutions of the elite population but for incompatibility reasons never appear in the same solution.

In an in-pattern of length n shared by the elite, average, and worst solution subpopulations, two meaningful situations can occur with respect to the pattern's frequency of appearance as we move from the elite to the average to the worst subpopulation. If the frequency increases, the in-pattern is *unpromising*; if it is stable or decreasing, the in-pattern is *promising*.

Pattern identification in global search

Consider an unpromising pattern of length n. To intensify the search around solutions with good attributes, we prohibit the arcs defining the pattern. However, fixing these arcs for several iterations diversifies the search relative to the current set of good attribute values. Symmetrically, given a promising pattern of length n, fixing the arcs of the pattern intensifies the search, while prohibiting them diversifies it.

The global search is the independent methods' cumulative search effort. To prevent independent methods from converging too rapidly, we promote diversification by prohibiting arcs during the global search's initial phases. Later in the search, we encourage intensification by fixing promising arcs, which enforces exploration of the solution space's promising regions. We also vary pattern length to modulate the intensity of global diversification and intensification phases and thus influence the evolution of the diversity and quality of solutions in the solution warehouse.

At the search's beginning, the search threads have gathered limited information because the solution space is insufficiently explored. Therefore, the solution warehouse is not representative of the solutions space, even when the population is diverse. So, we initially build increasingly long patterns from the average subpopulation so that we can identify promising patterns more rapidly. As the search progresses, we build patterns from the elite and worst subpopulations, as we described earlier. When we find a statistical pattern, we reduce its length until we obtain an in-pattern.

We have made several modifications to the initial framework. The solution warehouse now includes a process to identify and manage patterns. The process includes decisions on when to compute particular patterns and which metaheuristics to send them to. The process becomes, in fact, the new global metaheuristic corresponding to the global guided cooperative search. The solution warehouse now contains solutions and pattern information.

We also modified communications from the solution warehouse to the independent metaheuristics. The warehouse sends the appropriate pattern and instructions on fixing or prohibiting arcs with the solution (selected according to the original criteria). We thus modify each metaheuristic to cope with these instructions. Figure 2 illustrates the guided-cooperative-search framework as applied to the VRPTW.

Guided cooperative search

To demonstrate our mechanism, we developed a guided cooperative search for VRPTW, a well-known and extensively studied combinatorial problem that is thus well suited for benchmarking.

We address the single-depot VRPTW, which assumes a set of customers with known positive demands and specific time intervals during which service is available. A fleet of

homogeneous vehicles of known capacity resides at a given depot to perform the service. The objective is to find a set of closed routes (or tours) that start and end at the depot during its opening hours, while

- minimizing the total cost of performing the service,
- ensuring that customers receive exactly one service during their specified time windows, and
- avoiding overloading vehicles.

Figure 3 is an example of a best known solution for a problem of 600 clustered customers.

Jean-François Cordeau and his colleagues review VRPTW problem variants, formulations, and solution methods.[10] In the problem version we address, cost is a combination of the number of vehicles (routes) used and the total distance traveled. We associate high cost with vehicle use to force the search toward solutions with fewer vehicles. Each customer is visited exactly once. A vehicle cannot arrive later than the customer's closing time but can arrive before the customer's opening time, in which case it waits, without explicit penalty, until the customer is ready. Once the service starts, it continues until completion, even if the service ending time is later than the time window's expiration.

We previously developed a cooperative parallel method for the VRPTW based on our simple solution warehouse mechanism.[5] The cooperation involved

- the unified tabu[11] and Taburoute,[12] two tabu search methods that perform well sequentially;
- two simple evolutionary algorithms, one with order crossover (OX) and one with edge recombination (ER) crossover; and
- several postoptimization methods (2-opt, 3-opt, or-opt, and ejection chains) to reduce the number of vehicles and total distance traveled.

We used four simple construction algorithms to create an initial population.

