2010 18th Euromicro Conference on Parallel, Distributed and Network-based Processing

Artificial Bee Colony Inspired Algorithm Applied to Fusion Research in a Grid Computing Environment

Antonio Gómez-Iglesias*, Miguel A. Vega-Rodríguez†, Francisco Castejón*,
Miguel Cárdenas-Montes‡ and Enrique Morales-Ramos†
*National Fusion Laboratory - CIEMAT. Avda. Complutense, 22, Madrid 28040, Spain
Email: {antonio.gomez, francisco.castejon}@ciemat.es
†University of Extremadura, Cáceres, Spain
Email: mavega@unex.es, enmorales@alumnos.unex.es
‡CIEMAT. Avda. Complutense, 22, Madrid 28040, Spain
Email: miguel.cardenas@ciemat.es

***Abstract*—Artificial Bee Colony (ABC) algorithm is an optimisation algorithm based on the intelligent behaviour of honey bee swarm. In this work, ABC algorithm is used to optimise the equilibrium of confined plasma in a nuclear fusion device. Plasma physics research for fusion still presents open problems that need a large computing capacity to be solved. This optimisation process is a long time consuming process so an environment like grid computing has to be used, thus the first step is to adapt and extend the ABC algorithm to use the grid capabilities. In this work we present a modification of the original ABC algorithm, its adaption to a grid computing environment and the application of this algorithm to the equilibrium optimisation process.**

***Index Terms*—Artificial Bee Colony, Grid Computing, Equilibrium, Nuclear Fusion.**

I. Introduction

THE ABC algorithm is a distributed asynchronous process where different bees - computational resources - are looking for flowers - approximated solutions - and every time a good flower is found, more bees are allocated for similar flowers - more computational resources to explore solutions close to an approximated solution found - after the dancing of the bees in the colony, the process where bees exchange information. While it mimics the behaviour of swarms of honey bees, the algorithm uses concepts of swarm intelligence. In the case of grid computing, the computational resources are Working Node (WN) - in the terminology of gLite - and the waggle dance - sharing of information about the direction and distance to paths of flowers - is performed in the User Interface (UI), where the colony is. And, as long as we plan to improve the equilibrium of a nuclear fusion device, a flower is a configuration of the device; a good flower will be an approximated configuration of the fusion device where the equilibrium is better than the existing in current devices.

In the case of optimisation problems related to many scientific areas, the required time to get a fitness value is so long that traditional computation cannot be used to perform an exploration of the entire solution space. Modern paradigms, like grid computing, offer the computational resources and capabilities to carry out these optimisation problems, but they are not easy to use [1][2][3]. Until now, many investigations have been carried out in order to use parallel architectures and GAs (Genetic Algorithms) [4], but the coupling of grid capabilities and complex metaheuristics is still in an immature state. Furthermore, the use of swarm intelligence has not been yet deeply used in grid environments [5].

The grid has created a way that could potentially lead to an increase in the performance of these types of algorithms in terms of execution time and problem size. Nevertheless, a high level of expertise is required to develop and execute grid applications because many problems can arise due to the specifics of the grid infrastructure. Our goal consists of improving the equilibrium of plasma in a nuclear fusion device - TJ-II, a stellarator [6] located in Madrid - by means of the ABC algorithm and grid computing.

The rest of this paper is organised as follows: section II summarizes some of the related works and previous efforts done. Section III introduces some basic concepts of the ABC algorithm while section IV gives a description of how we have adapted the algorithm to the grid. Section V explains what equilibrium means and shows the required workflow to optimise nuclear fusion devices. In section VI, we collect the results obtained in our experiments. Finally, in section VII, we conclude the paper and summarise a variety of perspectives of the explained work.

II. Related Work

There are different efforts to use metaheuristics with the grid and apply these developments to several scientific fields. Many different investigations [4][7][8][9] have been carried out using the grid and metaheuristics and how to apply them to different scientific areas or investigations about how to implement a generic implementation that can be used to different purposes [10], although we have not found any related work where the objective function takes so long time as in our case, this being one of the critical problems we have to overcome.

There are also different frameworks that allow to implement and deploy parallel and distributed metaheuristics, and most of them are restricted to only parallel and distributed evolutionary algorithms [11][12] or using several different metaheuristics although they are not running on the grid [13].

