A new algorithm for a Dynamic Vehicle Routing Problem based on Ant Colony System*†

R. Montemanni, L.M. Gambardella, A.E. Rizzoli, A.V. Donati

Istituto Dalle Molle di Studi sull'Intelligenza Artificiale (IDSIA)

Corresponding author: Roberto Montemanni

IDSIA, Galleria 2, CH-6928 Manno, Switzerland

tel.: +41 (0)91 610 8568, fax.: +41 (0)91 610 8661, e.mail: roberto@idsia.ch

Abstract

Most of the literature available on vehicle routing problems is about static problems, where all data are known in advance. The technological advances of the last few years rise a new class of problems about dynamic vehicle routing, where new orders are received as time progresses and must be dynamically incorporated into an evolving schedule. In this work an algorithm for this problem, based on the Ant Colony System paradigm, is proposed.

Computational results on some problems derived from widely-available benchmarks confirm the efficiency of the method we propose.

A realistic case study, based on the road network of the city of Lugano (Switzerland) will be finally presented.

1 Introduction and problem definition

In the static *Vehicle Routing Problem* (*VRP*) a fleet of vehicles has to be routed in order to visit a set of customers at a minimum cost (generally the total travel time or the total travelled distance), where all the customers are known *a priori*.

In *Dynamic Vehicle Routing Problems* (*DVRP*s) new orders dynamically arrive when the vehicles have already started executing their tours, which consequently have to be replanned at run time in order to include these new orders. In our problem a set of orders is known in advance, and a first schedule is calculated for it. New orders are then received during tour execution and

*The full paper is available as Montemanni et al. [6].

†The work was co-funded by the European Commission IST project MOSCA, grant IST-2000-29557.

the tours have to be rearranged in order to serve them. Orders received after a given time are postponed to the next day.

In this work we consider applications where new orders can be assigned to vehicles which have already left the depot (e.g. parcel collection, feeder systems, fuel distribution, etc.).

2 The algorithm *ACS-DVRP*

The algorithm we present is based on the decomposition of the *DVRP* into a sequence of static *VRP*s. There are three main elements in the architecture we propose.

Event manager. It receives new orders and keeps track of the already served orders, of the position and residual capacity of each vehicle. This information is used to construct the sequence of static *VRP*-like instances. The working day is divided into time slices and for each of them a static *VRP*, which considers all the already received (but not yet executed) orders, is created. New orders received during a time slice are postponed until its end. At the end of each time slice, customers whose service time starts in the next time slice (according to the solution of the last static *VRP*) are assigned to the vehicles. They will not be taken into account in the following static *VRP*s.

***ACS* algorithm.** The *Ant Colony System* (*ACS*) algorithm is based on a computational paradigm inspired by the way real ant colonies function (see Dorigo et al. [1]). The medium used by ants to communicate information regarding shortest paths to food, consists of *pheromone trails*. A moving ant lays some pheromone on the ground, thus making a path by a trail of this substance. While an isolated ant moves practically at random, an ant encountering a previously laid trail can detect it and decide, with high probability, to follow it, thus reinforcing the trail with its own pheromone. The collective behavior that emerges is a form of *autocatalytic* process where the more the ants follow a trail, the more attractive that trail becomes to be followed. The process is thus characterized by a positive feedback loop, where the probability with which an ant chooses a path increases with the number of ants that previously chose the same path. The *ACS* paradigm is inspired by this process. We apply it to the static *VRP*s created from a *DVRP* in our approach. The method is similar to that described in Gambardella et al. [2], which solves very efficiently the *VRP* with time windows.

The main element of the algorithm is *ants*, simple computational agents that individually and iteratively construct solutions for the problem. At each step, every ant k computes a set of feasible expansions to its current partial solution and selects one of these probabilistically, according to the following probability distribution. For ant k the probability p_{ij}^k of visiting customer j after customer i, the last one of the current partial solution, depends on the combination of two values: the *attractiveness* μ_{ij}, computed by some heuristic indicating the *a priori* desirability of that move

and the *trail level* τ_{ij}, indicating how proficient it has been in the past to visit j right after i (it represents therefore an *a posteriori* indication of the desirability of that move). Trails are updated at each iteration, increasing the level of those associated with arcs contained in "good" solutions, while decreasing all the others.

Pheromone conservation. Once a time slice is over and the relative static problem has been solved, the pheromone matrix contains information about good solutions to this problems. As each static problem is potentially very similar to the next one, this information is passed on to the next problem. We perform this task very efficiently, following a strategy inspired by that suggested in Guntsch and Middendorf [4].

3 Computational results

The 21 benchmarks adopted in the present work are derived from those proposed in Kilby et al. [5][1], which are based on very popular static *VRP* benchmarks (originally used by Taillard, Christofides and Beasley, Fisher et al.).

