Solving a Real-World Structural Optimization Problem With a Distributed SMS-EMOA Algorithm

Francisco Luna*, Gustavo R. Zavala†, Antonio J. Nebro‡, Juan J. Durillo§ and Carlos A. Coello Coello¶
*Departamento de Informática, Universidad Carlos III of Madrid, Leganés, Madrid, SPAIN
e-mail:fluna@inf.uc3m.es
†Khaos Research Group, University of Málaga, SPAIN
email: grz@lcc.uma.es
‡Departamento de Lenguajes y Ciencias de la Computación, University of Málaga, SPAIN
email: antonio@lcc.uma.es
§Institute of Computer Science, University of Innsbruck (AUSTRIA)
email: juan@dps.uibk.ac.at
¶CINVESTAV-IPN, MÉXICO
email: ccoello@cs.cinvestav.mx

***Abstract*—This paper addresses a real-world optimization problem in civil engineering. It lies in the dimensioning of a 162m long bridge composed of 1584 bars so that both its weight and its deformation are to be minimized. Evaluating each possible configuration of the bridge takes several seconds and, as a consequence, running a metaheuristic for several thousands of evaluations would require many days on one single processor. Our approach has been to develop a distributed master/worker version of SMS-EMOA, an indicator-based multi-objective algorithm. By combining the Java implementation of the algorithm in jMetal with the Condor distributed scheduler, we have been able to use more than 350 cores to obtain accurate results in a reasonable amount of time.**

I. Introduction

Civil Engineering is a discipline dealing with the design, construction, and maintenance of bridges, canals, roads, and buildings, as well as many other built environments. In this field, as in other disciplines (telecommunications, biology, finance, etc.), optimization problems constantly arise. In most of the cases, these problems have a multi-objective nature, requiring to optimise two in conflict criteria. As a particular example, if we focus in civil engineering structures, there are usually two goals to be achieved in a project realisation: to minimise the investment (financial cost) and maximise the final design safety.

Our focus on this paper is a real-world civil engineering structural design problem. More specifically, our target is a cable-strayed bridge having two towers from which steel cables hold the bridge deck. Our aim is to design the bridge minimising the overall weight of the whole structure (cost) and the summation of deformations in fixed/certain parts of the deck and towers (safety). As the two aforementioned goals are in conflict, instead of a single design, the final solution consists of a set of designs trading off cost and safety. The total length of the bridge is 162m and it is composed of 837 nodes and 1584 bars. In this work, we adopt a relaxed version of the problem consisting in grouping the bars having the same shape, material, and similar placement in the structure, for reducing the problem complexity. The resulting bi-objective optimisation problem is characterized by 191 variables and 4942 side constraints.

In this work, we rely on multi-objective metaheuristics [1], and more specifically Evolutionary Algorithms (EAs), for computing the set of trade-off designs of the bridge. To this end, we have chosen the *S Metric Selection Evolutionary Multi-Objective Algorithm* (SMS-EMOA) [2], a recently proposed method based on the idea of using quality indicators, in this case the Hypervolume [3], within the reproductive cycle of the algorithm. This kind of indicator based algorithms are gaining popularity in the last few years despite their, in some cases, slowness in comparison with other techniques. In the problem targeted in this work, however, the computing time is dominated by the evaluation of the objective functions determining the cost and safety of each proposed design, being required weeks or event months to complete.

We propose and evaluate in this paper a distributed version of SMS-EMOA for solving the bridge design problem in order to reduce the time to a satisfactory solution. Our approach, based on the master/worker paradigm, takes as a basis the implementation of the sequential algorithm provided in jMetal [4], and extends it for taking advantage of large number of workers. The deployment of our master/worker infrastructure is carried out using the Condor distributed computing software [5], used with success in previous works [6][7]. The outcome of our proposal is a distributed SMS-EMOEA where reproductive cycle is done in the master, and the evaluation of the objective functions are done remotely on the workers. In our experiments, our implementation has run smoothly using up to 350 CPU cores, while providing solutions of high quality. With the aim of assessing its performance, we have compare it with a distributed version of NSGA-II following the same principle.

The list of contributions of this paper can be summarized as follows:

- Design and development of a distributed SMS-EMOA

algorithm using jMetal and Condor.
- To solve a real-world engineering bi-objective optimization problem.
- Deployment and evaluation of our distributed SMS-EMOA algorithm and a distributed version of NSGA-II in a cluster of computers summing up to 350 cores.

