Computer Science and Information Systems 15(3):733–750 https://doi.org/10.2298/CSIS180115032N

On the Use of Self-⋆ Island-based Evolutionary Computation Methods on Complex Environments*

Rafael Nogueras and Carlos Cotta✉

ETSI Informática, Campus de Teatinos,
Universidad de Málaga, 29071 Málaga, Spain
ccottap@lcc.uma.es

Abstract. We consider the use of island-based evolutionary algorithms (EAs) on fault-prone computational settings. More precisely, we consider scenarios plagued with correlated node failures. To this end, we use the sandpile model in order to induce such complex, correlated failures in the system. Several EA variants featuring self-adaptive capabilities aimed to alleviate the impact of node failures are considered, and their performance is studied in both correlated and non-correlated scenarios for increasingly large volatility rates. Simple island-based EAs are shown to have a significant performance degradation in the correlated scenario with respect to its uncorrelated counterpart. Resilience is however much improved via the use of self-⋆ properties (self-scaling and self-healing), which leads to a more gentle degradation profile. The inclusion of self-generation also contributes to boost performance, leading to negligible degradation in the scenarios considered.

Keywords: evolutionary algorithms, memetic algorithms, self-⋆ properties, ephemeral computing, sandpile model.

1. Introduction

The use of parallel environments is of paramount interest for tackling intensive computational tasks. This turns out to be one of the fundamental advantages of metaheuristics, and more specifically of its population-based variants, given their amenability for being implemented in parallel and distributed environments [2]. In particular, evolutionary algorithms (EAs) [14] have a long success story in this kind of environments, dating back to the 1980s [19,21,22]. In this sense, there has been during the last years an important focus on the use of EAs in emergent computational scenarios that depart from classical dedicated networks so common in the past. Among these we can cite cloud computing [35], P2P networks [37,60], or volunteer computing [9,51], just to name a few. The dynamic nature of the underlying computational substrate is one of the most distinguished features of some of these new scenarios –consider for example a P2P network in which nodes may enter or leave the system subject to some uncontrollable dynamics caused by user interventions, network disruptions, eventual crashes, etc. The term *churn* is used to denote this phenomenon [54]. Under some circumstances, a potential solution to this issue might be to hide these computational fluctuations under an intermediate layer, thus providing a virtual stable environment to algorithms running on it. Nonetheless, this can constitute a formidable challenge, mainly in situations in which the underlying substrate is composed

* This is an extended version of [49].

of nodes with low computing power just providing brief, ephemeral bursts of computation (think of, e.g., a large collection of low-end networked devices –cell phones, smart wearables, etc.– contributing their idle time). Making an effective use of such highly-volatile resources is the key idea that underlies the notion of *Ephemeral Computing* (Eph-C), which has been recently defined as the use and exploitation of computing resources whose availability is ephemeral (i.e., transient and short-lived) to carry out complex computational tasks [10].

In order to approach Eph-C, algorithms can be made fully aware of the complex dynamic nature of the underlying computational environment and adapted to work natively on these conditions. In this sense, EAs can fit very well to this scenario. To begin with, they are intrinsically resilient at a fine-grained scale (this has been shown in master-slave models, where a reliable node runs the central logic of the algorithm and individual operations –such as fitness calculations or applications of reproductive operators– are distributed on fault-prone nodes [28,29,30,31]). The situation is however somewhat different at a coarse grain level. Consider, for example, the so-called island model of evolutionary algorithms [55], whereby multiple populations evolve in parallel (be it just as an algorithmic construct or as truly physically-distributed processes) and occasionally exchange information (leading to better solutions as well as to a great reduction of the computational cost necessary to achieve these [1]). As it is well known, an unstable computational environment such as the one described can lead to the loss of the current best solution [23] and will also negatively impact genetic diversity and the progress of the search process. In order to address these problems effectively, several strategies have been proposed in the past, such as redundancy [15] or epidemic algorithms [16] in the case of fine-grained models, and checkpointing [34,45] in the case of coarse-grained models, to name just a few.

Although checkpointing (that is, saving periodic snapshots of the state of the volatile nodes), seems to be a very sensible option in advance, it is important to consider that in order for it to work properly, some type of access to external stable storage is required. The main drawback of this is that it can contribute a significant overhead, mainly when the frequency of these snapshots is required to be high, e.g., due to a high failure rate in the nodes [43], with the subsequent performance degradation. A much more attractive option is to provide the algorithms with (self-)adaptive strategies so that they can react autonomously to changes in the environment, adjusting their behavior or functioning accordingly. Such strategies are often captured under the general umbrella term of self-$\star$ properties [4,7], that encompasses all those different mechanisms through which a system can self-manage any aspect of its own functioning. In this particular context we are discussing, these strategies should preferably have a decentralized nature since centralized control strategies are less likely to achieve a consistent image of the state of the computational landscape at a given time, and therefore, the decisions that arise from them would lag the changing conditions of the latter, cf. [10]. Again, this fits well to the intrinsically decentralized nature of island-based models and with the amenability of EAs for self-adaptation [13,24]. Indeed, this capability has been prominently featured by these techniques since the initial developments of the paradigm. In particular, it is currently central in the area of memetic computing [50] which can be described as the study of complex and dynamic computing structures composed of interacting modules (memes) whose evolution dynamics is inspired by the diffusion of ideas [39]. This definition tries

to bridge the notion of memetic algorithms [40] –understood as optimization algorithms that combine local and global search strategies (often using an evolutionary search engine for the latter purpose) and orchestrate their interplay in a synergistic way– with some specific self-⋆ properties such as self-organization and self-generation [27].

Recent work has precisely studied the use of self-⋆ properties such as self-scaling [46] and self-healing [47] in this context, providing some evidence on the contribution of these techniques to the robustness of the algorithm when run on unstable computational environments. Quite interestingly, these previous studies have however only considered simple network models in which the dynamics of each node is independent of the rest of the network, that is, the availability of a computing node does not depend on the availability of other nodes. A more general situation would encompass complex correlated availability patterns, that is, situations in which the dynamics of each node might be affected by the dynamics of other nodes, see e.g., [25]. Overall, the presence of correlated failures puts to test the robustness and resilience of the EA, and hence studying it can provide a wider perspective on the usefulness of self-⋆ techniques to cope with computational instability.

In this work, our initial study carried out with EAs in [49] is extended to island-based multimemetic algorithms (MMAs). MMAs [26] are an extension of memetic algorithms in which computational representations of problem solving strategies (neighborhood definitions for a local search operator in this case) are explicitly stored and evolved as a part of solutions, very much in line with the concept of memetic computing anticipated above. By deploying these techniques in a scenario as described different self-⋆ properties (self-generation, self-scaling and self-healing) are layered and put to test in this demanding context. More precisely, we consider their deployment on a simulated computational environment that allows experimenting with different churn rates and, therefore, exploring their limits, see Sect. 2.1. We relied on previous work [42,44,46] to address the different self-⋆ properties considered, as described in Sect. 2.2. We present the results of a broad empirical evaluation of the strategies considered in Sect. 3. We close with conclusions and an outline of future work in Sect. 4.

