A Guided Cooperative Parallel Tabu Search for the Capacitated Vehicle Routing Problem

Jianyong Jin [1], Teodor Gabriel Crainic [2], and Arne Løkketangen[1]

[1]Molde University College, Specialized University in Logistics, Molde, Norway , jianyong.jin@hiMolde.no, arne.lokketangen@hiMolde.no
[2]Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT), Montreal, Canada , TeodorGabriel.Crainic@cirrelt.ca

Abstract

This paper presents a guided cooperative parallel tabu search algorithm for solving the capacitated vehicle routing problem. In the proposed algorithm, knowledge is extracted from the previously found solutions and applied to guide the diversification and intensification of individual search threads. The computational experiments on two sets of large scale benchmark instances show that the proposed guidance mechanism is able to improve the effectiveness of the algorithm. The proposed algorithm has generated solutions to the benchmark instances that are competitive or better than the best solutions previously reported in the literature.

Keywords: Parallel metaheuristic; Cooperative search; Guidance mechanism; Vehicle routing.

1 Introduction

In the last decade, an increasing number of researchers in the operations research society have adopted parallel multi-thread metaheuristics for solving a variety of combinatorial problems. Such algorithms usually use several (or many) processes working simultaneously on available processors, with varying degrees of cooperation, to solve a given problem instance. It has been demonstrated that such cooperative parallel algorithms are capable of both speeding up the search and improving the robustness (ability of providing equally good solutions to a large and varied set of problem instances) and the quality of the solutions obtained (Crainic, 2008). For detailed introduction to parallel metaheuristics, we refer to the book of Alba (2005). Moreover, Le Bouthillier et al. (2005) showed that the performance of cooperative multi-thread search can be improved even further with a guidance mechanism. Thus, it is meaningful to develop an effective guidance mechanism for a multi-thread metaheuristic. Nevertheless, in the literature, there are very few contributions addressing this domain. These facts make exploring effective guidance mechanisms in a cooperative search setting a very interesting research field.

As demonstrated by Le Bouthillier et al. (2005), for a cooperative parallel search with a central solution warehouse (also called central memory), the solutions kept in

This paper was presented at the NIK-2011 conference; see http://www.nik.no/.

the warehouse can be used to extract knowledge about the search space for guiding the diversification/intensification of individual search threads. In their algorithm for solving the vehicle routing problem with time windows ($VRPTW$), the evolution of the appearance frequency of the solution attributes (arcs for $VRPTW$) in three subsets of solutions at distinct quality levels are used to identify promising or unpromising patterns (collection of arcs) to guide each individual metaheuristic toward promising or unexplored regions of the search space. This kind of guiding approach can be termed as guidance mechanisms using the central memory. On the other hand, for tabu search (TS) metaheuristics, frequency based knowledge has often been extracted at an individual tabu search thread and applied for intensifying or diversifying its search (Glover and Laguna, 1997). Such frequency based knowledge can be divided into two types, i.e., residence frequency and transition frequency. The former refers to how often a solution attribute has appeared in solutions previously generated while the latter concerns how frequently a solution attribute has been modified during the recent past. Since such frequency based knowledge is extracted locally at an individual TS thread, this kind of guiding approach can be termed as guidance mechanisms using the local memory. Here, the local memory is a term coined to denote the source from which the frequency based knowledge is extracted. Like the central memory, the local memory can hold solutions generated by an individual search thread or other search history information of a search thread. In this paper, we perform an experimental study to explore the strategies of using the knowledge extracted from either the local memory or the central memory to guide a cooperative multi-thread search containing several TS threads and examine its effectiveness for solving the capacitated vehicle routing problem ($CVRP$).

