Effect of population size in heterogeneous and homogeneous machines in a distributed EA

Anonymous
Lost island
Unknown
Pacific Ocean
jack,sawyer,hurley@lost.com

Anonymous
Lost island
Unknown
Pacific Ocean
lock@lost.com

ABSTRACT

This paper shows a preliminary study about population size tuning in an distributed genetic algorithm. This adaptation is done taking into account the computational power of each node of an heterogeneous cluster. Two problems with distinct characteristics have been tested: the linearly-solvable OneMax problem and the deceptive and multimodal MMDP functions. Same parameters are also tested in an homogeneous scientific cluster. Results show that setting this parameter according computational power decreases the time to obtain the optimum in both problems in heterogeneous clusters.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; G.1.6 [**Mathematics of Computing**]: NUMERICAL ANALYSIS—*Optimization*

General Terms

Algorithms

Keywords

parameter setting, distributed algorithms, island model

1. INTRODUCTION

New trends such as Cloud Computing [4], GRID [3] or Service Oriented Science [8] are leading to heterogeneous computational devices working at the same time. Moreover, many laboratories do not count with classic clusters, but the usual workstations used by scientists can behave in group as a heterogeneous cluster. Distributed Evolutionary Algorithms (dEAs) have been tested in these systems with SUCCES?. These systems can take advantage of heterogeneous dEAs.

Heterogeneous dEAs can be divided in two categories: a dEA where the parameters are different in each island (heterogeneous parameters) or the same algorithm in heterogeneous hardware. It also have been proved that this kind of algorithms are even more efficient in heterogeneous hardware configurations, than in homogeneous devices [2]. This can be explained by different reasons, such as different memory access times, cache, or even implementation languages or compilers in each machine, leading to different explotation rate of the search space. The heterogeneous parameters configuration also have been proved as more efficient than a fixed set of parameters for different problems [11]. Our motivation in this work is to combine both ideas adapting the population size of the islands to the heterogeneous hardware. To calculate the computational power, the algorithm is executed in each machine and distribute a total size of individuals according the number of generations attained in each node during the same time. Two different problems (MMDP and OneMax) have been used as a benchmark.

In this work, a distributed system has been developed to solve the following questions:

- Can a distributed EA be adapted to take the most of the performance of a heterogeneous cluster?
- Does the proposed population size adaptation to the computational power scheme any effect in both systems?
- Is there any difference in homogeneous and heterogeneous clusters?

The rest of the work is structured as follows: after the state of the art, we present the developed algorithms and experimental setting. Then, the results of the experiments are shown (Section 4), followed by conclusions and suggestions for future work.

2. STATE OF THE ART

In the field of the Evolutionary Computing there are two different approaches about the algorithm parameter setting: parameter control and parameter tuning [7]. The first one refers to set a number of parameters of an EA and change these parameter while the algorithm is running. The parameter tuning consist in establish a good set of parameters before the run (and do not change them during runtime).

In [2] authors compare a distributed GA in homogeneous and heterogeneous clusters. Super-linear performance is obtained in the heterogeneous clusters, being more efficient that the same algorithm in homogeneous clusters. Some authors have expanded this idea adapting the algorithm to be

Table 1: Parameters used.

Name	Value
Total individuals	256
Population size in HoSi	64
Population size in HeSi	98, 84, 66, and 8
Crossover type	Uniform crossover
Crossover rate	0.5
Mutation rate	1/genome size
Selection	2-tournament
Replacement	Steady-state
Generations to migrate	64
Genome size for MMDP	150
Genome size for OneMax	5000

executed: in [6] a distributed meta-heuristic executes simpler algorithms in simpler nodes. In [12] different configurations of heterogeneous machines for a tree topology are studied. However, the heterogeneity is simulated in an homogeneous clusters. Load-balancing is also applied taking into account the computational load of the nodes in [9]: a small benchmark is executed in all nodes at the start of the algorithm to distribute individuals of an Evolutionary Strategy (ES). However, there is not communication between the nodes. In the area of heterogeneous parameters, but homogeneous hardware, setting each island a random set of parameters can also increase the performance of a distributed Genetic Algorithm (dGA), as explained in [11]. That model outperformed a tuned canonical dGA with the same parameters in all islands. Finally, adapting the migration period have been produced better results than homogeneous periods in homogeneous cluster, as claimed by [14].

