On the run-time dynamics of a Peer-to-Peer Evolutionary Algorithm

J.L.J. Laredo[1] and E.A. Eiben[2] and M. van Steen[2] and J.J. Merelo[1]

[1] Department of Architecture and Computer Technology
University of Granada, Spain
e-mail: {juanlu,jmerelo}@geneura.ugr.es
[2] Department of Computer Science
Vrije Universiteit Amsterdam, The Netherlands
e-mail: {gusz,steen}@cs.vu.nl

Abstract. In this paper we propose an improvement on a fully distributed Peer-to-Peer (P2P) Evolutionary Algorithm (EA) based on autonomous selection. Autonomous selection means that individuals decide on their own state of reproduction and survival without any central control, using instead estimations about the global population state for decision making. The population size varies at run-time as a consequence of such a decentralized reproduction and death of individuals. In order to keep it stable, we propose a self-adjusting mechanism which has been shown successful in three different search landscapes. Key are the estimations about fitness and size of the population as provided by a gossiping algorithm. Such an algorithm requires several rounds to collect the information while the individuals have to wait for synchronization. As an improvement, we propose a completely asynchronous EA which does not need waiting times. The results show that our approach outperforms qualitatively the execution time of the synchronous version.

1 Introduction

Spare cycles among interconnected nodes constitute a free and powerful source for high performance computing, Peer-to-Peer (P2P) systems form an alternative to jointly constitute a single virtual computer. Nowadays, there are successful cases of virtual supercomputers based on volunteers sharing their CPU idle cycles (e.g. the BOINC project [1]).

However, Evolutionary Computing has just recently entered this arena and there are still many challenging issues. The DREAM project was one of the pioneers on distributed P2P EAs coming up in with the equally named DREAM framework [2]. Despite the P2P approach, the island-based parallelization of DREAM was shown in [9] to be insufficient for tackling large-scale decentralized scenarios.

Two of our most recent works, [12] and [8], have moved the focus from distributed P2P EAs into finer-grained approaches within the field of spatially structured EAs. As stated in [11], a spatially structured EA can be modeled as a graph in which the vertices are individuals and edges represent relationships between them. Obviously, a graph can be easily mapped to a network topology and consequently a spatially structured EA can be easily distributed.

In this paper, we analyse a distributed P2P EA in which the population structure is defined by a P2P overlay network. The network keeps small-world properties by means of the gossiping protocol Newscast [6]. Such a kind of small-world graphs have been shown, by Giacobini et al. in [4], to be suitable as population structure for an EA, outperforming *panmictic* approaches.

But population structure is not the only issue in a P2P EA. The absence of a central control requires a mechanism to convey global estimations into each local individual. This way, individuals can make local decisions about their status in a decentralized evolution. In order to get estimates about population fitness and size, we follow the counting algorithm proposed in [12] which deploys the aggregation protocol described in [5]. It consists of an iterative gathering of information that after several time steps (or rounds) becomes accurate enough to proceed with local decision making.

Making a local decision in EAs is not straightforward when the parallelization grain is a single individual. Despite crossover and mutation operators requiring one or two individuals, selection involves all of them, or at least a few (such as in tournament selection).

In fact, the autonomous selection presented by Eiben et al. in [3] uses locally available information about global estimations (e.g. those provided by the counting algorithm) and determines the selection probabilities for each individual with a locally executable function based on its own fitness against averaged global fitness. Subsequently, the fittest individuals survive for reproduction while the worst are erased from population. The consequence of such a decentralized process may lead to a run-time population resizing which could get out of control.

In [12], the authors overcame the issue of population size implosions/explosions using an self-adjusting mechanism for controlling the parameters of a sigmoid function (i.e. the seminal function for autonomous selection). Unfortunately, preliminary experiments in different search landscapes have shown sigmoid to be very sensitive under different roughness conditions. Hence, it turns out that keeping the population size under control requires of a hand-made calibration of the self-adjusting mechanism.

