Computing and Informatics, Vol. 31, 2012, 31–43

STELLARATOR OPTIMIZATION USING A DISTRIBUTED SWARM INTELLIGENCE-BASED ALGORITHM

Antonio Gómez-Iglesias, Francisco Castejón

National Fusion Laboratory
CIEMAT
Avda. Complutense 22
28040, Madrid, Spain
e-mail: {antonio.gomez, francisco.castejon}@ciemat.es

Miguel A. Vega-Rodríguez

Dep. of Technologies of Computers and Communications
Escuela Politecnica. Campus Universitario s/n 10003, Caceres, Spain
e-mail: mavega@unex.es

Abstract. The design of enhanced fusion devices constitutes a key element for the development of fusion as a commercial source of energy. Stellarator optimization presents high computational requirements because of the complexity of the numerical methods needed as well as the size of the solution space regarding all the possible configurations satisfying the characteristics of a feasible reactor. The size of the solution space does not allow to explore every single feasible configuration. Hence, a metaheuristic approach is used to achieve optimized configurations without evaluating the whole solution space. In this paper we present a distributed algorithm that mimics the foraging behaviour of bees. This behaviour has manifested its efficiency in dealing with complex problems.

Keywords: Large-scale optimization problems, swarm intelligence, distributed computing

1 INTRODUCTION

Nuclear fusion research presents open problems which must be solved in order to be able to design commercial reactors. Some of the limitations of the fusion research are due to the knowledge required to carry out some experiments; but other limitations are related to the computational requirements of the problems being solved. Thus, several of the challenges regarding nuclear fusion can be classified as large-scale problems. The techniques required to solve some of these issues may be challenging.

In the case of optimization problems, the main issues are related to the exploration of the solution space and the exploitation of approximated solutions for the problem being solved. The use of distributed metaheuristics is an excellent approach for these optimizations. Swarm intelligence-based algorithms [1], as for example those based on bees, can be implemented using the capabilities of large computational infrastructures like the grid to carry out large-scale optimization problems [2]. Large-scale problems present various peculiarities such as long execution times or large requirements in terms of storage. In the case of the problem presented here, the execution time needed for a single evaluation makes the optimization process challenging.

Swarm intelligence-based algorithms constitute a large research area in which the optimization techniques try to simulate the behaviour of swarms (originally of insects, but not limited to those animals). More specifically, they try to imitate specific characteristics of the swarm as, for example, the bees foraging behaviour. Honey bees forage for nectar in a changing environment. Discovery of new food sources may mean abandoning others previously known. Bees are able to adapt their behaviour to these changes to maximize their productivity.

Swarm intelligence based algorithms consist typically of a population of individuals (or agents) interacting with each other and with the environment. The individuals must follow a set or rules, usually very simple. The knowledge about the problem is distributed among all the individuals. However, the interaction between those individuals creates a global intelligent behaviour. This global intelligence is unknown to the different individuals, since it is extracted from all the knowledge in the entire population. This is known as self-organisation in social systems [1].

Bees constitute a hierarchical society in which the role of each individual is defined. The division of labour improves the efficiency of the colony [3]. Individuals do not usually switch among tasks, being specialized on a given task. Scouts search for new sources and provide information to the rest of the colony. This is performed by means of a decentralized system without any global decision-making. Individuals select the source with the best ration of gain to cost from all of the available nectar sources.

The decisions bees make regarding their movements are based on the communication among bees [4]. Individuals can move in groups in which a bee plays a leader role. This bee recruits more individuals in the colony during the waggle dance. In this dance, the leader exchange information with other bees waiting in the colony

about food sources. This information is related to the location and quality of the sources. Bees can reach an agreement and follow the leader. However, the recruits will not have all the information about the source. Hence, they will follow the leader but will introduce small changes in their paths. This helps the exploration of the space and also prevents from over-harvesting.

The rest of this paper is organized as follows: Section 2 summarizes our previous efforts and introduces a novel algorithm based on the foraging behaviour of bees while Section 3 details the grid-based implementation of the algorithm. Section 4 explains the problem being optimized, whereas the results of this optimization are shown in Section 5. Finally, Section 6 concludes the paper, summarizes the main achievements, and proposes some future lines of work.

2 OPTIMIZATION ALGORITHM

2.1 Previous Efforts

Previously, we focused on the adaptation and use of well-known optimization techniques [5]. Genetic Algorithms (GAs) and Scatter Search (SS) algorithm were considered.

