Theoretical and Applied Informatics
ISSN 1896–5334
Vol.24 (2012), no. 3
pp. 191–203
DOI: 10.2478/v10179-012-0012-5

Adaptive threads co-operation schemes in a parallel heuristic algorithm for the vehicle routing problem with time windows

JAKUB NALEPA, ZBIGNIEW J. CZECH

Institute of Informatics, Silesian University of Technology
ul. Akademicka 16, 44-100 Gliwice, Poland
{Jakub.Nalepa,Zbigniew.Czech}@polsl.pl

Received 30 May 2012, Revised 10 September 2012, Accepted 17 September 2012.

Abstract: The influence of the co-operation frequency of threads in a parallel heuristic algorithm to solve the vehicle routing problem with time windows on the accuracy of solutions is investigated. The accuracy of solutions is defined as their proximity to the best known solutions of Gehring and Homberger's benchmarking tests. Two adaptive co-operation schemes are proposed and experimentally evaluated.

Keywords: VRPTW, parallel algorithm, co-operation frequency, OpenMP interface

1. Introduction

The vehicle routing problem with time windows (VRPTW) is an NP-hard discrete optimization problem. It consists in finding a schedule for a fleet of homogenous vehicles servicing a set of geographically scattered customers. The fleet is composed of the vehicles with equal capacities which cannot be exceeded. All customers must be visited during the time windows to ensure the feasibility of the routing plan. The VRPTW is considered as a hierarchical optimization problem. The first objective is to minimize the total fleet size and the second one is to minimize the total traveled distance.

The practical applications of the VRPTW include the bus route planning, post and parcels delivering, food delivering, cash delivering to banks and ATM terminals and many more. A number of algorithms including exact and heuristic methods were introduced for the VRPTW. Due to the large complexity of the VRPTW and its wide practical applicability the heuristic and metaheuristic methods capable of producing high-quality feasible solutions in reasonable time are of main importance.

The exact algorithms incorporating, among others, dynamic programming, branch-and-bound algorithms and greedy approaches, were proposed by Bard et al. [1], Irnich and Villeneuve [14], Jepsen et al. [15], Kellehauge et al. [16] and Chabrier [5]. It is worth noting that only nine instances out of 300 belonging to the Gehring and Homberger's benchmark [13] have been solved to optimality [22].

The heuristic algorithms can be divided into the improvement and the construction methods. In the construction heuristic algorithms the customers are iteratively inserted into a partial solution without violating the time windows and capacity constraints to build a feasible solution. Several construction heuristics were proposed by Solomon [30], Potvin and Rousseau [25] and recently by Pang [24]. The improvement heuristics modify an initial solution and explore the search space S by performing a number of local search moves in order to decrease the fleet size and the total distance. The examples of such heuristics can be found in Thompson et al. [32], Russell [28] and Potvin and Rousseau [26].

The metaheuristic algorithms incorporate mechanisms to explore the search space and to exploit the most promising regions of the search space. The metaheuristics allow infeasible intermediate solutions and the solutions deteriorating during the search process in order to escape the local minima. A number of sequential and parallel algorithms were introduced during the recent years. The simulated annealing was successfully applied by Zhong and Pan [34], Debudaj-Grabysz and Czech [8] and Skinderowicz [29]. The tabu searches were proposed by Cordeau et al. [7] and Sin et al. [11]. The ant colony approaches can be found in Xuan et al. [31] and Qi and Sun [27]. The genetic algorithms were successfully applied by Cheng and Wang [6], Kamkar et al. [17] and Ursani et al. [33]. The evolution strategies were proposed by Gehring and Homberger [10,12], Mester et al. [20] and Kanoh et al. [18]. The sequential and parallel memetic algorithms (MAs) combining the evolutionary algorithms for more distant search with the local optimization algorithms for local exploitation were described by Berger and Barkaoui [2], Labadi et al. [19] and Nagata and Bräysy [22].

In this work we analyze a parallel heuristic algorithm to minimize the number of routes in the VRPTW. The algorithm is based on the improved sequential heuristics proposed by Nagata and Bräysy [21]. The objective of the work is to investigate the influence of the threads co-operation frequency on the accuracy of solutions. The adaptive co-operation frequency adjustment schemes are proposed. The work extends our previous research [23] and Błocho and Czech's improvements of the sequential algorithm [3].

