A Parallel Hybrid Evolutionary Metaheuristic for the Period Vehicle Routing Problem

Dalessandro Soares Vianna, Luiz S. Ochi
and Lúcia M. A. Drummond

Universidade Federal Fluminense - Departamento de Ciência da Computação
Travessa Capitão Zeferino 56/808
Niterói - Rio de Janeiro - Brazil - 24220-230
e-mail: dada@pgcc.uff.br, {satoru,lucia}@dcc.uff.br

Abstract. This paper presents a Parallel Hybrid Evolutionary Metaheuristic for the Period Vehicle Routing Problem (PVRP). The PRVP generalizes the classical Vehicle Routing Problem by extending the planning period from a single day to M days. The algorithm proposed is based on concepts used in Parallel Genetic Algorithms and Local Search Heuristics. The algorithm employs the island model in which the migration frequency must not be very high. The results of computational experiments carried out on problems taken from the literature indicate that the proposed approach outperforms existing heuristics in most cases.

1 Introduction

The classical Vehicle Routing Problem (VRP) is defined as follows: vehicles with a fixed capacity Q must deliver order quantities q_i $(i = 1, \ldots, n)$ of goods to n customers from a single depot $(i = 0)$. Knowing the distance d_{ij} between customers i and j $(i, j = 0, \ldots, n)$, the objective of the problem is to minimize the total distance travelled by the vehicles in such a way that only one vehicle handles the deliveries for a given customer and the total quantity of goods that a single vehicle delivers do not be larger than Q.

In classical VRPs, typically the planning period is a single day. The period Vehicle Routing Problem (PVRP) generalizes the classical VRP by extending the planning period to M days. Over the M-day period, each customer must be visited at least once during the considered period. The classical PVRP consists of a homogeneous vehicle fleet (vehicles with same capacities) which must visit a group of customers from a depot where the vehicles must start and return to at the end of their journeys. Each vehicle has a fixed capacity that cannot be exceeded and each customer has a known daily demand that must be completely satisfied in only one visit by exactly one vehicle. The planning period is M days. If M = 1, then PVRP becomes an instance of the classical VRP. Each customer in PVRP must be visited k times , where $1 \leq k \leq$ M. In the classical model of PVRP, the daily demand of a customer is always fixed. The PVRP can be seen as a problem of generating a group of routes for each day so that the constraints involved are satisfied and the global costs are minimized.

PVRP can also be seen as a multi-level combinatorial optimization problem. In the first level, the objective is to generate a group of feasible alternatives (combinations) for each customer. For example, if the planning period has $t = 3$ days $\{d_1, d_2, d_3\}$ then the possible combinations are: $0 \rightarrow 000$; $1 \rightarrow 001$; $2 \rightarrow 010$; $3 \rightarrow 011$; $4 \rightarrow 100$; $5 \rightarrow 101$; $6 \rightarrow 110$ and $7 \rightarrow 111$. If a customer requests two visits, then this customer has the following visiting alternatives: $\{d_1, d_2\}$, $\{d_1, d_3\}$, and $\{d_2, d_3\}$ (or the options: 3, 5 and 6 of Table 1). In the second level, we must select one of the alternatives for each customer, so that the daily constraints are satisfied. Thus we must select the customers to be visited in each day. In the third level, we solve the vehicle routing problem for each day. In this paper, our algorithm is applied to the basic model of PVRP with an additional constraint: the number of vehicles is limited, although this limit is not necessarily the same every day. The technique proposed here can be applied to various models of PVRP.

Customer	Diary Demand	# of Visits	# of Combinations	Possible Combinations
1	30	1	3	1, 2 e 4
2	20	2	3	3, 5 e 6
3	20	2	3	3, 5 e 6
4	30	1	3	1, 2 e 4
5	10	3	1	7

Table 1. A PVRP with t=3 days

In the last years, Evolutionary Metaheuristics and in particular Genetic Algorithms (GA) have been used successfully in solution of NP-Complete and NP-HARD problems of high dimensions. Hard problems require large search spaces resulting in high computational costs. In this context, Evolutionary Metaheuristics including GA may require a large amount of time to find good feasible solutions, encouranging the use of parallel techniques [10]. Although the main goal of a Parallel Evolutionary Metaheuristic is the reduction of the execution time necessary to find an acceptable solution, sometimes it can also be used to improve the results obtained by sequential versions.