The rest of this paper is structured as follows. Section II is devoted to briefly introduce background concepts and related works. The bridge design problem is detailed in Section III. Section IV presents both SMS-EMOA and its parallelization. The obtained results are analyzed in Section V. Finally, Section VI summarizes the paper and outlines some lines of further research.

II. Background and Related Work

In this section, we provide the reader with a brief background of multi-objective optimization problems, metaheuristics, and how these latter algorithms can be paralellized.

Multi-objective optimization refers to the process of optimizing two or more conflicting objective functions of a given problem. By conflicting objective it is meant that improving one of them implies to make the others worse. In this kind of problems, the set of solutions for which do not exist any solution improving all the objective functions is referred as *Pareto optimal Set*, and its corresponding mapping onto the objective space is referred as the *Pareto front*. Solutions within the Pareto optimal set are said to be non-dominated, since none of them is better than the others for all the criteria. More formal and detailed definitions of these concepts can be found in [1][8]. The main goal of multi-objective optimisation is to find an accurate approximation of the Pareto front in terms of convergence (closeness to the Pareto optimal set) and diversity (solutions uniformly distributed along the Pareto front).

Different techniques have been proposed in the research community to address multi-objective optimization problems (MOPs). Unlike classical mathematical programming approaches, metaheuristics [9] in general, and Evolutionary Algorithms (EAs) [10] in particular, have attracted growing attention over the last decade in the multi-objective community because of two main facts [1]. On the one hand, MOEAs are able to approximate the Pareto optimal set in one single run, being capable of dealing with different front shapes. On the other hand, as randomized black-box algorithms, MOEAs can address optimization problems with non-linear, non-differentiable, or noisy objective functions. Most of multi-objective metaheuristic proposals are evolutionary algorithms like NSGA-II [11], SPEA2 [12] or the more recent solvers SMS-EMOA [2] or MOEA/D [13].

Despite their advantages, these algorithms may be computationally expensive. On the one hand, they need to explore larger portions of the search space seeking for the entire Pareto front, and this usually means to perform a lot of function evaluations; on the other hand, and even more important, many real-world problems typically use computationally expensive methods for computing the objective functions and constraints.

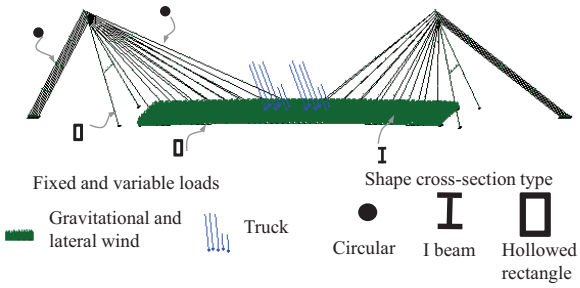

Fig. 1. Description of the bridge and its main components.

These issues are usually addressed in different ways, being the utilization of parallel and distributed computing platforms one of the most popular approaches to speedup the EA search [14]. Due to their population-based approach, EAs are suitable for parallelization because their main operations (i.e., crossover, mutation, and in particular function evaluation) can be carried out independently on different individuals. There is a vast amount of literature on how to parallelize EAs; the reader is referred to [15], [14] for surveys on this topic. In these scenarios where the computation of the objective functions demands high computational resources, the most simple and effective model to parallelize an EA (and a MOEA) is based on the master/worker paradigm [6][7]. This is the approach used in this work to introduce, to the best of our knowledge, the first master/worker parallelization of the SMS-EMOA algorithm. The parallelization has been carefully designed to enable the algorithm to profit from hundreds of workers (up to 350 here).

III. Description of the Problem

The target problem of our work is the design of a cable-strayed bridge with two pillars (towers), which is illustrated in Fig. 1. The bridge has a total length of 162m and the deck length and width are 90m and 9m, respectively. It has a two-way roadway and a pedestrian circulation lane.

From a mechanical point of view, the structure is composed of 837 nodes and 1584 elements, and two materials having different elastic properties are used. The element can have one out three different cross-sections: hollowed rectangle, I-beam, and circular.