This paper is composed of six sections. Section 2 formulates the VRPTW. The parallel heuristic algorithm and the co-operation of threads are discussed in Section 3. Section 4 describes the co-operation schemes. The experimental results are discussed in Section 5. Section 6 concludes the paper.

2. Problem formulation

Let $G = (V, E)$ be a directed graph with a set V of $N + 1$ vertices representing the customers and the depot, together with a set of edges E. The vertex v_0 represents the depot which is the start and the finish point of each route. The travel costs between each pair of travel points are given as $c_{i,j}$, where $i \neq j$, $i, j \in \{0, 1, ..., N\}$. The non-negative demands d_i, $i \in \{1, 2, ..., N\}$, are known for each customer. Each customer and the depot define their time windows $[e_i, l_i]$, $i \in \{0, 1, ..., N\}$, during which the service must be started. The customers have their own service times s_i, $i \in \{1, 2, ..., N\}$. There is a set of vehicles of size K with a constant capacity Q providing the service. The route is defined as a set of customers serviced by a single vehicle $(v_0, v_1, ..., v_{n+1})$, where $v_0 = v_{n+1}$. The total amount of goods delivered to the customers within a single route cannot exceed the maximal vehicle capacity, and the service of a customer should be started before the time window elapses.

The primary objective of the VRPTW is to minimize the total fleet size, i.e. the number of vehicles K servicing the customers. The secondary objective is to minimize the total travel distance, or the travel cost in general, covered by the vehicles.

3. Parallel heuristic algorithm

The parallel algorithm consists of p threads denoted as $P_0, P_1, ..., P_{p-1}$. The initial solution σ_{init} is constructed (Fig. 1, line 2) for each thread in the team. The number of routes in a feasible solution is reduced by one at a time until either the maximal execution time is exceeded or the minimal number of routes $K_{min} = \lceil \frac{\sum_{i=1}^{N} d_i}{Q} \rceil$ is obtained (line 29). A random route is selected and the V customers removed from the route are inserted into the ejection pool (EP) (lines 4-5). The penalty counters $p[i]$, where $i \in \{1, 2, ..., N\}$, indicating the re-insertion difficulty are reset (line 6).

The customers residing in the EP must be reinserted into the partial solution without violating the time windows and the capacity constraints. A single customer v is taken from the EP in each iteration (line 8), and a set of feasible insertion positions $N(v, \sigma)$ is constructed. If the set is not empty then a random insertion position forming a new solution σ' is selected (line 10). Otherwise, the next attempt to re-insert v is performed with the *Squeeze* method allowing for the temporary infeasible solutions (line 12). The infeasible solutions are sorted according to the value of the penalty function $F_p(\sigma)$ [21] and the solution with the minimal value is selected. A number of local search moves are performed in order to restore the feasibility of the solution. If the squeezing fails then the penalty counter $p[v]$ of the customer v is increased (line 15) and the ejections of other customers from the partial solution are tested (up to the limit k_{max} [3]) to form the set of partial solutions with the ejected customers $N_e(v, \sigma)$. The sum of the penalty

1: **parfor** P_i, $i = 0, 1, ..., p-1$ **do**
2: Create an initial solution σ_{init};
3: **while not** *solutionReady* **do**
4: Select and remove a random route from σ;
5: Initialize EP with a random permutation of V removed customers;
6: Initialize the penalty counters $p[i] := 1$, $i \in \{1, 2, ..., N\}$;
7: **while** $EP \neq \emptyset$ **and not** *isTerminated* **do**
8: Select and remove customer v from the EP;
9: **if** $N(v, \sigma) \neq \emptyset$ **then**
10: $\sigma := \sigma'$ selected randomly from $N(v, \sigma)$;
11: **else**
12: σ := *Squeeze*(v, σ);
13: **end if**
14: **if** v is not in σ **then**
15: $p[v] := p[v] + 1$; {increasing the penalty counter}
16: Select $\sigma' \in N_e(v, \sigma)$ such that P_{sum} is minimal;
17: $\sigma := \sigma'$;
18: Insert the ejected customers $\{v_{out}^{(1)}, ..., v_{out}^{(k)}\}$ into EP;
19: σ := *Perturb*(σ);
20: **end if**
21: *isTerminated* := *CheckTerminationConditions*(σ);
22: **end while**
23: **if** $EP \neq \emptyset$ **then**
24: Restore σ to the previous feasible solution;
25: **end if**
26: **if** *canCooperate* **then**
27: Call Co-operation procedure;
28: **end if**
29: *solutionReady* := *CheckIfSolutionIsReady*(σ);
30: **end while**
31: **end parfor**