The $CVRP$, as the classical version of the vehicle routing problem (VRP), is defined on a graph $G = (N, A)$ where $N = \{0, \ldots, n\}$ is a vertex set and $A = \{(i, j) : i, j \in N\}$ is an arc set. Vertex 0 is the depot where the vehicles depart from and return to. The other vertices are the customers which have a certain demand d to be delivered (or picked up). The travel cost between customer i and j is defined by $c_{ij} > 0$. The vehicles are identical. Each vehicle has a capacity of Q. The objective is to design a least cost set of routes, all starting and ending at the depot. Each customer is visited exactly once. The total demand of all customers on a route must be within the vehicle capacity Q. Some $CVRP$ instances may have an additional route duration limit constraint, restricting the duration (or length) of any route not to exceed a preset bound D. The detailed introduction of the VRP, its solution methods and the latest advances can be found in the books of Toth and Vigo (2002), Golden et al. (2008) and the survey paper of Laporte (2009). Due to its computational difficulty and practical importance, the $CVRP$ is selected for examining the effectiveness of the proposed guided cooperative tabu search algorithm ($GCTS$).

The main contribution of this paper is the development of a parallel tabu search with a guidance mechanism that utilizes the knowledge extracted from both the local memory and the central memory to guide the global search's diversification and intensification. The computational experiments on 32 large scale $CVRP$ benchmarks demonstrate that the proposed guidance mechanism is able to enhance the performance of the algorithm. The suggested algorithm provides new best solutions to 9 benchmark instances while the rest of the results are highly competitive

with the best solutions reported in the literature.

The remainder of this paper is organized as follows. In the next section the description of the proposed algorithm is introduced. Then Section 3 reports the computational results. Finally, conclusions are presented in Section 4.

2 Description of the guided cooperative search

In this section, we describe the proposed algorithm and its main components. First we introduce the cooperative search framework, and then we present the description of the tabu search threads and the guidance mechanism.

Cooperative search framework

In the cooperative search, which is illustrated in Figure 1, four TS threads are used in parallel for solving a given problem instance. These tabu search threads use distinct neighborhood combinations and thus follow different search strategies. A solution warehouse, which can be called a solution pool or the central memory, is used to hold solutions sent by the TS threads. The search threads cooperate by exchanging their current best solutions periodically through the solution pool. Each search thread decides when to exchange solutions with the central memory according to its own search trajectory; the communication is asynchronous. No direct communication takes place between the search threads.

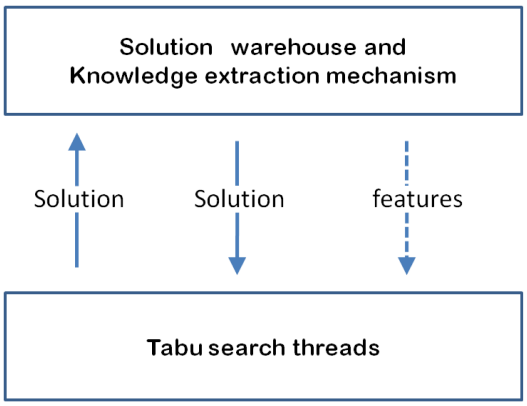

Figure 1: The framework of the cooperative search

The solution warehouse holds a central position in the cooperation mechanism. Its functions are threefold. First, it keeps the solutions sent from the TS threads and it is dynamically updated by the TS threads. The warehouse orders solutions according to the solution's objective value. Duplicate solutions are discarded. Second, it selects and sends solutions when the TS threads request a solution. The solution selection is based on the solution quality, a solution with lower objective value will be selected first. Each solution is only sent to each TS thread once. In addition, the solution warehouse will also extract knowledge from the solutions kept there and send to the TS threads when the central memory is selected to extract knowledge from for guiding the global search.

In terms of the taxonomy introduced by Crainic and Nourredine (2005) for parallel metaheuristics, this algorithm fits into the $pC/C/MPDS$ classification. The first dimension pC indicates the global search is controlled by multiple cooperative threads. The second dimension C stands for collegial information exchange and

refers to the fact multiple threads share information asynchronously. The last dimension *MPDS* indicates that multiple search threads start from different points in the solution space and follow different search strategies.

Tabu search threads

The tabu search threads included in this algorithm are based on those previously developed in (Jin et al., 2010). Each tabu search thread contains two phases. The goal of the first phase is to create a feasible starting solution for the second phase. The second phase attempts to improve the starting solution. In the second phase, distinct neighborhoods or neighborhood combinations are implemented for performing inter-route moves, they thus follow different search strategies. The neighborhoods used in the TS threads of this algorithm are listed below.