We have developed our own solution, that using grid computing, allows us to solve large scale scientific computational

1066-6192/10 $26.00 © 2010 IEEE
DOI 10.1109/PDP.2010.50

problems. We started with the implementation of different genetic algorithms [14][15] that, using the capabilities of grid computing, tried to improve plasma confinement. The required time to get optimised configurations was still long and the improvement, even when it showed to represent a clear advantage compared to existing configurations, was clearly worse than the one obtained with our next implementation [16]. In that case, we used an evolutionary algorithm (scatter search algorithm [17]) adapted to the grid.

III. Artificial Bee Colony Algorithm

A. Original Algorithm

The ABC algorithm was first introduced by Karaboga in 2005 [18]. According to the original implementation, and following the behaviour of a bee colony, there are three different bees in the algorithm:

1) *Employed*: these bees evaluate an approximated configuration. They are useful to maintain certain levels of convergence in the system.
2) *Scouts*: they explore the solution space. There are different techniques to explore the solution space. They introduce the dispersion component into the system.
3) *Onlookers*: these bees evaluate the results obtained by *employed* and *scouts* bees and choose next configurations to evaluate. They are the master elements of the system. The communication between the different bees - *waggle dance* according to the behaviour of bees in nature - is performed in the master process.

ABC algorithm has been used in different areas [19][20][21], with several modifications, but always with the same basic concepts and steps orginally introduced by Karaboga.

The algorithm assumes that there is only one *employed* for every food source. The *employed* of an abandoned food source becomes a *scout* and it becomes *employed* as soon as it finds a new food source. We have modified this approach taking into account the computational characteristics of the grid and the kind of problems that we plan to face.

B. Extensions

Taking into account the grid paradigm, the number of computational resources and the access to these resources, we have extended the original algorithm by adding:

- Two levels of *employed*:
  1) *Elites*: perform a wide search using an approximated configuration. This approximated configuration has been previously found by a *scout*.
  2) *Workers* (associated to an *elite* bee): by using a local search procedure, they explore deeply the best configuration found by *elite* bees.
- Two levels of *scouts*:
  1) *Rovers*: they use diversification methods to explore the solution space.
  2) *Cubs* (associated to a *rover* bee): random exploration changing configuration parameters based on a good solution, in terms of dispersion, found by a *rover*.

The different bees, and how they are associated, can be seen in Figure 1.

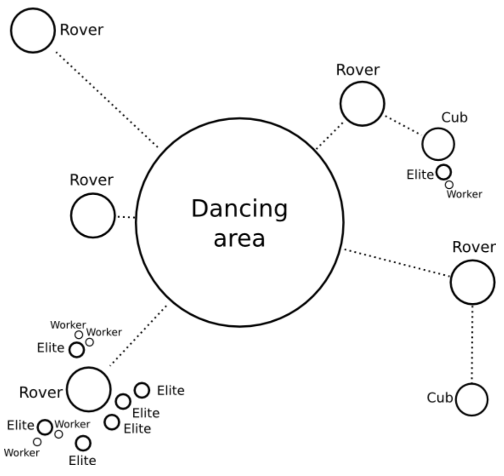

Fig. 1. ABC extension with four different kind of bees.

Although this is an iterative algorithm, our idea is to achieve a distributed asynchronous algorithm where, as soon as a new approximated solution is found, new computational resources are allocated to explore new solutions, without waiting for a single step to finish, since in our implementation the steps do not exist anymore.

C. Characteristics

Each type of bee is created and maintained using different criteria always trying to maintain certain levels of convergence and dispersion to get an optimal exploration of the solution space.

1) Creation of new elites: We have chosen a mutation-based procedure to mix individuals and obtain a new one. This mutation-based procedure uses the sample standard deviation of each chromosome in the whole population to perform the mutation using the best configuration found in the moment this procedure takes place.Each selected chromosome is added or subtracted with a value between 0 and the standard deviation for that value.

2) Creation of new workers: The *workers* are created using an improvement method of the best *elite* found. This improvement method we have to use cannot perform many evaluations because of the long execution time needed by each of these evaluations. For this reason, we have developed a local search method that introduces small variations in the values of the solution and checks if the solution is better than the original one.

3) Creation of scouts: The scouts are created using random modifications. To maintain certain levels of dispersion we have to measure the diversity among the elements in the population. In our case we use the normalised value of each chromosome of any individual.

