Service Oriented Evolutionary Framework: applications and results

Pablo García-Sánchez
Dept. of Computer Architecture
and Computer Technology
University of Granada, Spain
pgarcia@atc.ugr.es

ABSTRACT

This paper shows the stage of development of a Service Oriented Architecture for Evolutionary Algorithms and the first results attained in two different areas: Parameter Adaptation and Evolutionary Art. An abstract architecture is presented, with its assocciated implementation using a wide-used technology. Results attained in experiments with parameter adaptation in distributed heterogeneous machines are presented and the usage of the architecture in Evolutionary Art is also applied.

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

Ian Foster in [**?**], defined the term "Service-Oriented Science", that is, scientific research using interoperable and distributed networks of services, being the key of success the uniformity of interfaces, so researchers can discover and access to services without develop specific code for each data source, program or sensor. Therefore, this paradigm has the potential to increase scientific productivity thanks to those available distributed tools, and also increasing the automation of computation data analysis. On the other way, other trends such as Cloud Computing [3] or GRID [2] 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.

GECCO'13, July 6-10, 2013, Amsterdam, The Netherlands.

Nowadays, there exist a lot of frameworks for Evolutionary Algorithms (EAs), but all of them are Object Oriented statically programmed, not taking advantages that the Service Oriented Computing can offer.

As services must be well-defined, encapsulated and reusable, we need to abstract enough to have a good evolutionary algorithm design. In [**?**] authors discuss about genericity in evolutionary computation software tools. Although it is based in Object Oriented programming, the genericity of a EA framework can be applied to develop EC services, and extended with new functionalities.

The proposed work [5] shows an abstract Service Oriented Architecture for Evolutionary Algorithms (SOA-EA), whith guidelines and steps to migrate from traditional development to SOA. It also presents an specific implementation, called OSGiLiath: an environment for the development of distributed algorithms extensible via plug-ins architecture and based in a wide-accepted software specification (OSGi: Open Services Gateway Initiative [**?**]).

This work shows how this implementation have been used to obtain results in two areas: EL PATATIN Y EL PATATAN.

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 **??**), followed by conclusions and suggestions for future work.

2. STATE OF THE ART

Even as SOA is used extensively in software development, it is not widely accepted in the main EA software. Firstly, there exist Object Oriented frameworks, like Algorithm::Evolutionary, JCLEC or jMetal. Users implement specific interfaces of these frameworks (like *individual* or *crossover*) and they group them in the source code. For example, creating an operator object that groups several operators. However, these frameworks are not compatible among them. For example, the operators created in JCLEC can not be used in jMetal (despite both are programmed in Java). Also, they can not control the services (operators) outside the source code. Parallelism and distribution are added in other frameworks, like MALLBA [**?**], DREAM or ECJ, but using external libraries (such as MPI or DRM), so the code that uses these libraries is mixed with the algoritmh's code.

Even being distributed, these frameworks can not communicate with each other. HeuristicLab is one of the few plug-in and service oriented frameworks. It uses web services for communication, but just to distribute the load, after consulting a central database of available jobs. The

work [?] contains a comparasion and the references of the previous algorithms.

3. SERVICE ORIENTED ARCHITECTURE FOR EVOLUTIONARY ALGORITHMS

In [5] we presented an abstract architecture formed by loosely coupled, highly configurable and language-independent services for Evolutionary Computation. As an example of an implementation of this architecture, a complete process development using a specific service oriented technology (OSGi) was explained. With this implementation, less effort than classical development in integration, distribution mechanisms and execution time management has been attained. In addition, steps, ideas, advantages and disadvantages, and guidelines to create service oriented evolutionary algorithms were explained.