- Reinsertion (Savelsbergh, 1992) refers to moving a customer node from one position to another. The two positions can be within the same route or in two different routes.
- 2-opt (Flood, 1956) eliminates two edges and adds two new edges within the same route.
- Exchange (Osman, 1993) swaps two nodes from two routes.
- 2-opt* (Potvin and Rousseau, 1995) combines two routes so that the last customers of a given route are introduced after the first customers of another route, i.e., exchanging the head or tail part of two routes.
- CROSS-exchange (Taillard et al., 1997) swaps two route segments between two routes. In this algorithm the route segments can contain 1-3 nodes.

In the TS threads, at each iteration, first an inter-route move is performed to modify two routes, then an intra-route refinement procedure that uses reinsertion and 2-opt neighborhood alternately is triggered to improve the two modified routes. Table 1 shows the neighborhood used in each thread. When a TS thread incorporates more than one neighborhoods for inter-route moves, the neighborhoods are randomly selected according to a certain probability at each iteration.

Table 1: The neighborhoods used in each search thread

<table>
<tr><th rowspan="2">Thread</th><th colspan="2">Neighborhood used for inter-route moves</th></tr>
<tr><td>Phase2</td><td>Phase 1</td></tr>
<tr><td>Thread 1</td><td>Reinsertion, 2-opt*, CROSS-exchange.</td><td rowspan="4">Reinsertion
Exchange
2-opt*</td></tr>
<tr><td>Thread 2</td><td>2-opt*.</td></tr>
<tr><td>Thread 3</td><td>CROSS-exchange</td></tr>
<tr><td>Thread 4</td><td>Reinsertion, 2-opt*.</td></tr>
<tr><td colspan="3">All threads use reinsertion and 2-opt neighborhood for intra-route refinement</td></tr>
</table>

In addition, a granular neighborhood is adopted in these TS threads. The granular neighborhood was first introduced by Toth and Vigo (2003) for reducing the computational effort, especially for large instances by not considering some of the unpromising solution components. In this method, when considering moving a

customer node, only moves involving one member of the node's nearest neighbors set will be considered. The size of the nearest neighbors set is a parameter.

Besides, to explore the solution space more thoroughly, infeasible intermediate solutions are allowed. For this purpose, capacity and route length constraints are relaxed and their violations are penalized in the objective function. A penalty mechanism similar to the one described in Toth and Vigo (2003) is used to manage the evolvement between the feasible and infeasible search space. The detailed description of the TS threads can be found in Jin et al. (2010).

During the improvement phase, whenever a tabu search thread fails to improve its current best solution after a certain number of iterations, an information exchange procedure is activated. During the procedure, the TS thread sends its current best solution to the solution warehouse, and receives a solution from the solution warehouse and uses it as its starting solution to resume the search again. The search effort between two solution exchanges is termed a search period. When the first TS thread (Thread 1) fails to find better solutions after a certain number of periods, it terminates its search and sends a stop signal to the solution warehouse. The solution warehouse then sends a signal to other TS threads to terminate these threads. With such information exchange, each TS thread can take advantage of the best solutions or the diverse solutions other threads have generated.

Guidance mechanism

For a metaheuristic algorithm, intensification drives the search to intensively investigate the vicinity of good solutions while diversification encourages searching the unexplored search space. Glover and Laguna (1997) suggest that intensification can be realized by adding solution elements which often occur in good solutions or by dropping solution elements which often occur in poor solutions but barely in good solutions. In addition, they also advise to modify the solution elements that have less frequently been modified during the recent past for diversifying the search. In this algorithm, some of these suggested principles are adopted.

Knowledge extraction

Here the knowledge is classified in terms of two dimensions. The first dimension is the purpose, i.e., whether the knowledge is used for intensification or diversification. The other dimension is the source, i.e., whether the knowledge is to be extracted from the local memory or the central memory.