2. Methodology

We consider an island-based EA running on a simulated unstable environment. Each island runs on a computational node of the system, whose availability fluctuates along time. When a computational node goes down, its contents are lost. Similarly, when a computational node is reactivated, the island running on it must be created anew in some way. In the following subsections we shall describe in more detail the model of the computational scenario and the mechanisms used by the EA to cope with instability.

2.1. Network Model

Let us consider a network composed on n_ι nodes interconnected following a certain topology. More precisely, we consider a regular toroidal lattice with von Neumann connectivity (virtual topology used for the purposes of migration in the island model) overlaid on a scale-free network (underlying topology for the purposes of failure correlation) as it is often the case in P2P networks, e.g., [33] – see Fig. 1. In the latter, node degrees are distributed following a power-law (i.e., the fraction $p(d)$ of nodes with d neighbors goes as

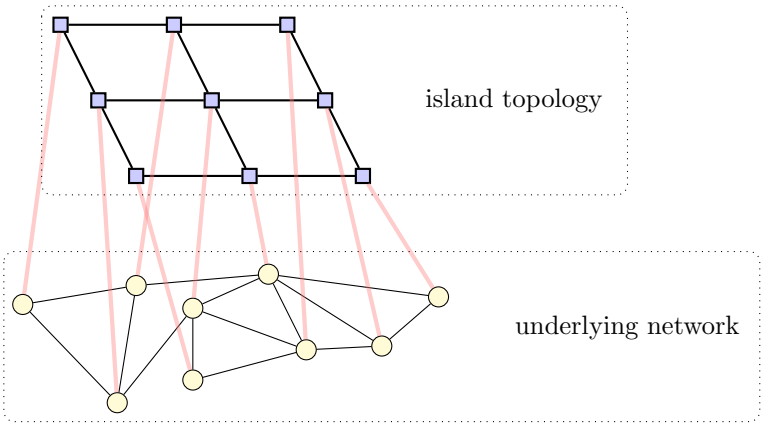

Fig. 1. Depiction of the network model: a toroidal von Neumann 2D grid (wrap-around links not shown to avoid clutter) is overlaid on an underlying scale-free network.

$p(d) \sim d^{-\gamma}$ for some constant parameter γ) and hence there will be a few hubs with large connectivity and increasingly more nodes with a smaller number of neighbors. To generate this kind of networks we use the Barabási-Albert model [3], whereby the network is grown from a clique of $m + 1$ nodes by adding a node at a time, connecting it to m of the nodes previously added (selected with probability proportional to their degree – the so-called, preferential attachment mechanism [6]) where m is a parameter of the model.

As stated before, these nodes are volatile, and may abandon the system and re-enter it at a later time, eventually repeating the process over and over again. To model this instability we consider two scenarios: (i) independent or non-correlated failures and (ii) correlated failures. The first one is the simplest model. Therein, the dynamics of each node is independent of other nodes. Each of them can switch from active to inactive or vice versa independently of other nodes with some probability that only depends on the time it has been in its current state. More precisely, let $p(t)$ be the probability of remaining in the same state after time t. Following previous work, as well as the commonly observed behavior of e.g., P2P systems [54], $p(t)$ follows a Weibull distribution:

$$p(t) = \exp\left(-(t/\beta)^{\eta}\right) \tag{1}$$

This distribution is controlled by two parameters β and η. The first one is the scale parameter and captures the spread of the distribution. The larger this parameter, the less frequent failure events are. The second one is the shape parameter and captures the effect that time has on failure events: for $\eta > 1$ (resp. $\eta < 1$), the longer the time elapsed, the more (resp. less) likely a failure event will be. If η was exactly 1, failures would be time-independent (i.e., the hazard function would be constant) and hence the time to failure would be exponentially distributed.

As to the correlated scenario, it features node failures that will be influenced by neighboring nodes. Consider for example the case of sensor networks in which nodes with a large number of active neighbors have their energy depleted faster due to the increased

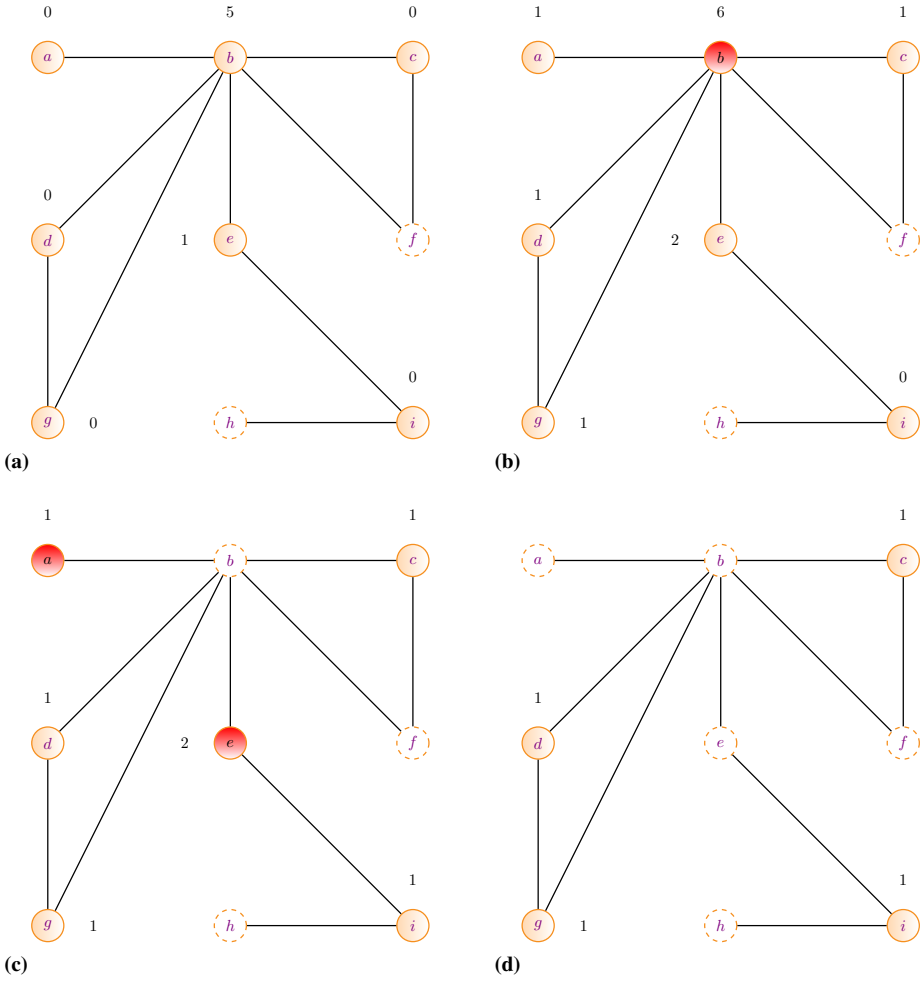

Fig. 2. Example of failure propagation in the sandpile model. Active (resp. inactive) nodes are depicted with solid (resp. dashed) borders. The numbers next to each active node indicate the cumulated number of failure events. The threshold θ_i for each node equals here its degree. (a) Initial state (b) Failure on node b (c) Failure propagation to nodes a and e (d) Final state.