2.1.1 Genetic Algorithms

We implemented three different GAs regarding the operator used to create a new offspring and the replacement mechanism. The three implementations achieved optimal results in terms of optimization but the usage of the computational results was far from optimal. The reason is that GAs consider a large number of candidate solutions to be generated and, only when all of the solutions have been evaluated, a new set of solutions is generated. This model leads to bottlenecks since the algorithm can be waiting for a long time for a single solution to finish without performing any other computation. The implementation of GAs showed to be easy for being implemented in a heterogeneous and distributed infrastructure following a master-slave model.

2.1.2 Scatter Search

The SS algorithm was developed following the implementation proposed by Martí [6]. This implementation was adapted to the grid and achieved a better optimization than those obtained with the GAs. The number of candidate solutions being handled at any time is smaller in this case, making the relevance of bottlenecks smaller. However, due to the iterative model of the algorithm and the number of improvement processes required, the number of bottlenecks and dependencies among jobs increased. Thus, the usage of resources was also poor.

2.2 Distributed and Asynchronous Algorithm

Due to the issues explained, we decided to implement a new algorithm, specifically designed for being executed on distributed platforms, taking advantage of the characteristics of these infrastructures. The synchronization required by the previous algorithms is replaced by an asynchronous model. In this a new model, the creation of new solution does not require the evaluation of another candidate solution. This generation is performed based on the status of the exploration of the solution space and the exploitation of optimal solutions.

The algorithm is called Distributed and Asynchronous Bees (DAB) algorithm since it is based on the foraging behaviour of bees and follows an asynchronous and decentralized model. It has been designed to run on large and distributed computational platforms. The algorithm has proved to be an optimal and efficient system to carry out large-scale optimization problems in distributed environments [7].

Four different types of bees, or processes, can be found in the algorithm:

- Two levels of *scouts* (devoted to the exploration of the solution space):
  1. *Rovers*: they use diversification methods to explore the solution space.
  2. *Cubs* (associated to a *rover* bee): random exploration changing variables based on a good solution, in terms of dispersion, found by a *rover*.
- Two levels of *employed* (devoted to the exploitation of known solutions):
  1. *Elites*: perform a wide search using an approximated solution previously found by a *scout*.
  2. *Workers* (associated to an *elite* bee): by using a local search procedure [8], they explore in-depth the best solution found by *elite* bees.

The pseudocode of the resulting DAB algorithm is as shown in Algorithm 1. As can be seen, not all of the bees start foraging simultaneously. The creation of bees depends on the evolution of the optimisation process. Furthermore (and not included in Algorithm 1), the DAB algorithm checks whether the bees (jobs) are performing their tasks. In case of failure on the grid infrastructure, the number of bees or jobs can be automatically readjusted. Moreover, if the user changes the configuration of the algorithm during its execution, the number of processes can be also modified to adapt the algorithm to the new configuration.

2.2.1 Exploration

In order to perform an optimal exploration of the solution space, the resources devoted to this task try to explore the most diverse areas of the space. Thus, they generate a set of possible solutions and select the one with the highest distance to the previously evaluated solutions. This distance is calculated after performing a normalization of the variables involved in the optimization. The selected solution will then be evaluated.

Algorithm 1: The DAB Algorithm Pseudocode

```
1  Initialise the population of solutions and create bees;
2  while Stop criterion not reached do
3      foreach Evaluate solution x by bee B do
4          Obtain the probability of the x to be selected;
5          if x is the global best then
6              if B is an elite then
7                  Create new workers;
8              else if Idle Elites then
9                  Create elite;
10         else if B is a Rover then
11             if Idle Cubs then
12                 Create new solutions with an in-depth local search;
13             Send B to explore new solutions;
14         else if Idle Elites then
15             Create elite;
16     Create new solution for the bee B based on its type;
```

During the generation of new solutions the algorithm must ensure that the value of each variable is within the limits of that variable for the problem being solved. If the value is out of the limits, the algorithm introduces a new value using a boundary factor. This helps enhancing the exploration of the solution space.

2.2.2 Exploitation

The exploitation is based on approximated solutions previously found by means of the exploration processes. In the tests presented here, the *elite* bees use a local search over the improved solutions considering a wide modification of the variables defining the solution. The *workers* introduce smaller modifications to perform a more focused enhancement.