Fig. 1. A parallel heuristic route minimization algorithm

counters of the ejected customers P_{sum} is calculated and taken into account while choosing the optimal insertion-ejections positions and selecting a new partial solution σ' from $N_e(v, \sigma)$ (line 16). The P_{sum} value should be minimized for ejecting the customers which are relatively easy to re-insert to the partial solution. The removed customers are inserted into the EP and the partial solution is perturbed with a number of feasible local search moves (line 19) to avoid getting stuck in a local minimum. The main loop of the

algorithm (lines 7-22) executes until either the customers are re-inserted into the partial solution or the termination condition is met (line 21) [23].

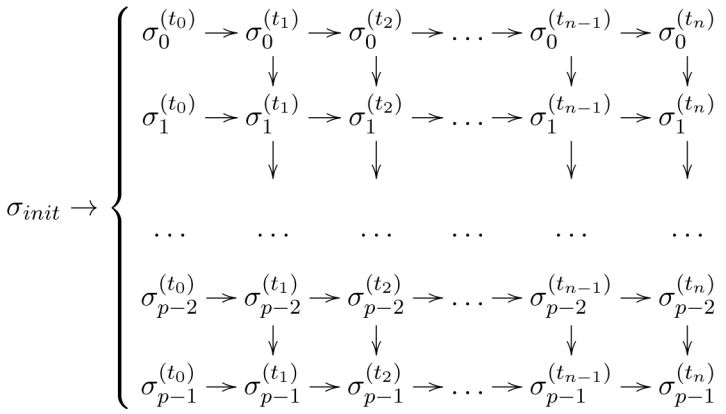

Fig. 2. Co-operation scheme

Threads co-operate periodically to exchange the best solutions found up-to-date (line 27). The solutions are assessed according to their costs. The solutions with the smaller number of routes *K* are preferred. If the number of routes is equal then the solution with the shorter total travel distance is considered better. The co-operation scheme among the threads is given in Fig. 2. The co-operation starts from thread P_0. Thread P_1 receives the solution σ_0 from thread P_0 and compares the costs of solution σ_1 with the received one. The solution with the smaller cost is chosen and replaces the current solution of thread P_1. Thread P_1 sends the updated solution to thread P_2. Similarly, thread P_{p-1} compares the solution σ_{p-1} with σ_{p-2} received from thread P_{p-2}. Finally, the best solution is held by thread P_{p-1}. It may be noticed that during the co-operation the best solution propagates towards thread P_{p-1}. If the best solution is found by thread P_0 then all the threads get σ_0. The probability of such situation depends on the number of co-operating threads.

The number of steps that are executed in parallel by the working threads before the co-operation must be determined sensitively. If the threads co-operate frequently then the search is guided towards the optima, however the parallel overhead becomes significant. Additionally, the set of solutions may become saturated with the individuals of similar characteristics, which in turn leads to getting stuck in the local minimum. The rare co-operation in case of problems containing relatively small number of customers reduces the parallel overhead and keeps the search diversified.

4. Adaptive schemes of co-operation

Four threads co-operation schemes are considered. In the frequent co-operation the threads exchange the best solutions every $N/10$ steps, whereas in the rare co-operation every $N/4$ steps, where N is the problem size, i.e. the number of customers to be serviced. The frequency of co-operation is constant during the algorithm's execution.

In case of the adaptive co-operation, the frequency f is defined as a ratio of N and the constant initial co-operation factor $c_i = 4$ at the beginning of the algorithm's execution. After f steps the frequency is divided by $c_d = 2$ until it reaches a defined minimal number of parallel steps between the co-operation, i.e. $f_{min} = 20$.

