Parallel Numerics '05, 109-118 M. Vajteršic, R. Trobec, P. Zinterhof, A. Uhl (Eds.)
Chapter 5: Optimization and Classification ISBN 961-6303-67-8

A parallel version of the D-Ant algorithm for the Vehicle Routing Problem

Karl F. Doerner[1,*], **Richard F. Hartl**[1], **Maria Lucka**[2]

[1] Institute for Management Science, University of Vienna,
Brünner Strasse 72, A-1210 Vienna

[2] Institute of Scientific Computing, University of Vienna,
Nordbergstrasse 15/C/3, A-1090 Vienna

In this paper we study a parallel implementation of the D-Ant algorithm developed by Reimann, Doerner and Hartl [9] for solving the Vehicle Routing Problem. The main idea in this algorithm is to speed up the search by letting the ants solve only sub-problems rather than the whole problem. This algorithm is well suited for parallelization. We propose a mixed parallelization strategy which combines fine-grained with coarse-grained parallelism.

1 Introduction

In this paper we study a mixed parallelization strategy, which is a combination of coarse-grained and fine-grained parallelization for the D-Ant algorithm applied to the Vehicle Routing Problem. The Vehicle Routing Problem (VRP) involves the construction of a set of vehicle tours starting and ending at a single depot and satisfying the demands of a set of customers, where each customer is served by exactly one vehicle and neither vehicle capacities nor maximum tour lengths are violated. Therefore no efficient exact solution methods are available, and the existing solution approaches are of a heuristic nature. In the standard Savings based Ant System [8] one population of Ants for solving the whole problem instance is used. This is reasonable for only small problem

*Corresponding author. E-mail: karl.doerner@univie.ac.at instances. The runtime for large problem instances is prohibitive and the solution quality decreases. Therefore, the main idea is to split up large problems into a number of smaller problems that can be solved both more effectively and more efficiently. We use an own population of ants for each sub-problem. This algorithm works better than the standard Savings based Ant System. Extensive computational results were published in [9]. The design of the algorithm is well suited for the development of a parallel variant. Recently some possible parallelization strategies for ACO have been proposed,which can be classified into *fine-grained* and *coarse-grained* strategies [5]. In fine-grained parallelization strategies usually several artificial ants of a colony are assigned to each processor and therefore frequent information exchange between the small sub-colonies of ants (i.e. an information exchange between the processors) takes place [4]. Coarse-grained parallelization schemes run several colonies in parallel. This strategy is also referred to as *multi colony approach.* The information exchange among colonies is done at certain intervals or numbers of iterations [1].

The outline of the remainder of this paper is as follows. In the next section we give a problem formulation of the VRP. In Section 3 we repeat the Savings based Ant System from [8]. In Section 4 we repeat the decomposition for Vehicle Routing Problems from [9]. We will present results for the speedup and efficiency of our mixed parallelization strategy in section 5 before we conclude with an outlook on future research.

2 Problem Formulation of the Vehicle Routing Problem

The VRP can be formulated in the following way. This formulation is closely related to the formulation presented in [3]. Let $G = (V, E, c)$ be a complete graph, with $n+1$ nodes $(v_0, ..., v_N)$ corresponding to the customers $i = 1, ..., N$ and the depot $i = 0$, and the edge set $((v_i, v_j) \in E \; \forall \; v_i, v_j \in V)$. With each edge $(v_i, v_j) \in E$ is associated a non-negative weight c_{ij}, which refers to the travel costs between nodes v_i and v_j and a non-negative weight t_{ij}, which refers to the distance between the nodes. Furthermore, with each node $v_i, i = 1, ..., N$ is associated a non-negative demand d_i, which has to be satisfied, as well as a service time δ_i. The service time at the depot is set to $\delta_0 = 0$. At the depot a fleet of size K is available, where each vehicle has a capacity of Q^k and the maximum driving time for each vehicle is T^k.

