

Addressing Churn in a Peer-to-Peer Evolutionary Algorithm

J.L.J. Laredo, P.A. Castillo, A.M. Mora, C. Fernandes, J.J. Merelo
Department of Architecture and Computer Technology
University of Granada, Spain
juanlu,pedro,amorag,cfernandes,jmerelo@geneura.ugr.es

Abstract—In this paper we analyse the robustness of a Peer-to-Peer (P2P) Evolutionary Algorithm (EA) subject to the following dynamics: peers leave the system independently from each other causing a collective effect known as *churn*. The algorithm has been designed to tackle large instances of computationally expensive problems and, in this paper, we will assess its behavior under *churn*. To that end, we have performed a scalability analysis in five different scenarios using the Massively Multimodal Deceptive Problem as a benchmark. In all cases, the P2P EA reaches the success criterion without a penalty on the response time. The key to the algorithm robustness is to ensure enough peers at the beginning of the experiment. Some of them leave but those that remain are enough to guarantee a reliable convergence.

I. INTRODUCTION

A Peer-to-Peer (P2P) Evolutionary Algorithm (EA) is a spatially structured EA in which the population structure is defined by a P2P overlay network [13]. As a spatially structured EA (see [23] for a survey), it can be modeled as a graph in which the vertices are individuals and edges represent relationships between them. A graph can be easily mapped to a network topology, consequently, a P2P EA can be easily distributed.

P2P EAs have been shown in [14] to be a suitable approach for tackling large instances of hard optimization problems via massive scalability of P2P systems. As a general property of optimization problems, the evaluation cost scales with respect to the size. Hence, large instances imply a bigger computational cost on the evaluation function. Additionally, the problem complexity increases with size, making the problem more difficult to solve. Resolution methods based on population, as EAs, require larger population sizes in order to tackle such instances with enough reliability and P2P systems are a large and mostly free source of computing resources.

However, Evolutionary Computing has recently entered this area and there are still many challenging issues. A P2P system is subject to the peers dynamics: Peers join the system, contribute some resources and leave it afterwards [17]. The independent arrival and departure of thousands of peers causes a collective effect called *churn* which has been modeled by Stutzbach and Rejaie in [22]. *Churn* is an inherent property of P2P systems and has to be taken into account in the design of any P2P application. In this paper we will assess a P2P EA under different *churn* scenarios.

The key to our study is the Evolvable Agent Model (EvAg) presented in [13]. It consists of a fine grained approach for parallelizing EAs in which there is a population of concurrent and self-scheduled agents performing the evolutionary steps of selection, variation and evaluation of individuals. The population structure is based on the the gossiping protocol newscast, a P2P protocol which builds an overlay network with small-world properties [11], [25]. Such a kind of small-world graphs have been shown, by Giacobini et al. in [5], to be suitable as population structure for an EA.

When tackling an optimization problem, the population size of an EA (i.e. number of nodes-individuals¹ in the graph) shows a dependence on the population structure and scales according to the given optimization problem. Therefore, we analyse the adequacy of our algorithm for large problem instances by scaling the Massively Multimodal Deceptive Problem (MMDP) [7] from small to large sizes. In order to assess the conditions in which our algorithm is fault tolerant, we have performed a scalability analysis using five scenarios. In the first one all nodes stay during the whole experiment while we set several intensities of *churn* in the rest. Note that we do not add any special feature to make *EvAg* fault tolerant, rather this study focuses on fault tolerance as an emergent property of the collective dynamics.

In all the scenarios, we have established that the algorithm has to reach success (i.e. find the problem optimum) with a 98% of reliability. To that end, we obtain the adequate population size using a method based on bisection [19] which establishes the minimum population size able to solve a problem with a given reliability (a success rate of 0.98 in our case). In spite of the departure of nodes, possibly containing valid solutions, our approach is able to reach success in all the cases (in the worst case the experiment ends with a 0.4% of the initial population). In all the test cases, the population size scales with curves of the same order. Nevertheless, the curves are shifted by a constant which depends on the *churn*, the more extreme the *churn* conditions, the bigger the constant is. The fact of yielding the problem optimum with enough reliability points to the redundancy of nodes as a natural mechanism for fault tolerance. Assuming no restrictions in the amount of available peers, the response

¹In 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

time of the algorithm scales independently of either the *churn* scenario and the population size, which confirms that the EvAg model is robust and fault tolerant.

Investigating scalability is of extreme importance when changing from a "toy problem" test environment to real-world problems which may require very large chromosomes to codify the solutions. However, real environments are restrictive for the estimation of an adequate population size (e.g. using bisection method). Additionally to the previous study, we have tried to gain some insight on the algorithm performance when the optimum population size is unknown in order to address practitioners on this issue. We have analysed how the success rate varies depending on the problem instance and the *churn* intensity. As a very general rule, the increase of the population size always benefits the search without an additional cost on the response time.

The rest of the paper is structured as follows. Section II introduces some related work in distributed Evolutionary Computing. The overall description of the EvAg model is presented in Section III. We expose, in Section IV, the methodology and the experimental setup. In Section V, we analyse the results of the experiments. Finally, conclusions and some future work lines are proposed in Section VI.

II. RELATED WORK

The idea of distributed Evolutionary Algorithms has been proposed from early (e.g. Grefenstette in [9]). Nowadays, parallel EAs are approached mainly under three methodologies: master-slave, islands and fine grained spatially structured EAs. Tomassini makes a good review in [23] of the different state of the art models with a special focus in fine grained approaches as the one presented in this paper.

Nevertheless, P2P EAs are more recent and not all the models fit with the issues involving P2P systems, such as decentralization, massive scalability or fault tolerance.

In the master-slave mode the algorithm runs on the master node and the individuals are sent for evaluation to the slaves, in an approach usually called *farming*. Such an architecture does not match decentralized structures and the master represents a single point of failure. In the island model, several panmictic EAs (islands) process their own population and exchange individuals between islands with a certain rate and frequency [2]. This coarse grained approach has been shown in [15] to be very sensitive to parameter calibration and P2P systems do not provide a priori knowledge of the global environment that an island model would need in order to set parameters such as the number of islands, the population size per island and the migration rate.

Finally, fine grained approaches are more suitable for decentralization as stated in [26], [12], where the key underlying idea is that individuals evolve within a defined set of neighbours. Following this line, we presented in [13] a formal model for P2P EAs, it is the *Evolvable Agent model* that we analyse in this paper under *churn* conditions. The model uses the gossiping protocol newscast [11] as population structure. Newscast was proposed within the DREAM project [1],

one of the pioneers in distributed P2P EAs. The island-based parallelization of DREAM was shown in [16] to be insufficient for tackling large-scale decentralized scenarios. Nevertheless, the gossiping protocol newscast forms a communication graph with small-world features and such a kind of graphs have been shown to be suitable as population structure for Genetic Algorithms in [6], [18], [5].

As a gossiping protocol, newscast is scalable and robust [25], however, there are no evidences about if an EA based on it might be fault tolerant or scalable by extension, hence the study that we are proposing.

There are some works addressing fault tolerance in distributed EAs, Fernández de la Vega in [24] advanced that Evolutionary Algorithms are fault tolerant because its nature and design. However, the employed methodologies, so far, do not have a direct correspondence with a real non-forced scenario. E.g. Hidalgo et al. analyse the fault tolerance of the island model in [10] dying out up to the 50% of the computational resources. Lombráña and Fernández de la Vega in [8] use two rates for processors failures, respectively 2% and 10% every generation. Scriven et al. propose a distributed Multi-Objective Particle Swarm Optimisation in [20] and a P2P approach in [21]. Nevertheless, they use failure rates of 0%, 5%, 10% and 20% which, at the end, are ad-hoc values.

Our study is based on the analysis of Stutzbach and Rejaie of the peers dynamics in [22]. That is, we model faults following the behaviour of real P2P systems. Therefore, the key contribution of this paper is proving that our proposal allows massive scalability of a distributed EA being robust under *churn*.

III. OVERALL MODEL DESCRIPTION

The overall procedure of our approach consists of a population of Evolvable Agents (EvAg), described in Section III-A, whose main design objective is to carry out the main steps of evolutionary computation: selection, variation and evaluation of individuals [4]. Each EvAg is a node within a newscast topology in which the edges define its neighborhood. For the sake of simplicity, we assume a newscast node as a peer. However, a peer could hold several nodes in practice.

A. Evolvable Agent

An *Evolvable Agent (EvAg)* itself is an EA composed of a single individual [12], [13]. In spite of the model not having a *population* in the canonical sense, adjacent EvAgs provide each other with the genetic material that they require to evolve. Therefore, we talk about a population of EvAgs instead of a population of individuals.

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

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

Algorithm 1 Evolvable Agent

```
 $S_t \leftarrow$  Initialize Agent
loop
  Sols  $\leftarrow$  Local Selection(newscast) See algorithm 2
   $S_{t+1} \leftarrow$  Recombination(Sols)
   $S_{t+1} \leftarrow$  Mutation( $S_{t+1}$ )
  Evaluate( $S_{t+1}$ )
  if  $S_{t+1}$  better than  $S_t$  then
     $S_t \leftarrow S_{t+1}$ 
  end if
end loop
```

newly generated solution S_{t+1} is better than the old one S_t , it replaces the current solution.

B. Population structure

In principle, our method places no restrictions in the choice of population structure, although this choice will have an impact on the dynamics of the algorithm since it establishes the environmental selection pressure. As it has been previously said, we apply the newscast protocol as graph structure. Within this section we do not enter on the dynamics but on its functioning elements (see [11], [25] for further details). Algorithm 2 shows the pseudo-code of the main tasks in the self-organized process which builds the newscast graph. Each node maintains a cache with one entry per node in the network at most. Each entry provides information about a foreign node: A time-stamp of the entry creation (it allows the replacement of old items), and an agent identifier.

Algorithm 2 Newscast protocol in node $EvAg_i$

```
Active Thread
while  $EvAg_i$  not finished do
  sleep  $\Delta T$ 
   $EvAg_j \leftarrow$  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 while

Passive Thread
while  $EvAg_i$  not finished do
  wait  $Cache_j$  from  $EvAg_j$ 
  send  $Cache_i$  to  $EvAg_j$ 
   $Cache_i \leftarrow$  Aggregate ( $Cache_i, Cache_j$ )
end while

Local Selection(newscast)
 $[EvAg_h, EvAg_k] \leftarrow$  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 ($EvAg_h(S_t)$ and $EvAg_k(S_t)$). After ΔT time each $EvAg_i$ initiates a communication process (active thread). It selects randomly an $EvAg_j$ 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 items (newscast) for each cache entry in $Cache_i$, $Cache_j$ and merging them into a single cache that $EvAg_i$ and $EvAg_j$ will share. We have fixed ΔT to once per evaluation.

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. Therefore, the cache size is smaller than the number of nodes (typically around $\log(n)$) in order to get small-world features such as a small characteristic path length and a high clustering coefficient (for further details on the dynamics refer to [25]). We have fixed the cache size to 20 based on the study of performance for different cache sizes in [13].

IV. METHODOLOGY AND EXPERIMENTAL SETUP

In this paper we try to assess the impact of the *churn* in our P2P EA approach when tackling the Massively Multimodal Deceptive Problem (MMDP).

We have focused on the Success Rate (SR) as a metric of the performance of an EA (i.e. times that the algorithm finds the optimum solution out of all trials [4]). In this study, there are three variables that affect the SR: The size of the problem (k), which will conduct to a scalability analysis, the intensity of *churn* (λ) described in Section IV-A and the population size (P).

Any of the variables have the following influence on the SR if we assume a fixed value for the rest of them. In the case of the problem instance, the bigger the size, the lower the SR. With respect to *churn*, the more departures of nodes, the lower the SR. Finally, the bigger the population size, the higher the SR.

If we consider λ and k as two independent variables under the condition of obtaining a SR of 0.98, the population size can be expressed as a function $f(\lambda, k) = P$. In order to find a fixed condition and obtain P , the bisection method has been used (explained in Section IV-C). We will analyse the curves produced by $f(\lambda, k)$ when $\lambda = 1, 5, 10, 50, \infty$ (where ∞ stands for no churn) and $k = 2, 4, 8, 16, 32, 64$.

Furthermore, the response time of the algorithm is also analysed as a function of the three variables $g(\lambda, k, P)$ since one of the goals in any distributed EA is to reduce the time for obtaining a solution.

Finally, this empirical study has been carried out using a benchmark problem in which estimating the optimal population size is possible. Nevertheless, such an estimation could turn into unapproachable when switching to hard time consuming real problems. Therefore, we have performed a last case of study using three different policies for the population size and analysing how the SR varies.

A. Modeling Churn

Following the work by Stutzbach and Rejaie in [22], there are two main group-level properties of *churn* which characterize the behaviour of all participating peers: The *inter-arrival time* and the *session length*, respectively, the time between two sessions and the time from the beginning to the end of a session.

In this study we have assumed that all nodes start at the same time with a certain *session length*. The *session length* can be modeled randomly from a Weibull distribution using the following formula:

$$X = \lambda(-\ln(U))^{\frac{1}{s}} \quad (1)$$

where U is drawn from the uniform distribution, s stands for the shape and λ for the scale of the Weibull distribution. The analysis in [22] exposes that the *session length* of different P2P systems fit with a shape of $s \approx 0.40$ but the values of λ differ. Additionally, the simulator driven experiments define the time unit as a simulator cycle which could apply for different time metrics in real time. Hence, we setup the following values for $\lambda = 1, 5, 10, 50, \infty$ depicted in Figure 1. It shows the complementary cumulative distribution functions (CCDF), representing the percentage of remaining nodes at each moment of the experiment for the different values of λ (e.g. in the cycle 10, $\sim 8\%$ of the peers remain for $\lambda = 1$ and $\sim 100\%$ for $\lambda = 50$).

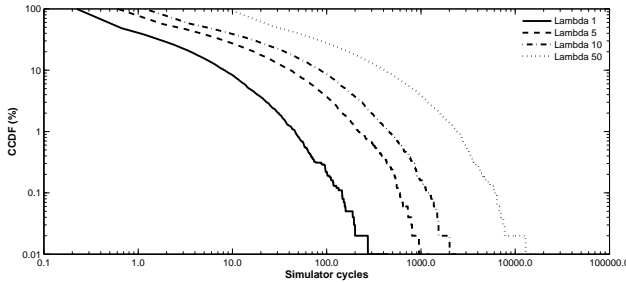


Fig. 1. Complementary cumulative distribution functions

In spite of the initial assumption, the set of *churn* scenarios allows a worst case analysis in which the system loses peers to zero. Once that a node leaves the experiment, it does not rejoin again (i.e. there is no *inter-arrival*).

B. Massively Multimodal Deceptive Problem (MMDP)

The Massively Multimodal Deceptive Problem (MMDP) [7] is designed to be difficult for an EA which has to find the optimum in spite of deceptive attractors. 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 I.

The fitness value is defined as the sum of the s_i subproblems with an optimum of k (equation 2). 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} .

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

TABLE I
BASIC DECEPTIVE BIPOLAR FUNCTION (s_i) FOR MMDP

Additionally, such a complexity is specially remarkable under deceptive conditions.

We consider several instances from low to high difficulty using $k = 2, 4, 6, 8, 10, 16, 32, 64$.

$$f_{MMDP}(\vec{s}) = \sum_{i=1}^k fitness_{s_i} \quad (2)$$

C. A method for estimating the population size

The bisection method [19] was used to determine the optimal *EvAg* population size P , that is, the lowest P for which 98% of the runs solve the MMDP. To find it, mutation rate is set to 0, so as to search a minimum population size such that using random initialization it is able to provide enough building blocks to converge to the optimum without other mechanism than recombination.

Algorithm 3 depicts the method based on bisection. 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 (0.98 of Success Rate). After that, the interval (min, max) is halved several times and the population size adjusted within such a range. min and max stand respectively for the minimum and maximum population size estimated.

Algorithm 3 Method based on Bisection

```

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

```

D. Experimental Setup

Standard operators and parameters in EAs were used for the experiments. The recombination operator is DPX with $p_c = 1.0$ and the selection of parents uses binary tournament (see e.g. [4]). All results are averaged over 50 independent runs. The experiments were conducted in PeerSim Simulator².

²<http://peersim.sourceforge.net/>. Accessed on July 2008.

V. RESULTS

Figure 2 shows the respective response times of the algorithm for $k = 2, 4, 8, 16, 32, 64$. They are the result of averaging 50 runs in every possible *churn* scenario (i.e. $\lambda = 1, 5, 10, 50$). That way, every intersection between a vertical line and a CCDF represents the percentage of individuals in P for which the algorithm is expected to end on a given instance k and a given scenario λ . The effects of *churn* are more pernicious as the instances scale since the response time increases (e.g. for a $\lambda = 50$, $k = 2$ ends with a 100% of the initial population while $k = 64$ ends with a $\sim 35\%$ of P). Additionally, the lower the λ , the more departures of nodes (e.g. when $k = 64$, $\lambda = 50$ ends with a $\sim 35\%$ of the initial population and $\lambda = 1$ with a $\sim 0.4\%$).

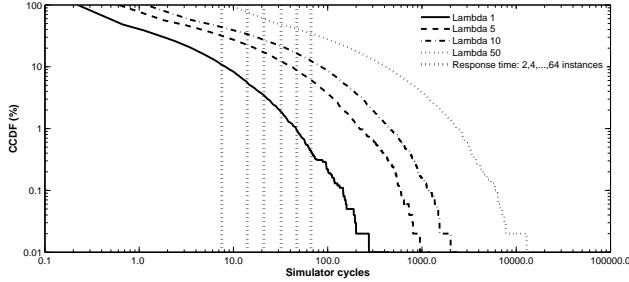


Fig. 2. Complementary cumulative distribution functions and average response time

Under these conditions, Figure 3 shows the scalability of the population size (P) as different curves of k , that is, k scales and λ remains fixed in $f(\lambda, k) = P$. The scalability of $f(\lambda, k)$ fits with a complexity order of $O(k^2)$ in any of the scenarios. Hence, *churn* does not damage the scalability order (i.e. the curves are just shifted by a constant which is *churn* dependent) and a reliable convergence can be guaranteed by ensuring enough resources. This fact points to the robustness of the Evolvable Agent model.

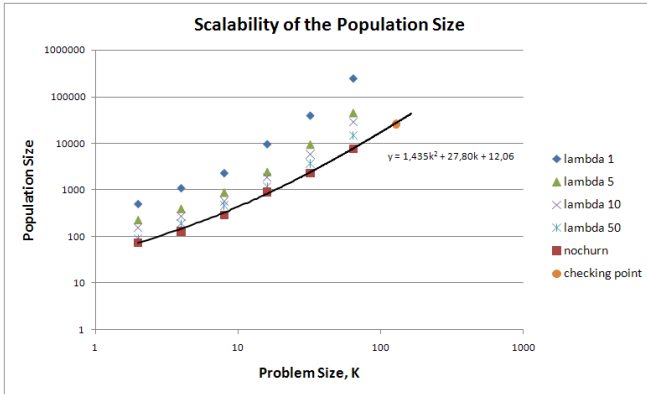


Fig. 3. Scalability of the population size $f(\lambda, k) = P$ for the MMDP in 5 scenarios of churn (λ)

Additionally, Figure 4 shows the response time of the algorithm $g(\lambda, k, P)$. g is independent from λ and P , with an order $O(k^{0.6})$. In this case, *churn* and the population size

do not affect the scalability (neither shifting the curves with a constant) and the response time is completely dependent on the problem instance k . That means that we can expect the same response time under any *churn* scenario if we ensure enough resources to satisfy the 0.98 of SR condition. Therefore, the algorithm is robust under *churn*.

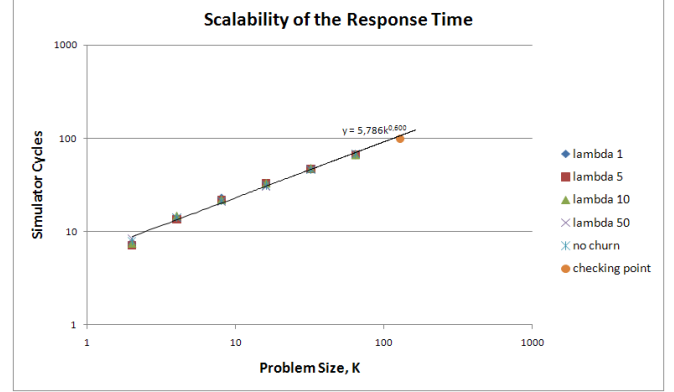


Fig. 4. Scalability of the response time $g(\lambda, k, P)$ for the MMDP in 5 scenarios of churn (λ) using P estimated in $f(\lambda, k)$

The Figure 5 shows the scalability of the computational effort with *no churn*: population size remains constant along of a run. Such a scenario is the one that spends less evaluations since *no churn* needs smaller population sizes. However, the algorithm in a *churn* scenario loses nodes very quickly as k scales (e.g. Figure 2), then, it might scale better than using a fixed population size. If that would be the case, *churn* would turn into a good alternative to save computing resources in large-scale problem instances as when using Plagues in Genetic Algorithms [3]. We find this hypothesis of the maximum interest and will try to address it in future works.

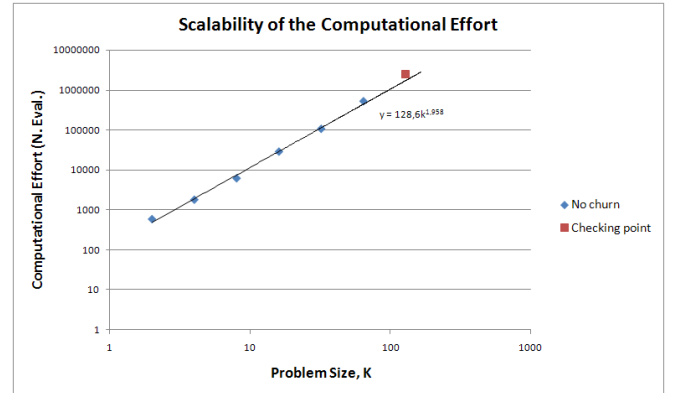


Fig. 5. Scalability of the computational effort (or number of evaluations) for the no churn case

Finally, we have performed a study assuming that the optimal population size can not be estimated under *churn*. This way, we intend to address how the SR varies depending on the *churn* (λ) and the problem instances (k).

We have considered three different policies to establish the population size P .

- 1) Assuming that the population size with *no churn* can be estimated. P is the optimal population size in a *no churn* scenario.
- 2) P is four times the optimal size in a *no churn* scenario.
- 3) Assuming that P can not be estimated but there are many resources in the P2P system. P is an ad-hoc big population size i.e. $P = 200000$.

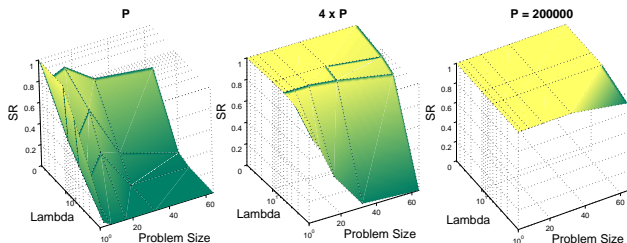


Fig. 6. Success Rate (SR) as a function of λ , k and P

Figure 6 shows three graphs of the respective policies. They show that the SR decreases with respect to both λ and k . But the change is specially noticeable for small values of λ (e.g. for $\lambda = 5, 10, 50$ policies 2 and 3 show a reliable convergence to the optimum which decreases with $\lambda = 1$). Therefore, it would be very convenient to analyse the *churn* of a P2P system before deploying a P2P EA. If we pay attention to the policy 3 (i.e. $P = 200000$) we can see how the increase of the population size always benefits the search, the worst value here is a SR of 0.9 with $k = 64$ and $\lambda = 1$.

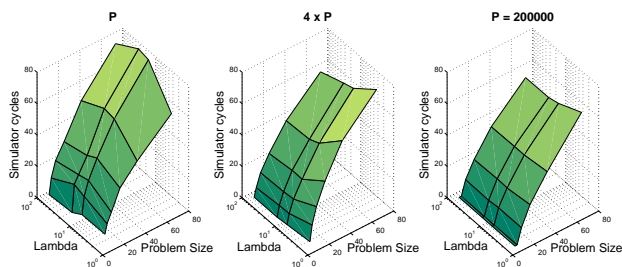


Fig. 7. Response time

Besides, Figure 7 shows the response times for the three policies. It confirms the results on Figure 4, the response time of the algorithm is independent from λ , P and the SR. Therefore, it can be expressed as a function of k and do not depend on the availability of resources (P). However, the increase of P guarantees a better SR in any scenario.

VI. CONCLUSIONS

In this paper we have analysed a P2P Evolutionary Algorithm and proved its adequacy for tackling large problem instances under five *churn* scenarios. We have used the Massively Multimodal Deceptive Problem as a benchmark, but, given that it has been designed to be difficult for EAs, these results should be easily extended to more general discrete or combinatorial optimization problems.

Our approach deals with P2P issues such as decentralization, large-scalability and *churn*. To this end, the population structure is managed by the newscast gossiping protocol. Through the experimental results we conclude that large instances of hard optimization problems can be tackled in P2P systems using the Evolvable Agent (EvAg) model in spite of aggressive *churn* conditions.

The population size scales with polynomial order with respect to the problem size which demands for a big amount of resources. Besides, the expected response time of the algorithm scales with fractional power with respect to the problem size which makes the algorithm efficient. The approach shows to be robust with respect to *churn*, once that the estimated population size guarantees a reliable convergence to the problem optimum, the departure of nodes does not inflict a penalization on the response time.

As future lines of work, we intend to assess the impact of both, the latency between peers and the use of the gossiping algorithm, on the run-time performance. We expect that the idle processing time decreases as the problem instances scale (i.e. bigger instances require a bigger computational time while the communication time can be assumed as fixed). Additionally, we will analyse our approach taking into account bootstrapping and heterogeneous nodes. Finally, through the course of this investigation, we have formulated an hypothesis about the loss of nodes as a good alternative to save computational effort in large-scale problems. Such an hypothesis will be tested using different probability distributions.

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] Erick Cantú-Paz. *Efficient and Accurate Parallel Genetic Algorithms*. Kluwer Academic Publishers, Norwell, MA, USA, 2000.
- [3] F. Fernández de Vega, Erik Cantú-Paz, J.I. López, and T. Manzano. Saving resources with plagues in genetic algorithms. In *Parallel Problem Solving from Nature - PPSN VIII*, LNCS, pages 272–281, 2004.
- [4] Agoston E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing*. Springer-Verlag, 2003.
- [5] Mario Giacobini, Mike Preuss, and Marco Tomassini. Effects of scale-free and small-world topologies on binary coded self-adaptive CEA. In *Evolutionary Computation in Combinatorial Optimization - EvoCOP 2006*, volume 3906 of LNCS, pages 85–96, Budapest, 10-12 April 2006. Springer Verlag.
- [6] 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.
- [7] David E. Goldberg, Kalyanmoy Deb, and Jeffrey Horn. Massive multimodality, deception, and genetic algorithms. In *Parallel Problem Solving from Nature, 2*, Amsterdam, 1992. Elsevier Science Publishers, B. V.

- [8] Daniel Lombrana Gonzalez and Francisco Fernandez de Vega. On the intrinsic fault-tolerance nature of parallel genetic programming. In *PDP '07: Proceedings of the 15th Euromicro International Conference on Parallel, Distributed and Network-Based Processing*, pages 450–458, Washington, DC, USA, 2007. IEEE Computer Society.
- [9] J.J. Grefenstette. Parallel adaptive algorithms for function optimization. Technical Report CS-81-19, Vanderbilt University, Computer Science Department, Nashville, 1981.
- [10] J. Ignacio Hidalgo, Juan Lanchares, Francisco Fernández de Vega, and Daniel Lombrana. Is the island model fault tolerant? In *GECCO '07: Proceedings of the 2007 GECCO conference companion on Genetic and evolutionary computation*, pages 2737–2744, New York, NY, USA, 2007. ACM.
- [11] 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.
