The Eleventh International Conference on
Engineering Computational Technology
23–25 August, 2022 | Montpellier, France

Proceedings of the Eleventh International Conference on
Engineering Computational Technology
Edited by B.H.V. Topping and P. Iványi
Civil-Comp Conferences, Volume 2, Paper 8.1
Civil-Comp Press, Edinburgh, United Kingdom, 2022, doi: 10.4203/ccc.2.8.1

Solving Optimization Problems with MPI-ACO

J. Banda-Almeida[1,2] and I. Pineda[1,2]

[1]School of Mathematical and Computational Sciences, Yachay Tech University, Urcuquí, Ecuador
[2]Yachay Scientific Computing Group, Ecuador

Abstract

Parallelization of metaheuristics using High-Performance Computing (HPC) provides a suitable environment to approximate large NP-hard combinatorial optimization problems (COPs). The Ant Colony Optimization (ACO) is a population metaheuristic with outstanding time and performance results. This algorithm mimics the indirect communication and self-organization capabilities of ants. In ACO, each ant is an autonomous construction procedure, and builds partial solutions to share with the rest of the colony. The combination of ants results in inherent parallel behaviour that enables the evaluation of complex problems. This behaviour has motivated the creation of algorithms that exploit parallel architectures. This paper accelerates ACO using Message Passing Interface (MPI) and HPC infrastructure. The MPI-ACO parallelization follows the master-slave model with coarse granularity. The algorithm divides the number of ants into different processors that simultaneously create local solutions and iteratively approximate the optimal solution. We evaluate the algorithm using three COPs, the travelling salesman problem (TSP), the job shop scheduling problem (JSP), and image segmentation. Each problem is encoded in MPI-ACO using the appropriate heuristic information and selection policies. The speedup, efficiency, and Karp-Flatt metric are used to evaluate the acceleration of the MPI-ACO using up to 32 cores, demonstrating the scalability of the MPI-ACO algorithm.

Keywords: metaheuristics, parallel computing, high-performance computing, ant colony optimization, message passing interface, combinatorial optimization problems.

1 Introduction

Optimization algorithms are widely used to solve engineering, energy, economics, management, and logistics problems. Soft methods such as metaheuristics can efficiently approximate large and complex optimization problems. Ant Colony Optimization (ACO) is a population metaheuristic inspired by the food foraging process of ants [1]. During this process, ants deposit a chemical known as pheromone on the traversed path from the nest to the food source. The shorter routes accumulate more pheromone and attract other ants, converging in the optimum route.

In the ACO algorithm, artificial ants are simple agents that share a common memory space and build a solution following an iterative process that adds components incrementally to a partial solution [2]. The algorithm consists of initialization, solution construction, and pheromone reinforcement [3]. In the initialization, the problem is encoded with a graph representation, and we configure the initial parameters: pheromone influence, heuristic coefficient, evaporation ratio, and the population size of ants.

In the solution construction phase, ants build a solution following an iterative probabilistic policy that determines the edges chosen by ants at each step, using

$$p_{ij}^{k} = \frac{[\tau_{ij}]^{\alpha}+[\eta_{ij}]^{\beta}}{\sum_{j\in\Lambda}[\tau_{ij}]^{\alpha}+[\eta_{ij}]^{\beta}}, \quad (1)$$

where τ is the pheromone information, η is the heuristic information, and Λ represents the nodes not yet visited. Finally, the pheromone reinforcement process removes some amount of pheromone from the edges traversed by ants to enable the exploration of different paths.

The inherent parallel behaviour of ants allows them to cooperate and solve complex problems. This behaviour has motivated the implementation of parallel ACO algorithms to speed up the evaluation of COPs. Moreover, High-Performance Computing (HPC) can leverage parallel algorithms to overcome traditional computing capabilities limitations.