Let x_{ij}^k denote the binary decision variables with the following interpretation:

$$x_{ij}^k = \begin{cases} 1 & \text{if vehicle } k \text{ visits node } v_j \\ & \text{immediately after node } v_i \\ 0 & \text{otherwise.} \end{cases}$$

Then the objective can be written as

$$\mathit{minimize} \sum_{i=0}^{N} \sum_{j=0}^{N} \sum_{k=1}^{K} c_{ij} x_{ij}^k \tag{1}$$

under the following restrictions

$$\sum_{i=1}^{N} \sum_{j=1}^{N} x_{ij}^k d_i \leq Q^k \quad 1 \leq k \leq K \tag{2}$$

$$\sum_{i=0}^{N} \sum_{j=0}^{N} x_{ij}^k (t_{ij} + \delta_i) \leq T^k \quad 1 \leq k \leq K \tag{3}$$

$$\sum_{i=0}^{N} x_{ij}^k - \sum_{l=0}^{N} x_{jl}^k = 0 \quad 1 \leq k \leq K, 0 \leq j \leq N \tag{4}$$

$$\sum_{i=0}^{N} \sum_{k=1}^{K} x_{ij}^k = \begin{cases} 1 & 1 \leq j \leq N \\ K & j = 0 \end{cases} \tag{5}$$

$$\sum_{i \in S} \sum_{j \in S} x_{ij}^k \leq S - 1 \quad \forall S \subseteq \{1, ..., N\}, 1 \leq k \leq K \tag{6}$$

$$x_{ij}^k \in \{0, 1\} \quad 1 \leq k \leq K, 0 \leq i, j \leq N \tag{7}$$

The objective (1) is to minimize the total travel costs. Constraints (2) ensure that no vehicle is overloaded. Constraints (3) require that the maximum driving time for each vehicle is respected. Constraints (4) ensure that if a vehicle visits a customer it also leaves the customer. Constraints (5) require that all customers are visited once, and that the depot is left K times. Subtour elimination is ensured through constraints (6). Finally, constraints (7) are the usual binary constraints.

3 Savings based ACO algorithms for the VRP

The Savings based ACO algorithm published in [8] and repeated here mainly consists of the iteration of three steps: (1) generation of solutions by ants according to private and pheromone information; (2) application of a local search to the ants' solutions, and (3) update of the pheromone information. Solutions are constructed based on the well known Savings Algorithm. In this algorithm the initial solution consists of the assignment of each customer to a separate tour. After that for each pair of customers i and j the following savings values are calculated:

$$s_{ij} = d_{i0} + d_{0j} - d_{ij}, \tag{8}$$

where d_{ij} denotes the distance between locations i and j, the index 0 denotes the depot, and s_{ij} represent the savings of combining two customers i and j on one tour contrary to serving them on two different tours. In the iterative phase, customers or partial tours are combined by sequentially choosing feasible entries from the list of saving values. A combination is infeasible if it violates either the capacity or the tour length constraints. The decision making about combining customers is based on a probabilistic rule taking into account both savings values and the pheromone information. Let τ_{ij} denote the pheromone concentration on the arc connecting customers i and j telling us how good the combination of these two customers i and j was in previous iterations. In each decision step of an ant, we consider the k best combinations still available, where k is a parameter of the algorithm which we will refer to as 'neighborhood' below. Let Ω_k denote the set of k neighbors, i.e. the k feasible combinations (i, j) yielding the largest savings, considered in a given decision step, then the decision rule is given by equation (9).

$$\mathcal{P}_{ij} = \begin{cases} \frac{s_{ij}^{\beta}\tau_{ij}^{\alpha}}{\sum_{(h,l)\in\Omega_k} s_{hl}^{\beta}\tau_{hl}^{\alpha}} & \text{if } (i,j) \in \Omega_k \\ \\ 0 & \text{otherwise.} \end{cases} \tag{9}$$

In (9), $\mathcal{P}_{ij}$ is the probability of choosing to combine customers i and j on one tour, while α and β bias the relative influence of the pheromone trails and the savings values, respectively. This algorithm results in a (sub-)optimal set of tours through all customers, once no more feasible savings values are available. The used pheromone update rule was proposed in [2] and its pheromone management centers around two concepts borrowed from Genetic Algorithms, namely ranking and elitism to deal with the trade-off between exploration and exploitation. In [8] this paradigm was used for solving the VRP. Thus, we will just briefly depict the pheromone update scheme here. Let $0 \leq \rho \leq 1$ be the trail persistence and E the number of elitists. Then, the pheromone update scheme can formally be written as

$$\tau_{ij} := \rho\tau_{ij} + \sum_{r=1}^{E-1} \Delta\tau_{ij}^{r} + E\Delta\tau_{ij}^{*} \tag{10}$$