The bridge design is formulated as a bi-objective problem. The first objective is to minimize the total weight of the structure; the second one is to minimize the summation of the deformations in certain points of the deck and the columns. To simplify the problem, we optimize a group of 56 bars having into account the design, size and structural performance (see Table I for a summary of the details). The elements composing the groups have the same shape, material and similar position in the structure. The total number of decision variables is 191, which correspond to the geometrical size of the cross-section. The total number of side-constraints is 4948 (the sum of 3 mechanical constraints per element, 4 geometrical constraints

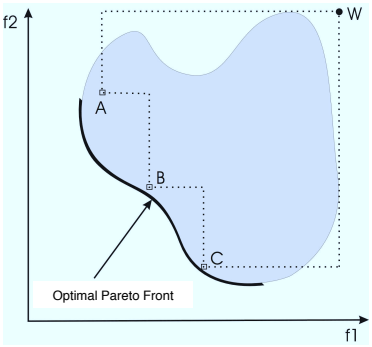

Fig. 2. Hypervolume enclosed by the non-dominated solutions A, B, and C.

per groups of I-beam and hollowed rectangle elements, and 16 deflection constraints).

IV. Parallel SMS-EMOA

This section is devoted to presenting the SMS-EMOA algorithm and the strategy used for its parallelization. Additinally, the last part of the section describes the encoding of the tentative solutions and the genetic operators used by SMS-EMOA.

A. SMS-EMOA

As commented before, the chosen algorithm to address the optimization of the bridge design problem is SMS-EMOA. The general idea is to use a quality indicator to guide the search of the algorithm. In other words, the algorithm aims to compute a Pareto front optimising the vlue of that quality indicator. SMS-EMOA makes use of the Hypervolume, which is commonly accepted as one of the most reliable metric to assess the performance of multi-objective algorithms.

The Hypervolume indicator, illustrated in Fig. 2, calculates the volume, in objective function space, covered by members of a non-dominated set of solutions $Q = A, B, C$ and a reference point W, e.g., the region enclosed into the discontinuous line in the figure, for problems where all the objectives are to be minimized [3]. Mathematically, for each solution $i \in Q$, a hypercube v_i is constructed with a W and the solution i as the diagonal corners of the hypercube. The reference point can be simply found by constructing a vector with the worst objective function values. Thereafter, the union of all hypercubes is computed and its hypervolume (I_{HV}) is calculated as:

$$I_{HV} = \text{volume}\left(\bigcup_{i=1}^{|Q|} v_i\right). \quad (1)$$

The higher the value of I_{HV}, the better the approximated Pareto front is.

SMS-EMOA is based on the NSGA-II algorithm, but introduces two main modifications: firstly, SMS-EMOA is a steady-state evolutionary algorithm, while NSGA-II is a generational one; secondly, instead of using the crowding distance as density estimator, SMS-EMOA considers the contribution of the solutions to the hypervolume of the current approximation front. This way, in every iteration the algorithm discards the solution contributing the less to the hypervolume. Algorithm 1 includes the Pseudo-code of SMS-EMOA.

Algorithm 1 Pseudo-code of the SMS-EMOA algorithm

```
population ← GenerateInitialPopulation()
while (not stopping condition is met) do
    parents ← selection(pop)
    newSol ← SBX(parents)
    newSol ← PolynomialMut(newSol)
    auxPop ← population ∪{newSol}
    computeHVcontributions(auxPop)
    population ← removeLessHVContributor(auxPop)
end while
return population
```

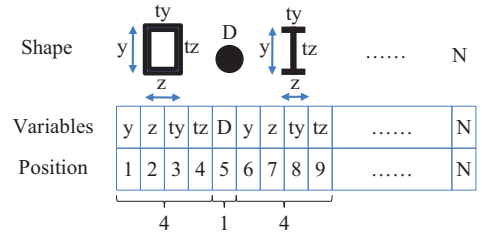

Fig. 3. Encoding of solutions.