Our proposal focuses on the following improvements over the work presented in [12]:

1. We propose a self-adjusting mechanism able to keep the population size stable in different search landscapes. Instead of the sigmoid, we use a simple linear function and a single adjustable parameter, ρ, which self-regulates the local selection pressure by controlling the function slope.
2. In order to get accurate global estimations, the counting algorithm spends several rounds in which the adaptation stage has to wait. In spite of such a necessary synchronization for autonomous selection, we propose an asynchronous EA that uses the counting rounds to evolve a population of individuals' replicas. Each replica evolves with those in its neighbourhood using

tournament selection. If the original individual survives the autonomous selection process, the evolved replica replaces it. As a consequence of reducing the ratio of rounds per evaluations, the execution time of the algorithm is improved.

The efficiency of our approach quantitatively outperforms previous large-scale distributed EAs. The key is the combination of local evolution of individuals' replicas and global resynchronization of the decentralized EA.

The rest of the paper is structured as follows. The overall model is presented in Section 2. We propose, in Section 3, a test suite composed of three real-coding functions with different roughness degree. In Section 4, we deduce from the run-time dynamics that the accuracy of the estimations is good enough to keep the population size stable. Finally, we reach some conclusions and propose some future work lines in Section 5.

2 Proposed Model

Algorithms 1 and 2 show respectively the pseudo-code of the algorithm and the work-flow of an iteration.

Algorithm 1 Outline of the self-adjusting distributed evolutionary algorithm

```
initialize $individual$
$individual_{replica} \Leftarrow individual$
repeat
    if adaptation stage then
        exchange information by gossiping
        estimate population size, average fitness and best fitness
        evolve $individual_{replica}$ within the neighbourhood
        update selection parameters by adaptation
    end if
    if resetting stage then
        if $individual$ is not able to survive then
            die
        else
            $individual \Leftarrow individual_{replica}$
            new $individual \Leftarrow reproduction(individual + random\ \ individual)$
        end if
    end if
until die or another stop criterion
```

During the adaptation stage (n+1 first rounds), each $individual_{replica}$ evolves within its neighbourhood using tournament selection. This stage is required for estimations over the decentralized population. In the resetting stage ($n + 2$ time step), the fittest *individuals* survive, acquire the evolved genome of their own $individual_{replica}$ and generate a new *individual*. Additionally, the values of the counting algorithm are reset and the number of rounds (n) is estimated for the next iteration.

Algorithm 2 Outline of an iteration of the distributed algorithm

```
1 to n time steps: Gossiping rounds
  Exchange information with neighbours
  evolve $individual_{replica}$ within the neighbourhood
  Perform the counting algorithm
n + 1 time step: Adaptation
  Call the adaptation process
  Update the parameters for selection
n + 2 time step: Resetting
  Call survive process
  Either die or $individual \Leftarrow individual_{replica}$
  Reset the values for the counting algorithm
  Calculate steps needed (n) for next iteration
```

2.1 Counting algorithm

The counting algorithm was presented in [12] and extends a P2P aggregation mechanism described in [5]. It provides estimations about the current best fitness, average fitness and total size of the population to each individual. The information is iteratively flooded among the nodes (individuals) and after several iterations (rounds) estimations are available to the nodes. The number of rounds needed is estimated as a logarithmic function of the total size of the population, growing as the size increases (i.e. a population size of 100 individuals would need 12 rounds for estimations while 1000 would need of 18).

2.2 Adaptation and Survival

The autonomous selection determines the probability of survival for each individual by the following equation:

$$P(x) = linear_{\rho}(\Delta f(x)) = \frac{1-\rho}{\Delta f(x_{best})}\Delta f(x) + \rho \tag{1}$$

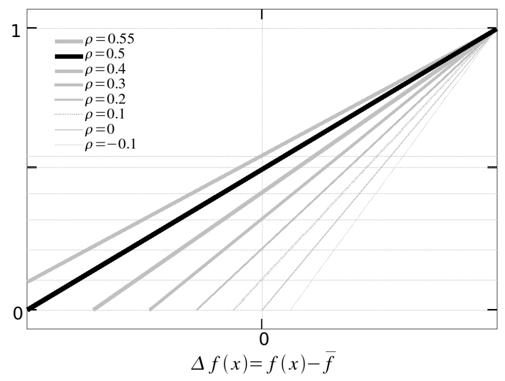

Fig. 1. Linear function for different values of the adjustable parameter ρ

where $\Delta f(x)$ is the deviation of the fitness with respect to the average fitness, $\Delta f(x) = f(x) - \overline{f}$. The function assigns a probability of survival equal to 1.0 for the best individual $P(\Delta f(x_{best})) = 1.0$.