First, the best solution found by the ants up to the current iteration is updated as if E ants had traversed it. The amount of pheromone laid by the elitists is $\Delta\tau_{ij}^{*} = 1/L^{*}$ if (ij) belongs to the best solution so far, 0 otherwise, where L^{*} is the objective value of the best solution found so far. Second, the $E-1$ best ants of the current iteration are allowed to lay pheromone on the arcs they traversed. The quantity laid by these ants depends on their rank r as well as their solution quality L^{r}, such that the r-th best ant lays $\Delta\tau_{ij}^{r} = (E-r)/L^{r}$ on the arcs it traverses. Arcs belonging to neither of those solutions just face a pheromone decay at the rate $(1-\rho)$, which constitutes the trail evaporation. A solution obtained by the above mentioned procedure can then be subjected to a local search in order to ensure local optimality. In our algorithms we sequentially apply the *swap* neighborhood between tours to improve the clustering and the 2-opt algorithm within tours to improve the routing.

4 Decomposing the VRP

The D-Ant algorithm published in [9] is repeated in the following section. The idea is based on Taillard's decomposition algorithm ([10]).

Initially an Savings based Ant System solves the problem for a given number of iterations. Given the best found solution so far our algorithm determines for each route of this solution the center of gravity. We then cluster these route centers using the Sweep algorithm as proposed by Gillett and Miller [6]. Each of the resulting clusters is then solved independently by applying our Savings based Ant System for a given number of iterations (see Figure 1).

After all sub-problems have been solved we re-assemble the global solution and update the global pheromone information and if applicable the global best solution. The steps described above are repeated until a pre-specified time limit is reached. The detailed pseudocode is given below.

The main notion with respect to the communication between the different processes is *master pheromone information.* In fact, we use one global memory for our algorithm. The communication between the Ant Systems for the sub-problems and the Ant System for the master process is based on

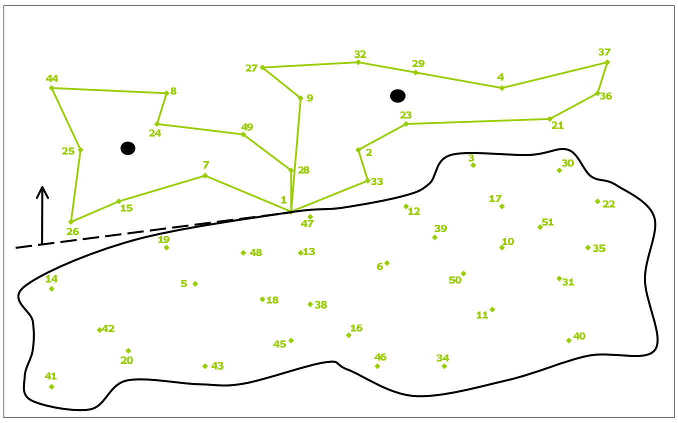

Figure 1: Solve the sub-problems with the Savings based Ant System

two important components. The first important component is, each Ant System which solves a sub-problem is using the associated part of the master pheromone information. In other words, each sub-problem receives only those parts of the master pheromone information necessary to solve its part of the problem. The sub-problems then change this pheromone information *locally* as they iteratively solve their instances. The second important component is, after all the sub-problem have been solved, the corresponding processes return the best found solution. The solution for the problem is computed and compared with the previous best found solution. If an improvement was achieved, the best found solution is updated and pheromone reinforcement in the master pheromone information occurs. Otherwise, only negative reinforcement in terms of pheromone evaporation is applied to the master pheromone information.

5 Parallelization and Computation Experiments

A common procedure for implementing parallel programs (especially on distributed memory machines) is to employ low level message passing systems such as MPI ([7]) in conjunction with a standard sequential language such as Fortran or C. With this approach explicit communication calls for non local data access have to be inserted into the sequential program.