Most of Evolutionary Metaheuristics, such as GA, Scatter Search, Ant Systems and Neural Nets, are easy to parallelize because of their intrinsic parallelism. There are differents ways to parallelize GA. The generally used classification divides parallel GAs in three categories: Island or stepping stone, Fine Grain and Panmitic Models [10].

In this paper, we propose a parallel hybrid evolutionary metaheuristic based on parallel GA, scatter search and local search methods. The algorithm is based on the Island Model and it was implemented on a cluster of workstations with 4 RISC/6000 processors.

The remainder of this paper is organized as follows. Section 2 presents the

related literature about PVRP. Section 3 presents the algorithm proposed. Experimental results are shown in Section 4. Finally, Section 5 concludes the paper.

2 Related Literature

Although the PVRP has been used in many applications, it has not been extensively studied in related literature. Golden, Chao and Wasil [7] developed a heuristic based on the concept of "record-to-record" proposed by Dueck [5]. To the best of our knowledge, there are not many papers using Metaheuristics to solve PVRP. Cordeau, Gendreau and Laporte [4] presented a Tabu Search Metaheuristic to solve this problem. Computational results taken from the literature indicate that their algorithm outperforms all previous conventional heuristics.

GA have been used with great success to solve several problems with high degrees of complexity in Combinatorial Optimization, including models of daily VRP ([1], [8], [9], [10]). More recently, Rocha, Ochi and Glover[11] proposed a Hybrid Genetic Algorithm (HGA) for the PVRP. Their algorithm is based on concepts of GA and Local Search heuristics. Computational experiments using tests available in literature showed the superiority of this method when it is compared with other existing metaheuristics.

3 A Parallel Hybrid Evolutionary Metaheuristic

This paper presents a Parallel Hybrid Evolutionary Metaheuristic (PAR-HEM) for the PVRP. The parallel algorithm is based on the Island Model and it was implemented on a cluster of workstations with 4 RISC/6000 processors. In order to reduce communication overhead, which is usual in this kind of model [9] and ensure the occupation of all processors during the execution of the algorithm, a new criteria of migration and termination were employed. In Island Model, the population of chromosomes of Genetic Algorithms is partitioned into several subpopulations, which evolve in parallel and periodically migrate their individuals(chromosomes) among themselves. Because of the high cost of communication in this model, migration frequency must not be very high. Thus migration is only executed when subpopulation renewal is necessary. The criterion of termination of processes is based on a global condition, which involves every process that composes the PAR-HEM, in order to prevent a process from being idle while the others are still executing.

In our algorithm, each processor executes a pair of tasks m_i and q_i. Each task m_i executes the following steps, based on the sequential algorithm proposed by Rocha, Ochi and Glover [11], described in next subsections: generation of a feasible alternative for each customer; selection of an alternative for each customer; representation of solutions through chromosomes; generation of an initial population of chromosomes; evaluation and reproduction; and diversification.

In order to execute the diversification step, m_i sends a migration request to q_i if population renewal is required. The task q_i is responsible for migration and termination of the algorithm above. See [9] for more details.

3.1 Representation of Chromosomes

The representation of a feasible solution in a chromosome structure may be much more complex for the PVRP than it is for the VRP. In addition to the problem of finding an optimal assignment of customers to vehicles and defining the best route for each vehicle, there is also the problem of distributing the number of visits required by each customer in the planning horizon, satisfying the daily constraints. Each chromosome in our GA, is represented by two vectors. The first vector is associated with the n customers of PVRP, as shown in Table 2. The i^{th} position (i^{th} gene) of this vector is a non-negative integer k, such that $0 \leq k \leq 2t-1$, where t is the number of days in the current planning horizon. The number k represents a feasible alternative (combination) of visits to the associated customer. The binary representation of k represents the days when the associated customer receives a visit. The second vector (associated to the first) corresponds to the accumulated demand of each day in the planning horizon. For example, considering 3 days, we could have the following served daily demand {60, 30, 50}, which means that in the first day 60 goods were delivered, in the second day 30 goods and finally in the last day 50 goods were delivered.