- [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] J.L.J. Laredo, P.A. Castillo, A.M. Mora, and J.J. Merelo. Exploring population structures for locally concurrent and massively parallel evolutionary algorithms. In *IEEE Congress on Evolutionary Computation (CEC2008), WCCI2008 Proceedings*, pages 2610–2617. IEEE Press, Hong Kong, June 2008.
- [14] J.L.J. Laredo, A.E. Eiben, M. van Steen, P.A. Castillo, A.M. Mora, and J.J. Merelo. P2P Evolutionary Algorithms: A suitable approach for tackling large instances in hard optimization problems. In Emilio Luque, Tomas Margalef, and Domingo Benitez, editors, *Euro-Par*, volume 5168 of *Lecture Notes in Computer Science*, pages 622–631. Springer, 2008.
- [15] Juan Luis Jiménez Laredo, Pedro A. Castillo, Antonio Miguel Mora, and Juan Julián Merelo Guervós. Evolvable agents, a fine grained approach for distributed evolutionary computing: walking towards the peer-to-peer computing frontiers. *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, 12(12):1145–1156, 2008.
- [16] Juan Luis 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.
- [17] J.J. Merelo, P.A. Castillo, J.L.J. Laredo, A.M. Mora, and A. Prieto. Asynchronous distributed genetic algorithms with Javascript and JSON. In *IEEE Congress on Evolutionary Computation (CEC2008), WCCI2008 Proceedings*, pages 1372–1379. IEEE Press, Hong Kong, June 2008.
- [18] Mike Preuss and Christian Lasarczyk. On the importance of information speed in structured populations. In *PPSN*, volume 3242, pages 91–100. Springer, 2004.
- [19] K. Sastry. Evaluation-relaxation schemes for genetic and evolutionary algorithms. Technical Report 2002004, University of Illinois at Urbana-Champaign, Urbana, IL., 2001.
- [20] I. Scriven, D. Ireland, A. Lewis, S. Mostaghim, and J. Branke. Asynchronous multiple objective particle swarm optimisation in unreliable distributed environments. In *IEEE Congress on Evolutionary Computation (CEC2008), WCCI2008 Proceedings*. IEEE Press, Hong Kong, June 2008.
- [21] I. Scriven, A. Lewis, D. Ireland, and J. Lu. Decentralised distributed multiple objective particle swarm optimisation using peer to peer networks. In *IEEE Congress on Evolutionary Computation (CEC2008), WCCI2008 Proceedings*. IEEE Press, Hong Kong, June 2008.
- [22] Daniel Stutzbach and Reza Rejaie. Understanding churn in peer-to-peer networks. In *Proceedings of the 6th ACM SIGCOMM on Internet measurement (IMC 2006)*, pages 189–202, New York, NY, USA, 2006. ACM Press.
- [23] Marco Tomassini. *Spatially Structured Evolutionary Algorithms: Artificial Evolution in Space and Time (Natural Computing Series)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.
- [24] Francisco Fernandez De Vega. A fault tolerant optimization algorithm based on evolutionary computation. In *DEPCOS-RELCOMEX '06: Proceedings of the International Conference on Dependability of Computer Systems*, pages 335–342, Washington, DC, USA, 2006. IEEE Computer Society.
- [25] Spyros Voulgaris, Márk Jelasity, and Maarten van Steen. *Agents and Peer-to-Peer Computing*, volume 2872 of *Lecture Notes in Computer Science (LNCS)*, chapter A Robust and Scalable Peer-to-Peer Gossiping Protocol, pages 47–58. Springer Berlin / Heidelberg, 2004.
- [26] 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.