We applied our cooperation framework to our cooperative parallel method.[5] For the VRPTW, defining an in-pattern of length n is straightforward. We define the problem on a network, where an arc corresponds to a possible movement between two customers or between a customer and a depot. We use the procedures introduced in the previous section to define patterns.

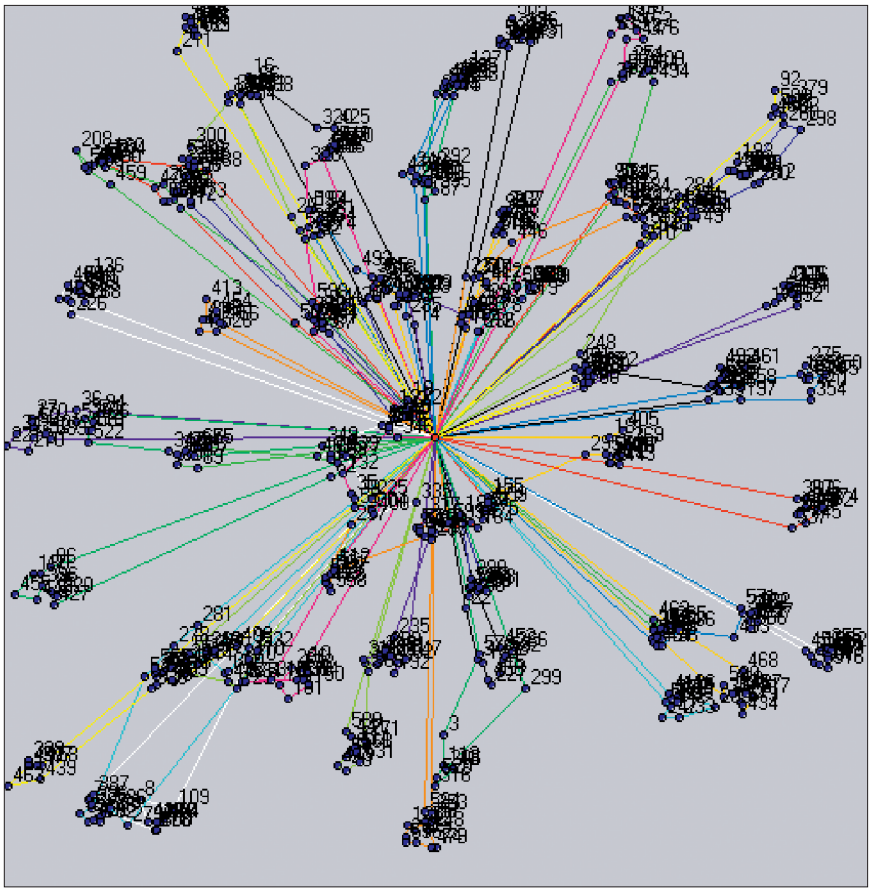

Figure 3. Best known solution for problem C1-6-4, which involves 600 clustered customers. The depot appears at the image's center. The different routes for serving customers are in different colors.

Fixing and prohibiting arcs in the solutions explored by the four construction algorithms is also straightforward. Fixing arcs always leads to a nonempty solution space that, at the most extreme, reduces to a single solution representing a too-long pattern. Prohibiting arcs can lead to an empty feasible solution space when patterns are too long. To avoid both situations and strike a balance between the number of feasible solutions and the constrained solution space's size, we limit the pattern length to 25 percent of the problem's size.

We divide the computing time (wall-clock time) allocated to the cooperative method into four phases: two phases of diversification at the beginning to broaden the search, and then two intensification phases to focus the search around promising regions. The phases proceed as follows:

1. Use unpromising in-patterns of frequent arcs in the average subpopulation, and prohibit them in the independent metaheuristics.
2. Prohibit arcs from frequent unpromising in-patterns from the worst subpopulation.
3. Work with the average subpopulation, and fix arcs from frequent promising in-patterns.
4. Use frequent promising in-patterns from the elite subpopulation, and fix the arcs for the metaheuristic searches.