This work presents a distributed Ant Colony Optimization algorithm with coarse granularity. The parallelization of ACO follows a master-slave model and divides the total number of ants into processing nodes. This division creates several subcolonies that execute the solution construction process independently. The algorithm was evaluated with three discrete combinatorial optimization problems: the TSP, the JSP, and image segmentation. The heuristic information and selection policies are adapted according to each problem constraint. The speedup, efficiency, and Karp-Flatt metric [4] indicate a scalable performance of the MPI-ACO.

2 Methods

The parallelization of the ACO algorithm implements a synchronous communication model with coarse granularity using the MPI library. The master-slave model is the most common process communication approach in message-passing applications [5]. At the beginning of the procedure, the master process initialises the parameters and the pheromone structure. Also, it loads the problem instance and constructs a complete weighted graph. The next step initialises the MPI communicator with k spawned processes. At the beginning of each iteration, a copy of the pheromone matrix τ is broadcasted to each process. Slave processes construct local solutions simultaneously and send back their results. Then, the master process updates the global pheromone structure and broadcasts the information again until the end condition is satisfied.

The implemented algorithm was evaluated on three COPs. The TSP is an NP-hard problem that minimises the total cost of constructing a Hamiltonian cycle on a weighted graph. A TSP instance is represented in the MPI-ACO as a complete graph $G = (V, E)$ where V and E are the sets of nodes and edges, respectively. The heuristic information in this problem corresponds to the inverse of the euclidean distance between the cities.

The JSP is also an NP-hard problem and aims to process a finite set of jobs on a finite set of machines. Each machine can handle only one job at a time, and each job consists of m machine operations with a predefined order and a deterministic processing time. The objective is to find a feasible schedule that minimises the makespan of running all the jobs and satisfying all the constraints. The heuristic value η for these problems is $1/C_{to}$, where C_{to} symbolises the completion time of an operation [7].

Image segmentation consists in extracting a region of interest from an image. Our problem extracts the optic disc (OD) from retinal images. This process automates and increases the precision of detecting several eye diseases [8]. For this problem, the heuristic information corresponds to gradient intensity values. A colony of ants is initialised for each retinal image, and they move to neighbouring pixels until reaching the stop criterion.

We have evaluated the MPI-ACO with the described experimental scenarios using an HPC cluster based on the Nvidia DGX A100 system with a MIMD NUMA architecture and 128 dual cores AMD ROME 7742 with 2.25 GHz.

3 Results

We analyse the behaviour of the MPI-ACO algorithm on the TSP using up to 2, 4, 8, 16 and 32 cores. Table 1 compares the speedup, efficiency, and Karp-Flatt metric between three TSP instances (rat575, rl1889, and fl3795**)** from the TSPLIB

benchmark library [6]. Each value corresponds to the average of 30 executions. The speedup grows steadily from 1 to 8 cores, and then it increases gradually from 8 to 32 cores. The efficiency presents a degradation related to the sequential part of the algorithm. Some procedures, such as the global pheromone update, are performed only by the master process. Therefore, the efficiency of the algorithm decreases continuously. The Karp-Flatt metric shows a non-uniform behaviour as the number of cores increments. This behaviour suggests that the work distribution between processors is not perfectly load-balanced. It is related to defining a fixed colony size without considering the specific number of processors at each time.

Instance	Cores	Speedup	Efficiency	Karp-Flatt
rat575	2	1.92721	0.96360	0.03776
	4	3.63288	0.90822	0.03368
	8	6.85830	0.85728	0.02378
	16	12.1685	0.76053	0.02099
	32	15.98429	0.49950	0.03232
rl1889	2	1.80992	0.90496	0.10501
	4	3.82795	0.95698	0.01498
	8	7.32637	0.91579	0.01313
	16	11.84103	0.74006	0.02341
	32	16.43636	0.51363	0.03054
fl3795	2	1.84950	0.92475	0.08137
	4	3.62668	0.90667	0.03431
	8	6.83410	0.85426	0.02437
	16	11.33247	0.70827	0.02745
	32	17.09831	0.53432	0.02811

Table 1: MPI-ACO evaluation on TSP.

