A unified shared-memory scheme for metaheuristics *

F. Almeida[1], D. Giménez[2], and J. J. López-Espín[3]

[1] Departamento de Estadística, Investigación Operativa y Computación, University of La Laguna, Spain
falmeida@ull.es
[2] Departamento de Informática y Sistemas, University of Murcia, Spain
domingo@um.es
[3] Centro de Investigación Operativa, University Miguel Hernández de Elche, Spain
jlopez@umh.es

1 Introduction

The use of a unified scheme for metaheuristics [3] facilitates the easy development of new metaheuristics or hybrid metaheuristics to experiment and adapt them to a particular problem. In the process of obtaining a good-tailored metaheuristic for a problem, it is necessary to experiment with a large number of metaheuristics and parameters, and the time dedicated to the experiments can become very large. To alleviate this problem, parallel versions of the methods can be developed. There are a large number of studies on the parallelization of metaheuristics [1]. Each metaheuristic may have a different parallelization scheme, and some of them could follow a different paradigm. In our approach, and as main contribution, we consider the common development of parallel versions by using a unified metaheuristic scheme to obtain a unified parallel scheme for metaheuristics. Nowadays, most computational systems are formed by multicore components that can be programmed using OpenMP [4]. Thus, this work presents a unified shared-memory metaheuristic scheme implemented on OpenMP, and studies its applicability with different metaheuristics (GRASP, genetic algorithm, scatter search and their combinations) to solve the problem of obtaining Simultaneous Equation Models from a set of values of variables [2].

2 A unified shared-memory scheme for metaheuristics

The unified metaheuristic scheme in algorithm 1-left can be used to develop the corresponding unified shared-memory scheme. Two parallel patterns are identified in the basic functions of the scheme (algorithm 1-right). The first pattern is a data parallelism pattern which appears in the combination and evaluation of parts of the reference set, in functions `Initialize` and `Combine`. The second is a nested parallelism pattern, that can be used in some metaheuristics in the `Initialize` function when it includes an improvement part (greedy or local search), which can be implemented in the same (or similar) way as the improvement in `Improve`. The algorithm 1 scheme has been parallelized by parallelizing each one of the basic functions, using the two identified strategies for functions with the same structure. So, shared-memory versions of GRASP, genetic algorithm, scatter search and their combinations are easily obtained, facilitating the development of parallel versions of the metaheuristics and consequently the experimentation and tailoring of the metaheuristics to the problem we are working on.

3 Preliminary experimental results

The applicability of the methodology has been tested with different problems (knapsack 0/1, task-to-processors assignation...), and the results shown here correspond to the problem of obtaining Simultaneous Equation Models from a set of values of the variables. Similar results have been obtained with the other problems. Experiments have been carried out in nodes with 4 and 8 cores. The 8-core system (8CORE) is a Fujitsu Primergy RXI600 with 4 Dual-Core processors Intel

* The experiments have been carried out in the systems of the Parallel Computing Group at the Polytechnic University of Valencia. Partially supported by Fundación Séneca, Comunidad Autónoma de la Región de Murcia, 08763/PI/08, and Ministerio de Educación of Spain, TIN2008-06570-C04.

```
Initialize(S)                          -->  omp_set_num_threads(one-loop-threads)
while (not EndCondition(S)) {               #pragma omp parallel for
    SS = Select(S)                          loop in solutions
    if(|SS| > 1) SS1 = Combine(SS)            treat-solution
    else SS1 = SS
    SS2 = Improve(SS1)          -->  omp_set_num_threads(first-level-threads)
    S = Include(SS2)                #pragma omp parallel for
}                                   loop in solutions
                                      treat-solution-second-level(first-level-threads)

                                    treat-solution-second-level(first-level-threads):
                                    omp_set_num_threads(second-level-threads(first-level-
                                        threads))
                                    #pragma omp parallel for
                                    loop in element
                                      treat-element
```

Algorithm 1. General metaheuristic scheme

Itanium2 at 1.4 GHz and a shared RAM memory of 8 GByte. The 4-core system (4CORE) is an Intel Core 2 Quad, each core at 2.66 GHz and with a shared RAM memory of 4 GByte. Figure 1 shows the speed-up achieved with different metaheuristics in the two systems. The speed-up has been obtained for each method as the quotient of the execution time with one core with respect to that obtained varying the number of cores. The methods considered are: GRASP, a genetic algorithm (GA), a scatter search (SS), and two hybrid methods with GRASP followed by genetic (GRASP+GA) and scatter search (GRASP+SS). Our goal is to easily obtain parallel versions of multiple metaheuristics and combinations, but the best method and number of cores has to be obtained by experimentation for each particular problem.

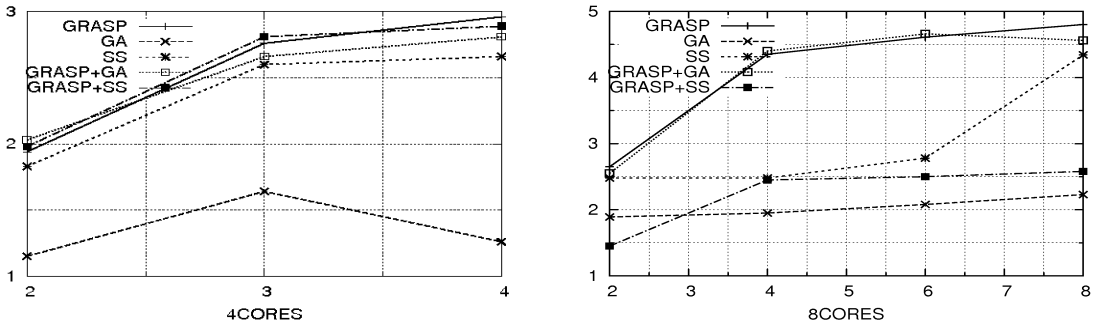

Fig. 1. Speed-up of different metaheuristics, varying the number of threads.

4 Conclusions and future research

The use of a unified parallel metaheuristic scheme allows us to easily develop different metaheuristics and their shared-memory versions. A reduction in the execution time is achieved with the parallel versions, but the reduction depends on the metaheuristic and the parallel system where it is applied. To optimise the parallel versions it is necessary to adjust them to the metaheuristic and the system, but an alternative is to develop a common optimisation strategy applicable to the metaheuristics sharing the common scheme. At present, we are working on the development of that strategy. The development of a unified message-passing scheme is another research line.

References

1. E. Alba. *Parallel Metaheuristics: A New Class of Algorithms.* Wiley-Interscience, 2005.
2. J. J. López-Espín and D. Giménez. Genetic algorithms for simultaneous equation models. In *DCAI*, pages 215–224, 2008.
3. G. R. Raidl. A unified view on hybrid metaheuristics. In *Hybrid Metaheuristics, Third International Workshop, LNCS*, volume 4030, pages 1–12, October 2006.
4. OpenMP web page. `http://openmp.org/wp/`.