Our work presents a combination of previous ideas, where a parameter tuning given by the computational cost of the machines is performed. For our knowledge, there are not works that modify parameters of the GA depending of the node where the island is being executed.

3. EXPERIMENTAL SETUP

This section presents the parameters and systems to conduce the experiments.

The algorithm to improve is a distributed Genetic Algorithm (dGA). Parameters are described in Table 1. The algorithm is steady-state: the offspring is mixed with the parents and the worst individuals are removed. A ring topology has been used, and the best individual is sent after a fixed number of generations. Two different parameter configurations have been used: 64 individuals per node (homogeneous size) and a different number of individuals proportional to the number of generations attained in this first homogeneous size execution (heterogeneous size).

The problems to evaluate are the Massively Multimodal Deceptive Problem (MMDP) [10] and the OneMax problem [15]. Each one requires different actions/abilities by the GA at the level of population sizing, individual selection and building-blocks mixing. The MMDP is designed to be difficult for an EA, due to its multimodality and deceptiveness. Deceptive problems are functions where low-order building-blocks do not combine to form higher order building-blocks. Instead, low-order building-blocks may mislead the search towards local optima, thus challenging search mechanisms. MMDP it is composed of k subproblems of 6 bits each one (s_i). Depending of the number of ones (unitation) s_i takes the values shown in Table 3.

Table 2: Basic deceptive bipolar function (s_i) for MMDP.

`Unitation`	`Subfunction value`
0	1.000000
1	0.000000
2	0.360384
3	0.640576
4	0.360384
5	0.000000
6	1.000000

The fitness value is defined as the sum of the s_i subproblems with an optimum of k (equation 1). The search space is composed of 2^{6k} combinations from which there are only 2^k global solutions with 22^k deceptive attractors. Hence, a search method will have to find a global solution out of 2^{5k} additionally to deceptiveness. In this work $k = 25$.

$$f_{MMDP}(\vec{s}) = \sum_{i=1}^{k} fitness_{s_i} \quad (1)$$

Onemax is a simple linear problem that consists in maximising the number of ones in a binary string. That is, maximize the expression:

$$f_{OneMax}(\vec{x}) = \sum_{i=1}^{N} x_i \quad (2)$$

To test the algorithm two different computational systems have been used: an *heterogeneous cluster* and an *homogeneous cluster*. The first one is formed by 4 different computers of our lab with different processors, operating systems and memory. The latter is a dedicated scientific cluster formed by homogeneous nodes. Table 3 shows the features of each system.

Because the operating system and architecture heterogeneity the ANONYMOUS framework [13], based in Java, has been used. This is a service-oriented evolutionary framework that automatically configures services to use and be used in a local network. In this case, each node offers a migration buffer to accept foreign individuals. Also, to avoid bottlenecks in distributed executions, asynchronous communication has been provided to avoid idle time using reception buffers (that is, the algorithm does not wait until new individuals arrive). This kind of communication offers excellent performance when working with different nodes and operating systems, as demonstrated by [2]. The transmission mechanism is based in ECF Generic server (over TCP) [1]. The source code of the algorithms used in this work is available in `http://anonymous` under a GPL V3 License.

Each different configuration has been tested 30 times. Acronyms for each configuration are HoSi (homogeneous population size), HeSi (heterogeneous population size), HoHa (homogeneous hardware) and HeHa (heterogeneous hardware). The population sizes are obtained after the execution of the HoSi/HeHa version of the MMDP and divide the

[1]`http://www.eclipse.org/ecf/`

Table 3: Details of the clusters used.

Name	Processor	Memory	Operating System	Network
Homogeneous cluster				
Cluster node	Intel(R) Xeon(R) CPU E5320 @ 1.86GHz	4GB	CentOS 6.7	??
Heterogeneous cluster				
N1	Intel(R) Core(TM)2 Quad CPU Q6600 @ 2.40GHz	4GB	Ubuntu 11.10 (64 bits)	Gigabit Ethernet
N2	Intel(R) Core(TM)2 Quad CPU Q6600 @ 2.40GHz	4GB	Ubuntu 11.04 (64 bits)	?? Ethernet
N3	AMD Phenom(tm) 9950 Quad-Core Processor @ 1.30Ghz	3GB	Ubuntu 10.10 (32 bits)	?? Ethernet
N4	Intel (R) Pentium 3 @ 800MHz	768 MB	Ubuntu 10.10 (32 bits)	

total number of individuals (256) proportional to the average number of generations attained in each node. Thus, the HeSi configuration uses 98, 84, 66, and 8 individuals (from N1 to N4). Note that, having two nodes with the same processors and memory (N1 and N2), they have different computational power.