Table 2 summarises the evaluation of JSP. The table shows the speedup, efficiency, and Karp-Flatt of three JSP instances (ta20, ta25, and ta40) using up to 32 cores. The speedup values are lower than the TSP results because of the increased complexity of the JSP problem. The speedup rises slightly with the number of cores because the work is divided into more executing units. However, the efficiency decays significantly with the problem size because the search space increases, and the algorithm is constantly trapped in local minima. The Karp-Flatt metric determines the elements that contribute to efficiency degradation. The values corresponding to the Karp-Flatt metric are nonuniform with the increase in cores.

Instance	Cores	Speedup	Efficiency	Karp-Flatt
ta20	2	1.53894	0.76947	0.29958
	4	3.03597	0.76947	0.10584
	8	4.80084	0.60010	0.09519
	16	5.25612	0.32850	0.13627
	32	5.25612	0.32850	0.12083
ta25	2	1.47105	0.73552	0.35956
	4	2.13451	0.53362	0.29132
	8	3.53994	0.44249	0.17998
	16	4.68174	0.29260	0.16116
	32	4.98962	0.20790	0.16565
ta40	2	1.52749	0.76374	0.30933
	4	2.22525	0.55631	0.26584
	8	3.85673	0.48209	0.15347
	16	3.75218	0.23451	0.21761
	32	4.40606	0.18358	0.19334

Table 2: MPI-ACO evaluation on JSP.

In the image segmentation problem, we perform experimental tests on 290 images from the RIGA [9] dataset and 40 images from the DRIVE [10] dataset. Three resulting images from the segmentation task are depicted in Figure 1. It shows the contours of the optic disc detected, the region of interest and the corresponding ground truths. It can be seen that the ACO-based segmentation algorithm localises and effectively extracts the optic disc on the evaluated images.

The ACO algorithm for OD segmentation presents average metric values over the 80th percentile. The accuracy obtained is 90.2%, the specificity is 91.3%, and the sensitivity corresponds to 87%. The sensitivity value is lower than the other metrics due to the false-positive pixels classified as part of the OD.

4 Conclusions and Contributions

We have implemented a distributed ACO algorithm using MPI. The algorithm was evaluated using three COPs: the TSP, the JSP, and image segmentation using up to 32 execution units on HPC distributed infrastructure. The main contributions of this work are:

- A distributed ACO algorithm that can be adapted to different COPs and scales appropriately when the number of execution units increases.
- A comprehensive evaluation of the parallel algorithm to understand the potential issues and opportunities for maximising speedup.