In the time-adaptive co-operation the initial frequency f is set analogously to the adaptive co-operation scheme. After f steps it is divided by a ratio of the last average threads execution time $\bar{t}$ and the previous average time $\bar{t}_{prev}$, if $\bar{t}_{prev} \neq 0$. The frequency f is updated until it reaches f_{min}.

5. Experimental results

The parallel heuristic algorithm was implemented in C++ language using the OpenMP interface and was tested on selected Gehring and Homberger's benchmarking tests. The set contains 300 problem instances in total with various sizes and structures [13]. The code was compiled using Intel C++ Compiler 10.1.015 with `-fast` and `-openmp` flags. The computations were carried out on a single node of Galera supercomputer at the Academic Computer Center in Gdańsk [9]. The computations were performed on the nodes with 16 GB RAM (2 GB/core) equipped with Intel Xeon Quad Core (2.33 GHz) processors (8 cores/node) with 12 MB of level 3 cache.

All the Gehring and Homberger's instances are split into different problem classes, containing customers grouped into clusters (C-class), dispersed randomly on the map (R-class), and containing both clustered and randomized customers (RC-class). Among these classes, there are tests with smaller vehicle capacities and considerably short time windows (C1, R1, RC1), and the problems with larger vehicle capacities and longer scheduling horizon (C2, R2, RC2). A larger number of vehicles will service the customers in the first subclass.

The computations were performed on the R1_10_10, R2_4_2, C1_2_7, C2_4_8, RC1_6_3, RC2_8_8 tests to investigate the influence of the threads co-operation frequency on the instances from each class and of each size. A number of improvements have been proposed in [23] and [3] concerning the neighborhood size, solutions perturbation, additional termination conditions and more. The parameters used during the experiments are described and discussed in [23].