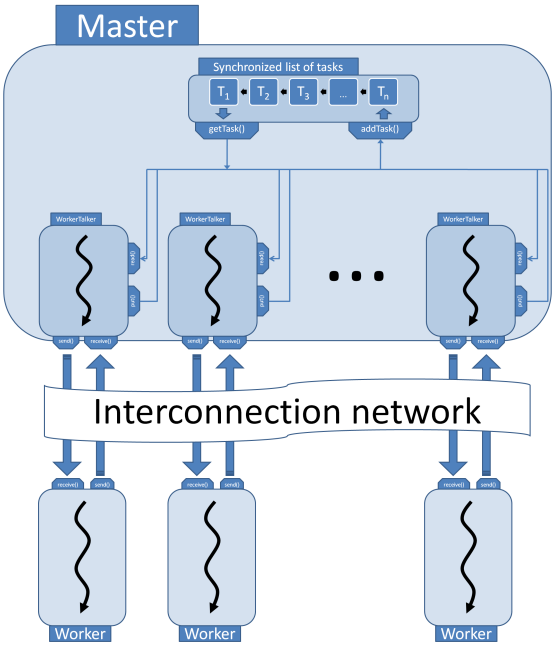

Fig. 4. Underlying parallel software architecture for both SMS-EMOA and NSGA-II

The encoding used to represent the solutions is depicted in Fig. 3. Each bar, depending on their shape, has a number of parameters to optimize; these parameters (diameter, height, etc,.) are values in domain of real numbers. Therefore, in this work SMS-EMOA considers the simulated binary (SBX) crossover and polynomial mutation, typically used when dealing with real-valued codifications.

B. Parallel algorithms

The parallelization of a steaty-state algorithm such as SMS-EMOA poses several challenges since only one solution is generated in the evolutionary loop of the algorithm. In this

TABLE I
VARIABLES AND CONSTRAINS

Shape	Elements	Groups	Variables	Geom. constraints (per group)	Mech. constraints (per element)	Defl. contraints
Circle	42	11	1		3	
I-beam	1 392	41	4	4	3	16
Hollowed rectangle	150	4	4	4	3	
Total	1 584	56	191	4948		

case, only one processor is required at most, thus being the parallelism only applicable to the computation of the objective function, but not to the algorithm itself. Our approach to enable SMS-EMOA to profit from a (potentially large) parallel or distributed computing platform has been to break down the synchronization requirements imposed by its evolutionary loop.

The general architecture of the parallel algorithm is outlined in Fig. 4. The idea is to follow a master/worker distributed scheme. As it can be seen, a multi-threaded master has been devised where there is a talker thread handling the communication with each worker. Talkers and workers communicate via tasks, which are just containers of tentative solutions to be evaluated remotely. Tasks are stored in a shared FIFO list which concurrently accessed by all the talkers (in mutual exclusion). A talker can both, remove and add, tasks to this list. If this list gets empty it means that all the tasks have been processed, i.e., all the solutions have been evaluated by the workers, and the master stops (as well as all the workers).

Algorithm 2 Pseudo-code of the master thread

```
1: population ← GenerateInitialPopulation()
2: taskList ← addTasks(population)
3: threads ← runTalkerThreads(population, taskList)
4: waitForAllThreadsToComplete(threads)
```

The mapping of the parallel SMS-EMOA onto this architecture is as follows. Initially, the master randomly generates as many parallel tasks (with the initial solutions) as the population size to create the initial population (lines 1 and 2 in Algorithm 2). That is, the evaluation of the initial population is also performed in parallel. Then, the processing is moved to the talker threads which manage the communications concurrently with the workers (thus, the master can profit from multi-core computers). These talkers pick tasks from the list, send them to the workers (using sockets), and wait for the evaluated individual to be returned. The operation in the workers is fairly simple: upon reception of a task, the enclosed solution is extracted, evaluated, and sent back to the master (a pseudo-code is shown in Algorithm 3).

Algorithm 3 Pseudo-code of a worker

```
1: while (not receive finalization notification) do
2:     task ← receiveFromTalker()
3:     solution ← task.solution
4:     evaluate(solution)
5:     newTask ← new Task(solution)
6:     sendToTalker(newTask)
7: end while
```

The two key issues of the implementation are: on the one hand, all the MOEA components (population, archive, etc.) are shared by all the talker threads; on the other hand, all the genetic operations within the evolutionary loop are performed in parallel as shown in Algorithm 4. A talker gets (and removes) the first task of the master's task list (line 2), sends it out to its corresponding worker for evaluation (line 3), and waits for the result (line 4). Then, the SMS-EMOA loop starts. It does not matter whether the solution belongs to a past iteration or not, the algorithm proceeds as in sequential. When a newly evaluated solution arrives, its contribution to the hypervolumen indicator is measured. This also implies recomputing the contributions of the solutions already in the population. A ranking procedure is applied and the solution in the population that less contribute to the hypervolume is removed so as to keep the population size constant. If there are more evaluations to perform and there are not enough tasks in the shared list, then a new individual is generated by crossover and mutation, and packaged into a task for its evaluation in a remote worker. Again, we have to remark that all these operations are performed concurrently by all the talker threads in the master node, thus using the appropriate mechanism for a reliable access to all the structures. With such an implementation, it can be seen that any number of workers may be involved in the parallel computation, regardless of the value of the population size of SMS-EMOA.