4. RESULTS

As claimed by [1] the number of evaluations can be misleading in the parallel algorithms area. In our case, for example, the evaluation time is different in each node of the heterogeneous cluster, and the real algorithm speed could not be reflected correctly. Also, the main interest in parallel programming is to reduce time. However, the number of evaluations has been added for comparison between the results of the HoHa system. It is difficult to compare between the HoHa and HeHa for the same reasons: the evaluation time is different in each system (and machine) ESTO DEBERIA JUSTIFICARLO MAS.

4.1 MMDP Problem

Table 4 shows the results for the MMDP problem. These results are also show in the boxplots of the Figure 2 (evaluations) and Figure 1 (time). Table 6 shows the statistical significance of the results. First, a Kolmogorov-Smirnov test is performed to asset the normality of the distributions. If the results are normal, then a Student's T-Test is run. Otherwise, the non-parametric test Wilcoxon signed rank is applied (see [5] for a tutorial for comparing EAs).

In the HeHa system, adapting the population to the computational power of each nodes makes the algorithm to end faster, but with the same number of evaluations (no statistical significance). This can be explained because the evaluation time is different in all nodes. On the other hand, in the HoHa system, setting the same population sizes makes no difference in time and evaluations. EXPLICAR.

To see the difference of how the evolution is being performed, the average fitness in each node of HeHa is shown in Figures 4 and 3. As can be seen, with the HeSi (Figure 3), the local optima are overtaken in less time than HoSi (Figure 4). This can be explained because in HeSi, the migration from N4 to N1 is performed faster, adding more heterogeneity to the whole system. In the HoHa systems, the populations are evolved at the same time, being the average fitness similar in all nodes during all run.

4.2 OneMax Problem

Results for this problem are shown in Table 5 and Figures 5 and 6. As before, in this problem, changing the population sizes decreases significantly the time in the heterogeneous cluster, and also the number of evaluations remains the same (see statistical significance in Table 6). In the homogeneous system, the effect of changing this sizes is more evident,

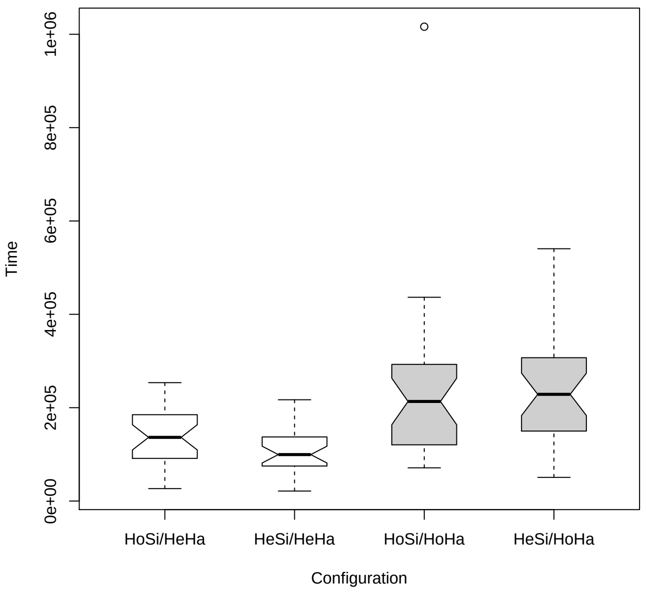

Figure 1: Time to obtain the optimum in the MMDP problem (milliseconds). White is the heterogeneous cluster and gray the homogeneous one.

Table 4: Results for the MMDP problem.