2.2.3 Parameters of the Algorithm

Taking into account the previous characteristics of the algorithm, and also considering the requirements of the problem being solved, the parameters of the algorithm are as shown in Table 1. While the parameters regarding the number of resources devoted to the optimization will exist for any problem, the other parameters have been introduced based on the problem being solved in this work.

Parameter	Symbol	Description
Grid Related		
Elites Number	EN	Maximum number of elites
Workers Number	WN	Maximum number of workers – per elite
Rovers Number	RN	Maximum number of rovers
Cubs Number	CN	Maximum number of cubs – per rover
Optimization Related		
Modification Rate	MR	Probability to modify a variable when a candidate solution is generated
Elite Solutions	ES	Maximum number of elite solutions considered
Candidate Factor	τ	Constant to modify the variables of new solutions
Boundaries Factor	φ	Constant to modify the variables of new solutions to fix within their boundaries
Workers Factor	ψ	Constant to modify the variables of new solutions when a Local Search is performed

Table 1. Parameters of the algorithm

2.2.4 Specialization

While in our previous efforts the exploitation of the known approximated solutions was based on a mutation process using the standard deviation of the variables and local searches, in the current research we only consider different levels of local searches.

Taking into account the division of labour and the specialization, as well as the fact that bees do not switch among tasks, the final design of the algorithm uses two different types of processes exploiting optimized solutions and two other procedures exploring the solution space.

2.2.5 Dynamic Reconfiguration

The algorithm can be reconfigured at any time just by modifying an XML file. Any of the configuration parameters can be changed including the maximum number of resources being used, the local search modification factor, or those related to the grid infrastructure itself. For example, the preferred CEs or SEs or the rank criteria to order the CEs can be specified.

Moreover, the algorithm itself is able to change the configuration related to the grid infrastructure based on the quality of service being provided by the sites on the VO.

3 GRID IMPLEMENTATION

The algorithm has been developed in Python. It has dependencies with the NumPy library and the lightweight DOM (Document Object Model) implementation library.

Depending on the version of the Python interpreter installed on the User Interface (UI) of the grid infrastructure it might be necessary to download and compile those libraries since they are not included in the old versions.

The algorithm makes use of a hash table to store the explored solutions in order to avoid evaluating twice the same configuration. A binary search tree is included in the table for collision resolution.

3.1 Long-time Jobs

In order to reduce the impact of the time waiting in queues on the final execution time, we have introduced the idea of long-life jobs in the design. These jobs read the SE waiting for input and, as soon as this input is available, they process the information and store the result in the SE. Another process will use this information and produce new input data. The jobs will be running as long as the security constraints of the grid infrastructure will allow. Only two types of processes (*rovers* and *elites*) use this feature. The other processes are short jobs which just evaluate a single input and finish. Using the long-time jobs for all submitted tasks may lead to problems with the LFC (Logical File Catalogue).

3.2 WN-UI Communication

As mentioned, the long-time jobs regularly require of new input. They ask a master process for this input. This master process runs on the UI of the grid. The implementation of this communication uses the LFC: the remote process updates a file on a location known by the local and the remote tasks, the master receives this update, downloads the data generated by the remote element, generates new input, and stores the input using a given path of the LFC.

In order to reduce the number of checks that the local process has to perform, all the remote processes update the same file. Hence, a barrier system has been developed to avoid that two or more tasks update the file at the same time overwriting the information stored on that file. Thus, the LFC and the SEs of the infrastructure are used as a shared-memory component of the infrastructure.

4 STELLARATOR OPTIMIZATION

The problem being solved in the enhancement of the magnetic confinement of plasma in a fusion device [9, 10]. By improving this confinement, the plasma may become more stable and the probabilities for obtaining fusion reactions increase. Therefore, the efficiency of the device increases. To perform this optimization, we consider the improvement of the Fourier modes describing the plasma boundary.

4.1 Costs of the Optimization Process

The fitness function of this problem requires the execution of an application workflow (shown in Figure 1) [11]. The required execution time of the workflow depends on the characteristics being modelled. For optimized configurations, this workflow has 40 458 792 Millions of Instructions (MI). This value changes depending on the input configuration because the grid used by the main code involved in the process can have more data. This code (called VMEC, Variational Moments Equilibrium Code [12]) could require of more iterations to find the equilibrium and the output file can have more data, so the objective function can take longer. Thus, the execution time may vary from several minutes to hours, showing a large variability. This variability introduces challenges in the optimization algorithm, since asynchronism is required and the communication among tasks needs to be performed without leading to continuous bottlenecks.