The first two phases explore pattern lengths in decreasing lengths; the latter two explore them in increasing lengths, by increments of one unit.

Computational experiments

By comparing our guided cooperative search (LCK05) to both the simple version (LC03) and the existing best-performing methods for the VRPTW, we aim to show that our method performs competitively in terms of solution quality and computational effort. We also aim to evaluate the impact of guiding the search toward or away from specific

patterns and performing diversification and intensification to control the evolution of the population.

We ran Taburoute, a unified tabu search algorithm, and two evolutionary algorithms with OX and ER crossover, on each of four processors. The solution warehouse, the postoptimization procedures, the pattern-identification mechanism, and the construction methods were assigned to a fifth processor.

For Taburoute, we used the original parameter settings for the vehicle routing problem.[12] We set tabu tag lengths to between five and 10 iterations, and evaluated 15 percent of the nodes in the *p*-neighborhood dimensions. We selected the initial solution from 15 initial generated solutions and a solution from the solution warehouse, using a penalty of 1 for frequently moved arcs. We used the parameters for the value function that Jean-François Cordeau, Gilbert Laporte, and Anne Mercier adopted (alpha = 1, beta = 1, gamma = 1) for the unified tabu.[11] Finally, we used an arc mutation probability of 1 percent on temporary copies of the parents for the crossovers used by the evolutionary algorithms.

We tested Marius Solomon's standard set of test problems, which contains 56 problems with 100 customers each.[13] We also used an extended set, which contains 300 problem instances with 200 to 1,000 customers.[8] We divided the Solomon set and extended problems into six classes: C1, C2, R1, R2, RC1, and RC2. For all problem instances, customers are distributed in a [0, 100] square unit. In the C sets, customers are clustered together, whereas they are randomly distributed in the R sets. Problems in the RC sets combine the two characteristics. Time windows at the depot are relatively small for C1, R1, and RC1 problems, so each route serves fewer customers. Time windows for C2, R2, and RC2 problems are larger. The service time is 10 units per customer for R and RC and 90 units for C.

We sort solutions in the solution warehouse first by the number of vehicles and second by a weighted sum, C(*p*), of attributes:

- the total time required to serve all customers,
- the associated total distance,
- the customers' total waiting time, and
- the sum of the slack in each time window,

$$C(p) = W1 * totalTime + W2 * totalDistance + W3 * totalWait + W4 * totalSlack.$$

We set the parameters weight, W1–W4, to 1 in all the reported experiments. We combined this measure with the number of vehicles to represent the solution quality (*totalTime* and *totalDistance*) and flexibility (*totalWait* and *totalSlack*). The last two measures indicate how much slack exists in the solution and how easily we can explore feasible neighboring solutions.

Previous research found that cooperative search provides faster results of equivalent or better quality than the independent searches performing without communication.[5] We therefore compare only simple (LC03) and guided parallel cooperative searches (LCK05).

Cooperative metaheuristics performed 12-minute wall-clock time runs for each of the 100-customer problems. We allowed longer running times, equal to those reported by Jörg Homberger and Hermann Gehring,[8] for the larger problem instances. These times increase up to 50-minute wall-clock time for the 1,000-customer problem.

Previous research found that cooperative search provides faster results of equivalent or better quality than the independent searches performing without communication.

To compare the wall-clock time in the simple cooperative search, we created virtual machines using VMware and forced their CPU clocks to emulate the machines used in the previous study.[5] Using faster virtual or physical machines only decreases the wall-clock time for the same results. We used a virtual cluster of five Pentium III 850-MHz CPU computers with 512 MBytes of RAM under Linux. We performed distance computations in double precision. The implementation is machine independent and can be run with MPICH (www-unix.mcs.anl.gov/mpi/mpich) on Unix, Windows, or Linux.

We allocated one-fourth of the total wall-clock execution time to each of the four global phases prohibiting or fixing arcs. In-pattern lengths were at most 25 percent of the problem size.