Additionally, the slope of the linear function is determined by the average fitness $\overline{f}$, with $\Delta\overline{f} = \overline{f} - \overline{f} = 0$, where the probability of survival is ρ, $(P(\Delta\overline{f}) = \rho)$.

ρ is an adjustable parameter within the range $[-0.1, 0.55]$. This range has been empirically calibrated in preliminary experiments to prevent population implosions/explosions. The self-adjusting procedure is shown in algorithm 3.

Algorithm 3 Outline of the self-adjusting procedure

```
$P \Leftarrow$ Initial Population Size
$\rho \Leftarrow 0.5$, $\rho \in [-0.1, 0.55]$
**repeat**
  **if** adaptation stage **then**
    $P_{estimated} \Leftarrow$ Counting Algorithm
    **if** $P_{estimated} > P$ **then**
      $\rho = \rho - 0.1$
    **else if** $P_{estimated} < P$ **then**
      $\rho = \rho + 0.1$
    **end if**
  **end if**
**until** stop criterion
```

Initially $\rho = 0.5$, which would probabilistically maintain the population size if we assume normality conditions in the fitness distribution. If the population size is bigger than the initial population, ρ is decreased by 0.1, otherwise ρ is increased by 0.1 (such a value has been empirically calibrated). From the different values of ρ (as shown in Figure 1) the algorithm self-adjusts the ratio of survival by changing the selection pressure.

3 Experimental setup

In order to test the run-time dynamics of the algorithm, we have conducted experiments in the P2P simulator PeerSim [7]. We have chosen as a benchmark three real-coding test functions from the test suite proposed by Suganthan et al. in [10]. This set includes different search landscapes derived from a sphere, the Schwefel problem and the Rastrigin multimodal function. It is important to note that our research objective is not to outperform existing results. Instead, we are initially interested in exploring the extent in which fully decentralized solutions can be successful. Therefore, to consider an EA run successful we allow an error margin of 1 for all test functions. Additionally, we have set the size of the problem instances to a medium degree of difficulty for a GA.

As a baseline for comparison, we have used the distributed P2P EA proposed in [12] to which we will refer as synchronous version from here on. The adaptation stage of the synchronous version has been set with the self-adjusting mechanism proposed in Section 2.2, the rest of the parameter setup is shown in Table 1.

Initial Population Size	200 individuals
Recombination	BLX-0.5 Crossover, $p_c = 1.0$
Mutation	BGA, $p_m = 0.01$
Initial value of ρ	0.5
Termination condition	optimum found with the required accuracy or 100000 evaluation spent or population size = 0 or population size > 600
Selection Parents (original)	Autonomous Selection
Selection Parents (replica)	Binary Tournament + individual

Table 1. Parameters of the algorithms

Shifted Sphere function. The *shifted sphere* function is a unimodal, scalable function:

$$F(x) = \sum_{i=1}^{D} z_i^2 + f_{bias} \tag{2}$$

$$\overline{z} = \overline{x} - \overline{o}$$

where D represents the number of dimensions, $-100 \leq x_i \leq 100$ and $o = [o_1, o_2, \ldots, o_D]$ the shifted global optimum. The optimum (minimum) is $f_{bias} = -450$. We have set D to 30 since it represents a medium degree of difficulty to be solved by a GA.

Schwefel function. This version of the Schwefel's problem is another unimodal function, whose definition is shown in equation 3

$$F(x) = \sum_{i=1}^{D} (\sum_{j=1}^{i} z_j)^2 + f_{bias} \tag{3}$$

$$\overline{z} = \overline{x} - \overline{o}$$

where D represents the number of dimensions, $-100 \leq x_i \leq 100$ and $o = [o_1, o_2, \ldots, o_D]$ the shifted global optimum. The optimum (minimum) is $f_{bias} = -450$. We have used $D = 10$.

