P2P Evolutionary Algorithms: A Suitable Approach for Tackling Large Instances in Hard Optimization Problems

J.L.J. Laredo[1], A.E. Eiben[2], Maarten van Steen[2], P.A. Castillo[1], A.M. Mora[1], and J.J. Merelo[1]

[1] Department of Architecture and Computer Technology
University of Granada, Spain
e-mail: {juanlu,pedro,amorag,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 present a distributed Evolutionary Algorithm (EA) whose population is structured using newscast, a gossiping protocol. This algorithm has been designed to deal with computationally expensive problems via massive scalability; therefore, we analyse the response time of the model using large instances of well-known hard optimization problems that require from EAs a (sometimes exponentially) bigger computational effort as these problems scale. Our approach has been matched against a sequential Genetic Algorithm (sGA) applied to the same set of problems, and we found that it needs less computational effort than the sGA in yielding success. Furthermore, the response time scale logarithmically with respect to the problem size, which makes it suitable to tackle large instances.

1 Introduction

Among the range of techniques used to solve hard optimization problems, *Soft Computing* population-based methods such as Particle Swarm Optimization, Ant Colony Systems, or Evolutionary Algorithms have lately become quite popular [2]. In this paper, we define each individual within the population as an agent which performs a given task and interacts with the rest of individuals. This parallel process leads to the optimization of a problem as a consequence of the iterative convergence of the population to the fittest regions within a search landscape (see e.g. [3] for a survey). Nevertheless, population based methods have been widely approached sequentially despite their intuitively parallel nature. The sequential approach defines by default a *panmictic* way of interaction between individuals, which means that any individual is likely to interact with any other (directly or by means of the environment) sometime. Such an interaction can be visualized as a graph that defines a population structure whose vertices are individuals and edges represent relationships between them.

Therefore, the sequential approach is represented as a complete graph whereas parallel approaches define a richer set of population structures described, for instance, by Tomassini in [15]. The impact of different population structures on the algorithm performance has been studied in addition for regular lattices [4], and different graph structures such as a toroid [5] or small-world [7, 13]. Giacobini and coauthors [6] show specifically that a Watts-Strogatz structured population yields better results than scale-free or complete graphs in the optimization of four different problems.

Besides, the population size (number of individuals) depends on the population structure and scales according to the given optimization problem. This way, larger instances of a problem require larger populations to be solved and additionally, the computational cost of evaluating the problem also scales depending on its computational order. Hence, large problem instances imply an avalanche effect on the computational cost. Such an effect discourage practitioners since the sequential approach is computationally expensive and not easily scalable for running in a distributed environment.

The challenge of tackling these large instances of a problem and the results regarding small-world structured populations in [6], drove us to analyze in this work the effects of a self-organized population using the gossiping protocol newscast [10, 9]. Newscast shares some small-world properties with the Watts-Strogatz model [17], such as a low average path length and a high clustering coefficient, and has been proved to scale to a large number of nodes [16].

Within the whole set of population based paradigms, we consider in this work Evolutionary Algorithms (EAs) for discrete optimization problems, our proposal is described in detail in Section 2. In order to assess our approach, we have used three discrete optimization problems proposed by Giacobini et al. in [6] and we have compared the results with a standard Genetic Algorithm (sGA). In addition, we analyse the adequacy of our algorithm for large problem instances by scaling the problems from small to large sizes. For each one of the instances, we fix a lower and upper bound for the population size in which the algorithm works.

We obtain the lower bound using a method based on bisection, which establishes the minimum population size able to solve the problem with a 98% of reliability, such a method is exposed in Section 3.3. Besides, we use an upper bound of 51200 individuals[3] which is reasonably very large. For further details, we describe the experimental methodology in Section 3.

The results show in Section 4 that our proposal yields better algorithmic results than the sGA. Meanwhile, the population size scales with polynomial order with respect to the different problem instances as in sequential GAs, whereas the response time does logarithmically. Finally, these results are discussed in Section 5 in which we expose some conclusions.

---

[3] In this work, we will refer equally to the terms individual and node, since each individual has its own schedule and could potentially be placed in a different node

2 Overall Model Description

The overall architecture of our approach consists of a population of Evolvable Agents (EvAg), described in Section 2.1, whose main design objective is to carry out the main steps of evolutionary computation: selection, variation and evaluation of individuals [3]. Each EvAg is a node within a newscast topology in which the edges define its neighbourhood.

2.1 Evolvable Agent

We deliberately leave an open definition for agent under the basic feature of just being an encapsulated processing unit. This way future works could extend easily the EvAg definition (i.e. behavioral learning between agents, self-adaptive population size adjustment on runtime [12, 18] or load balancing mechanisms among a real network [1]).

Algorithm 1 shows the pseudo-code of an EvAg where the agent owns an evolving solution (S_t).

Algorithm 1 Evolvable Agent

```
S_t ⇐ Initialize Agent
loop
  Sols ⇐ Local Selection(Newscast) See algorithm 2
  S_{t+1} ⇐ Recombination(Sols,P_c)
  S_{t+1} ⇐ Evaluate(S_{t+1})
  if S_{t+1} better than S_t then
    S_t ⇐ S_{t+1}
  end if
end loop
```

The selection takes place locally into a given neighborhood where each agent select other agents' current solutions (S_t). Selected solutions are stored in $Sols$ ready to be recombined. Within this process a new solution S_{t+1} is generated. If the newly generated solution S_{t+1} is better than the old one S_t, it replaces the current solution.

2.2 Population structure

In principle, our method places no restrictions in the choice of population structure, but this choice will have an impact on the dynamics of the algorithm. In this paper, we study the newscast protocol as neighbourhood policy and topology builder; however, we intend to assess the impact of other topologies in future works.

Newscast is a gossiping protocol for the maintenance of unstructured P2P overlay networks. Within this section we do not enter on the dynamics but on its procedure (see [10, 16] for further details). Algorithm 2 shows the pseudo-code of the main tasks in the communication process which build the newscast topology. Each node maintains a cache with one entry per node in the network at most. Each entry provides the following information about a foreign node: Address of the node, timestamp of the entry creation (it allows the replacement of old items), an agent identifier and specific application data.

Algorithm 2 Newscast protocol in node $EvAg_i$

Active Thread

loop
  sleep ΔT
  $EvAg_j \Leftarrow$ Random selected node from $Cache_i$
  send $Cache_i$ to $EvAg_j$
  receive $Cache_j$ from $EvAg_j$
  $Cache_i \Leftarrow$ Aggregate ($Cache_i$,$Cache_j$)
end loop

Passive Thread

loop
  wait $Cache_j$ from $EvAg_j$
  send $Cache_i$ to $EvAg_j$
  $Cache_i \Leftarrow$ Aggregate ($Cache_i$,$Cache_j$)
end loop

Local Selection(Newscast)

$[EvAg_j, EvAg_k] \Leftarrow$ Random selected nodes from $Cache_i$

There are two different tasks that the algorithm carries out within each node. The active thread which initiates communications and the passive thread that waits for the answer. In addition, the local selection procedure provides the EvAg with other agents' current solutions (S_t).

After ΔT time each $EvAg_i$ initiates a communication process (active thread). It selects randomly a $EvAg_j$ from $Cache_i$ with uniform probability. Both $EvAg_i$ and $EvAg_j$ exchange their caches and merge them following an aggregation function. In our case, the aggregation consists of picking up the newest item for each cache entry in $Cache_i$, $Cache_j$ and merging them into a single cache that $EvAg_i$ and $EvAg_j$ will share.

The cache size plays an important role in the newscast algorithm. It represents the maximum number of connections (edges) that a node could have. For example, a topology with n nodes and a cache size of n, will lead to a complete graph topology (after the bootstrapping cycles). Therefore, the cache size use to be smaller than the number of nodes (typically around logarithm of n) in order to get small-world features. We have fixed the cache size to 20 within the experimental setup.

3 Methodology and Experimental setup

The focus of the proposed experimentation is to find whether our approach is able to tackle large problem instances on a set of three discrete optimization problems presented in Section 3.1.

Firstly, we compare the EvAg model with a standard GA used as a baseline. To this end, we use a method based on bisection (Section 3.3) to establish the population size in both cases. Such a method guarantees a 98% of Success Rate (SR) on the results. Once the SR is fixed, we consider the Average Evaluations to Solution (AES) metric as a measure of the computational effort to reach success on the problems. Therefore, the more efficient algorithm is the one that guarantees a 98% SR using less computation.

Secondly, we tackle the scalability of the EvAg. We study how the population size and the computational effort (e.g. AES) increase as the problem size scales. Therefore, the response time of the approach will show the algorithm scalability since the computational effort is distributed among the nodes.

3.1 The benchmark

In this section we present the benchmark problems that we have used to evaluate our proposal. It consists of three discrete optimization problems presented in [6]: The massively multimodal deceptive problem (MMDP), the problem generator P-PEAKS and the deceptive version wP-PEAKS. They represent a set of difficult problems to be solved by an EA with different features such as multimodality, deceptiveness and problem generators.

Massively Multimodal Deceptive Problem (MMDP) The MMDP [8] is a deceptive problem compose of k subproblems of 6 bits each one (s_i). Depending of the number of ones (unitation) s_i takes the values depicted depicted in Table 1.

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

Table 1. Basic deceptive bipolar function (s_i) for MMDP

The fitness value is defined as the summatory of the s_i subproblems with an optimum of k (equation 1). The number of local optima is quite large (22^k), while there are only 2^k global solutions. We consider several instances from low to high difficulty using $k = 2, 4, 6, 8, 10, 16, 32, 64, 128$.

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

Multimodal Problem Generator (P-PEAKS and wP-PEAKS)

The wP-PEAKS and P-PEAKS problems are two multimodal problem generators. The wP-PEAKS is the modified version proposed in [6] of the problem generator P-PEAKS [11]. The idea is to generate P random $N - bit$ strings where the fitness value of a string $\boldsymbol{x}$ is the number of bits that $\boldsymbol{x}$ has in common with the nearest peak divided by N. The modified version consists in adding weights w_i with only $w_1 = 1.0$ and $w_{[2..P]} < 1.0$. Hence, despite P optima solutions as in the P-PEAKS, there is just one optima and $P - 1$ attractors. In P-PEAKS we consider $P = 100$ and $P = 10$ in wP-PEAKS with $w_1 = 1.0$ and $w_{[2..P]} = 0.99$ where the optimum fitness is 1.0. We consider an instance of $P = 10$ with $w_1 = 1.0$ and $w[2..P] = 0.99$ where the optimum fitness is 1.0 (equations 2 and 3).

$$f_{P-PEAKS}(\boldsymbol{x}) = \frac{1}{N} \max_{1 \leq i \leq p} \{N - HammingDistance(\boldsymbol{x}, Peak_i)\} \qquad (2)$$

$$f_{wP-PEAKS}(\boldsymbol{x}) = \frac{1}{N} \max_{1 \leq i \leq p} \{w_i N - HammingDistance(\boldsymbol{x}, Peak_i)\} \qquad (3)$$

In wP-PEAKS we scale the instances by sizing $\boldsymbol{x}$ to $2, 4, 6, 8, 10, 16, 32, 64, 128$. Meanwhile in P-PEAKS the values are $12, 24, 36, 48, 60, 96, 192$.

3.2 Experimental Setup

We have used for the experimentation two similar parametrized algorithms: EvAg with newscast neighborhood and a sGA. The recombination operator is DPX with $p_c = 1.0$ and for the selection of parents we use binary tournament [3]. All results are averaged over 50 independent runs. Finally, we have conducted the experiments in PeerSim Simulator[4]

3.3 A method for estimating the population size

The Algorithm 3 depicts the method based on bisection [14]. The method begins with a small population size which is doubled until the algorithm ensures a reliable convergence. We define the reliability criterion as the convergence of the algorithm to the optimum 49 out of 50 times (98% of Success Rate). After that, the interval (min, max) is halved several times and the population size adjusted within such a range.

[4] `http://peersim.sourceforge.net/`. Accessed on January 2008. All source code for the experiments is available from our Subversion repository at `https://forja.rediris.es/projects/geneura/`

Algorithm 3 Method based on Bisection

P = Initial Population Size
while Algorithm reliability < 98% **do**
  min = P ; max, P = Double (P)
end while
while $\frac{max-min}{min} > \frac{1}{16}$ **do**
  $P = \frac{max+min}{2}$
  (Algorithm reliability < 98%) ? min = P : max = P
end while

4 Results

Results of the first experiment are shown in Table 2, which shows at first glance that our approach needs less computational effort than the sGA to reach success in any of the problems as they scale. Therefore, our algorithm converges faster to a solution than the sGA which is significant in the algorithmic sense. However, such a result provides just an estimation on the algorithm performance since the EvAg is distributed whereas the sGA is not.

MMDP			wPPEAKS			PPEAKS		
Problem Size	sGA (AES)	EvAg (AES)	P. Size	sGA (AES)	EvAg (AES)	P. Size	sGA (AES)	EvAg (AES)
2	1167.3	**604.5**	2	**20**	100	12	**55.2**	128
4	4634.6	**1833**	4	**31.6**	120	24	1847.1	**669.3**
8	12029	**6243.8**	8	**316.2**	378	36	9429.8	**2838.7**
10	17933	**8779.5**	10	783	**551**	48	9806.1	**5160**
16	45186	**28796**	16	10403	**2015**	60	21669	**10584**
32	128310	**106290**	32	3604480	**23037**	96	42539	**30286**
64	562640	**518010**	64	-	**452740**	192	131200	**125030**
128	-	2517300	128	-	2099200	-	-	-

Table 2. Computational effort for the different problem instances: sGA vs. EvAg

More interesting are the results concerning the scalability of our approach. The analysis of the response time and population size are depicted in Figure 1 for the three problems under study. In any of them, the figures show that the estimated population size scales with polynomial order as the problems scale. Meanwhile, the response time (measured in simulator cycles) scales with logarithmic order showing a good adequacy of the algorithm for large problem instances. Therefore, the necessity of a huge amount of nodes for tackling large problem instances justifies the use of a P2P system in which such an amount would be available.

Finally, we have performed experiments with a network of 51200 nodes independently of the population size estimation. This way, we explore how the redundancy of nodes (individuals) affects on the algorithm dynamics. The results show that the response time decreases for small instances and gets closer to the one estimated with the bisection method for large instances. Such a result shows that the algorithm is robust concerning population size. Hence, we will explore the redundancy of nodes as a mechanism for fault tolerance.

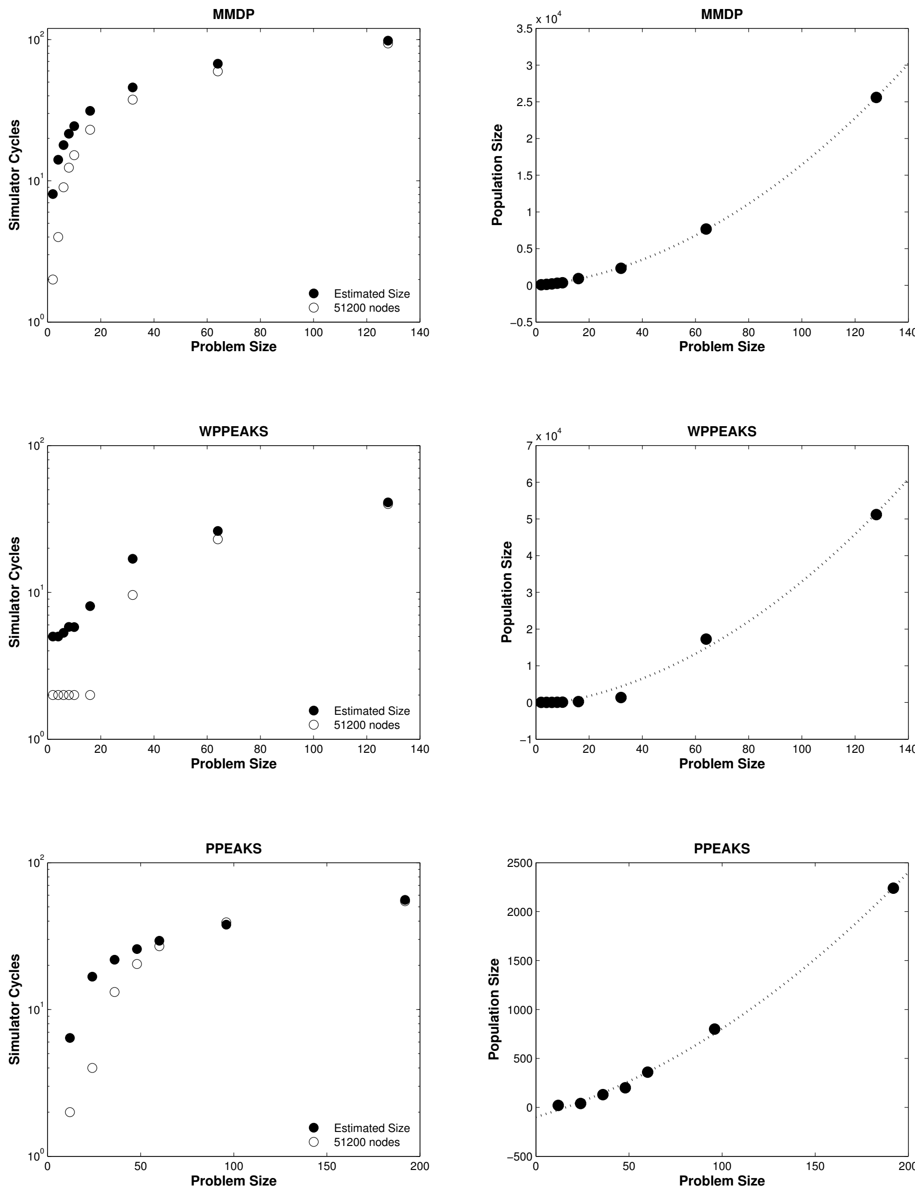

Fig. 1. Scalability analysis for the minimum population size (estimated) and an upper bound of 51200 individuals-nodes *(left)* and the estimated size *(right)* for a 98% of success rate using the method based on bisection