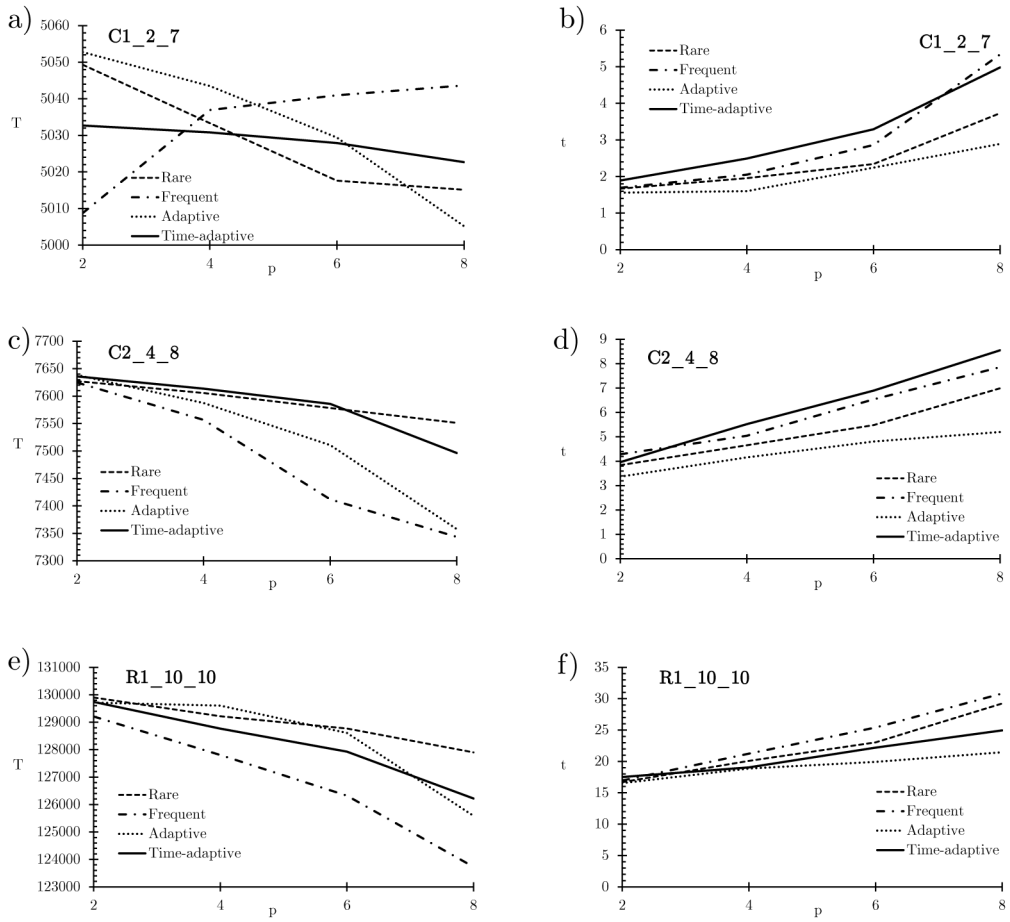

Fig. 3. Total travel distance T and average execution time t (in seconds) vs. number of threads p for tests C1_2_7 (a, b), C2_4_8 (c, d) and R1_10_10 (e, f)

The experimental results are given in Fig. 3 and Fig. 4. For each instance 500 experiments were performed to investigate the average traveled distance T and the average execution time t of the algorithm (given in seconds). The fleet sizes in the presented solutions are equal to the world's best results reported in [13].

It can be seen that increasing the number of co-operating threads for the instance with a small number of customers C1_2_7 (Fig. 3(a, b)) may lead to decreasing the accuracy of solutions if they co-operate frequently. The threads are guided to the local minima in the search space S which are difficult to avoid if the customers are grouped into clusters. If the threads co-operate frequently then the execution time increases and the threads work on a set of similar (close to locally optimal) solutions which are hard to improve

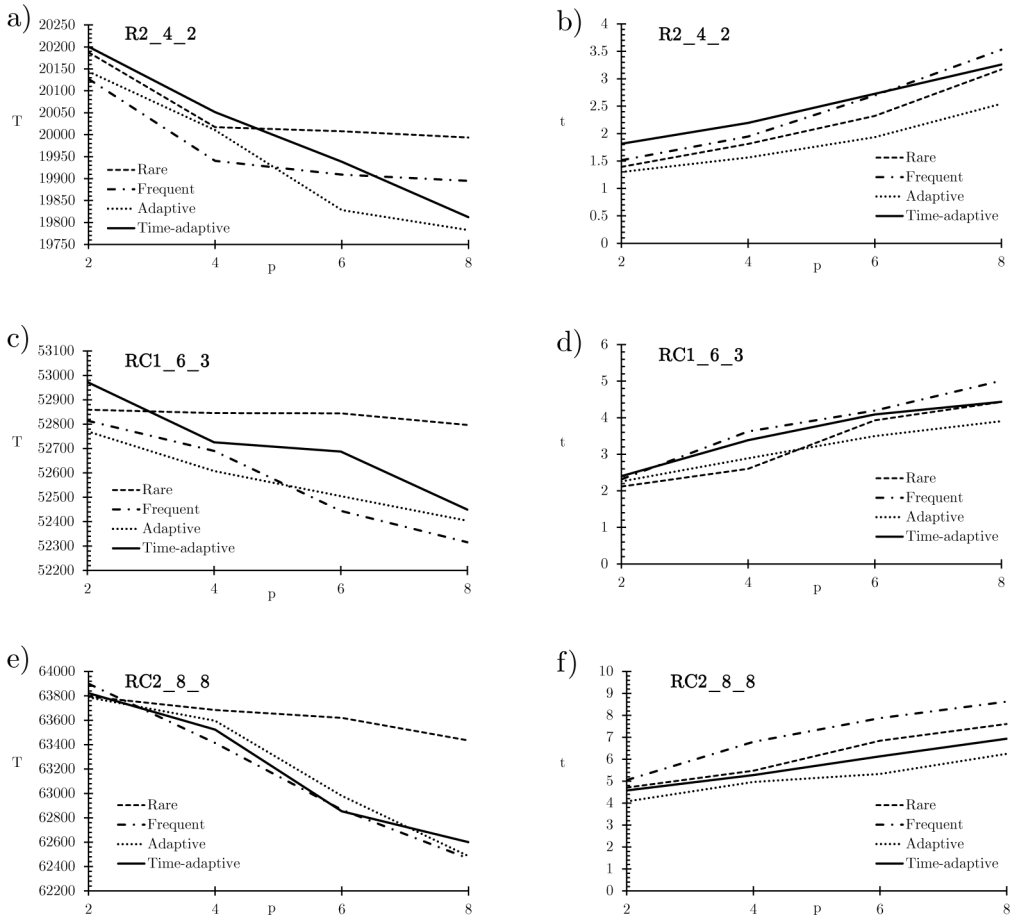

Fig. 4. Total travel distance T and average execution time t (in seconds) vs. number of threads p for tests R2_4_2 (a, b), RC1_6_3 (c, d) and RC2_8_8 (e, f)