IV. Grid Implementation

A. Interaction with the Middleware

To get a non-supervised system we have developed a set of wrappers around the command line interface that interact with the metascheduler and the proxy to manage all the required processes in a proper way. The fact that grid middleware is under active development, with some bugs and ongoing bug fixes, that usually affect the API (Application Programming Interface) and not the command line interface was why we chose these wrappers. The final algorithm is a distributed and asynchronous algorithm, where everytime a solution improving the previous best solution found is obtained, new resources are allocated to look for more solutions performing a local search.

1) Setting Up the Environment: The configuration of the algorithm is indicated by means of XML configuration files. In these files the user can indicate several features of the grid environment like the virtual organization, hosts for different elements in the infrastructure, the metascheduler used or any special requirement that the user could want to include. The system will read this configuration file and will perform the required actions. If the user does not supply any data for this file, a generic configuration is used.

Furthermore, the user can specify the number of chromosomes to be used, the extreme values for these chromosomes or the maximum number of evaluations to be performed among a large number of customizable characteristics. Furthermore, the maximum number of *employed* and *scouts* is indicated in a file, in terms of percentage of resources of the grid environment or an absolute number of computational resources. The master process will continuously check if there are less processes submitted than this maximum number in order to submit new tasks to the grid.

B. Types of Jobs

Considering the different types of bees that we have in the system, we have different jobs or tasks in the grid:

- *Onlookers*: this is the system running in the User Interface.
- *Elites* and *rovers*: these are jobs running in the WNs. The main idea behind these jobs is to achieve a deep use of grid capabilities, trying to minimize the time waiting in queues compared to the execution time that they are going to require. These jobs will try to run for the longest time possible. Hence, the way to proceed is:
  1) We send an empty job to the grid.
  2) Once this job starts, it looks for input data in the SE.
  3) When the job has processed all the input data, it tries to get new data from the SE.
- *Workers* and *cubs*: these jobs just process some input data and, after that, they finish their execution.

C. Communication Worker Nodes - User Interface

As previously said, there are two different jobs or bees that need some level of communication with the master node. Furthermore, the simple jobs taking all the input information in the job wrapper have to store their results in the SE and indicate the new result to the master element of the system.

The communication is done by means of the SE and it consists of different points:

- When the system starts, it generates the folder structure in the Logical File Catalog (LFC) that will store all the information of the different bees. Besides, some files to store data such as number of jobs, evolution of the best solution found or execution time, are uploaded to the SE. These files will be modified by the different jobs submitted to the grid. To avoid that two different jobs could modify the same file at the same time, we have developed a barrier based synchronization process:
  – When a job wants to have access to a file in the SE, looks whether the file is locked or not.
  – If the file is locked, the process waits until the process owning the file ends.
  – If the file is unlocked, the process locks the file, copies the file locally, modifies the content and uploads the file to the SE. When the file has been successfully uploaded, the process unlocks the file.
- When a job finishes, it stores the result in its folder in the LFC. It also updates a file, using the barrier system, to indicate the master process that new results have arrived.
- When the termination condition of the system is reached, the master process indicates this fact to all the different jobs in the system. The processes will detect the condition, finishing their evaluations.

This design helps to create a checkpoint system for the different jobs, that will store their information in the SE so they can resume the execution from the last checkpoint in case of failure.

D. Problems Faced

1) LFC: When working with small files in the SE, some problems can appear, specially in cases where many files have to be copied and removed several times from the their locations. During the early stages of this development, we implemented the following blocking method:

- A bee checks for new content in the SE. This means to copy the file from the SE to the UI.
- If new content is found, the bee has to lock the file so no other bee can modify the file. This means to create a new file locking the *content* file.
- When the bee has finished modifying the *content* file, it has to store the new file in the SE. This means, in fact, to remove the old version of the file and upload the new version.
- After uploading the new file, it has to unlock the file, by removing the locking file.

During this process, other bees can be requesting to access the file, so they are interacting at the same time with the LFC. This can make that, sometimes, the LFC fails so the files in the SE can get unregistered and still exist in the SE. This is why the number of write operations in the SE has to be minimized as much as possible. The way to proceed in the final system is:

- Instead of copying the file from the SE to the UI to see whether the file has been modified or not, now we check the file attributes.
- To lock the file, the bee only has to rename the appropriate locking file depending on the file to lock. Renaming files does not make the LFC to fail so often like removing and uploading files.