For intensification, we aim to increase the inclusion of the edges which have appeared in a certain number of best solutions found so far. The best solutions can be generated by one TS thread and kept at the local memory or are generated by all TS threads and held at the central memory. After selecting the best solutions, the appearance frequency of each edge in the selected solutions is counted. The selected best solutions and the appearance frequency are updated every search period.

For diversification, we attempt to increase the frequency of modifying the positions of the customer nodes that have less frequently been moved during the search so far. To this end, the node transition frequency is extracted from a transition recorder (the local memory) that counts the times each node has been relocated in the inter-route reinsertion moves at a TS thread. In order to avoid obtaining diverse solutions whose objective values are very poor, we only consider those less frequently moved nodes that are also connected to their distant neighbors

with long edges since long edges are less likely going to appear in high quality solutions and to destroy long edges may also improve the solution quality.

Organization of intensification and diversification

From the literature, there exist different ways of using frequency based knowledge. One approach is to add a frequency-based penalty or incentive to the move evaluation function (Glover and Laguna, 1997). Another method is to prohibit or fix some frequency based patterns (Le Bouthillier et al., 2005). The first approach is adopted in this algorithm so that incentives can be attached to the edges that have appeared in the best solutions previously found according to the appearance frequency, for the purpose of increasing the possibility of including these edges in the solutions. The formula for calculating the incentive is shown in equation 1.

$$Incentive = \frac{IC \times \Delta obj \times \sum Fre}{NumSol \times NE} \tag{1}$$

The components are defined as follows.

- IC: the incentive coefficient (negative number) is a parameter.
- Δobj: the average of the absolute solution's objective value change during last η iterations. η is a parameter.
- $\sum Fre$: the sum of the appearance frequency in the selected best solutions of the edges added in a move.
- $NumSol$: the number of best solutions selected.
- NE: the number of new edges added to the solutions in a move. In a move, reinsertion neighborhood changes 3 edges, 2-opt* changes 2 edges while CROSS-exchange changes 4 edges.

For diversification, a destroy-repair procedure is implemented to diversify the search. In the procedure, first, find a certain number of customer nodes that both are connected to their distant neighbors and have less frequently been moved during the search so far. Second, find which routes these nodes belong to. After that, remove a certain proportion of nodes in the middle part of these routes. At last, reinsert the nodes removed one by one to a position next to one of its nearest neighbors so as to cause the least route length increase. After destroy-repair, the new solution is improved by a TS procedure. The reason why the middle part of the routes is selected to be destroyed and repaired is twofold. First, these nodes are often less frequently moved according to the observation in initial preliminary experiments. Second, to remove and reinsert some consecutive nodes can change the solutions more thoroughly.

Unlike sequential algorithms, different threads are designated to perform intensification and diversification separately in this algorithm. The intensification procedure is implemented in Thread 1 while diversification is performed by Thread 4. The rationale behind this decision is twofold. First, the initial preliminary experiments showed that applying the suggested guidance mechanism to the other two TS threads did not bring about additional improvement. In addition, it makes the algorithm simpler.

At Thread 1, the intensification procedure is activated when the thread can not improve its best found solution after a certain number of periods since it is not so profitable to investigate the vicinity of the solutions generated at the beginning of the search. The destroy-repair procedure is triggered from the beginning of the second phase at Thread 4. During each period, the destroy-repair procedure is executed first, the modified solution is then improved by the normal TS procedure.

3 Computational results

In this section we describe the experimental platform, the test data sets, the algorithm configurations and compare the experimental results against the best results reported in the literature.

Experimental platform and implementation issues

The proposed algorithm is implemented in C++ and uses the message passing interface (*MPI*) for the inter-processor information exchange. The results were obtained by running the algorithm on a compute cluster in which each node consists of two Intel® Xeon 2.66 Ghz quad-core processors. We ran the four TS threads on each of four cores, the solution warehouse is assigned to a fifth core.