Algorithm 4 Pseudo-code of a talker thread

```
Require: population // the current SMS-EMOA population
Require: taskList // the list with the task to be remotely executed
 1: while (not stopping condition is met) do
 2:     task ← getTask(taskList)
 3:     sendToWorker(task)
 4:     processedTask ← receiveFromWorker()
 5:     s ← processedTaks.solution
 6:     auxPop ← population ∪ {s}
 7:     computeRanks(auxPop)
 8:     computeHVcontributions(auxPop)
 9:     population ← removeLessHVContributor(auxPop)
10:     if (taskList.size ≤ numWorkers) then
11:         parents ← selection(pop)
12:         newSol ← SBX(parents)
13:         newSol ← PolynomialMut(newSol)
14:         newTask ← new Task(newSol)
15:         taksList.add(newTaks)
16:     end if
17: end while
```

As to the NSGA-II parallelization developed to serve as a comparison basis, we have used the asynchronous generational NSGA-II, $NSGA{-}II^{asy}_{gen}$, version presented in [16] in order to

use the most standard, though parallel, version of NSGA-II. It operates in a generational fashion as it waits to fill the auxiliary population to proceed to the next generation, but generating as many individuals as workers in the parallel platform.

In both distributed algorithms, the Condor system is used to deploy the workers, which notably simplify finding idle cores. It is worth mentioning that, once familiarized with this distributed platform and with jMetal, developing a distributed version of a metaheuristic such as SMS-EMOA roughly takes about an hour, which is an additional advantage of our parallel infrastructure.

V. Experimentation

The commonly accepted methodology to assess the performance of multi-objective metaheuristics consists in performing a number of independent runs (a minimum of 30) and then to apply quality indicators measuring convergence and diversity; in this research work we have relaxed these issues. On the one hand, to run the distributed SMS-EMOA and NSGA-II we have needed a considerable amount of CPUs (cores) from our teaching labs, so we could not use them all the time; hence, we have completed only five independent runs at the time of writing this paper. On the other hand, we do not know the optimal Pareto front of the bridge design problem, being our interest to find an approximation set of valid solutions acceptable for a decision maker with expertise in the problem domain; this a challenge given the high number of side-constraints of the problem.

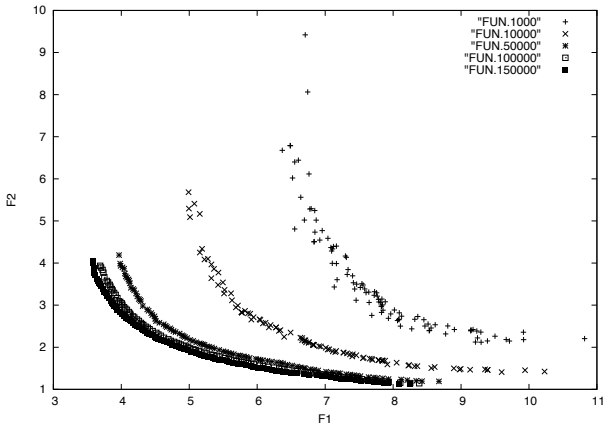

Fig. 5. Pareto Front approximation after 1000, 10000, 50000, 100000, and 150000 function evaluations.

After some pilot tests, we set the stopping condition to compute 150000 function evaluations. Figure 5 shows the Pareto front approximations obtained in one of the runs, depicting the solutions found at 1000, 10000, 50000, 100000, and 150000 evaluations. In particular, we include the solutions satisfying all the constraints. We observe that many solutions are dominated at the beginning of the iterations of the algorithm; however, the front is composed only of non-dominated solutions from 50000 evaluations. The shapes of the solution sets with 100000 and 150000 evaluations suggest that the algorithm has almost converged.