Note that our goal in parallelization of the ACO algorithm is to improve the execution time of the algorithm without altering its behavior. Possible improvements of ACO solution quality through the exploitation of parallelism will be analyzed in another project and are not discussed in this article.

```
procedure D-Ants
begin
  Read the input data;
  Initialize the system (parameters and global pheromone matrix);
  while a pre-specified stopping criterion is not met do
        for a pre-specified number of iterations do
              Solve the complete problem using the Savings based Ant System;
        end
        for the best solution found so far do
              Compute the center of gravity of each route;
        end
        Decompose the best solution into a pre-specified number of subproblems
        by applying the Sweep algorithm to the centers of gravity;
        for each subproblem do
              for a pre-specified number of iterations do
                    Solve the subproblem with the Savings based Ant System by
                    using the relevant part of the global pheromone matrix locally;
                    If applicable, update best solution; end
        end
        Update the global memory (global pheromone matrix);
  end
end
```

Figure 2: Pseudocode of the algorithm

We have parallelized the Savings based Ant System on the basis of individual ants. The results are published in [4]. In this section we report the results when using a mixed parallelization strategy for the D-Ant algorithm. We extended the fine-grained parallelization variant with a coarse-grained strategy. Then, the subpopulations of the decomposed problem are executed in parallel.

We have enriched our C implementation with MPI statements to exploit the parallelism in the algorithm.

5.1 Fine-grained approach

The input data are read by the root process and distributed to other processes. The different processes independently initialize the algorithm and compute the distance matrix. Each process constructs solutions, where the number of constructed solutions by one process is computed by the number of ants divided by the number of processes. As soon as the solutions of all processes are available the E-best solutions are searched and broadcasted to the root process. The reduction operation for the determination of the E-best ants is executed – then the solutions of the E-best ants are broadcasted from the root process to the different processes and the pheromone matrices are updated in each process.

5.2 Coarse-grained approach

The computation of several sub-arrays brings possibility for calculating them in parallel. The user can influence performance by setting the number of sub-problems that should be processed in parallel. Let us denote SUB the number of sub-arrays and SP the number of the subarrays that will be processed in parallel. The number SP must be of course smaller than the number SUB determining the number of sub-problems. After the parallel processing is initiated, the number of all processes divided by SP provides the number of processes available for the fine-grained parallelism of every sub-array.

5.3 Numerical Results

The problem instances used for our computational study are the larger Christofides instances [3]. All of the instances feature one central depot. The instances are capacity constraint as well as tour length constraint. Service times are equal for all customers and set to zero. The sizes of the instances vary between 150 and 199 customers. We use the following parameters: it_{master} denotes the number of iterations for solving the complete problem, it_{sub} denotes the number of iterations for solving the sub-problems, it_{tot} denotes the total number of iterations SUB denotes the number of sub-problems For the results presented below we used the following settings: $it_{master} = 1$, $it_{sub} = 150$. The total number of iterations was set to twice the number of customers. Finally, an important parameter is the number of sub-problems. For the computational study presented below we have partitioned each problem instance into $SUB = 2$ sub-problems. We chose to stick to the parameter values for the D-Ant that were found to be favourable [9]. The experiments with the parallel version of the D-Ants were run on a state-of-the art medium-sized Beowulf cluster comprised of 16 quad-board PCs equipped with Pentium IV processors and connected via Fast-Ethernet. Table 2 reports detailed results in runtime for the instances. The instances 4 and 9 have 150 customers and the instances 5 and 10 have 199 customers.

Table 1 summarizes our computational results. The number of used processors (P) is varied. The SUB indicates if the sub-problems are executed as one process sequentially ($SUB = 1$) or as two different processes ($SUP = 2$). Furthermore, the average time and the average solution quality is reported. It is obvious that for all the different settings we get about the same solution quality. Also the speed-up and the efficiency for using $1, 2, 4$ and 8 processors are reported. When we use 8 processes we have a slightly better speed up (3.85) and efficiency (0.48) by execution the sub-problems in parallel than executing the sub-problems sequentially and using all the 8 processors in a parallel execution of the population of ants (speedup 3.30 – efficiency 0.41).

P	SUB	SP	*Avg. Time*	*Avg. Solution*	*SpeedUp*	$Efficiency$
1	2	1	5,159.44	1,252.36	-	-
2	2	1	3,893.26	1,254.67	1.33	0.66
4	2	1	2,293.68	1,252.64	2.25	0.56
8	2	1	1,561.85	1,253.30	3.30	0.41
2	2	2	4,335.37	1,259.40	1.19	0.60
4	2	2	2,216.50	1,252.08	2.33	0.58
8	2	2	1,338.67	1,252.31	3.85	0.48

Table 1: Resume of the results for different configurations

6 Conclusions

In this paper we reported the efficiency and the speedup for the parallelization of the D-Ant algorithm. We are currently extending our endeavors concerning parallel implementations of ACO algorithms to more sophisticated (asynchronous) models of parallelization. The aim is to come up with more intelligent strategies that will lead to either better efficiency or better effectiveness (or possibly both). Also, the speed-up obtainable through parallelization should be highly important for solving larger problem instances in real time.

P	SUB	SP	instance 4	instance 5	instance 9	instance 10
1	2	1	1,758.14	7,212.14	2,185.82	9,481.65
2	2	1	1,390.54	5,593.39	1,660.97	6,928.13
4	2	1	879.77	3,281.14	1,055.36	3,958,44
8	2	1	720.75	2,214.93	808.93	2,502.81
2	2	2	1,455.72	6,987.41	1,664.09	7,234.26
4	2	2	797.20	3,411.90	940.94	3,715.97
8	2	2	529.52	1,910.34	621.54	2,293.30

Table 2: Detailed results. P denotes the number of parallel processes, SP means a number determining the number of sub-problems that were processed in parallel.

Acknowledgements

We would like to thank Elisabeth Mühlböck for running the numerical tests. The authors are grateful to Guenter Kiechle and Marc Reimann for providing part of the code. This work was supported by the Special Research Program SFB F011 "AURORA" of the Austrian Science Fund FWF.

References

[1] Benkner, S., Doerner, K. F., Hartl R. F., Kiechle, G. and Lucka, M. (2005). *Communication Strategies for Parallel Cooperative Ant Colony Optimization on Clusters and Grids* to appear: Lecture Notes in Computer Science, Conference Proceedings of PARA04.

[2] B. Bullnheimer, R. F. Hartl, and Ch. Strauss. *New Rank Based Version of the Ant System a computational study.* Central European Journal of Operations Research, 7(1):25–38, 1999.

[3] Christofides, N.; Mingozzi, A. and Toth, P.. *The vehicle routing problem.* In: Christofides, N., Mingozzi, A., Toth, P. and Sandi, C. (Eds.): Combinatorial Optimization. Wiley, Chicester (1979).

[4] K. F. Doerner, R. F. Hartl, G. Kiechle, M. Lucka, M. Reimann. *Parallel Ant Systems for the Capacitated Vehicle Routing Problem.*In: Evolutionary Computation in Combinatorial Optimization: 4th European Conference, EvoCOP 2004, LLNS 3004, Lecture Notes in Computer Science, pp. 72–83, Coimbra, Portugal, April 5-7, 2004. Springer.

[5] M. Dorigo, T. Stuetzle. *The Ant Colony Optimization Metaheuristic. Algorihtms, Applications, and Advances.* In: F. Glover, and G. A. Kochenberger, editors, *Handbook of Metaheurisitcs*, pp. 251–285, January 2003. Kluwer.

[6] Gillett, B. and Miller, L. (1974). *A heuristic algorithm for the vehicle dispatch problem.* Operations Research, **22**, 340–349.

[7] *MPI: A Message-passing Interface Standard Version 1.1.*MPI Forum, 1995.

[8] Reimann, M., Stummer, M. and Doerner, K. (2002). *A Savings based Ant System for the Vehicle Routing Problem.* In: Langdon, W. B. et al. (eds.): Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2002), Morgan Kaufmann, San Francisco, 1317–1325.

[9] Reimann, M., Doerner, K. F. and Hartl, R. F. *D-Ants. Savings Based Ants divide and conquer the vehicle routing problem.* Computers & Operations Research, Vol. 31 (4), pp. 563–591.

[10] Taillard, E. D. (1993). *Parallel iterative search methods for vehicle routing problems.* Networks **23** 661–673.