ICS 2011
12th INFORMS Computing Society Conference

© 2011 INFORMS | ISBN 978-0-9843378-1-1
DOI 10.1287/ics.2011.0003

Fast Local Search and Insertion Heuristics for Coordinated UGV Path Planning

Rich Kenefic
Raytheon, Fort Wayne, Indiana 46808, richard_j_kenefic@raytheon.com

Abstract This paper presents an ant-colony approach for solving the vehicle routing problem with time windows (VRPTW). The fast local search and insertion (FLS&I) methods used are based on well known heuristics, were implemented in C as Matlab mexfunctions, and were tested against the well known Solomon benchmark for the VRPTW. It is shown that solutions close to the best known results for the Solomon benchmark can be reliably obtained within the re-planning horizon of a typical military UGV application. The FLS&I algorithm and similar methods significantly improve upon agent based methods recently considered in the literature. It is also shown that, against the type C2 Solomon instances, the FLS&I heuristics used here outperform some recent methods that require much more processing time. The algorithm presented here has a feature that determines when an acceptable solution has been found to facilitate an early exit and save computer resources. With this feature, the average CPU time required to find the best known solution for the clustered Solomon instances is under one minute, well within the re-planning horizon.

Keywords vehicle routing; heuristics; path planning

Introduction

The importance of autonomous vehicles in future military operations is generally recognized (Albus et al. [2], Deyst et al. [9], McLaughlin et al. [13]). These vehicles include unmanned air vehicles (UAVs), unmanned ground vehicles (UGVs), and unmanned underwater vehicles (UUVs) with varying levels of autonomy and mission complexity. Both UAVs and UGVs have prominent roles in many planned military systems, and the control of multiple UAVs has been proposed as an AFOSR grand challenge (McLaughlin et al. [13, p. 14]). Recent work in this area has included the turn rate constraints of a UAV (Rathinam et al. [14]), but treats the problem as a turn rate limited traveling salesman problem (kTSP) rather than a vehicle routing problem (VRP) by omitting the UAV depot. Kinematic constraints for UGVs are typically left out of vehicle routing problems, and many of the VRPs that arise for UGVs in typical military applications can be considered variants of standard VRPs. A comprehensive survey of UGV systems and algorithms has recently been published in Anisi and Thunberg [3].

The VRPs of interest to military systems architects are known to be NP-hard, and in the absence of kinematic constraints, there are many heuristic methods available for them. Much research has focused on obtaining optimal solutions to standard instances, and the impression left in the minds of many military system architects is that these methods are too time consuming for use in a system that requires replanning on a relatively short time scale. Some have suggested that artificial intelligence methods may be useful for solving problems

of this type,[1] but the results reported for these methods on standard VRP instances have been disappointing.

The purpose of this paper is two-fold: first, to demonstrate that near-optimal solutions for important cases of the VRP relevant to mission planning can be computed within the replanning horizon for a typical military application. The methods used here are based on well known heuristics and were implemented in Matlab. Second, to show that an efficient use of the cross-exchange and minimum delay heuristics can find near optimal solutions more reliably and quickly than other methods recently employed on the type two Solomon instances of the vehicle routing problem with time windows (VRPTW). In addition, the algorithm employed here has a self-terminating feature that ends the search when it has been determined that a good solution has been found or when the allotted computer time has been exhausted. This feature saves computer resources on most of the clustered Solomon instances, terminating the search long before the maximum allotted CPU time has been reached.

Reference Ahmadzadeh et al. [1] was motivated by similar concerns, and used the hybrid projected gradient evolutionary search (HPGES) algorithm for the VRPTW for UAV path planning, but treated a limited set of standard instances and did not perform a Monte-Carlo study of the algorithm's performance. Reference Braysy et al. [7] makes use of heuristics similar to those used here for local search and insertion, and presents comparative performance results against a comprehensive set of test cases, including the Solomon benchmark. This reference also demonstrates that results close to the best known solutions can be obtained within the replanning horizon, but doesn't consider methods for recognizing when near optimal solutions have been found, and doesn't present detailed Monte-Carlo performance results. In this study it is shown that the ant colony meta-heuristic, combined with a slight variation of the cross-exchange for local search and minimum delay for insertion, quickly and reliably finds good solutions for all of the Solomon instances. The method presented here and the approach presented in Braysy et al. [7] both significantly outperform recently proposed agent based methods that have been applied against the Solomon benchmark.

UGV Path Planning

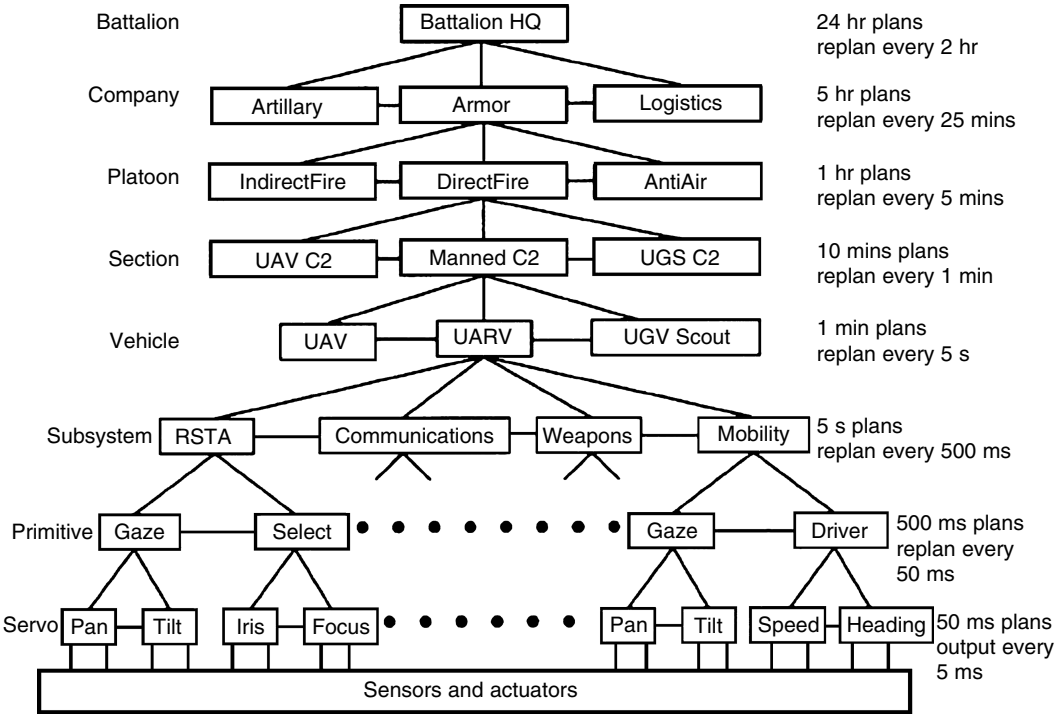
Large UGVs would be expected to efficiently perform supply missions at the battalion level over areas as large as 100 km by 100 km, with planning every 1–5 hours and replanning as often as every 5 minutes (Albus et al. [2]) as shown in Figure 1.

Assuming that planning makes use of all information available at the highest level to determine the tour order for every vehicle, the planning required for such missions resembles the well studied vehicle routing problem with time windows (VRPTW). In the most general case, where positions, demands, time windows, and travel times are not known apriori, or when the vehicle on a route may become disabled, dynamic versions of VRPTW should be considered.

In the VRPTW a single depot has a homogeneous fleet of vehicles, each of capacity C, required to service a set of customers, each with demand d_i and with available times $[e_i, l_i]$ in such a way that the vehicle capacity is not exceeded and the vehicle arrives at every customer on its route before time l_i. If a vehicle arrives before the customer opens at e_i, it must wait. When the customer opens, a service time s_i is required before the vehicle can leave for the next customer. The vehicle must return to the depot before it closes. The objective is to first minimize the number of vehicles required and then to minimize the total distance traveled. This problem and its many variants is known to be NP-hard and many researchers have investigated the performance of various meta-heuristic or evolutionary methods to obtain

[1] "Research is also needed to combine artificial intelligence methods with operations research tools to overcome inefficiencies in solving some mission critical air-force problems (e.g., scheduling in a distributed dynamic environment" (McLaughlin et al. [13], p. 33).

FIGURE 1. 4D-RCS reference architecture for a typical military system (Albus et al. [2]).



Note. Planning for the Battalion is assumed to take place at the support company level using information available at the vehicle level.

solutions. One of the most popular sets of test cases was proposed by Solomon [15], and is widely used in the literature as a basis for comparison. Much of the research emphasis has been on finding the best solutions for each of the Solomon instances, and an enormous amount of computing time has been spent attempting to find new best solutions.

A recent study of VRPTW (Bent and Van Hentenryck [4]) has resulted in several new best solutions to the VRPTW, but the processing times used were 1,800 and 7,200 CPU seconds, far beyond the replanning horizon for a typical military system. Results like these have prompted some researchers to dismiss the use of evolutionary methods to obtain results for real time systems.[2] One objective of this paper is to demonstrate the near optimal performance of an evolutionary approach to solving the VRPTW when the planning horizon is limited to 5 minutes.

Vehicle Routing with Time Windows

The vehicle routing problem with time windows (VRPTW) has been well studied for over 20 years, and several heuristics have been discovered that lead to good solutions. Recent attempts to deal with dynamic versions of VRPTW, where all of the information on the customer locations, demands, time windows, and travel times isn't known initially, have used multiagent coalitions (Boudali et al. [5, 6], Gorodetski et al. [11], Zargayouna [17]), but the performance on the Solomon benchmark when global information is available has been poor compared with known heuristics.[3] It is reasonable to expect heuristics of this type to yield

[2] "Searching high dimensional spaces can be accomplished using evolutionary algorithms . . . . However, these methods are typically too slow for real time use at levels where plans must be recomputed faster than once every few minutes" (Albus et al. [2, p. 56]).

[3] For example, Boudali et al. [6] considered the type 1 Solomon instances for 50 customers, and did not obtain the best known solution for any of the C1 instances.

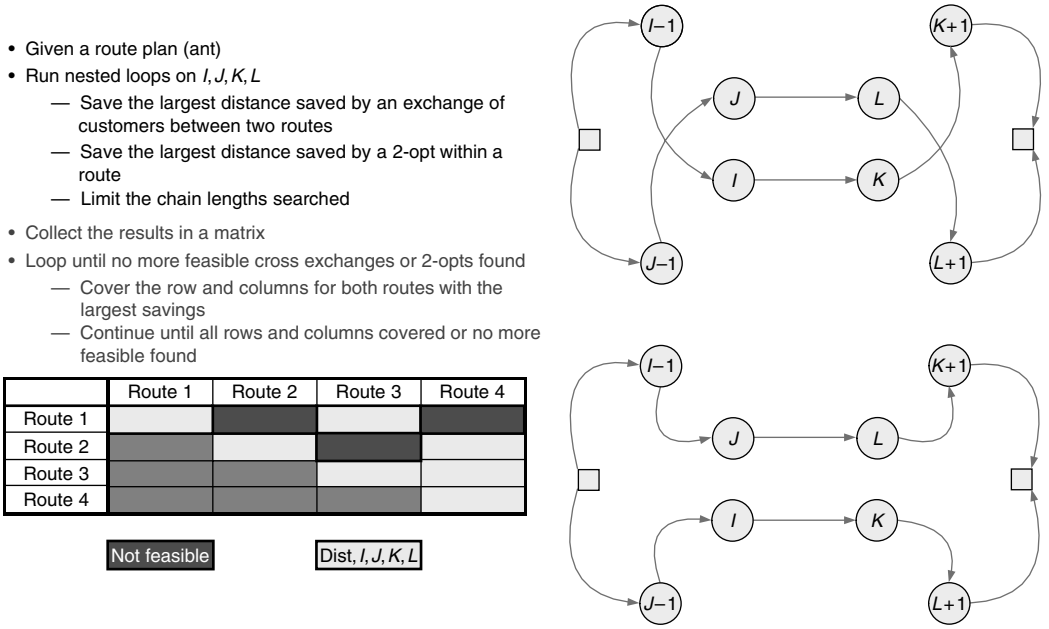
near optimal performance when global information is available, especially when, as shown here, the time to compute a good solution to the VRPTW is on the order of the travel time between vertices. For the case of incomplete initial information, the "local approaches" to solving dynamic VRPTW (Chen and Xu [8]) have been effective, and will yield near optimal solutions when global information is available.

The algorithm used here is based on the multiple ant colony system, MACS-VRPTW (Gambardella et al. [10]). Recently it has been shown that the ant colony meta-heuristic is closely related to other recent model-based optimization methods like stochastic gradient ascent and cross-entropy (Zlochin et al. [18]). Methods like simulated annealing and genetic algorithms are characterized as instance-based optimization methods in Zlochin et al. [18].

The MACS-VRPTW used here makes use of a modification to Taillard's cross exchange (Taillard et al. [16]) that speeds up the computation by limiting the length of the chains to examine for exchange and by finding the best disjoint set of exchanges to make at each step of a local search, as shown in Figure 2. At every step of the local search the best exchange within a route or between two routes is saved to a matrix and a greedy heuristic is used to perform all possible exchanges. The length of the chains to examine is incremented at every step and reinitialized if the number of vehicles is reduced by MACS-VEI or MACS-TIME. When the maximum length is reached the chain length gets no longer. The algorithm terminates and returns the best solution found when the maximum time has elapsed or when the best solution has been rediscovered at least twice. If the elapsed time is less than the maximum allowed before the start of any generation, then that generation is allowed to run to completion. The time limit used here was 200 seconds, and since partial generations are not allowed, the actual elapsed time can exceed this limit.

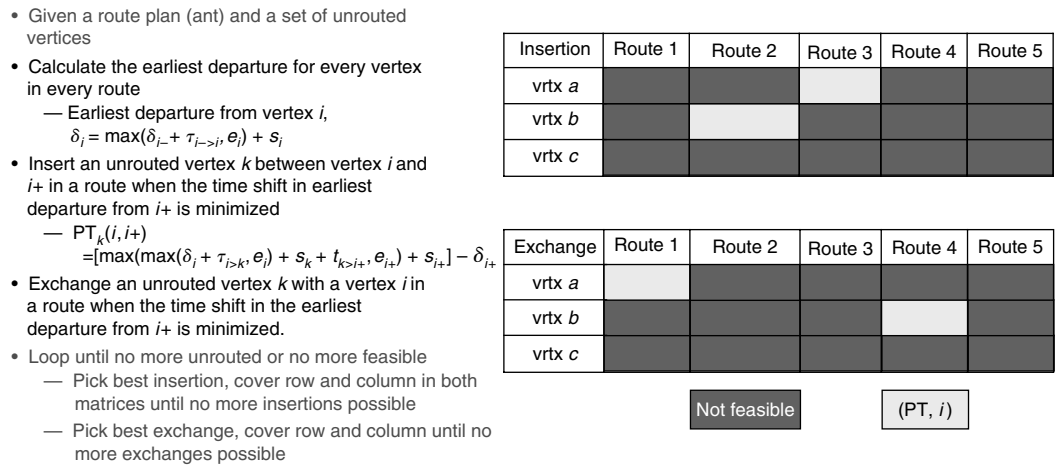
In addition to the cross exchange, which is effective in reducing the total length of the routing plan, the insertion heuristic uses minimal delay in order to reduce the number of vehicles (Homberger and Gehring [12]). The minimal delay heuristic was employed at the insertion step of the MACS-VRPTW and operates only on the unrouted customers, as shown in Figure 3. The heuristic first searches for feasible insertions of customers in routes

FIGURE 2. Modified cross-exchange.



Notes. The number of customers between I and K or J and L (chain length) is limited to speed up the search. New steps shown in blue text.

FIGURE 3. Modified minimum delay heuristic.



Note. New steps shown in blue text.

and saves the insertion that results in the least delay to a matrix. If a customer cannot be inserted in a route the heuristic looks for an exchange that will reduce the delay. Insertions always have precedence over exchanges, and if an exchange was performed, the heuristic will again look for insertions and exchanges until an iteration limit is reached. A tabu list is maintained for exchanges within routes to avoid cycling. The MACS approach falls within the hybrid methods discussed by Bent and Van Hentenryck [4].

Results for the Solomon Instances

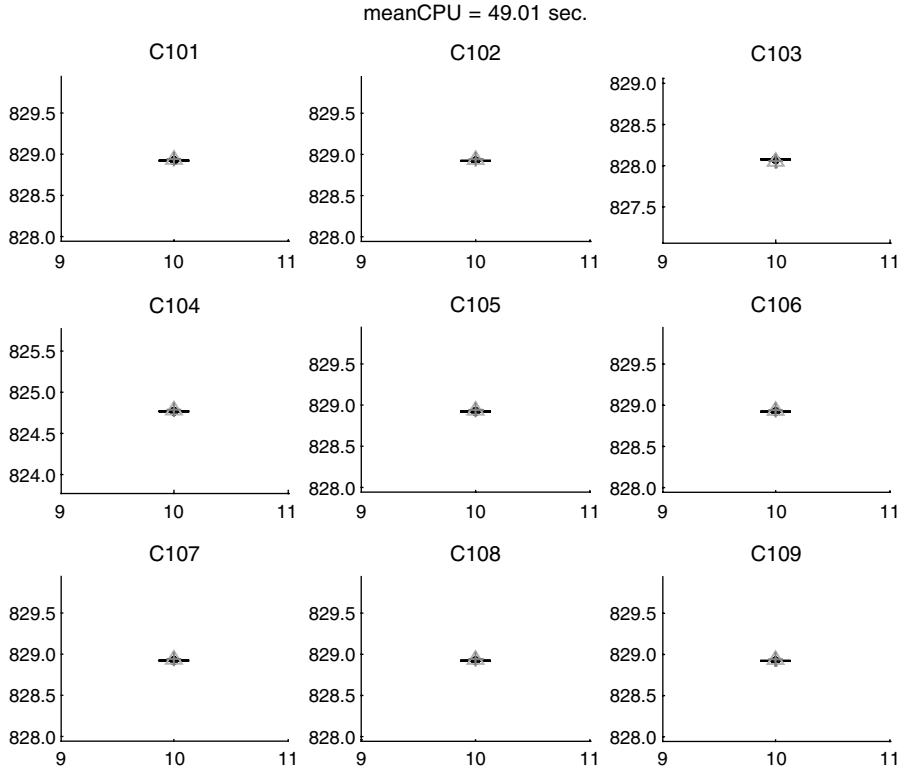
The Solomon instances are divided into two types and three classes within each type with 100 customers and no more than 25 vehicles at a single depot. In type one the scheduling times for the customers are tight and typically there will be few customers per vehicle and many vehicles required to satisfy the schedule. In type two the time windows are loose and vehicle capacity is large, so many customers can be assigned to each vehicle and few vehicles are needed to satisfy the schedule. In both types the geographical distribution of customers can be clustered, random, or mixed. The six categories are C1, R1, RC1, C2, R2, and RC2 where C1 is type one with clustered customers, etc.

In the figures presented below, 16 Monte-Carlo trials were performed for each instance within each category. The fast local search and insertion (FLS&I) results for each instance within a category are presented as a scatter plot (vehicles on the abscissa, distance on the ordinate) with a modified range-finder box plot, all in blue. The best solution known is presented with a magenta triangle, and the solutions found by Bent and Van Hentenryck (B&V-H) [4] are presented in cyan. The B&V-H results place a cross at the mean number of vehicles and mean distance with dashed lines from the minimum to maximum found for each (distance or vehicles), all in cyan. These results were obtained from Bent and Van Hentenryck [4] and through personal communication with the authors.

Type One Instances

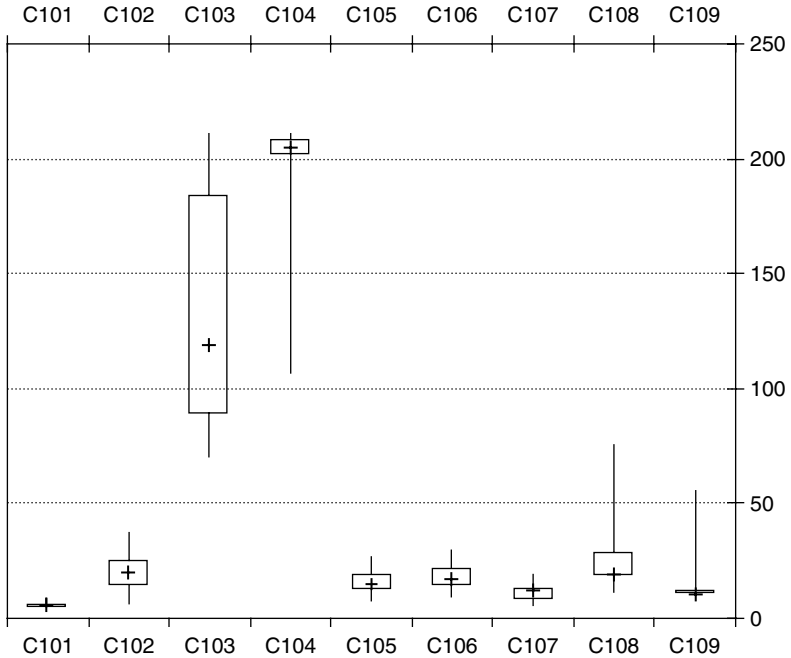
Type one results for FLS&I are shown in Figures 4, 5, 6, and 7 for the clustered, random, and mixed classes respectively. Note that, overall the B&V-H results are better, but require 1,800 seconds for each of the five trials in their study. For the C1 instances FLS&I finds the best known solutions on every trial. A box plot of the CPU time required for the C1 instances is shown in Figure 5. Note that most trials require less than 50 seconds to recognize that the best known solution has been found. Similar plots for R1 and RC1 are not included

FIGURE 4. Results for the type one clustered instances.



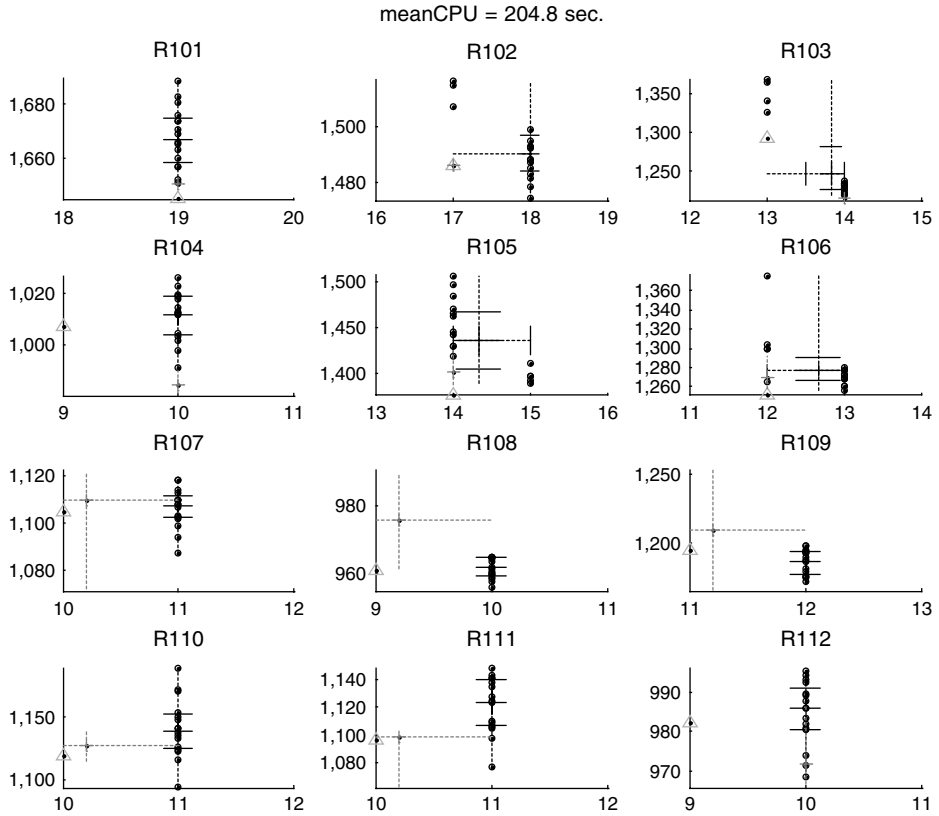
Notes. Results (Vehicles on the abscissa, distance on the ordinate) show that fast local search and insertion (FLS&I) finds the optimal solution in every trial for every instance. Average time required over all trials and instances is under 50 seconds (Δ = best known, + = B&V-H (Bent and Van Hentenryck [4]), -*o*- = FLS&I).

FIGURE 5. Box plots of CPU time (in seconds) required by FLS&I for the C1 instances.



Note. The time required to recognize optimality for most of the C1 instances is under one minute.

FIGURE 6. Scatter plot and range-finder box plot for R1 with FLS&I.



Note. Results are within one vehicle of the best known (Δ) for every trial and respectable compared with B&V-H (+ at the average).

because all of these cases required about 200 seconds per trial. For R1, the FLS&I results are within one vehicle of the best known solution for every trial. For RC1, the results are within one vehicle of the best known for all but a handful of trials (for RC105 and RC106). Given the much shorter processing time, the R1 and RC1 results are respectable compared to the results presented by B&V-H.

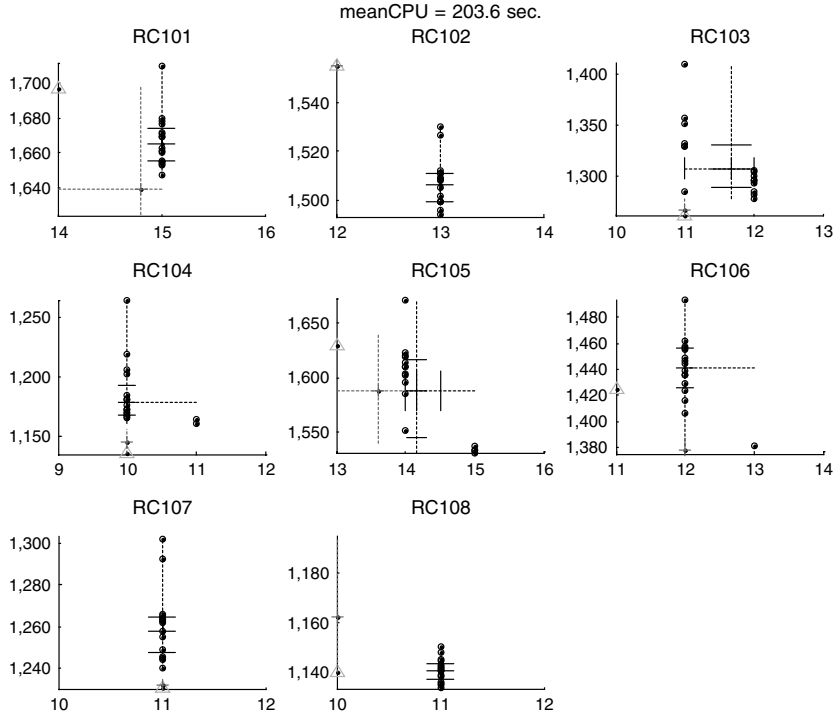
Type Two Instances

Type two results are shown in Figures 8, 9, 10, and 11 for the clustered, random, and mixed classes respectively. Note that, for the C2 instances, the local search and insertion heuristic used here significantly outperforms the results presented in Bent and Van Hentenryck [4] using far less processing time. A box plot of the CPU time required for the C2 instances is shown in Figure 9. There was one outlier at a local minimum in C202, all other trials rediscovered the best known solution at 591.56 twice and terminated the processing early. In C204, 11 of the 16 trials settled on the best known solution at 590.6: the remaining trials rediscovered a local minimum at 593.93, which terminated the processing early. Also note that FLS&I is competitive with the B&V-H results for many of the R2 and RC2 instances, requiring, on average, about 200 seconds per trial.

Conclusions

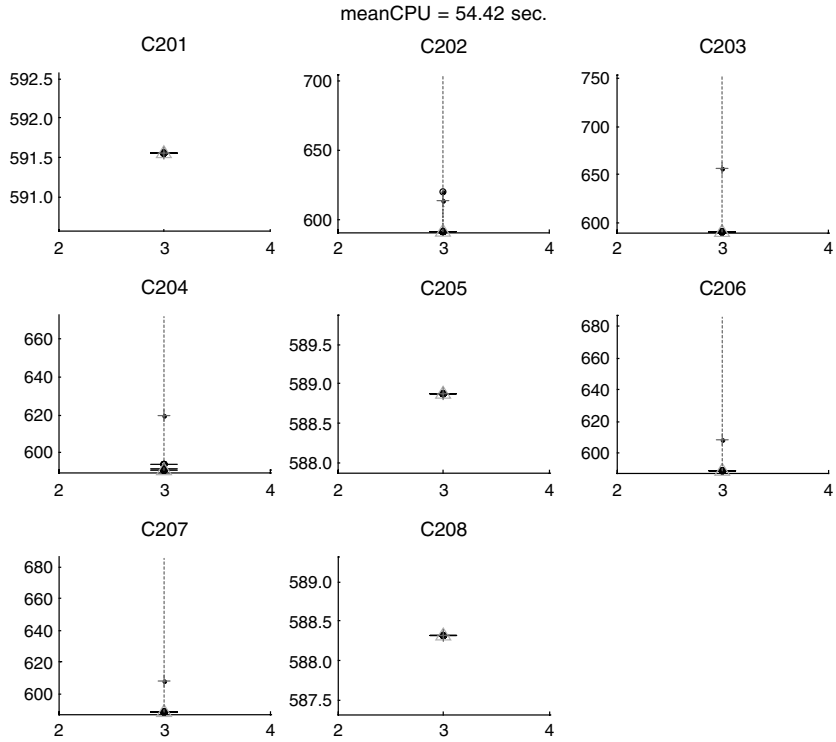
Results indicate that MACS-VRPTW with the fast local search and insertion heuristic finds solutions to the Solomon benchmark that are close to the best known solutions within the

FIGURE 7. Scatter plot and range-finder box plot for RC1 with FLS&I.



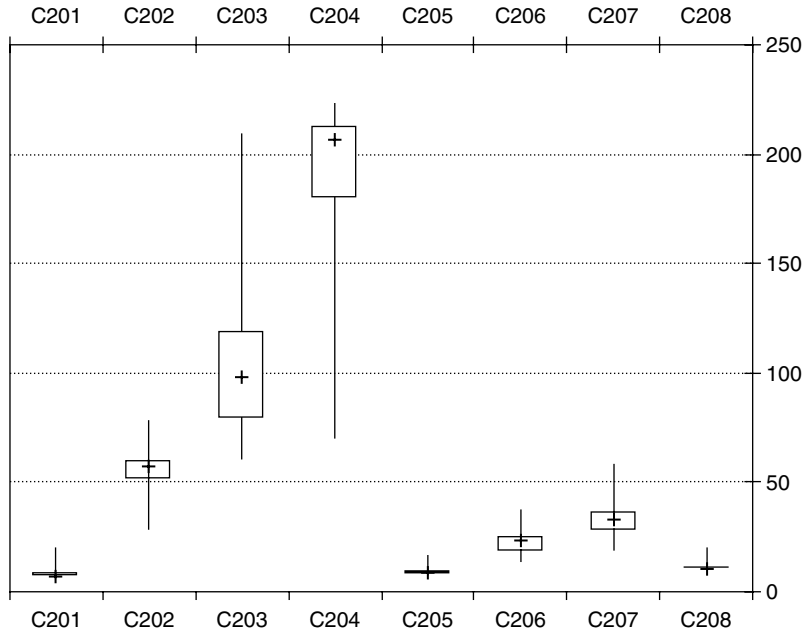
Note. Results are within one vehicle of the best known (Δ) for most trials and respectable compared with B&V-H (+ at the average).

FIGURE 8. Results for the type two clustered instances show that FLS&I finds the best known solution in every trial for 6 of 8 instances with an average CPU time under one minute.



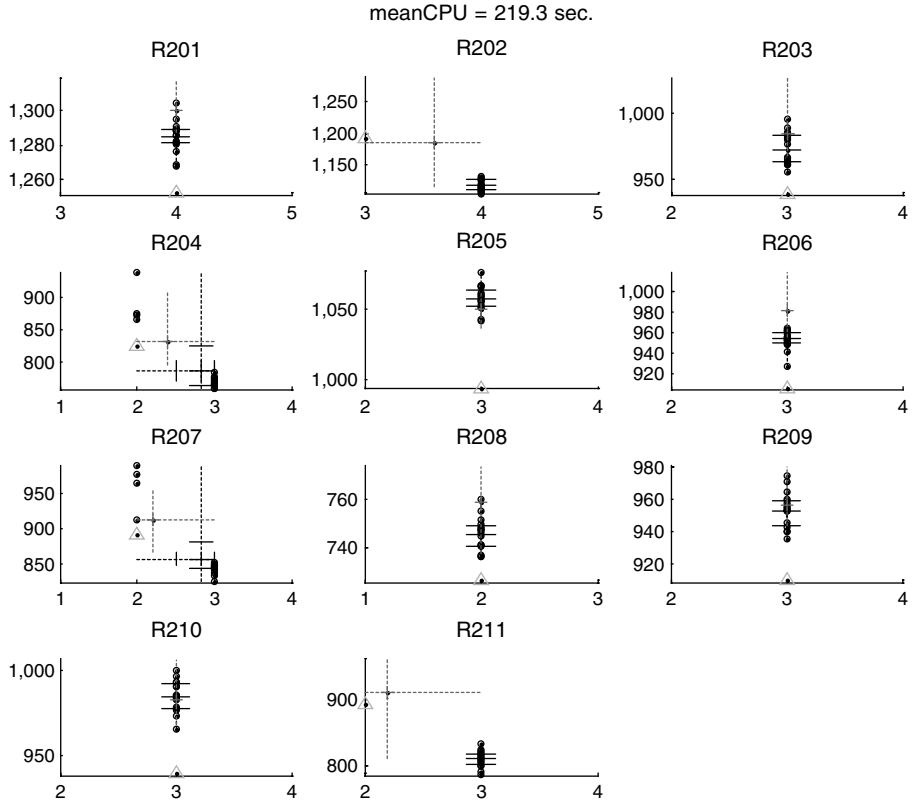
Note. Δ = best known, + = B&V-H, -*o*- = fast local search and insertion.

FIGURE 9. Box plots of CPU time (in seconds) required for the C2 instances.



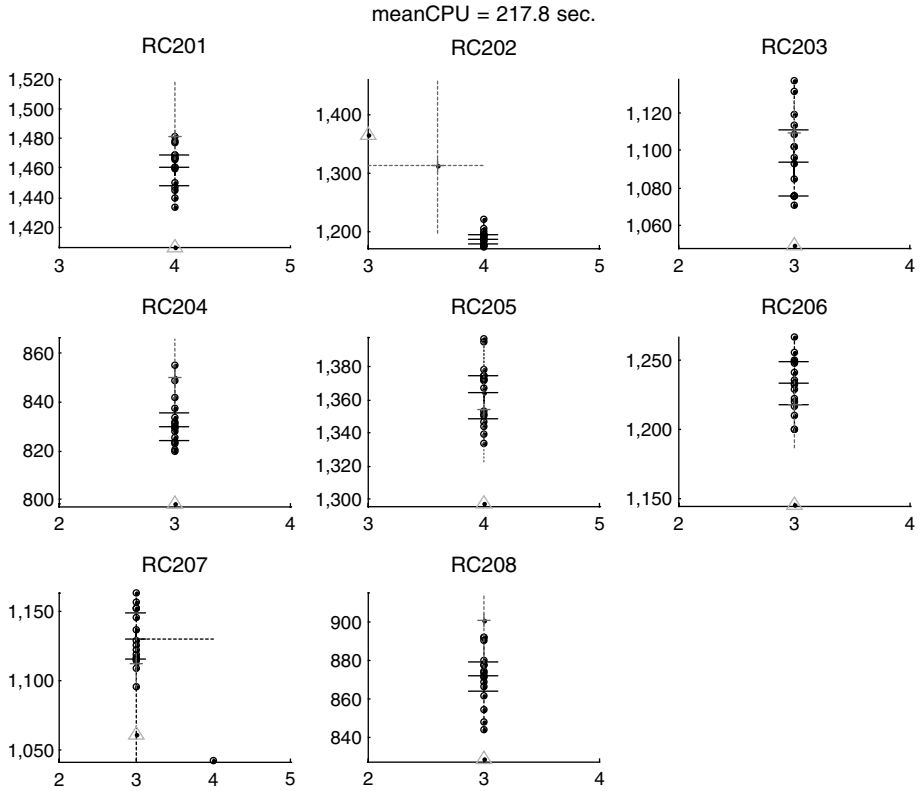
Note. The time required to recognize optimality for most of the C2 instances is under one minute.

FIGURE 10. Scatter plot and range-finder box plot for R2 with FLS&I.



Note. Results are within one vehicle of the best known (Δ) for every trial and comparable with B&V-H (+ at the average).

FIGURE 11. Scatter plot and range-finder box plot for RC2 with FLS&I.



Note. Results are within one vehicle of the best known (Δ) for every trial and comparable with B&V-H (+ at the average).

replanning horizon used here for a typical military application. Also, the method used to terminate the search finds the best known solutions to the C1 and C2 instances with an average CPU time under one minute.

The fast local search and insertion heuristics used here were both programmed in C as Matlab mex-functions and the remainder of the MACS-VRPTW method was programmed in Matlab. Elapsed time for each Monte-Carlo trial was determined using the tic and toc operators. It is expected that a dedicated C++ implementation will perform better than the one presented here.

Acknowledgements

This document does not contain technical data as defined by the International Traffic in Arms Regulations, 22 CFR 120.10(a), and is therefore authorized for publication.

References

[1] A. Ahmadzadeh, B. Sayyar-Roudsari, and A. Homaifar. A hybrid projected-gradient evolutionary search algorithm for capacitated multi-source multi-uavs scheduling with time windows. S. Butenko, R. Murphey, and P. Pardalos, eds. *Recent Developments in Cooperative Control and Optimization.* Kluwer, Norwell, MA, 1–21, 2004.

[2] J. Albus et al. *4D/RCS: A Reference Architecture for Unmanned Vehicle Systems, Version 2.0.* National Institute of Standards and Technology, Springfield, VA, 2002.

[3] D. Anisi and J. Thunberg. Survey of patrolling algorithms for surveillance UGVs. Report FOI-R–2266–SE, ISSN-1650-1942, Swedish Defense Research Agency, Stockholm, Sweden, 2007.

[4] R. Bent and P. Van Hentenryck. A two-stage hybrid local search for the vehicle routing problem with time windows. *Transportation Science* 38(4):515–530, 2004.

[5] I. Boudali, W. Fki, and K. Ghedira. How to deal with the VRPTW using multi-agent coalitions. *Proceedings of the Fourth International Conference on Hybrid Intelligent Systems*, IEEE Computer Society, Kitakyushu, Japan, 416–421, 2004.

[6] I. Boudali, W. Fki, and K. Ghedira. An interactive distributed approach for the VRP with time windows. *International Journal of Simulation* 6:48–59, 2006.

[7] O. Braysy, G. Hasle, and W. Dullaert. A multi-start local search algorithm for the vehicle routing problem with time windows. *European Journal of Operational Research* 159:586–605, 2004.

[8] Z. Chen and H. Xu. Dynamic column generation for dynamic vehicle routing with time windows. *Transportation Science* 40(February):74–88, 2006.

[9] J. J. Deyst et al. *Autonomous Vehicles in Support of Naval Operations*. The National Academies Press, Washington, DC, 2005.

[10] L. M. Gambardella, E. Taillard, and G. Agazzi. MACS-VRPTW: A multiple ant colony system for vehicle routing problems with time windows. *New Ideas in Optimization*. McGraw-Hill, London, 63–76, 1999.

[11] V. Gorodetski, O. Karsaev, and V. Konushy. Multi-agent system for resource allocation and scheduling. *Multi-Agent Systems and Applications III: Third International Central and Eastern European Conference on Multi-Agent Systems, Prague, Czech Republic*, Springer-Verlag, London, 1067–1077, 2003.

[12] J. Homberger and H. Gehring. Two evolutionary metaheuristics for the vehicle routing problem with time windows. *Information Systems and Operational Research* 37:297-318, 1999.

[13] A. J. McLaughlin et al. *Basic Research in Information Science and Technology for Air Force Needs*. The National Academies Press, Washington, DC, 2006.

[14] S. Rathinam, R. Sengupta, and S. Darbha. A resource allocation algorithm for multivehicle systems with nonholonomic constraints. *IEEE Transactions on Automation Science and Engineering* 4(January):98–104, 2007.

[15] Solomon's Benchmark Problems. Best known solutions identified by heuristics for Solomon's (1987) benchmark problems. http://www.sintef.no/static/am/opti/projects/top/vrp/bknown.html. Accessed July 18, 2007.

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

[17] M. Zargayouna. Time constrained VRP: An agent environment perception model. G. Brewka, S. Coradeschi, A. Perini, P. Traverso, eds. *Proceedings of the 17th European Conference on Artificial Intelligence, Riva del Garda, Italy*, IOS Press, The Netherlands, 849–850, 2006.

[18] M. Zlochin, M. Birattari, N. Meuleau, and M. Dorigo. Model-based search for combinatorial optimization: A critical survey. *Annals of Operations Research* 131:373–395, 2004.