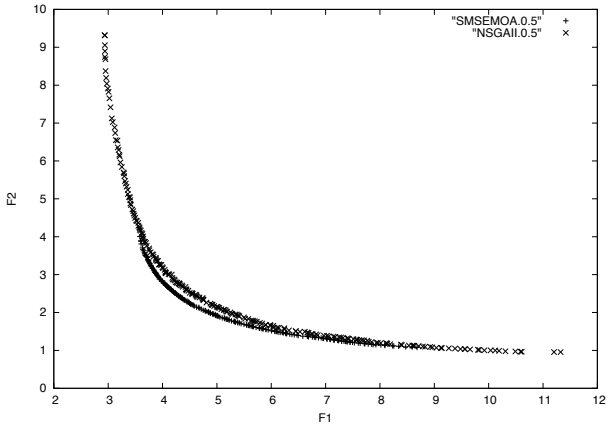

Fig. 6. Attainment surfaces of the Pareto front approximations yielded by SMS-EMOA and NSGA-II.

To analyze the performance of the evaluated algorithms we include the empirical attainment surfaces (EAF) [17] obtained with the approximated fronts computed in the five independent runs of each of them in Fig. 6. An EAF graph displays the expected performance and its variability over multiple runs of a multi-objective algorithm. In short, an EAF is a function α from the objective space $\mathbb{R}^n$ to the interval $[0, 1]$ that estimates for each vector in the objective space the probability of being dominated by the approximated Pareto front of one single run of the multi-objective algorithm. We can observe that SMS-EMOA is able to achieve a greater degree of convergence in the middle of the front of solutions while NSGA-II produces a wider coverage towards the extreme regions, so it is not trivial to decide which algorithm performs the best. An explanation of these differences is that the solutions in the central regions of the fronts usually have a higher contribution to the hypervolume than those in the extreme regions, so SMS-EMOA tends to discard solutions from these zones.

An important outcome of our research is the time reduction achieved with the proposed distributed algorithm. Having into account that we have used up to 350 cores and that the computers in the labs have different hardware features, we can make a rough estimation of the speed up of the parallel computations. The average wall clock time of running the algorithms until they have made 150000 function evaluations was about 10 hours. Assuming that the full set of cores have been used during that time, the total computing time would be 3500 hours, which means that 3500/24=146 days would have been needed by a sequential algorithm.

To illustrate the kind of solutions reported by the distributed SMS-EMOA, we include in Fig. 7 a partial view of the bridge according to a selected solution from the Pareto front approximation. We can observe the concrete dimensions of the elements of the bridge, which is the information that is useful to the civil engineer (i.e., the specialist in the problem).

VI. Conclusions and Future Work

We have faced solving a real-world problem from the civil engineering domain. In particular, the problem is the design of a cable-strayed bridge, in which two objectives are to be

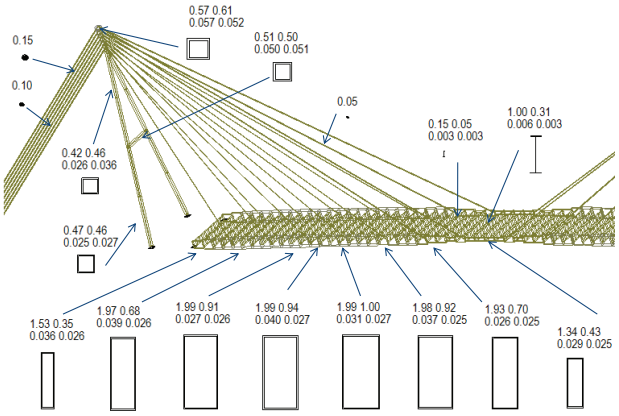

Fig. 7. Description of the bridge and its main components.

minimized: the total weight and the deformation in concrete points. The resulting optimization problem has 191 variables and 4948 side-constraints.

Our approach to optimize the problem has been to employ a modern multi-objective metaheuristic, SMS-EMOA. As the evaluation of each solution requires several seconds, we have developed a distributed version of the algorithm, with the goal in mind of taking advantage of hundreds of processors/cores. Taking the Java implementation of SMS-EMOA provided by the jMetal framework, we have applied a master/worker parallel model to SMS-EMOA, yielding as a result to a distributed algorithm where the master rules the logic of the metaheuristic and the workers perform the evaluations in parallel. By using the Condor system to deploy the workers, we have been able to use up to 350 cores, which has allowed us to get in about ten hours Pareto front approximations which otherwise would had required more that one hundred days of computation in a single computer.