2) Error recovering: Since the system uses deeply the job management system of the grid environment and the data management system, there are two critical points that can fail:

1) Job management: job failures are detected by the master process. As long as this process has the information corresponding to the different jobs in the system, it can resubmit a job that will resume the execution in the last checkpoint generated by the previous version of the job.
2) Data management: the master process will continuously check the smooth running of the system. The master process will generate replicas of the files in the different SE of the grid environment.

V. Equilibrium

What we pretend to measure in order to get a realistic idea of the equilibrium of our device is to minimise the value of $\vec{v}_d$ or drift velocity of the particles.

$$\vec{v}_d = \frac{\vec{E} \times \vec{B}}{B^2} + \frac{m}{2q}\left(2v_\perp^2 - v_\parallel^2\right)\frac{\vec{B} \times \vec{\nabla}|B|}{B^3} \qquad (1)$$

It is not possible to obtain the values for $\vec{E}$ and neither $2v_\perp^2 - v_\parallel^2$, though. So the possibility consists of minimising $\vec{v}_{\nabla B}$, which finally gives that the objective function we are looking for is given by the eq. 2 (a deeper explanation of this expression can be found in the related work [22]).

$$F_{targetfunction} = \sum_{i=1}^{N} \left\langle \left| \frac{\vec{B} \times \vec{\nabla}|B|}{B^3} \right| \right\rangle_i \qquad (2)$$

In this equation, i represents the different magnetic surfaces existing inside TJ-II whereas B, the magnetic field intensity of each magnetic surface. The goal is to minimise the value given by this function. To solve this equation, we have to execute an application workflow that it is also widely explained in the related work [22]. This workflow takes 45 minutes, on average, to run.

VI. Results

We have executed the system on the Euforia grid infrastructure [25] and the Fusion VO infrastructure. Table I shows the different configurations evaluated. The number of chromosomes of the individuals for all the test was set to 100.

Figure 2, Figure 3, Figure 4 and Figure 5 show the evolution of the mean and best values for the four tests during 24 hours.

Peaks in the mean values are due to the random exploration of the solution space: a random configuration is evaluated, with a extremely high value for the objective function so the mean values of the individuals being evaluated increase.

TABLE I
Configurations of the ABC

Test	No. Elites	No. Workers	No. Rovers	No. Cubs
1	10	10	8	8
2	60	20	50	50
3	80	50	60	60
4	100	20	50	30

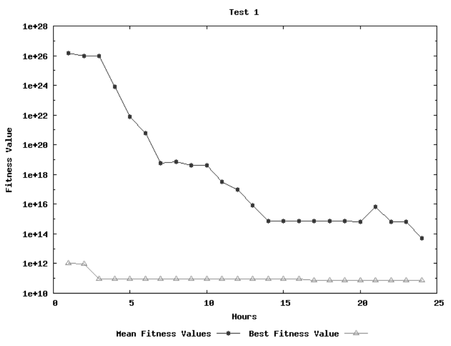

Fig. 2. Mean and best values for Test 1.

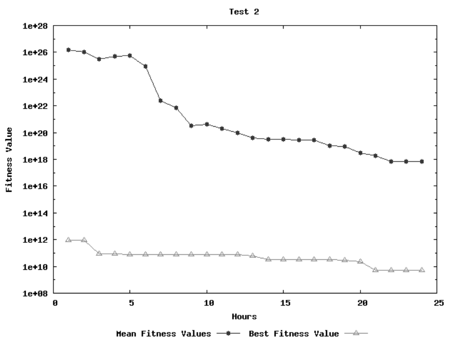

Fig. 3. Mean and best values for Test 2.

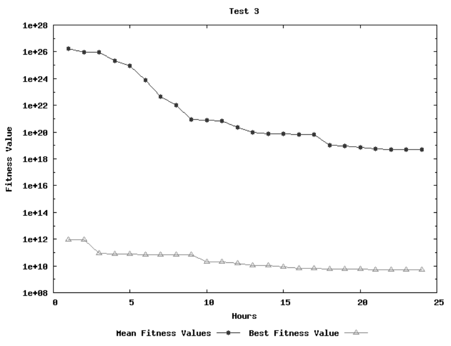

Fig. 4. Mean and best values for Test 3.