Configuration	Max. generations	Total generations	Total evaluations	Time (ms)
HoSi/HeHa	146401,48 ± 65699,69	380967,25 ± 168568,84	24382416,51 ± 10788405,87	136914,03 ± 60028,48
HeSi/HeHa	96051,5 ± 45110,90	289282,3 ± 135038,10	21784528,66 ± 10161989,38	109875,76 ± 49185,51
HoSi/HoHa	107334,46 ± 78167,19	393119,86 ± 241835,27	25273201,06 ± 15386663,12	237759,43 ± 178709,86
HeSi/HoHa	149732,6 ± 81983,74	438171,16 ± 240169,19	24430043,46 ± 13395037,34	245776,93 ± 134715,52

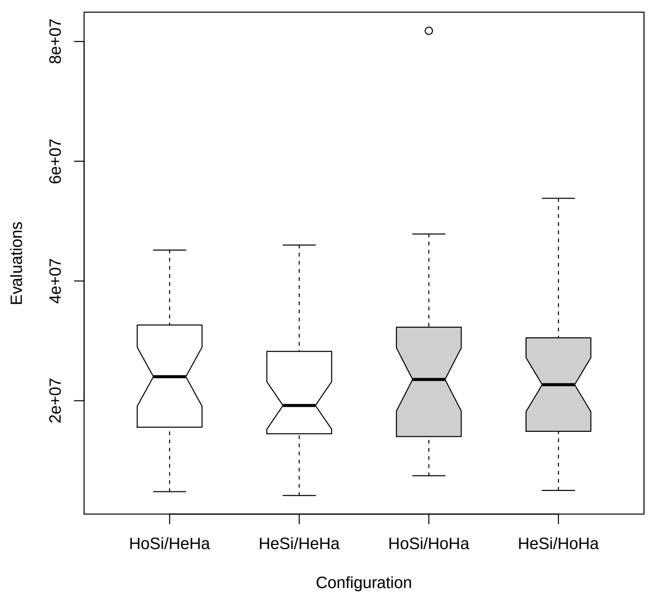

Figure 2: Number of evaluations for MMDP problem. White is the heterogeneous cluster and gray the homogeneous one.

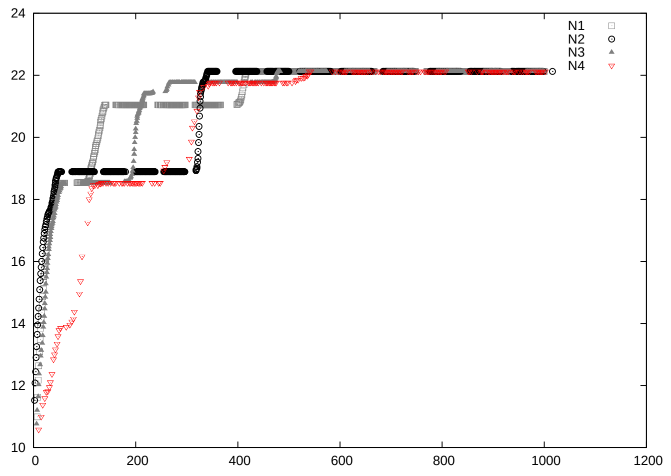

Figure 3: Average fitness in the first 1000 milliseconds of execution of the four nodes of the heterogeneous cluster with different population sizes (MMDP problem).

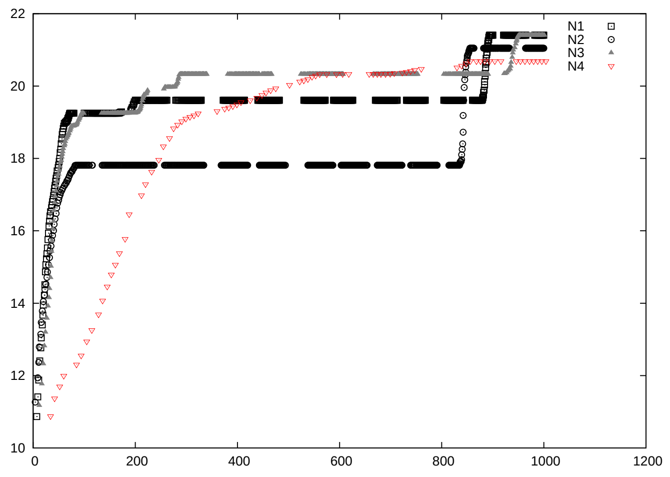

Figure 4: Average fitness in the first 1000 milliseconds of execution of the four nodes of the heterogeneous cluster with the same population sizes (MMDP problem).

and this time the evaluations (and therefore, the time) are reduced (both significally).