further. The experiments for a larger test C2_4_8 (Fig. 3(c, d)) show that the frequent co-operation results in a significant decrease of the traveled distance. In the time-adaptive scheme the co-operation frequency depends on the current state of search. If the threads get stuck then the co-operation becomes frequent and the probability of leaving a locally optimal region of the search space increases. In case of the instances with randomly scattered customers (R1_10_10 and R2_4_2, Fig. 3(e, f) and Fig. 4(a, b)) the larger co-operation frequencies lead to the solutions of better accuracy. The search space in case of R1_10_10 test is very large, thus the guided threads tend to explore different regions of space S. Tests from the RC-class have the characteristics of instances containing clusters of customers and random travel points. It can be seen that the frequent co-operations give the better results for both RC1_6_3 (Fig. 4(c, d)) and RC2_8_8 (Fig. 3(e, f)) tests.

During the co-operation the solutions are propagated in the team and the threads work on improving the solution characteristics and leaving the local optima if necessary which is especially important for the semi-clustered instances. A route servicing the customers from e.g. two neighboring clusters may be faster changed into a route servicing the customers belonging to a single cluster.

Increasing the number of working threads should either result in decreasing the execution time of the parallel algorithm or in increasing the accuracy of solutions obtained in a comparable (ideally constant) time. However, the parallel overhead may become significant and lead to the unacceptable increase of the execution time in case of frequent threads co-operation. It is worth noting that the proposed adaptive co-operation schemes significantly reduce the overhead imposed by the threads communication and synchronization for larger problem instances. The time-adaptive scheme introduces additional overhead of e.g. calculating the average threads execution time which turned to be expensive for relatively small tests (containing 200 and 400 customers). The experiments show that increasing the co-operation frequency results in enhancing the accuracy of the solutions in most cases. The accuracy of the solutions found using the adaptive co-operation schemes is similar to the accuracy obtained using the frequent co-operation. The adaptive co-operations outperformed the frequent co-operation for C1_2_7 and R2_4_2 tests.

6. Conclusions

The available processors may be either used to speed up the computations or to improve the accuracy of solutions found by a parallel algorithm. In the presented parallel heuristic algorithm threads co-operate periodically to guide the search towards the solutions of higher accuracy. The experimental results showed that increasing the co-operation frequency leads to obtaining the solutions of higher accuracy in most cases. However, the larger frequency for easier tests (e.g. C1_2_7) may lead to the saturation of the family of solutions with the locally optimal individuals. Our experiments show that it is advantageous to reduce the co-operation frequency for the instances that converge relatively fast to the solutions with a number of routes close to the optimum. The proposed adaptive schemes reduce the parallel overhead and allow finding the solutions of the accuracy comparable to accuracy of solutions obtained using the frequent co-operation scheme in shorter time.

A two-stage approach of solving the VRPTW makes it possible to design and combine the algorithms for both stages independently. Increasing the accuracy of feasible solutions with minimal number of routes in the first phase becomes important for the second stage heuristics (e.g. a memetic algorithm), which work on a population of individuals. Our ongoing research is focused on incorporating the edge-assembly crossover

operator to the parallel route minimization algorithm as proposed in [4] and combine it with our parallel memetic algorithm to minimize the traveled distance [23].

Acknowledgments

We thank the following computing centers where the computations of our project were carried out: Academic Computer Centre in Gdańsk TASK, Academic Computer Centre CYFRONET AGH, Kraków (computing grant 027/2004), Poznań Supercomputing and Networking Center, Interdisciplinary Centre for Mathematical and Computational Modeling, Warsaw University (computing grant G27-9), Wrocław Centre for Networking and Supercomputing (computing grant 30).