For comparison purposes, we have compared the distributed SMS-EMOA with a distributed NSGA-II algorithm. The results indicated that the first algorithm provided fronts with better convergence, while the latter returned set of solutions with a wider coverage. Our hypothesis is that the hypervolume indicator, used by SMS-EMOA to guide the search, promotes finding solutions in the central part of the front. This suggests as future research line to study parameter setting configurations of SMS-EMOA intended to foster the exploration in the extreme regions by adjusting the reference point used to calculate the hypervolume.

Other lines of further research include developing distributed versions of other state-of-the-art multi-objective algorithms (e.g., MOEA/D) and comparing them with our distributed SMS-EMOA. These techniques could be also applied to other engineering optimization problems requiring of parallel computing power to be solved in a short time.

Acknowledgment

Francisco Luna acknowledges support by the Spanish Ministry of Science and Innovation and FEDER under contract TIN2011-28336. Antonio J. Nebro is supported by projects TIN2011-25840 (Spanish Ministry of Education and Science) and P11-TIC-7529 (Innovation, Science and Enterprise Ministry of the regional government of the Junta de Andalucía). Juan J. Durillo acknowledges the Austrian Research Promotion Agency under contract nr. 834307 (AutoCore). Carlos A. Coello Coello acknowledges support from CONACyT project no. 103570.

References

[1] C. A. Coello Coello, G. B. Lamont, and D. A. Van Veldhuizen, *Evolutionary Algorithms for Solving Multi-Objective Problems*, 2nd ed. New York: Springer, September 2007.

[2] N. Beume, B. Naujoks, and M. Emmerich, "Sms-emoa: Multiobjective selection based on dominated hypervolume," *European Journal of Operational Research*, vol. 181, no. 3, pp. 1653–1669, 2007.

[3] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach," *IEEE Trans. Evolutionary Computation*, vol. 3, no. 4, pp. 257–271, 1999.

[4] J. J. Durillo and A. J. Nebro, "jmetal: A java framework for multi-objective optimization," *Adv. Eng. Softw.*, vol. 42, no. 10, pp. 760–771, 2011.

[5] D. Thain, T. Tannenbaum, and M. Livny, "Condor and the Grid," in *Grid Computing: Making the Global Infrastructure a Reality*, F. Berman, G. Fox, and T. Hey, Eds. John Wiley & Sons Inc., December 2002.

[6] A. J. Nebro, G. Luque, F. Luna, and E. Alba, "Dna fragment assembly using a grid-based genetic algorithm," *Computers & Operations Research*, vol. 35, no. 9, pp. 2776–2790, 2008.

[7] F. Luna, A. J. Nebro, E. Alba, and J. J. Durillo, "Solving large-scale real-world telecommunication problems using a grid-based genetic algorithm," *Engineering Optimization*, vol. 40, no. 11, pp. 1067–1084, November 2008.

[8] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*. New York, NY, USA: John Wiley & Sons, Inc., 2001.

[9] C. Blum and A. Roli, "Metaheuristics in combinatorial optimization: Overview and conceptual comparison," *ACM Computing Surveys*, vol. 35, no. 3, pp. 268–308, 2003.

[10] T. Bäck, D. Fogel, and Z. Michalewicz, Eds., *Handbook of evolutionary computation*. Oxford University Press, 1997.

[11] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[12] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength pareto evolutionary algorithm," Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, Tech. Rep. 103, 2001.

[13] Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Trans. Evolutionary Computation*, vol. 11, no. 6, pp. 712–731, 2007.

[14] G. Luque and E. Alba, *Parallel Genetic Algorithms*, ser. Studies in Computational Intelligence. Springer, 2011, vol. 367.

[15] E. Alba and M. Tomassini, "Parallelism and Evolutionary Algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 5, pp. 443–462, 2002.

[16] J. J. Durillo, A. J. Nebro, F. Luna, and E. Alba, "A Study of Master-Slave Approaches to Parallelize NSGA-II," in *IEEE Int. Symp. on Parallel and Distributed Processing, 2008 - IPDPS 2008*, 2008, pp. 1–8.

[17] J. Knowles, "A summary-attainment-surface plotting method for visualizing the performance of stochastic multiobjective optimizers," in *5th Int. Conf. on Intelligent Systems Design and Applications (ISDA'05)*, 2005, pp. 552 – 557.