energy toll for communications, or the case of networks that carry load and in which the failure of a node makes other ones absorb the load of the latter, eventually resulting in additional overload failures [25]. This can be modeled in different ways, e.g., [8,59]. In this work we have considered the sandpile model in order to induce cascading failures [12]. Much like in the previous case, we consider micro-failure events happening on each node with a certain probability. Now, each node i will have an associated threshold value θ_i, indicating the number of micro-failure events required for it to go down. When the number

of such micro-failures effectively equals this threshold, the node is disconnected from the system, and each of the active neighbors of this node receives an additional micro-failure event[1]. In case any of these neighbors now accumulated a number of micro-failures equal to its own threshold, it would go down as well, propagating in turn another micro-failure to its active neighbors, and so on (hence the possibility of cascading failures). Fig. 2 shows an example: after node b (which was in a critical state, i.e., one event short of going down) fails, neighboring nodes a and e (which were also in such a critical state) fail as well. We have considered for simplicity that the threshold θ_i of each node i is constant and equal to the number of neighbors (active or inactive) of the node. As to reactivation, just like in the non-correlated case a single event is required.

2.2. Algorithmic Model

The core algorithm considered is a steady-state EA with one-point crossover, bit-flip mutation, binary tournament selection and replacement of the worst parent. This algorithm is deployed on a computational environment such as described in the previous section by means of the island-model, whereby each node hosts an island that runs an instance of the previously mentioned EA. After each iteration of the basic EA, these islands perform migration (stochastically with probability p_{mig}) of a single individual to neighboring islands. In each migration event the migrant is randomly selected from the current population and the receiving island inserts it in its population by replacing the worst individual [41].

Four variants of the island-based EA are considered by endowing it with different sets of self-$\star$ properties:

- $\emptyset$: a simple island-based EA (simply termed EA in the following) in which every island has a fixed size and random reinitialization is used whenever a new node enters the system.
- $\{$self-generation$\}$: an island-based MMA (simply denoted as MMA in the following) obtained by endowing EA with self-generation capabilities.
- $\{$self-scaling, self-healing$\}$: a self-adaptive island-based EA (denoted as EA*) obtained by augmenting the EA with self-scaling and self-healing to re-size each island individually in response to fluctuations in the number of active neighbors and in the population sizes of these.
- $\{$self-scaling, self-healing, self-generation$\}$: a self-adaptive island-based MMA (termed MMA*) obtained by endowing the MMA with self-scaling and self-healing capabilities.

These self-$\star$ properties are described in the following.

Self-generation. Self-generation refers to the capability of the algorithm to redefine the search tools it uses, that is, to create and adapt these to the circumstances of the search process. We have considered a framework similar to that defined by Smith [52,53]. More specifically, each individual in the population contains a binary genotype and a single meme, the latter being represented as a rewriting rule. These rules have the form $A \rightarrow C$,

[1] It must be noted that these so-called micro-failures are not intended to represent any real phenomenon, but are just used as a means to introduce failure interdependencies.

where both A and C are strings of the same length taken from $\{0, 1, \#\}^*$, that is, the binary alphabet used to represent the solutions plus a wildcard ('#'). This meme defines a neighborhood relationship in the search space: given a genotype $g \in \{0, 1\}^n$ (i.e., a binary string of length n), its neighborhood $\mathcal{N}^{(M)}(g)$ induced by meme $M \equiv A \rightarrow C$ is defined as

$$\mathcal{N}^{(M)}(g) = \{g' \mid g \xrightarrow{M} g'\} \tag{2}$$

i.e., it is composed of all binary strings attainable by applying the rewriting rule M to g, that is, finding a match of the antecedent A in g (using the wildcard as "don't care" symbol) and replacing it with the consequent C (where the wildcard now means "don't change"). This neighborhood is subsequently used to optimize g by sampling $\mathcal{N}^{(M)}(g)$ w times (where w is a parameter used to keep the cost of the process under control) and keeping the best solution found (if better than the original genotype g). Note that since memes are a part of solutions, they are also subject to mutation [52], and they are transferred from parents to offspring through local selection (the offspring inherits the meme of the best parent).

Self-scaling. Self-scaling refers to the capability of the algorithm to adapt its functioning in response to changes in the scale parameters of the task being tackled, either of the problem under consideration or of the computational substrate on which the algorithm is run. In this context we focus precisely on this latter issue, and use self-scaling strategies aimed to attain a rather stable global population size across the islands that are active in each particular moment. To this end, each island periodically monitors the state of its neighbors to determine: whether they are active or not, their population sizes and the number of active neighbors they have in turn. When a neighboring island is detected to have just gone down, the island increases its own population size in order to compensate the loss of the former. This is done by calculating the fraction of the population size of the deactivated island corresponding to the number of active neighbors it had (e.g., if an island with population size μ and ν active neighbors went down, each of the latter would attempt to grow their populations by μ/ν individuals). On the other hand, if all neighboring islands are active then they exchange individuals in order to balance their population sizes. See [46] for details. Note that this is a completely autonomous and decentralized policy and therefore each node cannot comprehend the global state of the network, for instance, a node does not account for the simultaneous failure of nodes that are themselves neighbors, hence the interest of studying the robustness of the EA in the correlated scenario.

Self-healing. Self-healing refers to the capability of the system to repair or correct externally infringed damage [17]. EAs are intrinsically resilient as mentioned before (see also [38] for a theoretical analysis) thanks to its population-based nature providing built-in redundancy. Even more so, some explicit self-healing methods have been incorporated to EAs –even if just in a rudimentary form– since the earlier developments of the paradigm (consider for example the use of repairing functions to restore feasibility of solutions after the application of reproductive operators [36]). In this case, we specifically focus on the re-sizing of islands as a result of the self-scaling process, and more precisely on the situation in which an island has to grow as a result of a neighboring island having a crash.

A simple solution is to have the population grow by introducing new random solutions (the random immigrant strategy [20]). While this would certainly promote diversity by introducing completely new genetic material, it could also drag backwards the search process by moving the population away from promising regions of the search space that might have been identified. As an alternative a self-sampling procedure could be used. This amounts to maintaining within each island a probabilistic model of its current population in order to sample it whenever the population needs to grow. This has the advantage of introducing diversity due to the stochastic sampling, keeping as well the momentum of the search since the newly created individuals are coherent with the current state of the population (unlike the case of using random individuals to this end). In this work we have considered the use of a tree-like bivariate probabilistic model such as that used in the COMIT estimation of distribution algorithm [5] – see also [47] for details.

3. Experimentation

In order to study the resilience of the island-based EAs described in previous section, these have been put to test in a broad variety of scenarios. Before reporting the results obtained, next subsection describes in more detail the experimental setting considered.