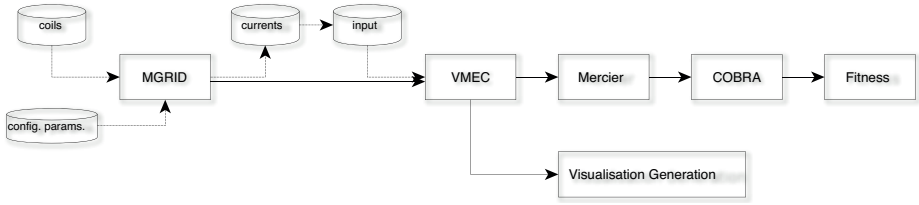

Fig. 1. Workflow to measure the quality of the confinement

The workflow only requires one CPU, although some of the components of the workflow are suitable for a parallel implementation. Since 32 and 64-bit machines may be found in the production grid infrastructures used, the applications involved in the optimization process must deal with these different architectures. Distinct versions of the libraries required by the applications are included in the optimization process.

4.1.1 Bandwidth

Load balancing is critical in other optimization algorithms running on the grid [13]. For the DAB algorithm this load balancing is not crucial, as synchronization is not required among the different processes involved in the optimization.

The amount of information transmitted through the network is high in this algorithm. Every bee must send not only the small files to communicate with the colony, but must also store all the results in the SE. The size of a single result is 5.5 MB. Since during an optimization process thousands of results are generated, various GBs are transferred from the WN to the SE. All this data has been previously analysed, which also shows the necessity of distributed environments for this problem.

4.1.2 CPU

As previously noted, the workflow required to calculate the fitness function of interest takes different time depending on the configuration being evaluated. One of the applications (VMEC) implements a Levenberg-Marquardt algorithm.

After VMEC, the workflow executes the fitness function that is given by Equation (1). This function uses the magnetic surfaces (i) of the confined plasma, and the intensity of the magnetic field (B) at each of these surfaces. The values involved in the expression are extracted from the output of VMEC. This function provides a value used for measuring the quality of the magnetic confinement of plasma in the fusion device. The physics involved in this expression, as well as the explanations of why this function is relevant can be also found in [11].

$$F_{\textit{fitness function}} = \sum_{i=1}^{N} \left\langle \left| \frac{\vec{B} \times \vec{\nabla} |B|}{B^3} \right| \right\rangle_i \tag{1}$$

Once the fitness value has been calculated, the workflow analyses the Mercier [14] and ballooning stabilities of the configuration [15]. If both criteria are satisfied, the configuration is valid.

5 RESULTS

Fusion VO has been used to carry out this optimization process. It is a production infrastructure running gLite with resources distributed through several locations in Europe.

Table 2 shows the configuration used for the optimization being carried out.

Grid Related		Optimization Related	
Parameter	Value	Parameter	Value
Elite Number	50	MR	0.20
Worker Number	8	ES	200
Rover Number	5	τ	0.0005
Cub Number	4	φ	0.02
		ψ	0.0001

Table 2. Parameters of the tests

5.1 Computational Results

The aggregated execution time of all the evaluations was 5 287 hours, with a wall time of 120 hours, which implies a speed-up of 44.1. During the execution of one of the tests performed (shown in Figure 2) the number of computational resources

was changed to show the automatic reconfiguration of the algorithm. It can be seen how the number of elite bees is increased up to 100 and then reduced again to 50.