The test data sets

The computational tests were carried out on the CVRP benchmarks of Golden et al. (1998) and Li et al. (2005). The 20 benchmark instances of Golden et al. (1998) have 200 to 483 customers. The first eight instances also have route length restrictions. Each instance is based on a simple geometric structure: eight instances have customers located in concentric circles around the depot, four instances have customers located in concentric squares with the depot located in one corner, four instances have customers located in concentric squares around the depot, and four instances have customers located in a six-pointed star around the depot. The benchmark instances of Li et al. (2005) have 560 to 1200 customers and route length restrictions, and their geometric structure is based on concentric circles around the depot. For each instance, the algorithm was executed 10 times with different random seeds and both the best and the average results are reported.

The algorithm configurations

The values of the parameters in the proposed algorithm are set according to the computational results of the initial preliminary experiments on the instances 1,4,5,8,9,12,13,16,17,20 of Golden et al. (1998). These instances are selected for parameter tuning because they differ in both instance size and geometric structure of customer location. The values of the parameters are summarized in Appendix A.

To demonstrate and compare the performance of the algorithm, the final experiments were executed under four different scenarios.

- BCTS: The algorithm was executed without the intensification and diversification procedure activated. This scenario examines the performance of the algorithm without the guidance mechanism and is named as the basic cooperative TS.

- LGCTS: The algorithm was executed with the intensification and diversification procedure activated. The intensification procedure extracted knowledge from the solutions generated by Thread 1. This scenario is to examine the performance of the guided cooperative TS using only the local memory.

- CGCTS: The algorithm was executed with the intensification and diversification procedure activated. The intensification procedure extracted knowledge from the solutions kept in the central memory. This scenario is to examine the performance of the guided cooperative TS using both the local and the central memory.

- CGCTSL: The algorithm was executed for 48 hours under the scenario CGCTS. This scenario is to examine the performance of the guided cooperative TS using both the local and the central memory with longer running time.

In next two subsections, the computational results of these scenarios are presented and compared against the best results reported in the literature. The best known results are collected from Groër et al. (2011). The current best known solutions have been updated to include the new best solutions identified by their algorithm. Our results are also compared against two state-of-art metaheuristics recently published.

Results for the benchmarks of Golden et al. (1998)

In Table 2 we compare the results for the 20 benchmark instances of Golden et al. (1998). In the table, the first column describes the instances (instance number and number of nodes). The second column lists the best known solutions previously reported in the literature. The third and fourth column provide the best results presented by Mester and Bräysy (2007) and Groër et al. (2011). The remaining columns give the average and best results of the proposed algorithm under the four scenarios. The second last row presents the average deviation of all instances from the best known solutions. The last row provides the average wall-clock time of all instances per run. From the table, we can see the performance of the proposed algorithm under the four scenarios, both the average and the best solutions, consistently improves from BCTS to LGCTS, CGCTS and CGCTSL. This fact proves that the proposed guidance mechanism is able to enhance the performance of the algorithm. New best solutions to six instances (numbers in bold font) have been identified. The average deviation of the best solutions identified by BCTS, LGCTS CGCTS and CGCTSL from the best known solutions is 0.35%, 0.22%, 0.13% and 0.04% respectively. In terms of this metric, the best results generated by LGCTS, CGCTS and CGCTSL are better than Mester and Bräysy (2007), the best results of CGCTSL match the quality of the solutions of Groër et al. (2011). In particular, the suggested algorithm has found better solutions than Mester and Bräysy (2007) and Groër et al. (2011) for most of the instances with both vehicle capacity and route length constraints (Instances 1-8).

Results for the benchmarks of Li et al. (2005)

The results for the 12 benchmark instances of Li et al. (2005) are presented in Table 3. The structure of this table is identical to the one of Table 2. For this set of instances, the proposed algorithm found new best solutions to three instances

Table 2: Comparison of results for the benchmarks of Golden et al. (1998)