3.1. Experimental Setting

We consider $n_\iota = 64$ islands whose initial size is $\mu = 32$ individuals and a total number of evaluations *maxevals* $= 250\ 000$. We use crossover probability $p_X = 1.0$, mutation probability $p_M = 1/\ell$, where ℓ is the genotype length, and migration probability $p_{mig} = 1/(5\mu) = 1/160$. Regarding the network parameters, we use $m = 2$ in the Barabási-Albert model in order to define the topology of the underlying network; as for node deactivation/reactivation, we use the shape parameter $\eta = 1.5$ (larger than 1 and hence implying an increasing hazard rate with time), and scale parameters $\beta = -1/\log(p)$ for $p = 1-1/(kn_\iota)$, $k \in \{1, 2, 5, 10, 20\}$. To interpret these parameters, note that they would correspond to an average of one micro-failure event every k cycles if the failure rate was constant. This provides different scenarios ranging from low volatility ($k = 20$) to very high volatility ($k = 1$). Notice thus there is an inverse relationship between the value of this parameter and stability: large values of this parameter correspond to great stability (actually, full stability for $k = \infty$, a scenario that has been also included in the experimentation to gauge the results) and small values correspond to high volatility (hitting a maximum for $k = 1/n_\iota$). In order to have a more meaningful comparison between both scenarios (accommodating the fact that several micro-failure events are required in order to take down a node in the correlated case but only one is needed in the non-correlated case), in the non-correlated scenario we adjust k values as $k' = k\tilde{\theta}$, where $\tilde{\theta}$ is the average of all θ_i values in the correlated scenario (which in this case is also the average degree of the network). Note at any rate that the main focus of the experimentation is the relative behavior of the algorithms considered in either scenario rather than a comparison between scenarios in absolute terms.

As stated in Sect. 2.2, we consider four algorithmic variants: EA (a standard island-based EA with fixed island sizes and random reinitialization of islands upon reactivation)

Table 1. Results (averaged for 25 runs) of the different EAs on the three problems considered under the network model with non-correlated failures. The median ($\tilde{x}$), mean ($\bar{x}$) and standard error of the mean ($\sigma_{\bar{x}}$) are indicated.

		TRAP		H-IFF		MMDP	
strategy	k	$\tilde{x}$	$\bar{x} \pm \sigma_{\bar{x}}$	$\tilde{x}$	$\bar{x} \pm \sigma_{\bar{x}}$	$\tilde{x}$	$\bar{x} \pm \sigma_{\bar{x}}$
EA	∞	0.00	0.00 ± 0.00	0.00	5.33 ± 1.49	1.50	1.50 ± 0.17
	20	0.00	0.10 ± 0.07	0.00	3.78 ± 1.27	1.50	1.84 ± 0.20
	10	0.00	0.25 ± 0.10	11.11	9.44 ± 1.42	3.00	2.73 ± 0.26
	5	1.25	1.20 ± 0.23	16.67	14.47 ± 1.53	4.49	4.74 ± 0.31
	2	10.00	9.20 ± 0.61	32.64	31.97 ± 0.96	13.15	13.21 ± 0.34
	1	30.00	29.88 ± 0.80	53.65	53.35 ± 0.58	28.96	28.25 ± 0.57
EA*	20	0.00	0.05 ± 0.05	0.00	6.22 ± 1.37	0.00	0.30 ± 0.12
	10	0.00	0.00 ± 0.00	11.11	9.11 ± 1.66	0.00	0.06 ± 0.06
	5	0.00	0.10 ± 0.07	16.67	13.00 ± 1.66	0.00	0.30 ± 0.12
	2	0.00	0.35 ± 0.11	19.44	18.22 ± 1.39	0.00	0.48 ± 0.14
	1	0.00	0.90 ± 0.22	22.22	22.28 ± 0.87	0.00	0.96 ± 0.23
MMA	∞	0.00	0.00 ± 0.00	0.00	0.00 ± 0.00	0.00	0.00 ± 0.00
	20	0.00	0.00 ± 0.00	0.00	0.00 ± 0.00	0.00	0.00 ± 0.00
	10	0.00	0.00 ± 0.00	0.00	0.00 ± 0.00	0.00	0.00 ± 0.00
	5	0.00	0.00 ± 0.00	0.00	0.00 ± 0.00	0.00	0.18 ± 0.10
	2	0.00	1.10 ± 0.30	0.00	0.00 ± 0.00	5.99	6.89 ± 0.54
	1	22.50	22.20 ± 0.90	22.05	19.72 ± 3.40	22.13	22.24 ± 0.61
MMA*	20	0.00	0.00 ± 0.00	0.00	0.00 ± 0.00	0.00	0.00 ± 0.00
	10	0.00	0.00 ± 0.00	0.00	0.00 ± 0.00	0.00	0.00 ± 0.00
	5	0.00	0.00 ± 0.00	0.00	0.00 ± 0.00	0.00	0.00 ± 0.00
	2	0.00	0.00 ± 0.00	0.00	0.00 ± 0.00	0.00	0.00 ± 0.00
	1	0.00	0.00 ± 0.00	0.00	0.00 ± 0.00	0.00	0.00 ± 0.00

EA* (the island-based EA endowed with self-healing and self-scaling), MMA (an island-based EA with self-generation) and MMA* (the island-based EA endowed with all three self-⋆ properties: self-generation, self-scaling and self-sampling). With regard to memes, as explained in Sect. 2.2, they are represented by rewriting rules. Their length varies between $l_{\min} = 3$ and $l_{\max} = 9$ and they have a mutation probability $p_r = 1/l_{\max}$, and are applied with parameter $w = 1$. The experimental benchmark comprises three test functions, namely Deb's trap function [11] (TRAP, concatenating 32 four-bit traps), Watson et al.'s Hierarchical-if-and-only-if function [58] (HIFF, using 128 bits) and Goldberg et al.'s Massively Multimodal Deceptive Problem [18] (MMDP, using 24 six-bit blocks). We perform 25 simulations for each algorithm, problem, volatility scenario and failure model.

3.2. Experimental Results

Fig. 3 shows a summary of the results (detailed numerical data for each problem, algorithm, and network model are provided in Tables 1–2). Let us firstly focus on variants without self-generation, namely EA and EA* (Fig. 3a–3b). As expected, the performance of the algorithm degrades as node volatility increases (that is, as we move to the right along the X axis): the increasing perturbation and loss of information caused by the disappearance of islands impairs the performance of the EAs. It is nevertheless interesting to