Shifted Rotated Rastrigin's function. The shifted rotated Rastrigin's function is a multimodal function with a huge number of local optima, it is defined by the following function:

$$F(x) = \sum_{i=1}^{D} (z_i^2 - 10cos(2\pi z_i) + 10) + f_{bias} \tag{4}$$

$$\overline{z} = \overline{x} - \overline{o}$$

where $-5 \leq x_i \leq 5$ and the global optimum is $f_{bias} = -330$. D has been set to 10.

4 Experimental Results

Figure 2 shows the dynamics of the counting algorithm and the self-adjusting mechanism in the control of the population size.

On one hand, the counting algorithm provides accurate estimations about the population size every n rounds (estimations are shown as circles). On the other hand, the self-adjusting mechanism keeps the population size stable by fluctuating around the pre-established initial size. From the observation we can see how the peaks grow when the algorithm is getting close to the problems' optima (the convergence is represented in Figure 3). The most probable hypothesis for these peaks is that the distribution of Δfs is biased by the global optimum. Hence, the distribution would lose normality conditions as it is getting close to the optimum with the consequent lack of effectiveness in the self-adjusting mechanism.

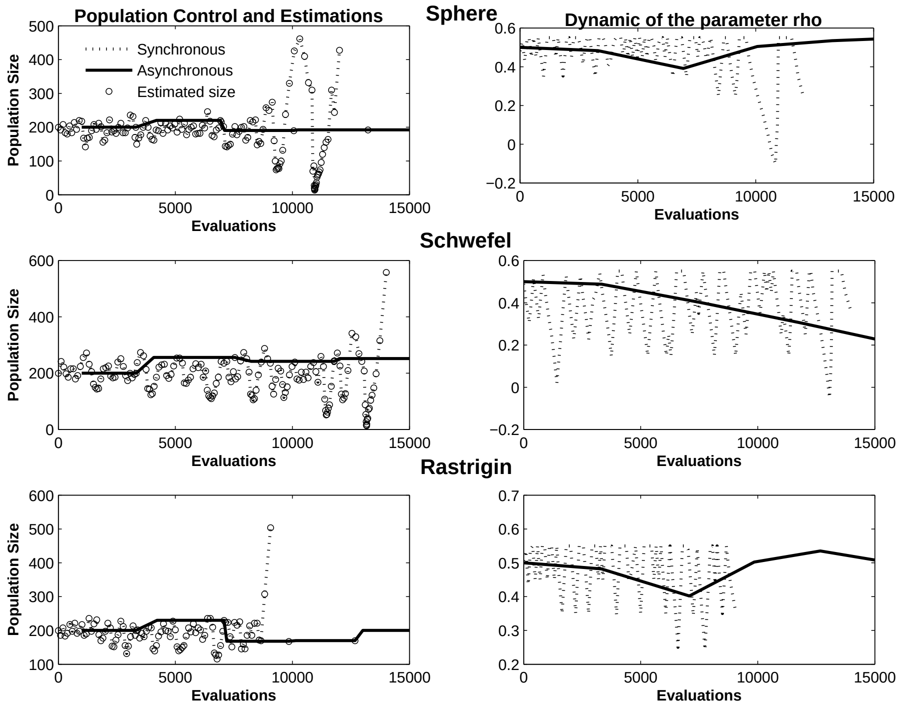

Fig. 2. Dynamic of the population size in the synchronous and asynchronous versions during one run *(left)* and respective values from ρ *(right)*. From *top* to *bottom* the run-time adjustment for the three functions. Circles represent the averaged estimation of the population size provided by the counting algorithm

There is an important difference between our proposal and the synchronous version. In our approach, the population size is fixed during the evolution of individuals' replicas. Afterwards, the population size adjusts in the resynchronization period. Figure 2 shows that the population size does not explode/implode, which is coherent with the previous formulated hypothesis: Once that the algorithm is approaching the problem optimum, local evolution yields success before an explosion in the population size. In fact, such an hypothesis will have to be validated in future works.