After a few hours the results obtained clearly improve the existing configurations and we can see the different evolutions of each test. This fact represents an improvement compared to other metaheuristics previously used [14][15] where the obtained results can only be compared with the results obtained with this implementation after a long time and a large number

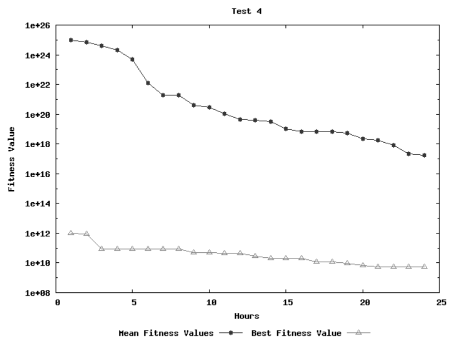

Fig. 5. Mean and best values for Test 4.

of iterations. It is advisable to maintain high dispersion levels for a better exploration of the solution space, and the way to achieve this goal is by adding more scouts to the system.

VII. Conclusions and Future Work

We have presented a grid implementation of the ABC algorithm and its application to plasma physics research by improving the equilibrium of an existing device.

The implementation proposed uses a barrier system to communicate the different elements in the system (different bees, that is, different computational resources) and coordinate the distributed execution of the system. This implementation is generic and can be used to optimise any other scientific problem by changing the configuration of a set of XML files and the job to be submitted to the grid.

The next step will be to increase the use of the grid infrastructure by means of increasing the number of bees in the system as well as considering a longer period of time to collect the results. Instead of using a fixed number of bees, we are going to use a percentage of the resources of the grid. With this deeper exploration we plan to compare the results with the previously obtained [14][15][16]. However, taking into account the results shown in the related work, the first results presented in this paper are very promising and clearly improve the results obtained with other metaheuristics with the same execution time.

Acknowledgment

This work has been funded in part through the Research Infrastructures initiatives of the 7th Framework Programme of the European Commission, grant agreement Number 211804 (EUFORIA project) and grant agreement Number 222667 (EGEE III).