L = (1	2	3	4	5) reference vector
P = (4	5	3	1	7) chromosome
⇕	⇕	⇕	⇕	⇕
100	101	011	001	111 } combination in binary form

Table 2. Representation of a chromosome

3.2 Initial Population

The initial population of our GA is generated as follows. Initially, a set of feasible alternative visits for each customer is generated. The customers are ordered according to their number of alternatives. For example, in a period of three days ($t = 3$), a customer which requires three visits has only one feasible alternative, while another which requires only one visit has three feasible alternatives. In order to generate an initial chromosome, we randomly select a customer from the set of customers which present the smallest number of feasible alternatives. Then one of the alternatives of this customer is chosen. The integer number associated to this alternative is placed in the corresponding gene. Then successively, an alternative is selected from each of the remaining customers with the smallest number of alternatives. For each alternative selected, we update the data in the second vector (vector of served daily demand). If the introduction of an alternative of a new customer violates the global capacity of the fleet in any day, this alternative is discarded and another alternative for this customer is

sought. When the customer does not have other alternatives, the chromosome is abandoned. The chromosome is complete when every gene has been filled.

In the first allocation, the customers selected hardly violate the maximum capacity constraint of the daily fleet, because few customers have been allocated. On the other hand, according to the addition of each new customer, the accumulated demand for each day increases and so does the risk of exceeding the maximum capacity allowed for any particular day. Therefore, starting the selection of customers with the one with greatest number of alternatives, may reduce the risk of discarding this chromosome.

3.3 Evaluation and Reproduction

The fitness level of a chromosome in our GA corresponds to the objective function value for the PVRP. The cost of the objective function is obtained by solving a set of VRPs, one for each day. Although this technique is expensive, it is the simplest form of evaluating a chromosome in most optimization problems. In order to minimize this effort, we implemented a fast and efficient heuristic, based on saving concepts [3] to solve a VRP for each day.

The reproduction mechanism in our GA uses classical crossover and mutation operators.

In the crossover operator, given two parents from the population, we choose randomly a position (gene) associated with a customer. Initially, we subtract from the second vector (the accumulated demand vector associated with this chromosome) the respective demand values corresponding to the selected alternatives. Next, it is tried to change the alternatives of this customer selected in each parent (if these alternatives are distinct). If this change does not produce a new feasible solution, considering the maximum daily capacity constraints of the fleet, another attempt with a new gene is made. This procedure is applied p times, where p is an input parameter of the problem.

The second operator used, is a Mutation operator, which executes feasible changes in p points selected randomly in the chromosome.

3.4 Diversification and Termination

In order to diversify a population with a small tax of renewal, our algorithm uses the migration operator.

The strategy adopted in this algorithm, consists of associating to each task m_i, which executes the PAR-HEM, a task q_i. A task q_i, for $1 \leq i \leq w$, where w is the number of processors, communicate only by message-passing.

A task q_i is activated by the associated task m_i in the following cases:

- When the renewal tax is less than 5 per cent (tax of replacement of parents by offspring's), triggering the migration of individuals.
- When the task m_i is capable of finishing, initiating the termination of the algorithm.

In the first case, q_i executes a broadcast of requests so that all tasks q_j ($\forall j : j \neq i$) send their local best solutions (LBS) generated so far. When this occurs, a new population is generated based on the k-LBS. If a chromosome received is better than one of the k-LBS, then the chromosome received replaces the worst of the k-LBS. Usually the number k is less than the size of a population. Thus, in order to mantain the same number of chromosomes in the new population, the following procedure is executed. Copies of the k-best solutions are generated using the roulette criterion, where the most suitable individuals give a bigger number of copies. For each copy, a "window" is opened in a random position and new feasible alternatives for each customer are selected swapping genes in these windows. Built a new population with characteristics from the LBS, the genetic operator is restarted.

After y migrations (y is an entry parameter), m_i becomes capable of finishing its execution and consequently it activates q_i, which keeps an array of n elements representing the state of the system. Each position i of the array of state may assume values true or false, indicating that the process m_i is capable of finishing or not. When q_i is activated, it updates the positon i in the array of state with true and initiates a broadcast so that the other arrays represent the current state of the system. This strategy of termination is based on a simplification of the algorithm for detection of stable conjunctive predicates presented in [6]. A task m_i will continue its execution, even when capable of finishing, until all processors are able to finish. This prevents a processor from being idle while other processors remain executing and still allows the improvement of the optimal solution while the application does not finish. When all elements of the array of state are true, termination is detected by q_i and m_i is informed to finish its execution. A task m_i informs to q_i the optimal solution so far whenever its champion chromosome is improved.

The strategy described above allows that the process, which is responsible for the PAR-HEM execution does not remain occupied with the communication required by migration of individuals, and termination of the PAR-HEM, simplifying its design and implementation.