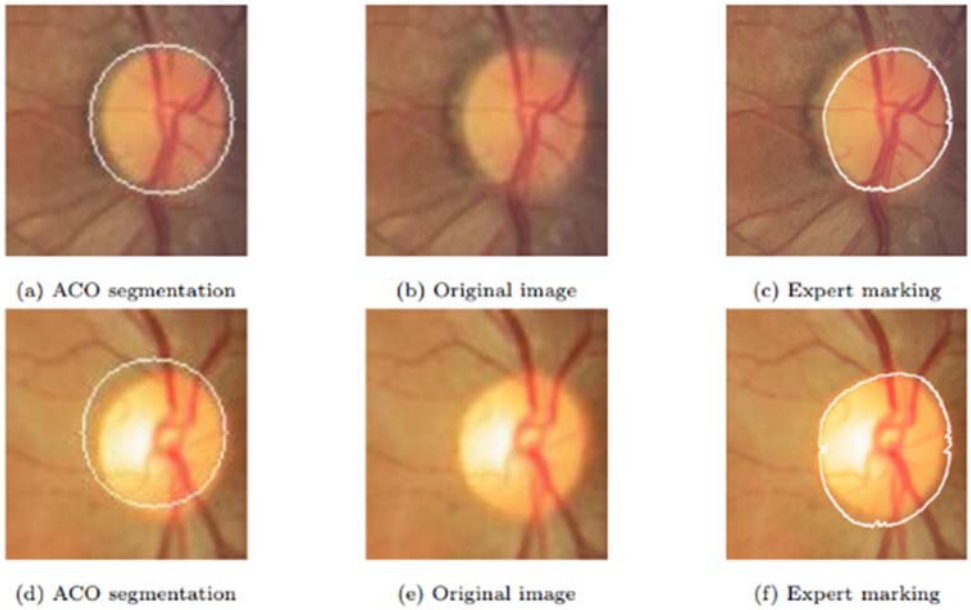

Figure 1: Optic disc segmentation with MPI-ACO.

MPI introduces a significant overhead when we increase the execution units to compute the solutions. This overhead occurs because the master process requires synchronising and managing more resources. The master-slave parallel model implemented in MPI-ACO scales well as each processor communicates only after a complete iteration. However, the drawback of this model is that the whole procedure depends on the master node to collect the pheromone information, update the pheromone structure, and broadcast the new information to all the other processes.

The parallel ACO evaluation demonstrated that the efficiency decreases when more execution units are evaluated. The efficiency degradation follows Amdahl's law that establishes a limit in the acceleration of parallel algorithms due to the inherent sequential part of the program. Moreover, the Karp-Flatt metric allowed us to relate the decrease in efficiency with a non-perfect distribution of the workload between the processes. The TSP problem achieved a higher speedup compared with the JSP. The complexity of JSP requires additional local search procedures to increase its performance. Finally, MPI-ACO accurately finds the OD in the image segmentation problem, demonstrating the flexibility of the algorithm to solve a variety of optimization problems.

References

[1] S.M. Dorigo, G. Di Caro, "The Ant Colony Optimization Meta-Heuristic", InD. Corne, M. Dorigo, and F. Glover, (Editors), New Ideas in Optimization, McGraw-Hill, London, 11–32, 1999.

[2] S.M. Dorigo, T. Stützle, "Ant colony optimization: overview and recent advances", Handbook of metaheuristics, 311-351, 2019.

[3] O. Guarnizo, I. Pineda, “Three Dimensional Adaptive Path Planning Simulation Based On Ant Colony Optimization Algorithm”. 2019 IEEE Latin American Conference on Computational Intelligence, LA-CCI, IEEE, 1-6, 2019.IEEE, 1-6, 2019.
[4] A.H. Karp, H.P. Flatt, "Measuring parallel processor performance." Communications of the ACM 33, no. 5, 539-543, 1990.
[5] C. Grelck, E. Niewiadomska-Szynkiewicz, M. Aldinucci, A. Bracciali, E. Larsson, “Why high-performance modelling and simulation for big data applications matters”, High-Performance Modelling and Simulation Big Data Applications, Springer, 445, 1–35, 2019.
[6] G. Reinelt, TSPLIB95 http://www.iwr.uni-heidelberg.de/groups/comopt/-software/tsplib95/index.html, 2004.
[7] J. Zhang, X. Hu, X Tan, J.H. Zhong, Q. Huang, "Implementation of an ant colony optimization technique for job shop scheduling problem”, Transactions of the Institute of Measurement and Control, 28, 93-108, 2006.
[8] C. Pereira, L. Gonçalves, M. Ferreira, “Optic disc detection in color fundus images using ant colony optimization”. Medical & biological engineering & computing 51, 295–303, 2013.
[9] A. Almazroa, S. Alodhayb, E. Osman, E. Ramadan, M. Hummadi, M. Dlaim, M. Alkatee, K. Raahemifar, V. Lakshminarayanan, “Retinal fundus images for glaucoma analysis: the RIGA dataset”, in “Medical Imaging, Imaging Informatics for Healthcare, Research, and Applications”, International Society for Optics and Photonics, 10579, 2018.
[10] M. Senn, http://www.isi.uu.nl/Research/Databases/DRIVE, 2009.