Instances	Previous best known	Mester and Bräysy (2007)	Groër et al. (2011) 129pc	BCTS aver.	BCTS best	LGCTS aver.	LGCTS best	CGCTS aver.	CGCTS best	CGCTSL aver.	CGCTSL best
1(240)	5623.47	5627.54	5623.47	5630.69	5625.74	5629.11	5623.47	5628.72	5623.47	5625.82	5623.47
2(320)	8431.66	8447.92	8435.00	8467.04	8452.21	8452.66	**8430.33**	8445.21	**8428.32**	8424.69	**8412.83**
3(400)	11036.22	11036.22	11036.22	11074.43	11043.40	11065.99	11043.40	11061.14	11041.00	11038.44	11036.22
4(480)	13592.88	13624.52	13624.52	13724.49	13693.10	13705.75	13630.50	13691.31	13632.90	13633.61	13602.90
5(200)	6460.98	6460.98	6460.98	6520.60	6460.98	6461.55	6460.98	6462.12	6460.98	6460.98	6460.98
6(280)	8404.26	8412.88	8412.90	8425.98	8414.28	8423.49	8413.82	8414.24	8413.82	8409.56	**8400.33**
7(360)	10156.58	10195.56	10195.59	10227.08	10191.00	10221.50	10186.70	10187.65	**10154.00**	10157.00	**10141.10**
8(440)	11643.90	11663.55	11649.89	11745.65	11710.30	11674.98	**11635.30**	11668.26	**11637.10**	11637.98	**11635.30**
9(255)	579.71	583.39	579.71	583.72	580.60	582.75	581.48	582.59	580.26	580.90	580.17
10(323)	737.28	741.56	737.28	742.06	739.36	740.59	737.41	740.02	**737.18**	738.29	**737.18**
11(399)	913.35	918.45	913.35	918.76	916.25	917.42	914.28	917.45	915.18	914.79	**913.30**
12(483)	1102.76	1107.19	1102.76	1112.15	1108.39	1110.99	1107.88	1110.68	1105.17	1107.39	1105.17
13(252)	857.19	859.11	857.19	863.62	860.30	863.70	861.18	861.37	857.19	858.68	857.19
14(320)	1080.55	1081.31	1080.55	1087.68	1085.17	1085.35	1081.44	1083.21	1080.55	1081.40	1080.55
15(396)	1338.00	1345.23	1338.00	1352.06	1348.51	1351.89	1345.12	1350.50	1345.95	1343.12	1341.50
16(480)	1613.66	1622.69	1613.66	1632.60	1626.47	1629.25	1625.83	1627.86	1621.27	1620.78	1617.97
17(240)	707.76	707.79	707.76	708.56	707.85	708.33	707.85	708.11	707.76	707.79	707.76
18(300)	995.13	998.73	995.13	1001.70	1000.00	1000.27	999.34	1000.44	999.20	998.62	998.01
19(360)	1365.60	1366.86	1365.60	1369.51	1368.04	1369.47	1367.78	1368.74	1366.94	1366.69	1366.00
20(420)	1818.25	1820.09	1818.25	1831.35	1827.63	1829.70	1826.29	1826.39	1820.54	1821.66	1819.58
Aver. deviation %		0.26	0.04	0.64	0.35	0.47	0.22	0.38	0.13	0.15	0.04
Aver. time min		24.40	5.00	38.34		45.80		47.07		2880	

(numbers in bold font). The average deviation of the best solutions found by BCTS, LGCTS CGCTS and CGCTSL from the best known solutions is 0.14%, 0.11%, 0.11% and 0.05% respectively. This trend also shows the effectiveness of the guidance mechanism. In terms of this metric, the best results of all scenarios are better than Mester and Bräysy (2007), the best results of CGCTSL are slightly better than the outcomes of Groër et al. (2011).

Table 3: Comparison of results for the benchmarks of Li et al. (2005)