Not only is our method able to keep in check the population size, but it is also able to find the optimum with the require accuracy. Figure 3 depicts the convergence curves of the best and average fitness of our proposal and the respective estimations by the counting algorithm. The counting algorithm shows a very accurate estimation for the best fitness while the estimated average fitness is not so accurate. The reason is that the decentralized aggregation of the average fitness is a more complex process than a simpler flooding of the best solution by gossiping. Nevertheless, it is important to note here that despite the estimation errors, the algorithm is robust enough to converge to the problem solutions.

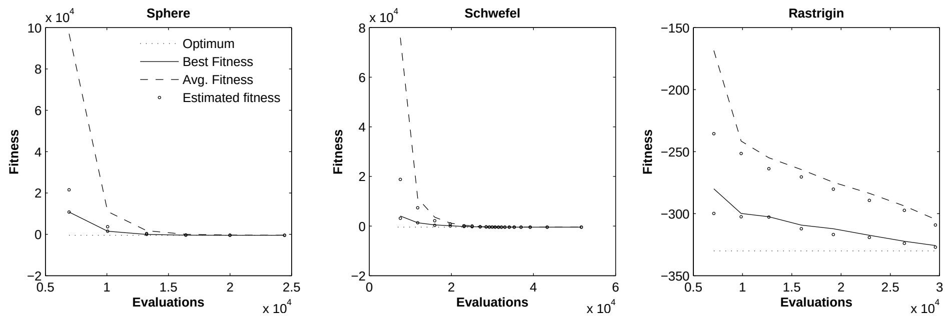

Fig. 3. Convergence curves in the sphere, Schwefel and Rastrigin test functions. Circles represent the averaged estimation of the best and average fitness

Finally, Figure 4 shows the best fitness curves of our asynchronous approach and the globally synchronized one presented in [12]. Each curve depicts the number of rounds needed to improve the fitness (i.e. number of cycles in simulator driven experiments). Both approaches reach success criteria but our proposal spends $\sim 90\%$ less time running. In fact, most of the cycles in [12] are employed in the counting algorithm, being useless from an evolutionary point of view. Therefore, the improvement consists in making those idle cycles useful, we use the local evolution of individuals' replicas to that end.

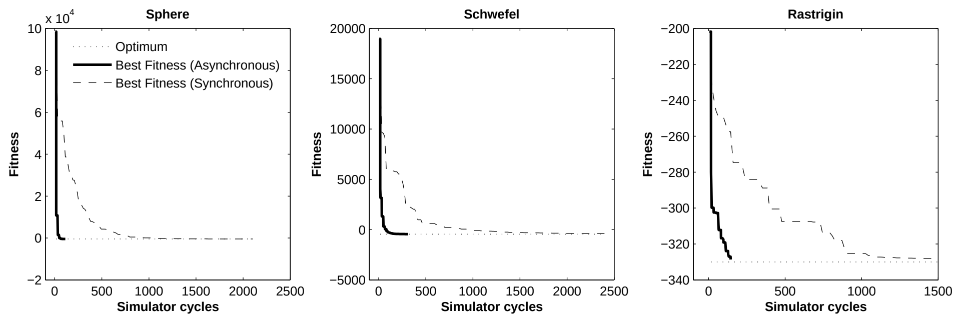

Fig. 4. Best fitness convergence of the synchronous and asynchronous versions in the sphere, Schwefel and Rastrigin test functions. On the *x-axis*, simulator cycles stand for the number of rounds that best fitness needs to improve

5 Conclusions and Future Works

In this paper we have proposed an asynchronous and distributed Peer-to-Peer Evolutionary Algorithm. The whole process is tackled in a decentralized manner in which every individual decides on its own state of reproduction and survival based on autonomous selection and global estimations. The variability on the population size is adjusted by a self-adjusting mechanism that maintains the size around the initial given value. In order to study the run-time dynamics of the algorithm, we have proposed a test suite of three different search landscape with different roughness degree. For all test functions our new EA (using adaptively controlled selection) was able to find the optimum with the required accuracy.