4 Computational Results

Our numerical experiments were run on a cluster of workstations with 4 RISC/6000 processors using the Programming Language C and MPI (Message-Passing Interface) for parallelism [12]. The performance of PAR-HEM was evaluated as follows. The algorithm was applied to several instances of the problems presented by Christofides and Beasley [2] and Golden, Chao and Wasil [7]. These test problems have dimensions ranging from 50 to 417 customers and planning period varying from 2 to 10 days as shown in Table 3.

We compared PAR-HEM with the following heuristics: CGL - Metaheuristic proposed by Cordeau, Gendreau e Laporte [4]; CGW - Heuristic proposed by Golden, Chao, e Wasil [7] and HGA - Metaheuristic proposed by Rocha, Ochi and Glover [11].

Problem	Customers	Period	Vehicles
2	50	5	3
3	50	5	1
4	75	2	5
5	75	5	6
6	75	10	1
7	100	2	4
8	100	5	5
9	100	8	1
10	100	5	4
11	126	5	4
12	163	5	3
13	417	7	9
14	20	4	2
15	38	4	2
16	56	4	2
17	40	4	4
18	76	4	4
19	112	4	4
20	184	4	4

Table 3. Problems

Problem	HEM	PAR-HEM
2	1355.37	1310.25
3	525.96	555.07
4	826.69	884.55
5	2102.83	2103.84
6	799.04	786.70
7	862.26	884.72
8	2158.83	2129.73
9	844.72	839.76
10	1717.60	1705.82
11	790.92	779.73
12	1227.37	1227.00
13	4558.57	4626.11
14	865.96	904.32
15	1792.14	1792.14
16	2749.68	2780.30
17	1631.53	1623.47
18	3180.04	3161.84
19	4809.64	4792.15
20	8570.94	8378.77

Table 4. Average cost: (HEM X PAR-HEM)

As showed in [11], their algorihtm (HGA) presents the best results in terms of quality of solution among the existing heuristics in literature. Thus, we compared PAR-HEM with an adapted version of HGA, which is called here as Hybrid Evolutionary Metaheuristic (HEM).

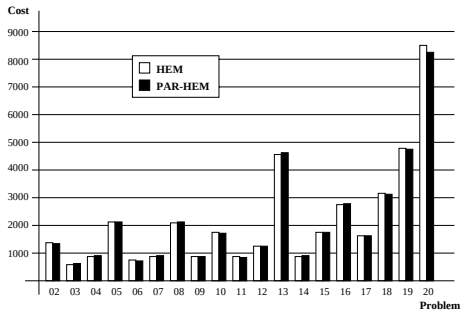

Fig. 1. Average cost: (HEM X PAR-HEM)

Figure 1 and Table 4 show a comparison in terms of average costs between the algorithms HEM and PAR-HEM. Each problem of the Table 3 was executed 3 times. Data corresponding to problem 01 were not avalibale. Figure 1 and

Table 4 show that PAR-HEM achieved the best results for 11 problems while HEM achieved the best results in 7 problems. Problem 15 presented identical results. Figure 2 and Table 5 show the average time in seconds required by the algorithms HEM and PAR-HEM. Figure 3 shows the speedup obtained by PAR-HEM. Sometimes the speedup exceeds the number of processors. In these cases the PAR-HEM converges faster than HEM. This may happen because the populations generated are different.

Problem	HEM	PAR-HEM
2	110.67	36.67
3	27.00	5.33
4	418.33	105.67
5	997.67	166.33
6	67.00	16.33
7	1365.67	362.67
8	1391.67	606.67
9	906.00	203.33
10	2470.67	950.33
11	2826.33	1415.33
12	10474.33	5844.33
13	370.33	169.00
14	14.00	4.00
15	67.00	20.33
16	269.67	87.00
17	108.67	34.33
18	541.00	215.67
19	3475.00	1205.50
20	13873.00	5444.00

Table 5. Average time: (HEM X PAR-HEM)

Problem	CGL	CGW	PAR-HEM
2	1330.09	1337.20	**1291.10**
3	524.61	**524.60**	533.91
4	**837.93**	860.90	871.71
5	**2061.36**	2089.00	2089.29
6	840.30	881.10	**770.82**
7	**829.37**	832.00	844.74
8	**2054.90**	2075.10	2112.96
9	**829.45**	829.90	836.74
10	**1629.96**	1633.20	1660.90
11	817.56	791.30	**775.89**
12	1239.58	1237.40	**1215.37**
13	**3602.76**	3629.80	4604.74
14	954.81	954.80	**864.06**
15	1862.63	1862.60	**1792.14**
16	2875.24	2875.20	**2749.68**
17	**1597.75**	1614.40	1613.67
18	3159.22	3217.70	**3143.23**
19	4902.64	4846.50	**4792.15**
20	8367.40	8367.40	**8299.71**

Table 6. Best Cost: (CGL X CGW X PAR-HEM)

We also compared PAR-HEM with the heuristics CGL and CGW. Table 6 presents a comparison among PAR-HEM and these heuristics in terms of cost.