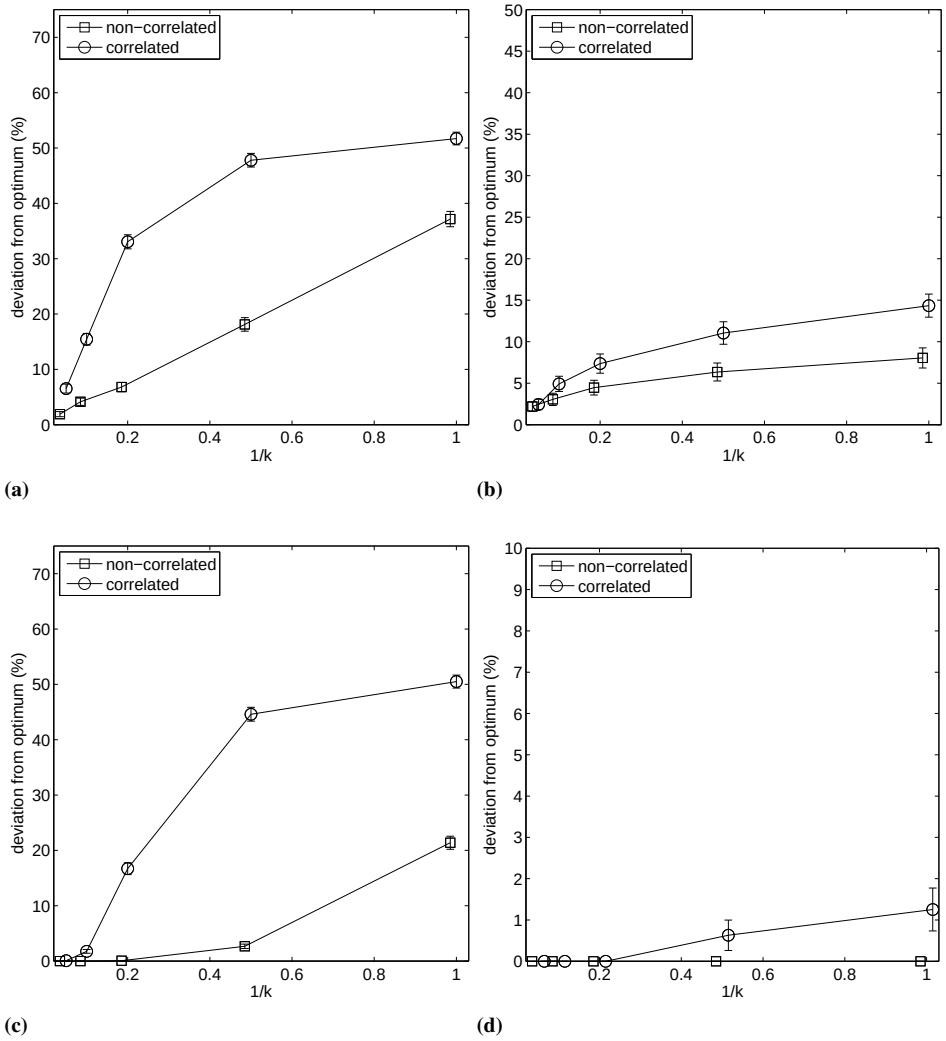

Fig. 3. Average deviation from the optimal solution across all problems for each algorithmic variant and network failure model. (a) EA (b) EA* (c) MMA (d) MMA*. Notice the different range of the Y axis in each case.

note how the degradation profile of EA is more marked in the correlated scenario. More frequent and simultaneous node failures have a clear toll on performance. If we now consider the case of EA*, two major observations stand out: on one hand, the performance of EA* is notably better than that of EA for the same volatility rate. This had been already observed in the non-correlated case (albeit for multimemetic algorithms – this behavior is hence extended for plain EAs as well) and is now confirmed in the correlated scenario, indicating than the self-$\star$ properties seem to keep providing robustness to the algorithm in

this case too. As a matter of fact –and this leads to the second observation– the degradation of performance in the correlated case is much less marked for EA* than it was for EA. More precisely, if we conduct a ranksum test on the results obtained by each algorithm on each problem and network scenario we observe that the performance of EA significantly (at level $\alpha = 0.01$) degrades in the correlated scenario with respect to the non-correlated one for all churn rates, whereas EA* is only significantly degraded for moderate and high churn rates ($k \leqslant 5$ for TRAP and HIFF and $k \leqslant 2$ for MMDP). This is not to say that EA* is not adversely affected by the new scenario (in the non-correlated case the performance of EA* was only significantly degraded with respect to the stable $k = \infty$ case for $k \leqslant 5$ in HIFF and $k \leqslant 2$ in TRAP, whereas in the correlated scenario there are statistically significant differences for $k \leqslant 2$ in MMDP, $k \leqslant 5$ in TRAP and $k \leqslant 10$ in HIFF) but this degradation is mostly in the most volatile cases (unlike EA, whose performance is degraded with respect to $k = \infty$ in the correlated case for all churn rates in all three problems) and not so large in magnitude as for EA. A result consistent with this can also be seen in Fig. 4, in which the genetic diversity of the population (measured using Shannon's entropy) is depicted for each algorithm and scenario (the data corresponds to the TRAP function, but the behavior is qualitatively similar in the remaining problems). Notice how EA faces increasingly large difficulties to converge as the volatility goes up, and how these difficulties are noticeable even for low-volatility settings in the correlated scenario. EA* can however maintain a better focus on the search, and seems mostly affected in the most volatile settings of the correlated scenario. The use of self-scaling seems crucial for this, since it damps perturbations in the overall size of the population and contributes to exchange genetic information among islands (it can be actually regarded as a self-adaptive, instability-driven migration process). This information exchange provides the convergence boost required to overcome to a great extent the perturbation caused by island losses (this perturbation is further alleviated by the use of self-healing, which helps avoiding having to create solutions from scratch, something that would contribute diversity but hamper convergence). This said, it is clear that a decentralized strategy such as the one considered, whereby each island takes decisions based on the interaction with its neighbors, can be more sensitive to simultaneous failures of neighboring nodes which is more frequent in the correlated scenario considered. Hence, while the EA* can be better equipped than plain EAs to withstand unstable scenarios, it is not immune to churn.

As for the EAs endowed with self-generation, we can draw similar observations. If we look firstly at MMA (Fig. 3c) we obtain the expected result, as in the case of the EA, i.e., the performance of the algorithm degrades as the volatility of the node increases, but this degradation is negligible in the non-correlated scenario except for the most volatile setting (k=1), unlike the much more marked degradation that takes place in the correlated scenario. In this sense, it must be taken into account the non-linear relationship between the value of k and instability, which makes the growth in churn when going from $k = 2$ to $k = 1$ much larger than, say, going from $k = 20$ to $k = 10$. Indeed, a certain increase in volatility will induce a steady degradation of performance, until the latter saturates (i.e., when the search degenerates completely and the algorithm is not capable of converging, there is not much further room for degradation – this can be seen for example in Fig. 4, in which for some low values of k the EA is not capable of converging). The onset of this degradation is also dictated by the inherent resilience of the algorithm. In this sense, the MMA is somewhat similar in behavior to the EA, although the fact that the

Table 2. Results (averaged for 25 runs) of the different EAs on the three problems considered under the network model with correlated failures. The median ($\tilde{x}$), mean ($\bar{x}$) and standard error of the mean ($\sigma_{\bar{x}}$) are indicated.