The efficiency on OneMax problems depends more on the ability to mix the building-blocs, and less on the genetic diversity and size of the population (as with MMDP). No genetic diversity is particularly required. When properly tuned, a simple Genetic Algorithm is able to solve OneMax in linear time. Sometimes, problems like OneMax are used as control functions, in order to check if very efficient algorithms on hard functions fail on easier functions. As can be seen in Figure 7, the HoSi/HeHa, the average fitness of all populations are increasing in linear way. However, the lower processor evaluates extremely less times. On the other side, in Figure 8, the adaptation of the population size makes that lower processors increase the number of evaluations, but the average fitness is also maintained in linear way (and in smaller increase rate). However, the other processors are still spending more number of evaluations. That is the reason why the number of evaluations is higher in HeHa, and lower in HoHa. Computational time is more efficiently used in faster processor, having more chance to mix the individuals. Also, because the larger size of the individuals in the OneMax problem (5000 bits vs. 150), the transmission time is larger (white gaps in the figures). That also implies for N4 send their best individual to N1 in a extremely large time when using HoSi (each 64 generations).

5. CONCLUSIONS

New trends, such as Cloud Computing or Service Oriented Architecture are providing a massively amount of hetero-

Table 5: Results for the OneMax problem.

Configuration	Max. generations	Total generations	Total evaluations	Time (ms)
HoSi/HeHa	4739,41± 305,32	12081,51± 776,35	773729,03± 49686,72	72152,32± 4994,71
HeSi/HeHa	3438,03 ± 149,47	11277,33± 471,77	794157,73± 31843,10	61870,2 ± 2518,74
HoSi/HoHa	3133,36± 101,70	12347,83± 394,99	790773,33± 25279,52	62105,03± 1964,75
HeSi/HoHa	13897,86± 625,27	20725,63± 929,43	651952,8 ± 29114,54	56120,53± 2491,92

Table 6: Statistical significance of the results.

Configuration	Normal	Test applied	P-value	Significant difference?
Time for MMDP				
HoSi/HeHa vs HeSi/HeHa	Yes	T-Test	**0.032**	Yes
HoSi/HoHa vs HeSi/HoHa	No	Wilcoxon	0.567	No
Evaluations for MMDP				
HoSi/HeHa vs HeSi/HeHa	Yes	T-Test	0.231	No
HoSi/HoHa vs HeSi/HoHa	No	Wilcoxon	0.958	No
Time for OneMax				
HoSi/HeHa vs HeSi/HeHa	Yes	T-Test	$\mathbf{9}\times10^{-15}$	Yes
HoSi/HoHa vs HeSi/HoHa	No	Wilcoxon	1×10^{-6}	Yes
Evaluations for OneMax				
HoSi/HeHa vs HeSi/HeHa	No	Wilcoxon	0.14	No
HoSi/HoHa vs HeSi/HoHa	Yes	T-Test	$2{*}\times10^{-27}$	Yes

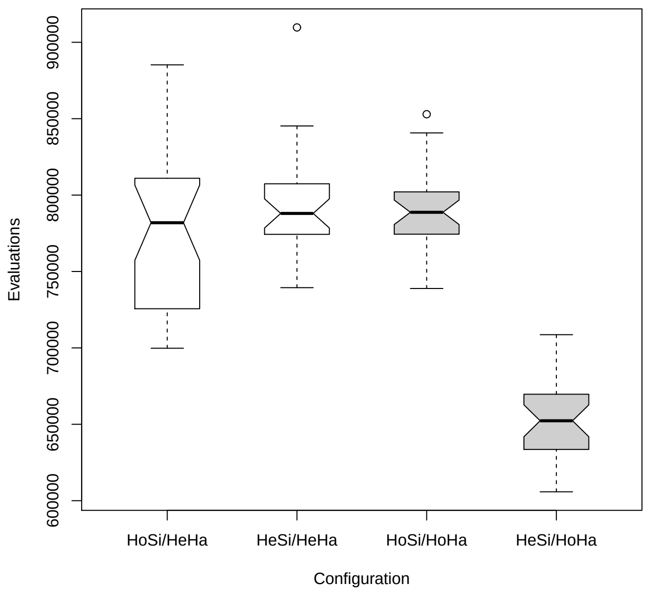

Figure 5: Number of evaluations for OneMax problem. White is the heterogeneous cluster and gray the homogeneous one.

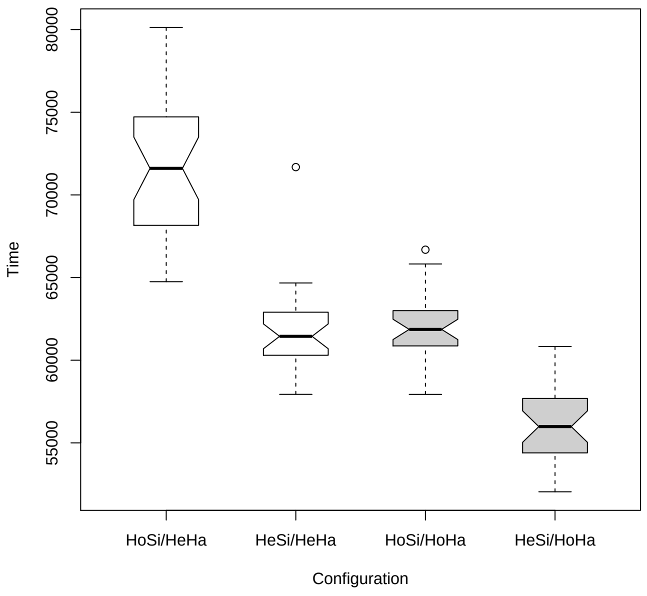

Figure 6: Time to obtain the optimum in the OneMax problem (milliseconds). White is the heterogeneous cluster and gray the homogeneous one.

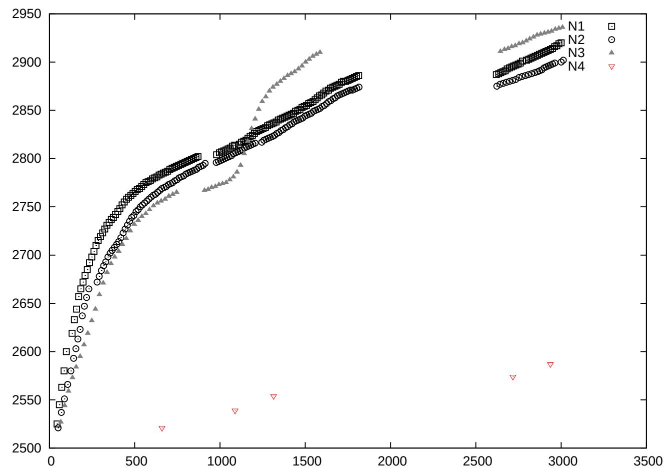

Figure 7: Average fitness in the first 3000 milliseconds of execution of the four nodes of the heterogeneous cluster with the same population sizes (OneMax problem).

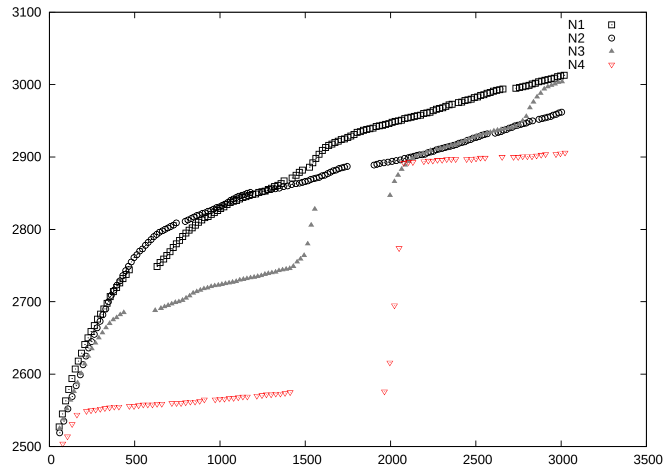

Figure 8: Average fitness in the first 3000 milliseconds of execution of the four nodes of the heterogeneous cluster with different population sizes (OneMax problem).

geneous computational devices. BLABLABLA This work shows a preliminary study about adapting the population size of an EA to computational power of different nodes in an heterogeneous cluster. Results show that adapting the population size decrease the execution time significantly in heterogeneous clusters, while changing this parameter in homogeneous clusters not always performs better. This is a promising start for adapting EAs to the computational power of each machine.

In future work a scalability study will be performed, with more computational nodes and larger problem instances. Also, other parameters such as migration rate or crossover probability will be adapted to the computational nodes. This studies will lead to automatic adaptation during runtime, whith different nodes entering or exiting in the topology during the algorithm execution or adapting to system load.

6. ACKNOWLEDGMENTS

This work has been supported in part

7. REFERENCES

[1] E. Alba and G. Luque. Evaluation of parallel metaheuristics. In Springer, editor, *Parallel Problem Solving from Nature (PPSN)*, volume 4193 of *LNCS*, pages 9–14, 2006.

[2] Enrique Alba, Antonio J. Nebro, and José M. Troya. Heterogeneous computing and parallel genetic algorithms. *Journal of Parallel and Distributed Computing*, 62(9):1362 – 1385, 2002.

[3] Mine Altunay, Paul Avery, Kent Blackburn, Brian Bockelman, Michael Ernst, Dan Fraser, Robert Quick, Robert Gardner, Sebastien Goasguen, Tanya Levshina, Miron Livny, John McGee, Doug Olson, Ruth Pordes, Maxim Potekhin, Abhishek Rana, Alain Roy, Chander Sehgal, Igor Sfiligoi, Frank Wuerthwein, and Open Sci Grid Executive Board. A Science Driven Production Cyberinfrastructure-the Open Science Grid. *Journal of GRID Computing*, 9(2, Sp. Iss. SI):201–218, JUN 2011.

[4] Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal, James Broberg, and Ivona Brandic. Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Gener. Comput. Syst.*, 25:599–616, June 2009.

[5] Joaquín Derrac, Salvador García, Daniel Molina, and Francisco Herrera. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*, 1(1):3–18, 2011.

[6] Julián Domínguez and Enrique Alba. HydroCM: A hybrid parallel search model for heterogeneous platforms. In El-Ghazali Talbi, editor, *Hybrid Metaheuristics*, volume 434 of *Studies in Computational Intelligence*, pages 219–235. Springer Berlin Heidelberg, 2013.

[7] A. E. Eiben and Selmar K. Smit. Parameter tuning for configuring and analyzing evolutionary algorithms. *Swarm and Evolutionary Computation*, 1(1):19–31, 2011.

[8] I Foster. Globus Toolkit version 4: Software for service-oriented systems. In Jin, H and Reed, D and

Jiang, W, editor, *Network and Parallel Computing Proceedings*, volume 3779 of *Lecture Notes in Computer Science*, pages 2–13, 2005.

[9] J.F. Garamendi and J.L. Bosque. Parallel implementation of evolutionary strategies on heterogeneous clusters with load balancing. In *Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International*, page 8 pp., april 2006.

[10] David E. Goldberg, Kalyanmoy Deb, and Jeffrey Horn. Massive multimodality, deception, and genetic algorithms. In R. Männer and B. Manderick, editors, *Parallel Problem Solving from Nature, 2*, pages 37–48, Amsterdam, 1992. Elsevier Science Publishers, B. V.

[11] Yiyuan Gong and Alex Fukunaga. Distributed island-model genetic algorithms using heterogeneous parameter settings. In *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2011, New Orleans, LA, USA, 5-8 June, 2011*, pages 820–827. IEEE, 2011.

[12] Yiyuan Gong, Morikazu Nakamura, and Shiro Tamaki. Parallel genetic algorithms on line topology of heterogeneous computing resources. In *Proceedings of the 2005 conference on Genetic and evolutionary computation*, GECCO '05, pages 1447–1454, New York, NY, USA, 2005. ACM.

[13] A. Nonymous. Anonymous framework. *Secret Journal*, 1(1):1–1, 1.

[14] Carolina Salto and Enrique Alba. Designing heterogeneous distributed gas by efficiently self-adapting the migration period. *Applied Intelligence*, 36:800–808, 2012.

[15] J.D. Schaffer and L.J. Eshelman. On Crossover as an Evolutionary Viable Strategy. In R.K. Belew and L.B. Booker, editors, *Proceedings of the 4th International Conference on Genetic Algorithms*, pages 61–68. Morgan Kaufmann, 1991.