In [?], six criteria for qualify EA frameworks were presented: generic representation, fitness, operator, model, parameters management and configurable output. In our previous work we shown how SOA follows these lines of genericity, but can also extend them:

- Genericity in the service interfaces: service interfaces are established to create new implementations. Furthermore, these interfaces must be abstract enough to avoid their modification.
- Programming language independence: for example, services implemented in Java can use services implemented in C++ and vice-versa.
- Distribution transparency: it is not mandatory to use a specific library for the distribution, or modify the code to adapt the existing operators.
- Flexibility: easy to add and remove elements to use the self-adaptation or other mechanisms.

4. ADAPTING THE POPULATION SIZE TO HARDWARE

One of the first experiments performed with OSGiLiath have been stablish the effect of the population size in homogeneous and heterogeneous clusters. The algorithm to improve is a dGA. Parameters are described in Table **??**. The algorithm is steady-state: the offspring is mixed with the parents and the worst individuals are removed. A ring topology is used, and the best individual is sent after a fixed number of generations of each node (64). 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). A uniform crossover is used (with a rate of 0.5) and a bit-flip mutation (with a probability of 1/genome size).

The problems to evaluate are the Massively Multimodal Deceptive Problem (MMDP) [4] and the OneMax problem [6]. Each one requires different actions/abilities by the GA at the level of population sizing, individual selection and building-blocks mixing. The chromosome length is 150 for MMDP and 5000 for OneMax.

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 size. The latter is a dedicated scientific cluster formed by homogeneous nodes. Table 4 shows the features of each system.

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 number of individuals in each node of the HeSi configuration is proportional to the average number of generations obtained in the nodes of the HoSi/HeHa configuration for the MMDP problem. 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.

4.1 Results

The objetives of the parallel programming are to tackle large computational problems, increase the performance of algorithms in a finite time, or reduce time. In this work we focus in the last objective. 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. However, the number of evaluations has been added in this section to better understand the results. Also, the total number of generations, and the maximum number of generations of the slower of the four nodes are shown. It is difficult to compare between the HoHa and HeHa for the same reasons: the evaluation time is different in each system (and also in each node).

4.1.1 MMDP Problem

Table 2 shows the results for the MMDP problem. These results are also shown in the boxplots of the Figure 1 (time). First, a Kolmogorov-Smirnov test is performed to asset the normality of the distributions. If the results fit a normal distribution, then a Student's T-Test is calculated. Otherwise, the non-parametric test Wilcoxon signed rank is applied.

In the HeHa system, adapting the population to the computational power of each node makes the algorithm to end significantly faster, but with the same number of evaluations (with 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, that is, changing this parameter does not influence the performance of the algorithm.

4.1.2 OneMax Problem

Results for this problem are shown in Table 3 and Figure 2. In this problem, adapting the population sizes decreases significantly the time for solving in the heterogeneous cluster, and, as before, the number of evaluations remains the same. In the homogeneous system, the effect of changing the population sizes is more evident, 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-blocks, and less on the genetic diversity and size of the population (as with MMDP). No genetic diversity is particularly required. When properly

Table 1: 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	Gigabit Ethernet
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)	Gigabit Ethernet
N3	AMD Phenom(tm) 9950 Quad-Core Processor @ 1.30Ghz	3GB	Ubuntu 10.10 (32 bits)	100MB Ethernet
N4	Intel (R) Pentium 3 @ 800MHz	768 MB	Ubuntu 10.10 (32 bits)	10MB Ethernet

Table 2: 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

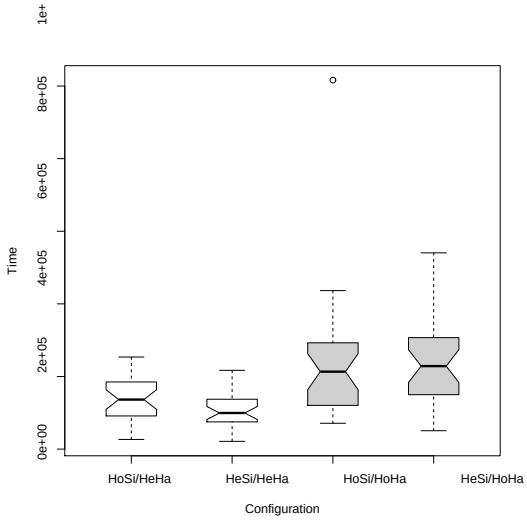

Figure 1: Time to obtain the optimum in the MMDP problem (milliseconds). White is the heterogeneous cluster and gray the homogeneous one.