References

[1] I. Foster and C. Kesselman. *The Grid: Blueprint for a New Computing Infrastructure.* Morgan Kaufmann, 1999.
[2] J. Joseph and C. Fellenstein. *Grid Computing.* Prentice Hall, 2003.
[3] Z. Juhász, P. Kacsuk and D. Kranzlmüller. *Distributed and Parallel Systems: Cluster and Grid Computing.* Springer Science, 2005.
[4] E. Alba, A.J. Nebro and J.M. Troya. *Heterogeneous Computing and Parallel Genetic Algorithms.* Journal of Parallel and Distributed Computing, 2002; **62**(9):1362–1385.
[5] M. Cárdenas-Montes, A. Gómez-Iglesias, MA. Vega-Rodríguez and E. Morales-Ramos. Exploration of the Conjecture of Bateman using Particle Swarm Optimisation and Grid Computing. 8^{th} *International Symposium on Parallel and Distributed Computing* 2009 **1**(1):143–150.
[6] F. Castejón, JM. Reynolds, JM. Fontdecaba, D. López-Bruna, R. Balbín, J. Guasp, D. Fernández-Fraile, LA. Fernández, V. Martín-Mayor and A. Tarancón. Ion Orbits and Ion Confinement Studies on ECRH Plasmas in TJ-II Stellarator. *Fusion Science and Technology* 2006 **50**(3):412–418.
[7] S. Cahou, E. Talbi and M. Melab. *ParadisEO: A Framework for the Reusable Design of Parallel and Distributed Metaheuristics.* Journal of Heuristics 2004; **10**(3):357–380.
[8] D. Lim, Y.S. Ong, Y. Jin, B. Sendhoff and B.S. Lee. *Efficient Hierarchical Parallel Genetic Algorithms Using Grid Computing.* Future Generation Computer Systems, 2007; **23**(4):658–670.
[9] N. Melab, S. Cahon and E. Talbi. *Grid Computing for Parallel Bioinspired Algorithms.* Journal of Parallel and Distributed Computing, 2006; **66**(8):1052–1061.
[10] J. Herrera, E. Huedo, R.S. Montero and I.M. Llorente. *A Grid-oriented Genetic Algorithm.* 3^{rd} European Grid Conference. LNCS. 2005; **3470**(1):315–322.
[11] M.G. Arenas, P. Collet, A.E. Eiben, M. Jelasity, J.J. Merelo, B. Paechter, M. Preußand M. Schoenauer. *A Framework for Distributed Evolutionary Algorithms.* Proceedings of the $7th$ International Conference on Parallel Problem Solving from Nature, 2002; **2439**(1):665–675.
[12] C. Gagné, M. Parizeau and M. Dubreuil. *Distributed BEAGLE: an Environment for Parallel and Distributed Evolutionary Computations.* Proceedings of the $17th$ Annual International Symposium on High Performance Computing Systems and Applications, 2003; **1**(1):201–208.
[13] E. Alba, F. Almeida, M. Blesa, J. Cabeza, C. Cotta, M. Díaz, I. Dorta, J. Gabarró, C. León, J. Luna, L. Moreno, J. Petit, A. Roas and F. Xhafa, *MALLBA: a Library of Skeletons for Combinatorial Optimization.* Proceedings of the EuroPar, 2002; **2400**(1):927–932.
[14] A. Gómez-Iglesias, MA. Vega-Rodríguez, F. Castejón, M. Cárdenas-Montes and E. Morales-Ramos. Using a Genetic Algorithm and the Grid to Improve Transport Levels in the TJ-II Stellarator. *International Symposium on Parallel and Distributed Computing. IEEE Computer Society* 2008; **1**(1):81–88.
[15] A. Gómez-Iglesias, MA. Vega-Rodríguez, F. Castejón, M. Cárdenas-Montes and E. Morales-Ramos. Grid-Enabled Mutation-Based Genetic Algorithm to Optimise Nuclear Fusion Devices. *Computer Aided Systems Theory - EUROCAST 2009. LNCS* 2009; **5717**(1):809–816.
[16] A. Gómez-Iglesias, MA. Vega-Rodríguez, M. Cárdenas-Montes, E. Morales-Ramos and F.. Castejón. Grid-Oriented Scatter Search Algorithm. *ICANNGA 2009, LNCS* 2009; **5495**(1):193–202.
[17] M. Laguna and R. Martí. *Scatter Search. Methodology and Implementations.* Kluwer Academic Publishers, 2003.
[18] D. Karaboga. *An Idea Based on Honey Bee Swarm for Numerical Optimization.* Technical Report,Erciyes University, Engineering Faculty, Computer Engineering Department, 2005.
[19] D. Karaboga and B. Akay. *Solving Large Scale Numerical Problems Using Artificial Bee Colony Algorithm.* 6^{th} International Symposium on Intelligent and Manufacturing Systems, 2006.
[20] D. Karaboga and B. Basturk. *Artificial Bee Colony (ABC) Optimization Algorithm for Solving Constrained Optimization Problems.* IFSA 2007. LNCS. 2007; **4529**(1):789–798.
[21] A. Singh. *An artificial bee colony algorithm for the leaf-constrained minimum spanning tree problem.* Applied Soft Computing. Elsevier Computing, 2009. **9**(2): 625–631.
[22] A. Gómez-Iglesias, MA. Vega-Rodríguez, F. Castejón, M. Rubio-del-Solar and M. Cárdenas-Montes. Grid Computing in Order to Implement a Three-Dimensional Magnetohydrodynamic Equilibrium Solver for Plasma Confinement. *16^{th} Euromicro International Conference on Parallel, Distributed and network-based Processing. IEEE Computer Society* 2008; **1**(1):435–439.
[23] PM. Bellan. *Fundamentals of Plasma Physics.* Cambridge University Press, 2006.
[24] J. Freidberg. *Plasma Physics and Fusion Energy.* Cambridge University Press, 2007.
[25] R. Stotzka, E. Morales-Ramos, M. Aspnäs, JÅ. Strooöm, M. Cárdenas-Montes, F. Castejón, JM. Cela, DP. Coster, A. Gómez-Iglesias, B. Guillerminet, A. Hammad, M. Hardt, L. Kos, D. Piccioni-Koch, I. Campos Plasencia, M. Plociennik, G. Poghosyan, L. Smith, L. Sonnendrücker, P. Strand and J. Westerholm. *EUFORIA - Simulation Environment for ITER Fusion Research.* Proceedings of the 34^{th}th EUROMICRO Conference on Software Engineering and Advanced Applications, 2008.