Instances	Previous best known	Mester and Bräysy (2007)	Groër et al. (2011) 129pc	BCTS aver.	BCTS best	LGCTS aver.	LGCTS best	CGCTS aver.	CGCTS best	CGCTSL aver.	CGCTSL best
21(560)	16212.74	16212.74	16212.83	16282.73	16223.60	16258.77	16222.20	16243.41	16220.00	16252.10	16214.00
22(600)	14584.42	14597.18	14584.42	14619.29	14606.60	14625.40	14613.30	14610.99	**14581.20**	14585.75	**14575.19**
23(640)	18801.12	18801.12	18801.13	18881.13	18809.50	18876.05	18811.90	18866.40	18810.70	18825.20	18802.30
24(720)	21389.33	21389.33	21389.43	21420.38	21400.20	21413.17	21397.80	21408.20	21396.60	21395.88	21390.60
25(760)	16763.72	17095.27	16763.72	16852.45	16814.70	16849.43	**16746.30**	16855.67	16804.20	16776.97	**16739.84**
26(800)	23971.74	23971.74	23977.73	24137.33	23987.30	23996.96	23984.90	23993.50	23986.10	23985.02	23982.50
27(840)	17433.69	17488.74	17433.69	17509.74	17448.6	17498.96	**17423.90**	17506.42	**17430.20**	17453.75	**17408.66**
28(880)	26565.92	26565.92	26566.03	26583.29	26574.40	26600.77	26574.40	26587.07	26575.60	26574.64	26573.20
29(960)	29154.34	29160.33	29154.34	29170.96	29159.10	29172.93	29162.30	19168.39	29161.50	29162.82	29161.50
30(1040)	31742.51	31742.51	31742.64	31765.90	31752.20	31764.94	31752.20	31760.66	31752.20	31754.76	31749.80
31(1120)	34330.84	34330.84	34330.94	34346.53	34340.50	34348.88	34340.50	34347.26	34340.50	34340.70	34339.30
32(1200)	36919.24	36928.70	37185.85	37296.97	37235.60	37300.17	37254.50	37276.22	37240.00	37337.09	37220.00
Aver. deviation %		0.20	0.06	0.35	0.14	0.28	0.11	0.26	0.11	0.16	0.05
Aver. time min		104.30	5.00	145.10		151.60		160.10		2880	

4 Conclusions and Perspectives

In this paper, we have presented a guided cooperative parallel tabu search algorithm for the capacitated vehicle routing problem. The computational experiments on the two sets of large scale CVRP benchmarks from the literature show that the suggested metaheuristic is effective and competitive in comparison to the best

heuristic solution methods from the literature. Moreover, the comparison among the designed scenarios demonstrates the presented guidance mechanism is able to further enhance the performance of the algorithm. The experiments also show that the guidance mechanism using both the local and the central memory outperforms the one using only the local memory since the quality of the solutions kept in the central memory is often better.

The reason the proposed method is capable of identifying high quality solutions can be attributed to two aspects. First, the proposed algorithm exploits the cooperative power of several tabu search threads which utilize different neighborhood combinations. Even without the guidance mechanism, it can generate good solutions to the benchmarks. Moreover, the suggested guidance mechanism which extracts knowledge from both the local and the central memory appears to be able to direct the TS threads to explore the search space in a more effective way.

The benchmarks used have been well-studied, even minute improvements are difficult to obtain. Our main comparisons are using normal computing power. The very long time tests show, as is often the case, that the algorithm is able to improve the results a little by using much more computing power.

Future work will focus on pursuing more sophisticated diversification approaches for the guidance mechanism so as to enhance both the effectiveness and the efficiency of a cooperative parallel search.

Acknowledgements

The authors thank Compute Canada and the Norwegian Metasenter for Computational Science (NOTUR) for providing computing resources to conduct the experiments of this research.

References

E. Alba, editor. *Parallel Metaheuristics: A New Class of Algorithms*. John Willey & Sons, Hoboken, NJ, 2005.

T.G. Crainic. Parallel solution methods for vehicle routing problems. In B. Golden, S. Raghavan, and E. Wasil, editors, *The Vehicle Routing Problem: Latest Advances and New Challenges*, pages 171–198, New York, 2008. Springer.

T.G. Crainic and H. Nourredine. Parallel metaheuristics applications. In E. Alba, editor, *Parallel Metaheuristics*, pages 447–494, Hoboken, NJ, 2005. John Willey & Sons.