		TRAP		H-IFF		MMDP	
strategy	k	$\tilde{x}$	$\bar{x} \pm \sigma_{\bar{x}}$	$\tilde{x}$	$\bar{x} \pm \sigma_{\bar{x}}$	$\tilde{x}$	$\bar{x} \pm \sigma_{\bar{x}}$
EA	∞	0.00	0.00 ± 0.00	0.00	5.33 ± 1.49	1.50	1.50 ± 0.17
	20	1.25	1.47 ± 0.21	16.67	13.18 ± 1.68	4.49	4.93 ± 0.41
	10	6.87	7.15 ± 0.41	25.87	26.97 ± 1.05	11.98	12.19 ± 0.43
	5	26.25	26.15 ± 0.85	47.40	47.67 ± 0.63	25.97	25.35 ± 0.48
	2	46.88	46.33 ± 0.58	61.46	61.19 ± 0.29	35.46	35.87 ± 0.40
	1	51.25	51.27 ± 0.50	63.72	63.80 ± 0.21	39.95	40.08 ± 0.36
EA*	20	0.00	0.05 ± 0.05	11.11	7.22 ± 1.49	0.00	0.06 ± 0.06
	10	0.00	0.10 ± 0.07	16.67	14.06 ± 1.57	0.00	0.60 ± 0.23
	5	0.00	0.70 ± 0.18	19.44	20.61 ± 1.19	0.00	0.78 ± 0.21
	2	2.50	2.10 ± 0.21	27.78	27.08 ± 0.83	4.49	3.95 ± 0.37
	1	5.00	5.68 ± 0.41	31.94	30.53 ± 1.04	7.49	6.83 ± 0.42
MMA	∞	0.00	0.00 ± 0.00	0.00	0.00 ± 0.00	0.00	0.00 ± 0.00
	20	0.00	0.00 ± 0.00	0.00	0.00 ± 0.00	0.00	0.18 ± 0.10
	10	0.00	0.55 ± 0.26	0.00	0.00 ± 0.00	4.49	4.68 ± 0.65
	5	17.50	17.82 ± 1.18	0.00	11.88 ± 2.61	21.47	20.39 ± 0.91
	2	41.25	41.45 ± 0.58	59.55	58.26 ± 1.02	33.97	34.10 ± 0.54
	1	50.63	50.45 ± 0.43	62.50	62.65 ± 0.28	38.46	38.44 ± 0.52
MMA*	20	0.00	0.00 ± 0.00	0.00	0.00 ± 0.00	0.00	0.00 ± 0.00
	10	0.00	0.00 ± 0.00	0.00	0.00 ± 0.00	0.00	0.00 ± 0.00
	5	0.00	0.00 ± 0.00	0.00	0.00 ± 0.00	0.00	0.00 ± 0.00
	2	0.00	0.00 ± 0.00	0.00	1.89 ± 1.07	0.00	0.00 ± 0.00
	1	0.00	0.00 ± 0.00	0.00	2.89 ± 1.36	0.00	0.87 ± 0.69

former includes local search contributes to improve slightly the results and counteracts the effects of degradation for large values of k, but cannot prevent it for small k. At any rate, the greater hardness of the correlated scenario makes degradation being qualitatively noticeable for larger values of k.

On the other hand, if we now consider the case of MMA*, three fundamental issues can be observed: firstly, its performance is considerably better than that of MMA in these latter settings; secondly, the degradation of performance is much more gentle for any value of k; thirdly, the performance is considerably better and more stable in the case of the MMA than in the EA, for all values of k and for both the correlated and the non-correlated cases, being in the latter case very close to 0, i.e., the MMA with the self-$\star$ properties is seemingly able to overcome the inconveniences of instability of the environment and alleviate the degradation of the algorithm. The increased resilience provided by self-scaling and self-healing –for the same reasons pointed out in the case of the EA– is in this case amplified by the better search capabilities of the MMA with respect to plain EAs thanks to their being endowed with local search. If a ranksum test is performed on the results obtained for the MMA by each algorithm on each problem and network scenario we can observe that the performance of MMA significantly (in the $\alpha = 0.01$ level) degrades in the correlated scenario with respect to the non-correlated one for moderate and

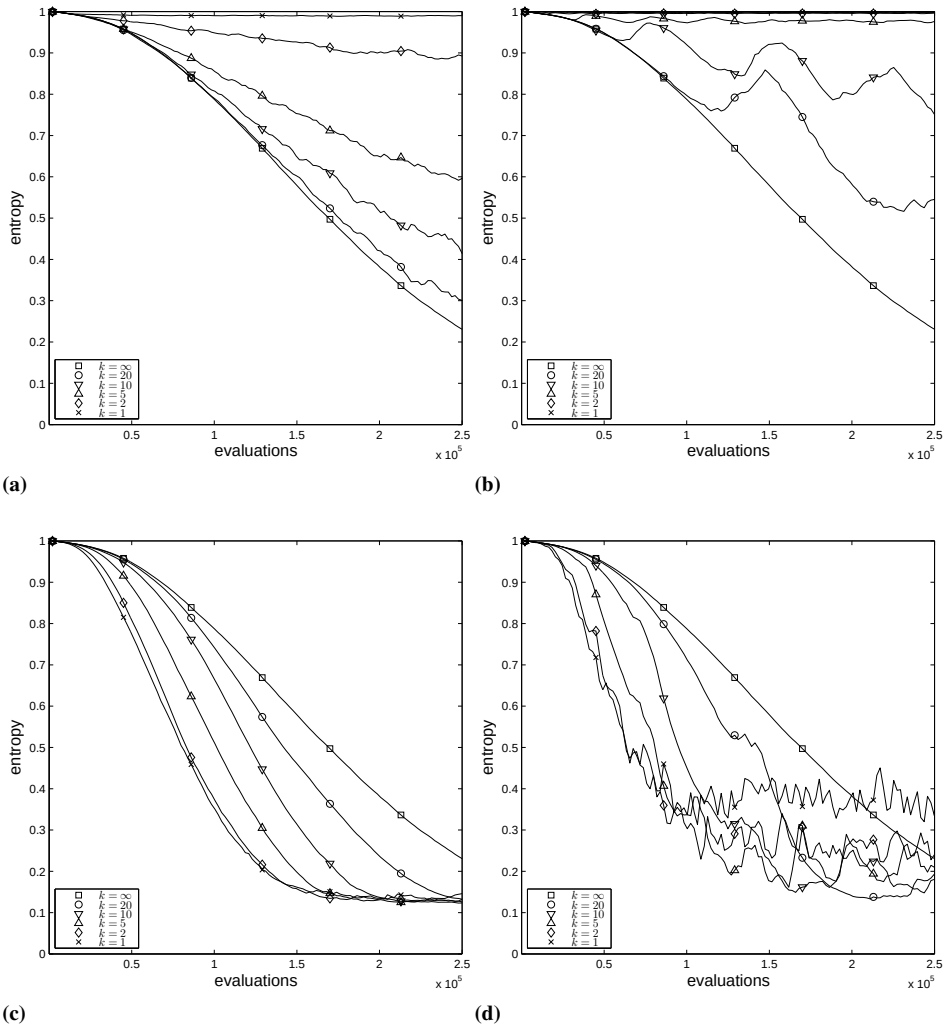

Fig. 4. Genetic diversity for the TRAP function. The top row corresponds to EA and the bottom row to EA*; the left column corresponds to non-correlated failures, and the right row to correlated failures.