The proposal also includes an asynchronous replica mechanism which avoids the global synchronization presented in [12]. The execution time has been clearly outperformed which is key in a parallel environment. Therefore, we conclude that our proposal is a feasible approach towards a fully decentralized EA with a special focus on P2P.

There are still many challenge to tackle concerning P2P EAs that will have to be studied in future works. We plan to dive in the algorithmic performance and compare our method with other spatially structured algorithms (not focused in P2P necessarily). Additionally, a study on a real environment would provide feedback on actual problems of full decentralization.

Acknowledgements

This work has been supported by the Spanish MICYT project TIN2007-68083-C02-01, the Junta de Andalucia CICE project P06-TIC-02025 and the Granada University PIUGR 9/11/06 project.

References

1. David P. Anderson, Jeff Cobb, Eric Korpela, Matt Lebofsky, and Dan Werthimer. SETI@home: an experiment in public-resource computing. *Commun. ACM*, 45(11):56–61, 2002.
2. M.G. Arenas, Pierre Collet, A.E. Eiben, M. Jelasity, J.J. Merelo, Ben Paechter, Mike Preuss, and Marc Schoenauer. A framework for distributed evolutionary algorithms. In J.J. Merelo Guervós, P. Adamidis, H.G. Beyer, J.L. Fernández-Villaca nas, and H.P. Schwefel, editors, *Parallel Problem Solving from Nature - PPSN VII, Granada, Spain*, number 2439 in Lecture Notes in Computer Science, LNCS, pages 665–675. Springer-Verlag, 2002.
3. A. E. Eiben, Marc Schoenauer, D. W. F. van Krevelen, M. C. Hobbelman, M. A. ten Hagen, and R. C. van het Schip. Autonomous selection in evolutionary algorithms. In *GECCO '07*, pages 1506–1506, New York, NY, USA, 2007. ACM Press.
4. Mario Giacobini, Mike Preuss, and Marco Tomassini. Effects of scale-free and small-world topologies on binary coded self-adaptive CEA. In Jens Gottlieb and Günther R. Raidl, editors, *Evolutionary Computation in Combinatorial Optimization – EvoCOP 2006*, volume 3906 of *LNCS*, pages 85–96, Budapest, 10-12 April 2006. Springer Verlag.
5. Márk Jelasity, Alberto Montresor, and Ozalp Babaoglu. Gossip-based aggregation in large dynamic networks. *ACM Trans. Comput. Syst.*, 23(3):219–252, 2005.
6. Márk Jelasity and Maarten van Steen. Large-scale newscast computing on the Internet. Technical Report IR-503, Vrije Universiteit Amsterdam, Department of Computer Science, Amsterdam, The Netherlands, October 2002.
7. G.P. Jesi. Peersim, a peer-to-peer simulator. `http://peersim.sourceforge.net/`.
8. J. L. J. Laredo, E. A. Eiben, M. Schoenauer, P. A. Castillo, A. M. Mora, and J. J. Merelo. Exploring selection mechanisms for an agent-based distributed evolutionary algorithm. In *GECCO '07*, pages 2801–2808, New York, NY, USA, 2007. ACM Press.
9. Juan Luís Jiménez Laredo, Pedro A. Castillo Valdivieso, Ben Paechter, Antonio Miguel Mora, Eva Alfaro-Cid, Anna Esparcia-Alcázar, and Juan Julián Merelo Guervós. Empirical validation of a gossiping communication mechanism for parallel eas. In *EvoWorkshops*, volume 4448 of *Lecture Notes in Computer Science*, pages 129–136. Springer, 2007.
10. P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y. P. Chen, A. Auger, and S. Tiwari. Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization. Technical report, Nanyang Technological University, Singapore, 2005.
11. Marco Tomassini. *Spatially Structured Evolutionary Algorithms: Artificial Evolution in Space and Time (Natural Computing Series)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.
12. W. R. M. U. K. Wickramasinghe, M. van Steen, and A. E. Eiben. Peer-to-Peer evolutionary algorithms with adaptive autonomous selection. In *GECCO '07*, pages 1460–1467, New York, NY, USA, 2007. ACM Press.