References

1. J.F. Bard, G. Kontoravdis, G. Yu: *A branch-and-cut procedure for the vehicle routing problem with time windows*. Transportation Science, 36(2), 250–269, 2002.
2. J. Berger, M. Barkaoui: *A parallel hybrid genetic algorithm for the vehicle routing problem with time windows*. Computers and Operations Research, pages 2037–2053, 2004.
3. M. Blocho, Z.J. Czech: *An improved route minimization algorithm for the vehicle routing problem with time windows*. Studia Informatica, 32(99), 5–19, 2010.
4. M. Blocho, Z.J. Czech: *A parallel EAX-based algorithm for minimizing the number of routes in the vehicle routing problem with time windows*. In Proceedings of the Fifth Symposium on Advances of High Performance Computing and Networking, AHPCN'12, 2012.
5. A. Chabrier: *Vehicle routing problem with elementary shortest path based column generation*. Comput. Oper. Res., 33(10), 2972–2990, 2006.
6. Ch.-B. Cheng, K.-P. Wang: *Solving a vehicle routing problem with time windows by a decomposition technique and a genetic algorithm*. Expert Syst. Appl., 36(4), 7758–7763, 2009.
7. J. Cordeau, G. Laporte, É. Hautes, É. Commerciales, L.C.D. Gerad: *A unified tabu search heuristic for vehicle routing problems with time windows*. Journal of the Operational Research Society, 52, 928–936, 2001.
8. A. Debudaj-Grabysz, Z.J. Czech: *A concurrent implementation of simulated annealing and its application to the VRPTW optimization problem*. In Distributed and Parallel Systems, volume 777 of The Kluwer International Series in Engineering and Computer Science, pages 201–209. Springer US, 2005.
9. CI TASK. Website, 2011. http://www.task.gda.pl/kdm/sprzet/ Galera.
10. H. Gehring, J. Homberger: *Parallelization of a two-phase metaheuristic for routing problems with time windows*. Journal of Heuristics, 8(3), 251–276, 2002.

11. S.C. Ho, D. Haugland: *A tabu search heuristic for the vehicle routing problem with time windows and split deliveries*. Computers and Operations Research, 31, 1947–1964, 2002.

12. J. Homberger, H. Gehring: *Two evolutionary metaheuristics for the vehicle routing problem with time windows*. INFOR, 37, 297–318, 1999.

13. *Problems and benchmarks*, VRPTW. Website, 2011. http://www.sintef.no/Projectweb/TOP/VRPTW/Homberger-benchmark/.

14. S. Irnich, D. Villeneuve: *The shortest-path problem with resource constraints and k-cycle elimination*. INFORMS J. on Computing, 18(3), 391–406, 2006.

15. M. Jepsen, B. Petersen, S. Spoorendonk, D. Pisinger: *A non-robust branchand- cut-and-price algorithm for the vehicle routing problem with time windows*. Technical Report 06-03, DIKU, University of Copenhagen, Denmark, 2006.

16. B. Kallehauge, J. Larsen, O.B.G. Madsen: *Lagrangian duality applied to the vehicle routing problem with time windows*. Comput. Oper. Res., 33(5), 1464–1487, 2006.

17. I. Kamkar, M. Poostchi, M.R.A. Totonchi: *A cellular genetic algorithm for solving the vehicle routing problem with time windows*. In Soft Computing in Industrial Applications, volume 75 of Advances in Intelligent and Soft Computing, pages 263–270. Springer, Berlin, Heidelberg, 2010.

18. H. Kanoh, S. Tsukahara: *Solving real-world vehicle routing problems with time windows using virus evolution strategy*. Int. J. Know.-Based Intell. Eng. Syst., 14(3), 115–126, 2010.

19. N. Labadi, Ch. Prins, M. Reghioui: *A memetic algorithm for the vehicle routing problem with time windows*. RAIRO – Operations Research, 42(3), 415–431, 2008.

20. D. Mester, O. Bräysy: *Active guided evolution strategies for large-scale vehicle routing problems with time windows*. Comput. Oper. Res., 32(6), 1593– 1614, 2005.

21. Y. Nagata, O. Bräysy: *A powerful route minimization heuristic for the vehicle routing problem with time windows*. Operation Research Letters, 37(5), 333–338, 2009.