We present here the average results per class for the cumulative number of vehicles (CNV) and cumulative total distance (CTD) for the standard Solomon problems (see Table 1) and the extended problem set (see Table 2). The best results are in boldface, and Table 1 lists methods in decreasing value of the CNV and CTD. We compare results from the cooperative search method and the guided parallel cooperative search to results from the best methods (published or not) for the VRPTW on the SINTEF benchmark site (www.sintef.no/static/am/opti/projects/top) on 15 February 2005. Detailed results on the standard and extended Solomon's benchmark problem sets appear in the technical report for LCK05.[6]

Our method yields good results, appearing in second place in both tables. For the 100-customer problems, our method yields a CNV of 405, which is the lowest number obtained so far and makes the guided cooperative search one of the best metaheuristics available for the VRPTW.

Our method reduced the number of vehicles by two and the distance by 52.38 units over the simple cooperative search for problems with 100 customers. Our cooperative guided search produced 24 of the best known results and reports the best known CNV. It reports a new best average for the R1 problems. For the C1 and C2 problems with 100 customers, almost all methods found the best known average number of vehicles and the total distance. For the other problem classes, the cooperative guided search found the best known number of vehicles in all instances and at most a 0.16 percent increase in distance over the other methods.

Table 2 displays the results for CNV and CTD values aggregated by problem size for the extended set of problems. Few researchers have addressed the entire problem set, which explains the limited number of entries in the table (as we mentioned before, we selected the best methods for comparison).

Our cooperative guided search obtained a new best known CTD value for 200-customer problems. For all of the problem classes, the difference between methods is less than 0.19 percent in terms of the CNV. In all cases, our approach improved upon the simple cooperative search by a total of 19 vehicles and 171,718 in distance. It reports a slightly higher total number of vehicles compared to HG, but it also shows a reduc-

Table 1. Average number of vehicles and distances of methods for the vehicle routing problem with time windows on 100-customer problems (CNV is the cumulative number of vehicles; CTD is the cumulative total distance). The best results are in boldface.

Method	R1	R2	C1	C2	RC1	RC2	CNV/CTD	Details
RT*	12.25 1,208.50	2.91 961.72	**10.00** **828.38**	**3.00** **589.86**	11.88 1,377.39	3.38 1,117.44	415 57,231	SG 100 MHz, 1 run, 92.2 minutes
CLM†	12.08 1,210.14	2.73 969.57	**10.00** **828.38**	**3.00** **589.86**	11.50 1,389.78	3.25 1,134.52	407 57,555	Sun U2 300 MHz, n/a
LC03‡	12.08 1,209.19	2.73 963.62	**10.00** **828.38**	**3.00** **589.86**	11.50 1,389.22	3.25 1,143.70	407 57,412	5xP850 MHz, 1 run, 12 minutes
GH01‖	12.00 1,217.57	2.73 961.29	**10.00** **828.63**	3.00 590.33	11.50 1,395.13	3.25 1,139.37	406 57,641	4 P400 MHz, 5 runs, 13.5 minutes
B01#	11.92 1,222.12	2.73 975.12	**10.00** **828.38**	**3.00** **589.86**	11.50 1,389.58	3.25 1,128.39	405 57,710	P200 MHz, 1 run, 82.5 minutes
LCK05**	**11.92** **1,214.20**	2.73 954.32	**10.00** **828.38**	**3.00** **589.86**	11.50 1,385.30	3.25 1,129.43	405 57,360	5xP850 MHz, 1 run, 12 minutes
BVH††	12.18 1,231.08	**2.73** **954.18**	**10.00** **828.38**	**3.00** **589.86**	**11.50** **1,384.17**	**3.25** **1,124.47**	**405** **57,272**	Sun U10 440 MHz, 5 runs, 120 minutes