high churn rates ($k \leqslant 5$ for TRAP and HIFF and $k \leqslant 2$ for MMDP), whereas MMA* is not significantly degraded for any churn rate. This is consistent with the meme dynamics shown in Fig. 5 for correlated scenarios. For low volatility, the length of memes in the MMA (Fig. 5a) seems to be converging, and the improvement rate (percentage of meme applications that result in an improvement) takes a U shape (Fig. 5b), a typical pattern observed in standard (panmictic) versions of the algorithm [44]. However, for high volatility memes are unable to converge (meme lengths fluctuate around the initial mean) and the improvement rate is rather flat (pointing to the population maintaining in a similar state

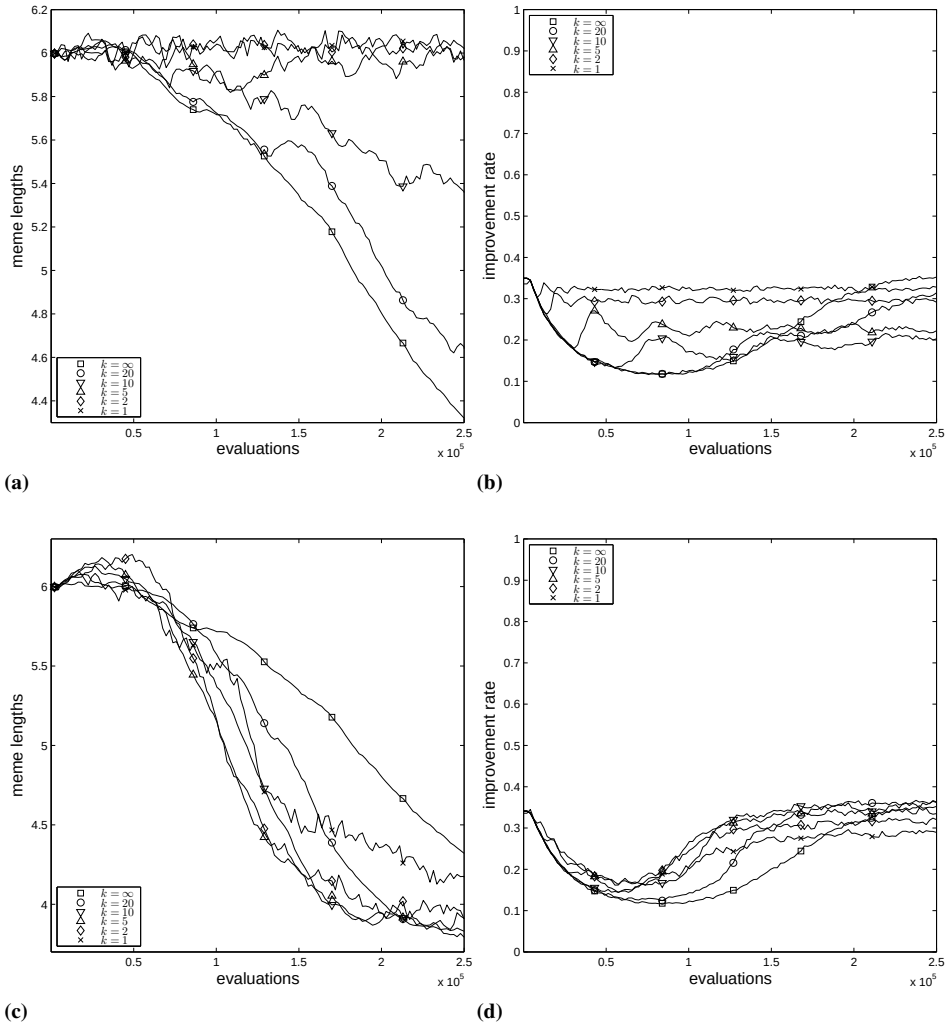

Fig. 5. Meme dynamics in the correlated scenario. The top row corresponds to MMA and the bottom row to MMA*. The left column shows meme lengths and the right column the improvement rate.

during the whole run). However, MMA* maintains a much more consistent profile (Fig. 5c–5d) across all configurations considered, indicating that the algorithm could keep the momentum of the search in spite of the larger instability.

4. Conclusions and Future Work

Unstable computational environments put to test the resilience of algorithms that run on them, even more so when volatility is high and follow complex patterns. In this work we

have analyzed how these unstable scenarios can affect the performance of several variants of island-based EAs, and observed a marked degradation of the results in absence of appropriate policies to deal with this harder setting. Endowing the EA with self-⋆ properties can however increase its resilience and make it able to withstand from low up to moderately high volatility. Stacking together self-generation, self-scaling, and self-healing redound in increased resilience and a much more stable behavior, leading in some cases to a negligible loss of performance in the scenarios considered.

There are several lines of research for future work. First of all, the network model could be expanded, either by trying new topologies or by considering other different models of correlated faults. In line with the latter, the use of dynamic thresholds could be considered (work is in progress in this area [48]) or even the utilization of completely different correlation models, e.g., [8,59]. A complementary issue of interest is the potential heterogeneity of the network, allowing a one-to-many mapping between some computing nodes and the islands of the algorithm. In the longer term, a related problem is the optimization of the network itself to cope with this kind of failures. Some recent work has addressed this issue [56], paving the way for other developments in this direction. Finally, other non-evolutionary algorithms could be deployed in this scenario. These works are currently underway.

Acknowledgments. This work is supported by the Spanish Ministerio de Economía and European FEDER under Projects EphemeCH (TIN2014-56494-C4-1-P –`http://ephemech.wordpress.com`) and DeepBIO (TIN2017-85727-C4-1-P) and by Universidad de Málaga, Campus de Excelencia Internacional Andalucía Tech.

References

1. Alba, E.: Parallel evolutionary algorithms can achieve super-linear performance. Information Processing Letters 82(1), 7–13 (2002)
2. Alba, E.: Parallel Metaheuristics: A New Class of Algorithms. Wiley-Interscience, Hoboken, New Jersey, USA (2005)
3. Albert, R., Barabási, A.L.: Statistical mechanics of complex networks. Review of Modern Physics 74(1), 47–97 (2002)
4. Babaoğlu, Ö., Jelasity, M., Montresor, A., Fetzer, C., Leonardi, S., van Moorsel, A., van Steen, M. (eds.): Self-star Properties in Complex Information Systems, Lecture Notes in Computer Science, vol. 3460. Springer-Verlag, Berlin Heidelberg (2005)
5. Baluja, S., Davies, S.: Using optimal dependency-trees for combinatorial optimization: Learning the structure of the search space. In: 14th International Conference on Machine Learning. pp. 30–38. Morgan Kaufmann Publishers (1997)
6. Barabási, A.L., Albert, R.: Emergence of scaling in random networks. Science 286(5439), 509–512 (1999)
7. Berns, A., Ghosh, S.: Dissecting self-⋆ properties. In: Third IEEE International Conference on Self-Adaptive and Self-Organizing Systems - SASO 2009. pp. 10–19. IEEE Press, San Francisco, CA (2009)
8. Böttcher, L., Luković, M., Nagler, J., Havlin, S., Herrmann, H.J.: Failure and recovery in dynamical networks. Scientific Reports 7, 41729 EP – (2 2017)
9. Cole, N., Desell, T., González, D.L., Fernández de Vega, F., Magdon-Ismail, M., Newberg, H., Szymanski, B., Varela, C.: Evolutionary algorithms on volunteer computing platforms: The milkyway@home project. In: Fernández de Vega, F., Cantú-Paz, E. (eds.) Parallel and Distributed Computational Intelligence, Studies in Computational Intelligence, vol. 269, pp. 63–90. Springer-Verlag, Berlin Heidelberg (2010)