5 Conclusions

In this paper we have presented a P2P Evolutionary Algorithm and proved its adequacy for tackling large problem instances. Specifically, we have studied three discrete optimization problems which have been designed to be difficult for EAs. Our approach is designed to deal with some P2P features such as decentralization and large-scalability. To this end, the population structure is managed by the gossiping protocol newscast. Through the experimental results we conclude that large instances of hard optimization problems can be tackled in P2P systems using the Evolvable Agent (EvAg) method. In this paper we have thus proved that:

- Our approach needs less computational effort than a standard GA to reach the same degree of success on the problems under study. Additionally, such a computational effort is distributed whereas in the sGA is not.
- The population size scales with polynomial order which demands for a big amount of resources.
- The expected response time of the algorithm scales logarithmically with respect to the problem size which makes it efficient despite large problem instances.
- The algorithm is robust with respect to the population size. Once we estimate the minimum population size that yields success, adding more nodes does not damage the response time.

As future lines of work, we intend to assess the impact of other population structures on the algorithm performance. Additionally, we will study the redundancy of individuals as a fault tolerance mechanism in Peer-to-Peer Evolutionary Algorithms.

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. 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 *Parallel Problem Solving from Nature - PPSN VII, Granada, Spain*, number 2439 in Lecture Notes in Computer Science, LNCS, pages 665–675. Springer-Verlag, 2002.
2. Johann Dréo, Patrick Siarry, Alain Pétrowski, and Eric Taillard. *Metaheuristics for Hard Optimization*, chapter Some Other Metaheuristics, pages 153–177. Springer Berlin Heidelberg, January 2006.

3. Agoston E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing*. SpringerVerlag, 2003.
4. M. Giacobini, M. Tomassini, A. Tettamanzi, and E. Alba. Selection intensity in cellular evolutionary algorithms for regular lattices. *IEEE Transactions on Evolutionary Computation*, 9(5):489–505, 2005.
5. Mario Giacobini, Enrique Alba, Andrea Tettamanzi, and Marco Tomassini. Modeling selection intensity for toroidal cellular evolutionary algorithms. In *GECCO '04: Proceedings of the 2004 conference on Genetic and evolutionary computation*, LNCS, pages 1138–1149. Springer Berlin / Heidelberg, 2004.
6. 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.
7. Mario Giacobini, Marco Tomassini, and Andrea Tettamanzi. Takeover time curves in random and small-world structured populations. In *GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation*, pages 1333–1340, New York, NY, USA, 2005. ACM.
8. 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*, Amsterdam, 1992. Elsevier Science Publishers, B. V.
9. Márk Jelasity, Alberto Montresor, and Ozalp Babaoglu. Gossip-based aggregation in large dynamic networks. *ACM Trans. Comput. Syst.*, 23(3):219–252, 2005.
10. 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.
11. Kenneth A. De Jong, Mitchell A. Potter, and William M. Spears. Using problem generators to explore the effects of epistasis. In Thomas Bäck, editor, *Proceedings of the Seventh International Conference on Genetic Algorithms (ICGA97)*, San Francisco, CA, 1997. Morgan Kaufmann.
12. 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.
13. Mike Preuss and Christian Lasarczyk. On the importance of information speed in structured populations. In *PPSN*, volume 3242, pages 91–100. Springer, 2004.
14. K. Sastry. Evaluation-relaxation schemes for genetic and evolutionary algorithms. Technical Report 2002004, University of Illinois at Urbana-Champaign, Urbana, IL., 2001.
15. Marco Tomassini. *Spatially Structured Evolutionary Algorithms: Artificial Evolution in Space and Time (Natural Computing Series)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.
16. Spyros Voulgaris, Márk Jelasity, and Maarten van Steen. *A Robust and Scalable Peer-to-Peer Gossiping Protocol*, volume 2872 of *Lecture Notes in Computer Science (LNCS)*, pages 47–58. Springer Berlin / Heidelberg, 2004.
17. D.J. Watts and S.H. Strogatz. Collective dynamics of "small-world" networks. *Nature*, 393:440–442, 1998.
18. 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.