5 Concluding Remarks

Our results presented in Tables and Figures above, show significant advantages for PAR-HEM, not only with respect to the running time but also with respect to the search quality.

The partial results exhibited by PAR-HEM qualify it as a promissing way to obtain good aproximate solutions of PVRP since the solutions shown in this paper were obtained by initial tests, without an accurate definition of the best parameters for PAR-HEM. On the other hand, the results presented by sequential heuristics were obtained by exhaustive tests with several parameters.

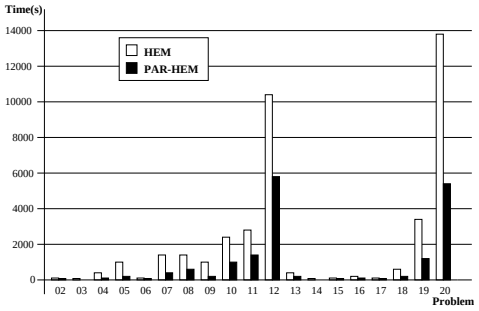

Fig. 2. Average time: (HEM X PAR-HEM)

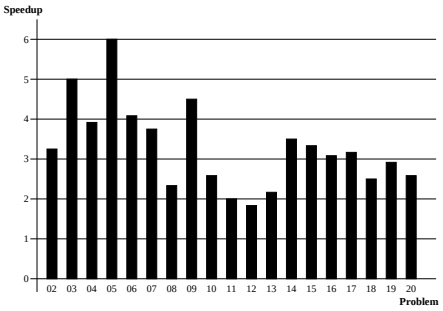

Fig. 3. Speedup

References

1. Back, T., Fogel, D.B., Michalewicz, Z.: Handbook of Evolutionary Computation, University Oxford Press, NY, (1996).
2. Christofides, N., Beasley, J.E.: The Period Routing Problem, Networks **14** (1984) 237 - 256.
3. Clarke, G., Wright, J.: Scheduling of Vehicles From a Central Depot to a Number of Delivery Points. Operations Research **12-4** (1964) 568-581.
4. Cordeau, J.F., Gendreau, M., Laporte, G.: A Tabu Search Heuristic for Periodic and Multi-Depot Vehicle Routing Problems. Technical Report 95-75, Center for Research on Transportation, Montréal (1995).
5. Dueck, G.: New Optimization Heuristics: The Great Deluge Algorithm and the Record-to-Record Travel, Scientific Center Technical Report, IBM Germany, Heidelberg Scientific Center (1990).
6. Drummond, L.M.A., Barbosa, V.C.: Distributed Breakpoint Detection in Message-passing programs, J. of Parallel and Distributed Computing **39** (1996) 153-167.
7. Golden, B.L., Chao, I.M., Wasil, E.: An Improved Heuristic for the Period Vehicle Routing Problem. Networks (1995), 25-44.
8. Ochi, L.S., Figueiredo, R.M.V., Drummond, L.M.A.: Design and Implementation of a Parallel Genetic Algorithm for the Travelling Purchaser Problem, ACM symposium on Applied Computing (1997) 257-263.
9. Ochi, L.S., Vianna, D.S., Drummond, L.M.A.: A Parallel Genetic Algorithm for the Vehicle Routing Problem with Heterogeneous Fleet, Proc. of the BioSP3, Lecture Notes in Computer Science, Springer-Verlag **1388** (1998) 187-195.
10. Ribeiro, J.L.: An Object-oriented Programming Environment for Parallel Genetic Algorithms, Ph.D. Thesis, Dep. of Comp. Science, Univ. College London (1995).
11. Rocha, M.L., Ochi, L.S., Glover, F.: A Hybrid Genetic Algorithm for the Periodic Vehicle Routing Problem (1998) - (To be published).
12. Snir, M., Otto, S.W., Huss-Lederman, S., Walker, D.W., Dongarra, J.: MPI: The Complete Reference, The MIT Press (1996).

This article was processed using the LaTeX macro package with LLNCS style