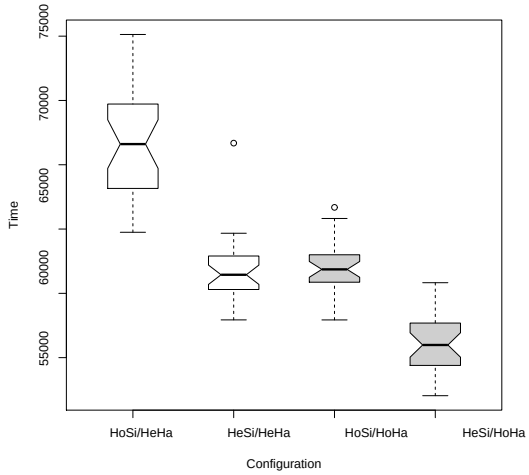

Figure 2: Time to obtain the optimum in the OneMax problem (milliseconds). White is the heterogeneous cluster and gray the homogeneous one.

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.

5. COMPARING RGB AND HSV HISTOGRAMS IN EVOLUTIONARY ART

OSGiLiath has also been used to study the differences of using the information of the HSV (Hue, Saturation, Value) and RGB (Red, Green, Blue) histograms during the evolution. A service to access to Processing [**?**], a programming framework designed for visual artists, have been performed. Also, services to measure the fitness, and implementations of individuals are also available in OSGiLiath. Processing is used inside the Evolutionary Algorithm (EA) to model the individuals, generate their associate images and extract information of them (HSV, RGB and Average histograms) to fit with the histograms of a test image.

Individuals are represented as a list of Processing prim-

Table 3: 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

itives (circles) and the fitness functions used are based in the similarity with an existent aesthetic image (the comparison is performed using the libraries available in Processing). Three different fitness functions using color histogram have been tested and added to OSGiLiath as services: difference between the HSV and RGB histograms, and an average difference of the two histograms at the same time. Experiments show that better results in terms of similarity are obtained using the HSV comparison (due to the noisy information provided by the RGB). Table 4 show the attained results. This is a basic image metric, only used by purposes of proof-of-concept and more complex measurements will be studied in next works.

6. CONCLUSIONS

Service Oriented Computing is a new trend where ... Also, other trends, such as Cloud Computing are providing a massively amount of heterogeneous computational devices. This has been the motivation to develop SOA-EA and OSGiLiath.

The first applications have been 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 execution node.

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 execution nodes. This studies will lead to automatic adaptation during runtime, with different nodes entering or exiting in the topology during the algorithm execution or adapting to the current load of the system.

The project development is explained in `http://www.osgiliath.org`.

7. ACKNOWLEDGMENTS

This work has been supported in part by FPU research grant AP2009-2942 and projects EvOrq (P08-TIC-03903), UGR PR-PP2011-5 and TIN2011-28627-C04-02.

8. 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] 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.

[3] 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.

[4] 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.

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

[6] 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.

Table 4: Results for the different fitness. Only one histogram type is used, but the other values obtained are also added.

Differences used	Obtained RGB	Obtained HSV	Obtained Average
RGB Histogram	*0.267 ± 0.012*	0.170 ± 0.010	0.218 ± 0.009
HSV Histogram	0.227 ± 0.017	*0.265 ± 0.021*	0.246 ± 0.010
Average Histogram	0.173 ± 0.012	0.294 ± 0.013	*0.234 ± 0.010*