* RT: Yves Rochat and Éric Taillard's active-guided-evolution strategy[9]
† CLM: Jean-François Cordeau, Gilbert Laporte, and Anne Mercier's unified tabu search[11]
‡ LC03: Simple cooperative search[5]
‖ GH01: Hermann Gehring and Jörg Homberger's active-guided-evolution strategy[14]
B01: Olli Bräysy's active-guided-evolution strategy[15]
** LCK05: Our guided parallel cooperative search method
†† BVH: Russell Bent and Pascal Van Hentenryck's two-stage hybrid local search[16]

	BVH*		MB†		LC03‡		LCK05‖		HG#	
Problem	CNV	CTD	CNV	CTD	CNV	CTD	CNV	CTD	CNV	CTD
100	**405**	**57,272**	n/a	n/a	407	57,412	405	57,360	405	57,715
200	697	171,715	694	168,573	694	173,062	**694**	**169,959**	694	173,313
400	1,393	410,112	1,389	390,386	1,390	410,330	1,389	396,612	**1,388**	**409,764**
600	2,091	858,040	2,082	796,172	2,088	840,583	2,086	809,494	**2,076**	**851,681**
800	2,778	1,469,790	2,765	1,361,586	2,766	1,475,436	2,761	1,443,400	**2,755**	**1,479,802**
1,000	3,468	2,266,959	3,446	2,078,110	3,451	2,225,367	3,442	2,133,645	**3,439**	**2,236,583**
Total	10,832	5,233,888	n/a	n/a	10,796	5,182,188	10,777	5,010,470	**10,757**	**5,208,858**

* BVH: Russell Bent and Pascal Van Hentenryck's two-stage hybrid local search[16]
† MB: David Mester and Olli Bräysy's active guided evolution strategy[17]
‡ LC03: Simple cooperative search[5]
‖ LCK05: Our guided parallel cooperative-search method
HG: Homberger and Gehring's evolutionary algorithm[8]

tion in the total cumulative distance by 3.96 percent for all classes.

We can construct patterns of attributes independently of particular solution structures and apply them to a wide range of combinatorial problems to increase robustness and speed of resolution. Patterns of attributes on explored solutions can enhance the guidance in noncooperating parallel methods as well as in traditional sequential methods. Future work includes identifying disjoint sets of solution elements in the explored solution space, which will require an enhanced pattern-identification mechanism. ⬒

Acknowledgments

We thank Jean-François Cordeau for sharing his unified tabu search code and Michel Gendreau for his assistance with the original version of Taburoute. The Natural Sciences and Engineering Council of Canada partially supported this research through its Discovery Grants program. The authors are affiliated with the Center for Research on Transportation and thank if for its support.

References

1. T.G. Crainic and M. Toulouse, "Parallel Strategies for Meta-heuristics," *State-of-the-Art Handbook in Metaheuristics*, F. Glover and G. Kochenberger, eds., Kluwer Academic, 2003, pp. 475–513.

2. T.G. Crainic and M. Gendreau, "Cooperative Parallel Tabu Search for Capacitated Network Design," *J. Heuristics*, vol. 8, no. 6, 2002, pp. 601–627.

3. T.G. Crainic, M. Toulouse, and M. Gendreau, "Parallel Asynchronous Tabu Search for Multicommodity Location-Allocation with Balancing Requirements," *Annals of Operations*

Research, vol. 60, no. 63, 1995, pp. 277–299.

4. R.M. Aiex et al., "Cooperative Multi-Thread Parallel Tabu Search with an Application to Circuit Partitioning," *Proc. 5th Int'l Symp. Solving Irregularly Structured Problems in Parallel* (IRREGULAR 98), LNCS 1457, Springer-Verlag, 1998, pp. 310–331.
5. A. Le Bouthillier and T.G. Crainic, "A Cooperative Parallel Meta-Heuristic for the Vehicle Routing Problem with Time Windows," *Computers & Operations Research*, vol. 32, no. 7, 2005, pp. 1685–1708.
6. A. Le Bouthillier, T.G. Crainic, and P. Kropf, *Towards a Guided Cooperative Search*, publication CRT-05-09, Center for Research on Transportation, Univ. de Montréal, 2005.
7. F.F. Glover, "Tabu Search—Part I," *ORSA J. Computing*, vol. 1, no. 3, 1989, pp. 190–206.
8. J. Homberger and H. Gehring, "Two Evolutionary Metaheuristics for the Vehicle Routing Problem with Time Windows," *INFOR*, vol. 37, Aug. 1999, pp. 297–318.
9. Y. Rochat and É.D. Taillard, "Probabilistic Diversification and Intensification in Local Search for Vehicle Routing," *J. Heuristics*, vol. 1, no. 1, 1995, pp. 147–167.
10. J.-F.Cordeau et al., "The VRP with Time Windows," *The Vehicle Routing Problem*, P. Toth and D. Vigo, eds., Soc. Industrial and Applied Mathematics, 2002, pp. 157–193.
11. J.-F. Cordeau, G. Laporte, and A. Mercier, "A Unified Tabu Search Heuristic for Vehicle Routing Problems with Time Windows," *J. Operational Research Soc.*, vol. 52, Aug. 2001, pp. 928–936.
12. M. Gendreau, A. Hertz, and G. Laporte, "A Tabu Search Heuristic for the Vehicle Routing Problem," *Management Science*, vol. 40, no. 10, 1994, pp. 1276–1290.
13. M.M. Solomon, "Time Window Constrained Routing and Scheduling Problems," *Operations Research*, vol. 35, 1987, pp. 254–265.
14. H. Gehring and J. Homberger, "A Parallel Two-Phase Metaheuristic for Routing Problems with Time Windows," *Asia-Pacific J. Operational Research*, 2001, vol. 18, no. 1, 2001, pp. 35–47.
15. O. Bräysy, "A Reactive Variable Neighborhood Search Algorithm for the Vehicle Routing Problem with Time Windows," *INFORMS J. Computing*, vol. 15, no. 4, 2003, pp. 347–368.
16. R. Bent and P. Van Hentenryck, "A Two-Stage Hybrid Local Search for the Vehicle Routing Problem with Time Windows," *Transportation Science,* vol. 38, no. 4, 2004, pp. 515–530.
17. D. Mester and O. Bräysy, "Active Guided Evolution Strategies for Large Scale Vehicle Routing Problems with Time Windows," *Computers & Operations Research*, vol. 32, no. 6, 2005, pp. 1593–1614.

The Authors

Alexandre Le Bouthillier is the founder of Omega Technologies, cofounder of Omega Optimisation, and a PhD student in computer science at University of Montreal. His research interests include parallel computing, artificial intelligence, and operations research. He received his master's in computer science from the University of Montreal. Contact him at the Center for Research on Transportation, Univ. de Montréal, PO Box 6128, Station Centre-Ville, Montréal, QC H3C 3J7, Canada; alexleb@crt.umontreal.ca.

Teodor G. Crainic is a professor in the School of Management Sciences at Université du Québec à Montréal, and the director of the Intelligent Transportation System Laboratory at the Center for Research on Transportation. He is also a fellow at CIRANO and an adjunct professor in the Université de Montréal, Molde University College, and the University of Manitoba, Alberta. His research interests include operations research with applications in transportation, electronic business, and telecommunications. He received his PhD in computer science and operations research from the University of Montréal. Contact him at the Center for Research on Transportation, Univ. de Montréal, PO Box 6128, Station Centre-Ville, Montréal, QC H3C 3J7, Canada; theo@crt.umontreal.ca.

Peter Kropf is a full professor and head of the Department of Computer Science at University of Neuchâtel and a researcher in the Center for Transportation Research. His research interests include parallel computing, distributed systems, communication protocols for distributed applications, ubiquitous computing, and e-commerce. He received his PhD in computer science from University of Berne. Contact him at the Institut d'informatique, Université de Neuchâtel, Faculté des sciences, Rue Emile-Argand 11 CH-2007 Neuchâtel, Suisse; peter.kropf@unine.ch.