M. M. Flood. The traveling-salesman problem. *Operations Research*, 4:61–75, 1956.

F. Glover and M. Laguna. *Tabu Search*. Kluwer, 1997.

B.L. Golden, E.A. Wasil, J.P. Kelly, and I.-M. Chao. The impact of metaheuristics on solving the vehicle routing problem: algorithms, problem sets, and computational results. In T. Crainic and G. Laporte, editors, *Fleet management and logistics*, pages 33–56, Boston, 1998. Kluwer.

B.L. Golden, S. Raghavan, and E.A. Wasil. *The Vehicle Routing Problem: Latest Advances and New Challenges*. New York, 2008. Springer.

C. Groër, B.L. Golden, and E.A. Wasil. A parallel algorithm for the vehicle routing problems. *INFORMS Journal on Computing*, 23:315–330, 2011.

J. Jin, T.G. Crainic, and A. Løkketangen. A parallel multi-neighborhood cooperative tabu search for capacitated vehicle routing problem. *Publication CIRRELT-2010-54*, Centre interuniversitaire de recherche sur les réseaux d'entreprise, la logistique et le transport, Université de Montréal., 2010.

G. Laporte. Fifty years of vehicle routing. *Transportation Science*, 43(4):408–416, 2009.

A. Le Bouthillier, T.G. Crainic, and P. Kropf. A guided cooperative search for the vehicle routing problem with time windows. *IEEE Intelligent Systems*, 20(4): 36–42, 2005.

F. Li, B.L. Golden, and E.A. Wasil. Very large-scale vehicle routing: New test problems, algorithms, and results. *Computers & Operations Research*, 32:1165–1179, 2005.

D. Mester and O. Bräysy. Active-guided evolution strategies for large-scale capacitated vehicle routing problems. *Computers & Operations Research*, 34: 2964–2975, 2007.

I. H. Osman. Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problems. *Annals of Operations Research*, 54:421–452, 1993.

J-Y. Potvin and J-M. Rousseau. An exchange heuristic for routing problems with time windows. *Journal of the Operational Research Society*, 46:1433–1446, 1995.

M. W. P. Savelsbergh. The vehicle routing problem with time windows: minimizing route duration. *INFORMS Journal on Computing*, 4:146–154, 1992.

E. Taillard, P. Badeau, M. Gendreau, F. Geurtin, and J-Y. Potvin. A tabu search heuristic for the vehicle routing problem with soft time windows. *Transportation Science*, 31:170–186, 1997.

P. Toth and D. Vigo. *The vehicle routing problem*. SIAM Monographs on Discrete Mathematics and Applications, Philadelphia, 2002. PA.

P. Toth and D. Vigo. The granular tabu search and its application to the vehicle routing problem. *INFORMS Journal on Computing*, 15:333–346, 2003.

A Search parameters and their values

The following table provides the parameters of the algorithm and their values.

Table 4: The parameters of the proposed algorithm

Parameters	**Value**
Tabu tenure	Reinsertion neighborhood: $0.07\|N\|$ Exchange neighborhood: $0.07\|N\|$ 2-opt* neighborhood: $0.135\|N\|$ CROSS-exchange neighborhood: $0.07\|N\|$
Neighborhood selection probability at Thread 1	Reinsertion neighborhood: $^5/_7$ 2-opt* neighborhood :$^1/_7$ CROSS-exchange neighborhood:$^1/_7$
Neighborhood selection probability at Thread 4	Reinsertion neighborhood: $^6/_7$ 2-opt* neighborhood :$^1/_7$
Nearest neighbors set size	Phase 1: 24 Phase 2: (10 + random [0, 10])
Time to trigger solution exchange	when a thread can not improve its best solution after $200 \times \sqrt{\|N\|}$ iterations.
termination criterion	when Thread 1 fails to improve its best found solution after 15 periods.
Intensification incentive coefficient	-2.0
Number of elite solutions for guiding intensification	2
Time to trigger intensification	when Thread 1 fails to improve its best solution for 5 consecutive periods
Iterations for calculating solution value change	10
Number of nodes selected for identifying the candidate routes for destroy-repair	the number of routes in a solution
Number of nodes removed and reinserted in a route	(0.3× the number of nodes in the route)

$|N|$ represents the instance size.