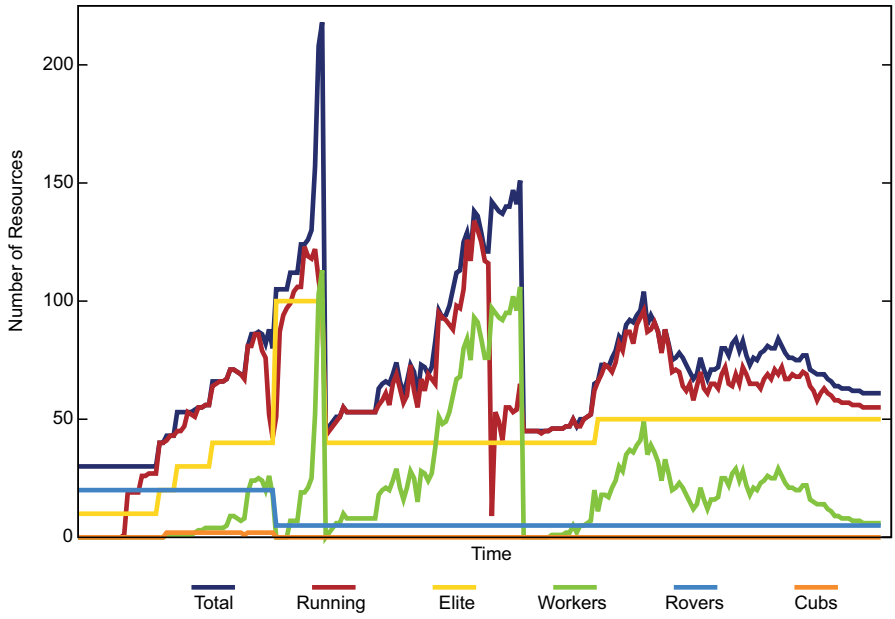

Fig. 2. Evolution of the number of bees

The variability on the number of tasks running was caused by a problem with the WMS (Workload Management System). As can be seen, the system reconfigured the number of resources and continued the execution automatically. It waited until all the workers had finished, without creating new tasks of this type. This reduced the load of the WMS.

5.2 Optimization Results

The first configuration found in the optimization process had a value of 156 106 868 for the expression in Equation (1). In this case, the algorithm does not consider an existing configuration to start the optimization, although any suitable configuration could be used as starting point. This latter scenario becomes especially interesting when the algorithm has been previously executed and a set of suitable and optimized configurations has been created. The best configuration achieved after the optimization processes carried out had a value of 1 717 129 for the same expression. Table 3 shows the best values found for different tests, revealing the reproducibility of the experiments. The different speed-up is due to the different evolution, leading to a different number of jobs being submitted, and also to the usage level of the grid infrastructure.

	Best Result	Wall Time	Execution Time	Speed-up
Test 1	1 739 083	120:08:41	5 296:12:58	44.1
Test 2	1 799 798	120:00:37	5 356:23:21	44.6
Test 3	1 769 937	120:01:05	5 005:58:49	41.7
Test 4	1 794 375	120:11:10	5 311:00:07	44.2
Test 5	1 717 129	120:03:31	5 287:10:25	44.1

Table 3. Tests performed

Figure 3 shows the cross-section of the confined plasma for the best configuration found. Each of the lines in the figure represents a magnetic surfaces, which corresponds to the values of i in Equation (1).

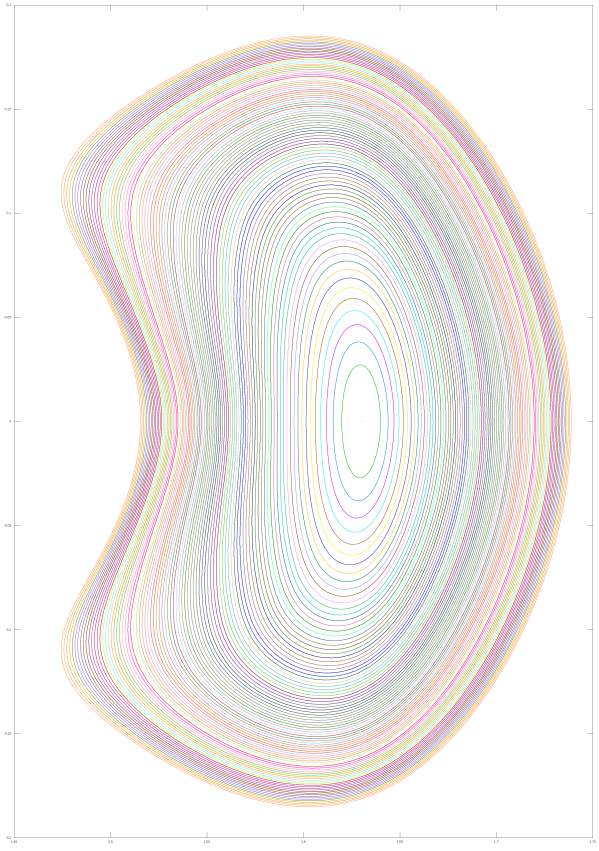

Fig. 3. Cross-section of the optimized configuration

6 CONCLUSIONS

As shown, the use of metaheuristics with the grid permits the achievement of optimized results in a reasonable time. However, the design of the optimization technique needs to consider the special features of the computational platform in which the enhancement process will take place.

Large-scale optimization problems present some characteristics that make challenging to design and develop automated optimization systems.

