Improving the Performance of MOACO Algorithms by Applying Colony-Level Parallelization

Antonio M. Mora (1), Ben Paechter (2), Emma Hart (2), Michele Weiland (3)

(1) Departamento de Arquitectura y Tecnología de los Computadores, University of Granada, Spain
(2) School of Computing, Napier University, Scotland
(3) EPCC, University of Edinburgh

amorg@geneura.ugr.es, B.Paechter@napier.ac.uk, E.Hart@napier.ac.uk, mweiland@epcc.ed.ac.uk

Abstract. In this work, the parallelization of some Multi-Objective Ant Colony Optimization (MOACO) algorithms has been performed. The aim is to get a better performance for these classical optimization metaheuristics, not only relating to the running time, usual objective when a parallelization method is applied, but also improving the quality of the yielded solutions.

1. Introduction and preliminary concepts

When one classical method is redesigned to implement a parallel revision of it, usually the aim is to yield good solutions improving the running time. But in addition, the parallelization will imply a different searching scheme in some metaheuristics, as in the case of Ant Colony Optimization [1] algorithms. They are based in a set of artificial agents (ants) which explore the search space cooperating to get the solution for a problem.

In addition the main feature of a good multi-objective (MO) algorithm [2] (devoted to find solutions for more than one objective function) is to get the maximum set of non-dominated solutions, the Pareto Set (PS), those which try to minimize all the functions in the problem. The ACO algorithms implemented to deal with some objectives are known as MOACOs (see [3] for a survey).

So, the idea addressed in this work has been the distribution of the ants into several computation nodes, being each one of these nodes focused into a different area of the searching space. This algorithm structure contributes to yield a better set of results, by including a more explorative behaviour.

Two different parallelization schemes have been tested, considering two different MOACOs from literature [4,5]. Both of them have been implemented considering a parallelization grain at colony level. In addition, the proposed models have been compared with the correspondent sequential approaches, applied to solve the same problem (a bicriteria TSP [6]). The profits in runtime and also in the quality of the solutions yielded have been studied.
2. Description of the method

The implemented model is based in a coarse-grain parallelization, that is, a parallelization at colony level, since every computation node contains a set of ants.

Two different distributed schemes have been proposed:

- **Space Specialized Colonies**: it consists on a group of independent colonies, each of them searching in a different area of the space of solutions. At the end of the process they merge their solutions (their Pareto sub-sets) to constitute a single Pareto Set (considering dominance criteria to build it, since the non-dominated solutions of a colony may be dominated by the solutions yielded by another colony). The split of the space is made through the use of some parameters which weights the objectives in the search for each ant in every colony.

- **Objective Specialized Colonies**: it consists again on a group of independent colonies, but this time, each one tries to optimize only one of the objectives. It does not consider the others in the search, but all of them are taking into account when the solutions are evaluated, so the colonies search, as in the previous model, in a multi-objective space of solutions. Again, at the end, all the PS are merged (considering the dominance criterion) into the final set.

These schemes have been applied considering two well-considered MOACO algorithms from the literature: the Multi-Objective Ant Colony System, MOACS [4], and the BiCriterion Ant, BIANT [5]. In both methods we have used a key parameter in the search (inside the State Transition Rule), $\lambda$, which let us to focus in a concrete area of the search space to explore. So in the first scheme, each colony considers a different value for this parameter.

Both algorithms have been adapted (in the work) to solve the bicriteria TSP.

3. Experiments and results

We have performed some experiments to test the approach value. Firstly each of the algorithms has been run in a single processor, considering the set of configuration parameters obtained through systematic experimentation.

After this, we have test the different approaches (using both schemes) in a different number of processors, considering the same parameters, yielding very good results as it is shown in the next tables and figures.

In Figure 1, it is shown an example of the results yielded by distributing the MOACS algorithm between 11 processors. So there are 11 space specialized colonies whose set of solutions can be seen in Figure 1 (top), in different colours.
Figure 1. Example of the results for MOACS distributed in space specialized colonies (11 processors)
As it is shown, each colony explores a different area of the space of solutions yielding a set of solutions different from the rest. At the end, all these sets are merged, getting a global Pareto Set, as it is show in Figure 1 (down). It is more diverse (overcoat in the edges), and closer to the ideal Pareto Set, than the set obtained by the sequential run (Mono-Proc), which is run considering a variable value for $\lambda$ (one value per ant).

A summary of results can be seen in Figures 2 and 3.

Figure 2. Results for MOACS distributed in space specialized colonies and also in objective specialized colonies, from 1 to 16 processors. They are compared with the mono processor approach.

Where it can be seen that the results are better than the sequential approach ones (mono Proc), but it is necessary to consider at least 8 processors to cover all the search space. Relating to the objectives specialized colonies, they are implemented using 2 processors (one per objective), and the results show that they get very good solutions, but just taking into account one of the objectives, so the solutions are located in the edges of the search space. It will work better in problems with more than 8 objectives, for the same reason that space specialized scheme shows.
The last analysis performed has been the runtime improving due to the parallelization, and the conclusion is clear. It is necessary a bit more time when we consider a higher amount of processors, but the performance relating to the value of the solutions is worthwhile.

The time scalability functions are shown in Figure 4. As can be seen, both algorithms follow the same progression, taking much less time (in average) to get the results for a number of processors lower than 16. In the last case, the average time is closer (but smaller) to the time taken by the sequential approach, but the quality of the set of solutions justifies this distribution.
Figure 4. Results for BIAN T distributed in space specialized colonies and also in objective specialized colonies, from 1 to 16 processors. They are compared with the mono processor approach.

4. Future work

The results yielded in this work are very promising, so the next objective is to implement a fine-grained parallelization approach (at ant level) in order to improve the time performing. The aim is to deal with very large instances of multi-objective problems.

Acknowledgements

This work was carried out under the HPC-EUROPA2 project (project number: 228398) with the support of the European Commission - Capacities Area - Research Infrastructures.

References