22. Y. Nagata, O. Bräysy, W. Dullaert: *A penalty-based edge assembly memetic algorithm for the vehicle routing problem with time windows*. Comput. Oper. Res., 37(4), 724–737, 2010.

23. J. Nalepa, Z.J. Czech: *A parallel heuristic algorithm to solve the vehicle routing problem with time windows*. Studia Informatica, 33(1), 91–106, 2012.

24. K.-W. Pang: *An adaptive parallel route construction heuristic for the vehicle routing problem with time windows constraints*. Expert Syst. Appl., 38(9), 11939– 11946, 2011.

25. J.-Y. Potvin, J.-M. Rousseau: *A parallel route building algorithm for the vehicle routing and scheduling problem with time windows*. European Journal of Operational Research, 66(3), 331–340, 1993.

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

27. Ch. Qi, Y. Sun: *An improved ant colony algorithm for VRPTW*. In Proceedings of the 2008 International Conference on Computer Science and Software Engineering – Volume 01, pages 455–458, Washington, DC, USA, 2008. IEEE Computer Society.

28. R.A. Russell: *Hybrid heuristics for the vehicle routing problem with time windows*. Transportation Science, 29(2), 156, 1995.

29. R. Skinderowicz: *Co-operative, parallel simulated annealing for the VRPTW*. In Proceedings of the Third international conference on Computational collective intelligence: technologies and applications – Volume Part II, ICCCI'11, pages 495– 504, Berlin, Heidelberg, 2011. Springer-Verlag.

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

31. X. Tan, X. Zhuo, J. Zhang: *Ant colony system for optimizing vehicle routing problem with time windows (VRPTW)*. In Proceedings of the 2006 international conference on Computational Intelligence and Bioinformatics – Volume Part III, ICIC'06, pages 33–38, Berlin, Heidelberg, 2006. Springer-Verlag.

32. P.M. Thompson, H.N. Psaraftis: *Cyclic transfer algorithms for multivehicle routing and scheduling problems*. Oper. Res., 41(5), 935–946, 1993.

33. Z. Ursani, D. Essam, D. Cornforth, R. Stocker: *Localized genetic algorithm for vehicle routing problem with time windows*. Appl. Soft Comput., 11(8), 5375–5390, 2011.

34. Y. Zhong, X. Pan: *A hybrid optimization solution to VRPTW based on simulated annealing*. In Automation and Logistics, 2007 IEEE International Conference on, pages 3113–3117, 2007.

Adaptacyjne schematy kooperacji wątków w równoległym heurystycznym algorytmie dla problemu trasowania pojazdów z oknami czasowymi

Streszczenie

Wyznaczanie tras dla pojazdów z oknami czasowymi (ang. *vehicle routing problem with time windows*) jest problemem optymalizacji dyskretnej należącym do klasy problemów NP-trudnych. Istnieją metody heurystyczne rozwiązywania problemu, pozwalające wyznaczyć w rozsądnym czasie rozwiązania nieoptymalne o koszcie bliskim kosztowi rozwiązania optymalnego, takie jak symulowane wyżarzanie, przeszukiwanie tabu, algorytmy genetyczne czy algorytmy memetyczne. W przypadku algorytmów dwustopniowych, w pierwszej fazie minimalizowana jest liczba tras, a w fazie drugiej całkowita przebyta odległość. Flota składa się z pojazdów o jednakowej, zdefiniowanej pojemności, która nie może zostać przekroczona, a obsługa klientów musi rozpocząć się w czasie trwania ich okien czasowych.

W artykule dokonano analizy wpływu częstotliwości kooperacji wątków na dokładność rozwiązań problemu trasowania pojazdów z oknami czasowymi, uzyskanych za pomocą równoległego algorytmu heurystycznego. Zostały przedstawione wyniki eksperymentów dla testów wzorcowych o różnej liczbie klientów, należących do każdej z klas testów opracowanych przez Gehringa i Hombergera. W artykule zostały zaproponowane dwa adaptacyjne schematy kooperacji, w których częstotliwość komunikacji między wątkami zależna jest od aktualnej fazy przeszukiwania przestrzeni rozwiązań S.