Application of parallel metaheuristics to an execution time-power consumption bi-objective problem

José M. Cruz-Zapata, Domingo Giménez, and Daniel Ruiz-García

Departamento de Informática y Sistemas, Universidad de Murcia, Spain.
{josemanuel.cruz,domingo,daniel.ruiz4}@um.es

1 Introduction

This work analyses the solution of a bi-objective optimization problem in which the execution time and the energy consumption of an algorithm in a heterogeneous system are minimized. The processes are assigned to the processors in the system, one process per processor. Each processes-to-processors mapping has associated theoretical models of the execution time and of the energy consumption [2, 3], and the bi-objective execution time-energy consumption problem is to obtain mappings which give pairs (time,energy) at the Pareto front. Genetic algorithms and Particle swarm optimization methods for the problem are developed, with sequential, shared-memory, message-passing and hybrid versions.

The algorithm used as test case is a simple master-slave scheme, and the simulated computational system presents heterogeneity in computation, communication and energy consumption. While the algorithmic scheme, the theoretical models of the time and energy, and the computational system simulated are very simple, the computational technique used to tackle the problem is general, and can be similarly applied in other more complex and realistic situations.

2 Metaheuristics for the bi-objective problem

A mapping is determined by a permutation $\pi = (\pi_0, \pi_1, \ldots, \pi_{p-1})$ of $(0, 1, \ldots, p-1)$, where π_i represents the processor to which process i is assigned. Given a mapping π, the time line of the execution of the algorithm (the p processes) in the particular system (the p processors with the tables representing time and energy consumptions) is simulated, and the modeled execution time and energy consumption corresponding to π are computed from this simulation. The energy consumption is closely related to the execution time, which means normally low execution times give low energy consumptions, which produces Pareto fronts with few pairs. The functions in the metaheuristics are implemented in the traditional way, and some considerations must be taken into account for the adaptation to the problem.

In a Genetic algorithm the individuals to be combined are selected with a roulette method with more probability for individuals with low values of time and energy, for which the probability of each individual is proportional to the inverse of the addition of time and energy. The elements generated are not, in general, permutations, and they are modified to obtain valid individuals. To do so, the repeated values in the element are substituted by values not in the element, which are selected randomly.

Particle swarm algorithms work with populations or sets of particles whose evolution depends on information local to each particle and on global information common to all the particles. Each particle is in a position in the search space, with the position given by the permutation representing the particle, and has a movement speed, which is determined by its position with respect to the best local and global positions.

Parallel versions are developed with an island scheme [1, 4]. In the OpenMP versions, the population of size $|S|$ is divided in subpopulations of $|S|/t$ individuals (t threads are considered). Each thread works independently during g iterations. After g iterations the threads share information from the subpopulations. Each thread stores in mutual exclusion the solutions with the lowest execution time and the lowest energy consumption from all the pairs at its Pareto front. Each thread includes in its subpopulation pairs from this shared structure which are better in time and energy than some pair in its subpopulation. When the iterations finish, the Pareto fronts in the different threads are combined to obtain the final Pareto front. The MPI versions also use the

island model and have a similar structure similar to that of the corresponding OpenMP version, the only difference being the structure with the best elements, which is now replicated.

Experiments with the two metaheuristics give similar results in relation to the goodness of the solutions obtained, and also satisfactory speed-ups for the parallel versions. Figure 1 shows the evolution of the Pareto front when the sequential Genetic algorithm is applied to a problem with 20 processes. A normal behavior is observed, with improvement of the Pareto front when the population size or the number of iterations increase, and with fluctuations due to the randomness of the algorithm. The speed-up achieved with the parallel implementations of the Genetic algorithm when varying the number of processes and threads is shown in Figure 2. The MPI version gives higher speed-up, which is caused by some parts of the algorithm with non linear cost, which produces an important reduction of the execution time when working with small populations.

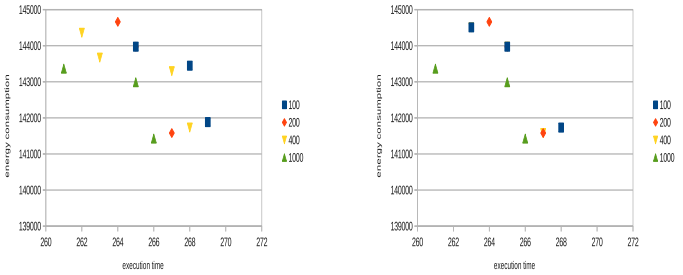

Fig. 1: Evolution of the Pareto front for the sequential Genetic algorithm. A problem with 20 processes is considered. For a population with 1000 individuals when the number of iterations varies (left). With the number of iterations fixed at 1000 and varying the population size (right).

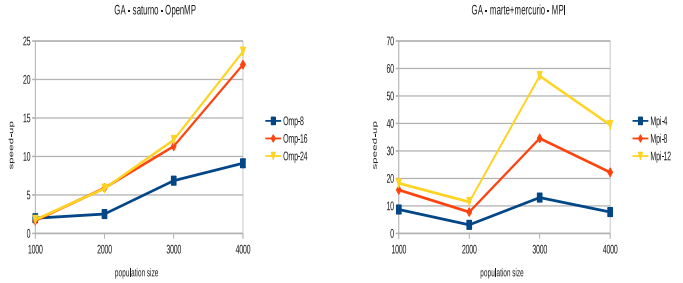

Fig. 2: Speed-up with the Genetic algorithm, for a problem with 40 tasks, 1000 iterations and varying the population size. In a NUMA system with 24 cores (left). In two hexa-cores (right).

Acknowledgements. This work was supported by the Spanish MINECO, as well as European Commission FEDER funds, under grant TIN2012-38341-C04-03.

References

1. E. Alba. *Parallel Metaheuristics: A New Class of Algorithms*. Wiley-Interscience, 2005.
2. F. Almeida, D. González, and L. M. Moreno. The master-slave paradigm on heterogeneous systems: A dynamic programming approach for the optimal mapping. *Journal of Systems Architecture*, 52(2):105–116, 2006.
3. A. Cabrera, F. Almeida, V. Blanco Pérez, and D. Giménez. Analytical modeling of the energy consumption for the high performance Linpack. In *PDP*, pages 343–350, 2013.
4. M.-S. Mezmaz, Y. Kessaci, Y. C. Lee, N. Melab, E.-G. Talbi, A. Y. Zomaya, and D. Tuyttens. A parallel island-based hybrid genetic algorithm for precedence-constrained applications to minimize energy consumption and makespan. In *GRID*, pages 274–281, 2010.