The optimization achieved demonstrates how the combination of large-scale computational platforms and distributed metaheuristics represents an optimal approach to carry out optimizations of large-scale problems.

As future work we plan on performing further optimizations and adding more target functions. This will lead to configurations improved considering additional relevant physics characteristics. The complexity of the optimizations will also increase. A new version of the algorithm adapted to use CREAM-CE is being developed [16]. Direct submission to the CE should avoid situations of problems with the load of the WMS.

Acknowledgements

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement number RI-261323 (EGI-InsPIRE).

REFERENCES

[1] Bonabeau, E.—Dorigo, M.—Theraulaz, G.: Swarm Intelligence: From Natural to Artificial Systems. Oxford 1999.

[2] Foster, I.—Kesselman, C.: The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann Publishers 1998.

[3] Robinson, G. E.: Regulation of Division of Labor in Insect Societies. Annual Review of Entomology, Vol. 37, 1992, No. 1, pp. 637–665.

[4] Couzin, I. D.—Krause, J.—Franks, N. R.—Levin, S. A: Effective Leadership and Decision-Making in Animal Groups on the Move. Nature, Vol. 433, 2005, No. 7025, pp. 513–516.

[5] Gómez-Iglesias, A.—Vega-Rodríguez, M. A.–Castejón, F.—Cárdenas-Montes, M.—Morales-Ramos, E.—Reynolds, J. M.: Grid-Based Metaheuristics to Improve a Nuclear Fusion Device. Concurrency and Computation: Practice and Experience, Vol. 22, 2009, No. 11, pp. 1476–1493.

[6] Laguna, M.—Martí, R.: Scatter Search, Methodology and Implementations. Kluwer Academic Publishers 2003.

[7] Gómez-Iglesias, A.–Vega-Rodríguez, M. A.–Castejón, F.—Cárdenas-Montes, M.: Distributed and Asynchronous Bees Algorithm: An Efficient Model for

Large Scale Problems Optimizations. In Advances in Intelligent and Soft Computing, Springer 2010, pp. 381–388.

[8] Hoos, H. H.–Stützle, T.: Stochastic Local Search: Foundations and Applications. Morgan Kaufman Publishers 2005.

[9] Bellan, P. M.: Fundamentals of Plasma Physics. Cambridge University Press 2006.

[10] Miyamoto, K.: Fundamentals of Plasma Physics and Controlled Fusion. Iwanami Book Service Center 1997.

[11] Gómez-Iglesias, A.–Vega-Rodríguez, M. A.–Castejón-Magaña, F.—Rubio-Solar, M.—Cárdenas-Montes, C.: Grid Computing in Order to Implement a Three-Dimensional Magnetohydrodynamic Equilibrium Solver for Plasma Confinement. in PDP 2008, pp. 435–439.

[12] Hirshman, S. P.—Neilson, G. H.: External Inductance of an Axisymmetric Plasma. Physics of Fluids, Vol. 29, 1986, No. 3, pp. 790–793.

[13] Luna, F.—Nebro, A. J.—Alba, E.: Observations in Using Grid-Enabled Technologies for Solving Multi-Objective Optimization Problems. Parallel Computing, Vol. 32, 2006, No. 5-6, pp. 377–393.

[14] Hegna, C. C.—Nakajima, N.: On the Stability of Mercier and Ballooning Modes in Stellarator Configurations. Physics of Plasmas, Vol. 5, 1998, No. 5, pp. 1336–1344.

[15] Sanchez, R.—Hirshman, S. P.—Whitson, J. C.—Ware, A. S.: COBRA: An Optimized Code for Fast Analysis of Ideal Ballooning Stability of Three-Dimensional Magnetic Equilibria. Journal of Computational Physics, Vol. 161, 2000, No. 2, pp. 576–588.

[16] Aiftimiei, C. et al.: Design and Implementation of the gLite CREAM Job Management Service. Future Generation Computer Systems, Vol. 26, No. 4, pp. 654–667.

Antonio Gómez-Iglesias is a research scientist in the National Fusion Laboratory in Madrid, Spain, and his main research lines include the use of evolutionary algorithms to optimise nuclear fusion devices by means of distributed environments such as grid computing, and porting and exploitation of different fusion codes in different computing paradigms. He received a M. Sc. in computing engineering in 2004 from the University of Extremadura, Spain, and is now working towards his Ph. D. in computer science at the same University. He has been involved on different projects funded by the 7^{th} Framework Programme.