10. Cotta, C., Fernández-Leiva, A.J., Fernández de Vega, F., Chávez, F., Merelo, J.J., Castillo, P.A., Bello, G., Camacho, D.: Ephemeral computing and bioinspired optimization - challenges and opportunities. In: 7th International Joint Conference on Evolutionary Computation Theory and Applications. pp. 319–324. SCITEPRESS, Lisboa, Portugal (2015)
11. Deb, K., Goldberg, D.: Analyzing deception in trap functions. In: Whitley, L. (ed.) Second Workshop on Foundations of Genetic Algorithms. pp. 93–108. Morgan Kaufmann Publishers, Vail, Colorado, USA (1993)
12. Dorogovtsev, S.N., Goltsev, A.V., Mendes, J.F.F.: Critical phenomena in complex networks. Rev. Mod. Phys. 80, 1275–1335 (Oct 2008)
13. Eiben, A.E.: Evolutionary computing and autonomic computing: Shared problems, shared solutions? In: Babaoğlu et al. [4], pp. 36–48
14. Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computation. Natural Computing Series, Springer-Verlag, Berlin Heidelberg (2003)
15. Gagné, C., Parizeau, M., Dubreuil, M.: Distributed beagle: An environment for parallel and distributed evolutionary computations. In: 17th Annual International Symposium on High Performance Computing Systems and Applications - HPCS 2003. pp. 201–208. Sherbrooke, Quebec, Canada (2013)
16. García Arenas, M., Collet, P., Eiben, A.E., Jelasity, M., Merelo Guervós, J.J., Paechter, B., Preuß, M., Schoenauer, M.: A framework for distributed evolutionary algorithms. In: Merelo Guervós, J.J., et al. (eds.) Parallel Problem Solving from Nature - PPSN VII. Lecture Notes in Computer Science, vol. 2439, pp. 665–675. Springer Verlag, Berlin Heidelberg (2002)
17. Ghosh, D., Sharman, R., Rao, H., Upadhyaya, S.: Self-healing systems - survey and synthesis. Decision Support Systems 42(4), 2164–2185 (2007)
18. Goldberg, D., Deb, K., Horn, J.: Massive multimodality, deception and genetic algorithms. In: Männer and Manderick [32], pp. 37–48
19. Gorges-Schleuter, M.: ASPARAGOS: an asynchronous parallel genetic optimization strategy. In: Schaffer, J. (ed.) Third International Conference on Genetic Algorithms. pp. 422–427. Morgan Kaufmann Publishers, San Francisco, CA (1989)
20. Grefenstette, J.: Genetic algorithms for changing environments. In: Männer and Manderick [32], pp. 137–144
21. Grefenstette, J.J.: Parallel adaptive algorithms for function optimization. Tech. Rep. CS-81-19, Vanderbilt University, Nashville, TN (1981)
22. Grosso, P.: Computer simulation of genetic adaptation: Parallel subcomponent interaction in a multilocus model. Ph.D. thesis, University of Michigan, Ann Arbor (1985)
23. Hidalgo, J., Lanchares, J., Fernández de Vega, F., Lombraña, D.: Is the island model fault tolerant? In: Thierens et al. [57], pp. 2737 – 2744
24. Hinterding, R., Michalewicz, Z., Eiben, A.: Adaptation in evolutionary computation: A survey. In: Fourth IEEE Conference on Evolutionary Computation. pp. 65–69. IEEE Press, Piscataway, New Jersey (1997)
25. Kong, Z., Yeh, E.M.: Correlated and cascading node failures in random geometric networks: A percolation view. In: 2012 Fourth International Conference on Ubiquitous and Future Networks (ICUFN). pp. 520–525. IEEE, Phuket, Thailand (July 2012)
26. Krasnogor, N., Blackburne, B., Burke, E., Hirst, J.: Multimeme algorithms for protein structure prediction. In: Merelo Guervós, J.J., et al. (eds.) Parallel Problem Solving From Nature - PPSN VII. Lecture Notes in Computer Science, vol. 2439, pp. 769–778. Springer Verlag, Berlin Heidelberg (2002)
27. Krasnogor, N., Gustafson, S.: A study on the use of "self-generation" in memetic algorithms. Natural Computing 3(1), 53–76 (2004)
28. Laredo, J., Castillo, P., Mora, A., Merelo, J.J., Fernandes, C.: Resilience to churn of a peer-to-peer evolutionary algorithm. International Journal of High Performance Systems Architecture 1(4), 260–268 (2008)

29. Lombraña González, D., Jiménez Laredo, J., Fernández de Vega, F., Merelo Guervós, J.J.: Characterizing fault-tolerance of genetic algorithms in desktop grid systems. In: Cowling, P., Merz, P. (eds.) Evolutionary Computation in Combinatorial Optimization. Lecture Notes in Computer Science, vol. 6022, pp. 131–142. Springer-Verlag, Berlin Heidelberg (2010)
30. Lombraña González, D., Jiménez Laredo, J., Fernández de Vega, F., Merelo Guervós, J.J.: Characterizing fault-tolerance in evolutionary algorithms. In: Fernández de Vega, F., et al. (eds.) Parallel Architectures and Bioinspired Algorithms, Studies in Computational Intelligence, vol. 415, pp. 77–99. Springer-Verlag, Berlin Heidelberg (2012)
31. Lombraña González, D., Fernández de Vega, F., Casanova., H.: Characterizing fault tolerance in genetic programming. Future Generation Computer Systems 26(6), 847–856 (2010)
32. Männer, R., Manderick, B. (eds.): Parallel Problem Solving from Nature - PPSN II. Elsevier Science Inc., New York, NY, USA (1992)
33. Matei, R., Iamnitchi, A., Foster, P.: Mapping the Gnutella network. IEEE Internet Computing 6(1), 50–57 (Jan 2002)
34. Melab, N., Cahon, S., Talbi, E.: Grid computing for parallel bioinspired algorithms. Journal of Parallel and Distributed Computing 66(8), 1052–1061 (2006)
35. Meri, K., Arenas, M., Mora, A., Merelo, J.J., