https://doi.org/10.26552/com.C.2009.4.5-8

Maria Lucka – Stanislav Piecka *

MULTI-THREADED ANT COLONY OPTIMIZATION WITH ASYNCHRONOUS COMMUNICATIONS FOR THE VEHICLE ROUTING PROBLEM

In this paper we study behaviour of Ant Colony Optimization algorithm for solving the Vehicle Routing Problem implemented by POSIX Threads in parallel cluster environment. The algorithm is based on a fine-grained parallelism strategy which uses asynchronous communication for cooperation in finding solutions. Our aim is to analyze the effect of proposed method on speedup, execution and communication time with respect to the quality of solution.

Keywords: Ant Colony Optimization, Parallel Metaheuristic, Vehicle Routing Problem, POSIX threads.

1. Introduction

In the field of combinatorial problems, the Vehicle Routing Problem (VRP) introduced by [1] is one of the most challenging. This optimization problem and its variants have multiple applications in telecommunication, transportation and logistics. Unfortunately the majority of these applications belong to NP-hard problems so in the worst case the exponential time is required to find the optimal solution. The VRP problem determines a set of vehicle routes starting and ending at the depot where each customer is visited exactly once. The demand of each customer is satisfied and both maximum tour lengths and vehicle capacities cannot be violated. The objective of the VRP is to minimize the total travel costs. Exact algorithms can be used only for relatively small instances. In practice, all known solutions of larger instances come from heuristic or metaheuristic algorithms. It seems that especially metaheuristic methods produce good quality solutions in relatively short calculation time.

The Ant Colony Optimization method (ACO) developed by [3] has become very successful for solving the VRP problem. The idea of the method is inspired by behaviour of real ants where each ant deposits pheromone on the ground as information that other ants should follow it. Deposited pheromone evaporates in time. Successfulness of ant in finding the food causes that certain path is passed more often and more pheromone is deposited. Therefore it is more likely that other ants choose the same path as well. This behaviour of ants is in computer science modeled iteratively by repeatedly called procedures which create solutions by exploring fully connected graph of customers. The artificial ant makes decision about the way it will continue in every vertex. This decision making is specific for concrete problem and is influenced by two factors: joint memory and heuristic information. When created, best solutions are used for updating of common memory according to the achieved quality. This updating is done after all artificial ants have finished their search. The whole procedure is called repeatedly as many times as required.

In our work we have rewritten the parallel Savings based ACO algorithm for the VRP described in [4] using synchronous communication model with Message Passing Interface to a thread based asynchronous model. We study characteristics of the algorithm when decentralized asynchronous communication is used, on four larger instances: C5 published by [2] and G18, G19, G20 published by [6]. For these instances there are still not exactly calculated optimal solutions.

This paper is organized as follows. In the following two sections we shortly describe the Vehicle Routing Problem and the ACO parallelization strategies and propose the asynchronous algorithm. In Section 4 gained computational results are shown. We present dependence of the gained speedup and efficiency on the number of threads used, whereby the solution quality, execution and communication time are also presented. The last section concludes with several remarks and outlooks concerning the future work.

2. Formulation of the Vehicle Routing Problem

According to [2] we can describe the Vehicle Routing Problem (VRP) as follows: Let $G = (V, E, c)$ be a complete graph which has $n + 1$ 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.

* **Maria Lucka[1], Stanislav Piecka[2]**
[1]Faculty of Education, University of Trnava, Slovakia, E-mail: mlucka@truni.sk
[2]Faculty of Controlling and Informatics, University of Zilina, Slovakia

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 which equals 1 if vehicle k visits node v_j immediately after node v_i and 0 otherwise.

The objective can be written as

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

Using the following restrictions

$$\sum_{i=1}^{N} \sum_{\substack{j=0 \\ j \neq 1}}^{N} x_{ij}^k d_i \leq Q^k \quad 1 \leq k \leq K \tag{2}$$

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

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

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

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

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

The objective (1) is to minimize the total travel costs. Constraints (2) ensure that no vehicle can be overloaded. Constrains (3) require that for each vehicle the maximum driving time is respected. Constraints (4) ensure that vehicle which visits a customer also leaves the customer. Constraints (5) require that all customers are visited exactly once, and that the depot is left K times. Through constraints (6) sub-tour elimination is ensured. Constraints (7) are the usual binary constraints.

3. Parallelization of ACO

Interesting feature of the ACO is its feasibility of parallelization. Each ant makes relatively simple and independent task. The only dependence is caused by using the same pheromone matrix and the same best solution found so far. The first issue is required for decision making process and the second one for measurement of quality of generated solution. We can identify several goals that can be achieved by parallelization of the ACO such as reduction of calculation time, increase of solution quality or speed of convergence. To achieve the reduction of calculation time we can split the colony of ants between processors and let each processor calculate its part of colony. This approach is known as functional decomposition [10]. Instead of this we can apply domain decomposition by dividing customers into subsets and let processors calculate solutions of sub-problems. Generally there are possibilities of synchronous or asynchronous communication between processors. Classification of parallelization of the ACO for the VRP can be found in [4], and [5]. In short, we can use: fine-grained, coarse-grained and mixed parallelization. Fine-grained parallelization splits ants of a colony between processors which often communicate when update the pheromone matrix and exchange the best solutions. By course-grained parallelization schemes run more colonies of ants in parallel whereby each colony is calculated on one processor. The information exchange between processors take place in certain time intervals and concern specified parts of information. The mixed approach is a combination of the first two.

In our approach we used fine-grained parallelization strategy with decentralized approach whereby one ant colony is proportionally divided among several computing threads. An execution thread [9] is a fork of a computer program into more tasks which can run concurrently. Those threads share memory and other resources, but they run independently. Considering the shared memory computational model of threads does not need to change the address space, inter-process communication of threads is faster than that for processes.

We suppose that all threads in our implementation have the same behavior and calculate homogeneous parts of the colony of ants. Number of threads is specified by the number of the processor's cores used. Each core runs exactly one thread. Every thread computes its own pheromone update by using information received from other threads. When possible, an inter-thread communication is done by using shared memory, otherwise the network is used. Concurrent access to the shared memory is secured by critical sections which are implemented by PThread mutexes [9]. Instead of using shared files proposed in [7] we used the user datagram protocol as a communication layer. If a thread has found better solution and has to use the network layer, the user datagram packet is sent only once per cooperating node. All received packets are stored in system buffers and are processed by first thread which reads them. After the packet is processed the thread publishes the best gained solution to all threads in its group in shared memory. This approach does not require dedication of a separate thread for communication. The pseudo-algorithm can be formulated as follows:

```
1:  Initialization;
2:  For i = 1; i <= IT do:
    /* IT is the number of iterations */
        For Ant = 1; Ant <= N / NoT + 1 do:
        /* NoT is the number of threads */
                Create Savings based Ant solution;
        Select elitist Ants;
        Download received solution if exists;
        Update and spread solution if better solution is found;
        Update pheromone matrix if needed;
        Update Savings list if needed;
    End do i;
3:  Finalization;
```

The speedup is defined for measurement of parallelization quality by the following formula:

$$S_p = T_1/T_p \quad (8)$$

where p is the number of processors, T_1 is the execution time of the sequential algorithm and T_p is the execution time of the parallel algorithm with p processors. According to dual-core architecture used in our experiments we are calculating with number of used cores instead of processors. Similarly the efficiency is a performance metric defined by the following formula:

$$E_p = S_p/p = T_1/pT_p \quad (9)$$

This value is typically between zero and one and estimates how well-utilized in solving the problem are processors, compared to how much effort is wasted in synchronization and communication.

4. Computational results

In our experiments we used the cluster consisting of 72 SUN X4100 nodes with two 64-bit dual core processors, each. Therefore we could use at most 4 threads per node working over the common shared memory. Reported results are average values gained over independent 15 runs for all instances. The number of customers is denoted as N and configurations mentioned below are used. Even we experienced better solution quality with different configurations, we used the same parameters settings as proposed in [8] and used in [4] to keep results comparable. We used N artificial ants for each instance, $\alpha = \beta = 5$ and $\sigma = 6$ elitist ants, the evaporation rate $\rho = 0.95$, and the neighborhood size $\lfloor N/2 \rfloor$. We ran the algorithm for 2N iterations for both instances. The algorithm did not send whole pheromone matrix between cores. Only the best σ solutions were chosen, compared with the best solutions found so far and spread between nodes every time better solution was found.

In Table 1 we can see that the time spent by communication does not increase linearly with using more cluster nodes. We can see that efficiency decreases with the increasing number of threads. The achieved efficiency for 32 cores for instance C5 is 0.59 and 0.73 for instance G19. This value is better than published in [4], where the gained efficiency is 0.37 and 0.39, respectively. So we can conclude that asynchronous communications are more suitable for the ACO as synchronous, especially for larger instances. Reduction of speedup of instance C5 against G19 is caused by the fact that time required for creating solution is smaller. Therefore, the ratio between communication and calculation is higher. Efficiency greater than 1 on the C5, G18 and G20 instances is caused by caching effect, where several threads run on separate cores of one processor. We can see that solution quality is decreasing with more threads; this is about 4% on the C15 and 3% on the G19 when 32 threads are used.

5. Conclusions

We presented parallel POSIX Threads based implementation of the ACO method using asynchronous cooperative approach for solving the VRP. We measured its speedup and efficiency in comparison with synchronous approach published in [4]. We showed that asynchronous communication increases efficiency of the ACO algorithm.

In our future work we would like to apply the presented asynchronous approach to the mixed, multi-colony ACO parallelization with focus on increasing quality of the solution. We would like to test the algorithm with different configurations. We also plan to test the dependency of the instance size on efficiency and solution quality in parallel asynchronous ACO algorithms.

Calculated average results according to the number of threads of each measured instance, where V denotes solution quality, t_c denotes the time spent by communication and synchronization calculated per node, t_r denotes the overall execution time including communication. Table 1

Instance	Threads	Nodes	V	t_c [ms]	t_r [s]	Speed	Efficiency
C5 (n=199)	1	1	1377.68	7.96	440.70	1.000	1.000
	2	1	1376.14	16.78	219.11	2.011	1.006
	4	1	1384.03	34.06	113.03	3.899	0.975
	8	2	1396.19	36,52	61.73	7.139	0.892
	16	4	1418.81	36.43	35.48	12.422	0.776
	32	8	1435.79	36.88	23.40	18.834	0.589
G18 (n=300)	1	1	1097.49	12.49	2307.50	1.000	1.000
	2	1	1097.52	24.90	1089.14	2.119	1.059
	4	1	1101.99	62.03	563.80	4.093	1.023
	8	2	1110.80	62.39	289.89	7.960	0.995
	16	4	1123.84	56.31	149.53	15.432	0.964
	32	8	1135.54	52.85	83.65	27.585	0.862
G19 (n=360)	1	1	1498.71	16.77	5825.99	1.000	1.000
	2	1	1501.60	38.45	2929.19	1.989	0.994
	4	1	1495.88	80.55	1470.60	3.962	0.990
	8	2	1509.38	86.44	790.26	7.372	0.922
	16	4	1521.93	86.81	420.89	13.842	0.865
	32	8	1546.23	85.74	250.26	23.280	0.728
G20 (n=420)	1	1	2013.44	18.67	12421.2	1.000	1.000
	2	1	2008.52	43.71	6093.24	2.039	1.019
	4	1	2006.99	98.14	3066.11	4.051	1.013
	8	2	2026.06	101.67	1540.71	8.062	1.008
	16	4	2049.92	91.86	814.73	15.246	0.953
	32	8	2074.81	87.87	436.65	28.447	0.889

Acknowledgement

The authors would like to thank Prof. Dr. Siegfried Benkner, Institute of Scientific Computing, University of Vienna, for the possibility to use the parallel computer cluster LUNA for running their experiments.

References

[1] DANTZIG, G., RAMSER, R.: *The truck dispatching problem.* Management Science 6, pp. 80–91, 1959.
[2] CHRISTOFIDES, N., MINGOZZI, A., TOTH, P.: *The Vehicle Routing Problem.* Combinatorial Optimization, Wiley, Chichester, 1979.
[3] DORIGO, M., GAMBARDELLA, L. M.: *Ant Colony System: A cooperative learning approach to the Travelling Salesman Problem.* IEEE Transactions on Evolutionary Computation, 1(1), pp. 53–66, 1997.
[4] DOERNER, K. F., HARTL, R. F., BENKNER, S., LUCKA, M.: *Cooperative Savings based Ant Colony Optimization – Multiple Search and Decomposition Approaches.* Parallel Processing Letters, 16(3), pp. 351–369, 2006.
[5] CRAINIC, T. G.: *Parallel Solution Methods for Vehicle Routing Problems.* CIRRELT- 2007-28, 2007.
[6] GOLDEN, B. L., WASIL, E. A., KELLEY, J. P., CHAO, K. M.: *The impact of metaheuristics on solving the vehicle routing problem: algorithms, problem sets, and computational results.* Fleet management and logistics, Norwell: Kluwer, 1998.
[7] LUCKA, M., PIECKA, S.: *Parallel POSIX Threads Based Ant Colony Optimization using asynchronous Communications.* Proceeding of 8th International Conference APLIMAT 2009, pp. 613–620, 2009.
[8] REIMANN, M., DOERNER, K. F., HARTL, R. F.: *D-Ants. Savings Based Ants divide and conquer the vehicle routing problem.* Computers & Operations Research, 31 (4) (2004), 563–591.
[9] ANDREWS, G. R.: *Foundations of Multithreaded, Parallel, and Distributed Programming.* Addison-Wesley, Reading, MA, 2000. http://www.cs.arizona.edu/people/greg/mpdbook.
[10] SCHROEDER-PREIKSCHAT, W.: *The logical design of parallel operating systems.* Prentice-Hall, Englewood Cliffs, NJ, pp. 29, 1994.