We first investigated how the results of *ACS-DVRP* are connected with pheromone conservation, then we compared *ACS-DVRP* with a method based on a *Multi-Start Local Search* algorithm (*MSLS*). In order to carry out the tests, we compressed the working day of each problem into 1500 seconds of CPU time of an Intel Pentium 4 1.5 GHz processor. Five runs for each of the 21 problems are considered, and the results obtained are summarized in Table 1.

Table 1: Computational results.

Problem	MSLS			ACS		
	Min	Max	Avg	Min	Max	Avg
c100	1080.33	1169.67	1124.04	973.26	1100.61	1066.16
c100b	978.39	1173.01	1040.99	944.23	1123.52	1023.60
c120	1546.50	1875.35	1752.31	1416.45	1622.12	1525.15
c150	1468.36	1541.54	1493.06	1345.73	1522.45	1455.50
c199	1774.33	1956.76	1898.20	1771.04	1998.87	1844.82
c50	693.82	756.89	722.15	631.30	756.17	681.86
c75	1066.59	1142.32	1098.85	1009.38	1086.65	1042.39
f134	16072.97	17325.73	16866.79	15135.51	17305.69	16083.56
f71	369.26	437.15	390.48	311.18	420.14	348.69
tai100a	2427.07	2583.02	2510.29	2375.92	2575.70	2428.38
tai100b	2302.95	2532.57	2406.91	2283.97	2455.55	2347.90
tai100c	1599.19	1800.85	1704.40	1562.30	1804.20	1655.91
tai100d	2026.82	2165.39	2109.54	2008.13	2141.67	2060.72
tai150a	3787.53	4165.42	3982.24	3644.78	4214.00	3840.18
tai150b	3313.03	3655.63	3485.79	3166.88	3451.69	3327.47
tai150c	3090.47	3635.17	3253.08	2811.48	3226.73	3016.14
tai150d	3159.21	3541.27	3323.57	3058.87	3382.73	3203.75
tai75a	1911.48	2140.57	2012.13	1843.08	2043.82	1945.20
tai75b	1634.83	1934.35	1782.46	1535.43	1923.64	1704.06
tai75c	1606.20	1886.24	1695.50	1574.98	1842.42	1653.58
tai75d	1545.21	1641.91	1588.73	1472.35	1647.15	1529.00

[1]Problems available at http://www.dcs.st-and.ac.uk/~apes/apedata.html.

The average results of *ACS-DVRP* are 4.37% better then those of *MSLS*. The best (worst) results are 4.86% (2.70%) better in average. The best solutions are always found by *ACS-DVRP*.

Some results obtained on a realistic scenario, based on the road network of the city of Lugano, will be finally presented.

4 Conclusion

A Dynamic Vehicle Routing Problem has been discussed. A solving strategy for this problem has been presented. It based on the partition of the working day into time slices. An Ant Colony System algorithm has been used to solve the static Vehicle Routing Problems arising from this partition. A technique to transfer information about good solutions, from a static problem to the following one has been developed.

Computational results confirm that the performance of our algorithm is strictly connected with the technique to transfer information about good solutions between consecutive static problems. Further computational results suggest that the algorithm we propose is very efficient, also when compared with other heuristic techniques.

A case study, set up in the city of Lugano, demonstrates that our method can be applied to real-world instances.

References

[1] M. Dorigo, G. Di Caro, L.M. Gambardella. Ant algorithms for discrete optimization. In *Artificial Life* 5, pages 137–172, 1999.

[2] L.M. Gambardella, É. Taillard, G. Agazzi. MACS-VRPTW: a multiple ant colony system for vehicle routing problems with time windows. In *New ideas in optimization*. McGraw-Hill. D. Corne et al. eds., pages 63-76, 1999.

[3] M. Gendreau, J.-Y. Potvin. Dynamic vehicle routing and dispatching. In *Fleet management and logistic*. Kluwer Academic Publishers. T.G. Crainic, G. Laporte eds., pages 115-226, 1998.

[4] M. Guntsch, M. Middendorf. Pheromone modification strategies for ant algorithms applied to dynamic TSP. In *Lecture Notes in Computer Science* 2037, pages 213–222, 2001.

[5] P. Kilby, P. Prosser, P. Shaw. Dynamic VRPs: a study of scenarios. Technical Report APES-06-1998, University of Strathclyde, September 1998.

[6] R. Montemanni, L.M. Gambardella, A.E. Rizzoli, A.V. Donati. A new algorithm for a Dynamic Vehicle Routing Problem based on Ant Colony System. Technical Report IDSIA-23-02, IDSIA, November 2002. Available at ftp://ftp.idsia.ch/pub/techrep/IDSIA-23-02.pdf.gz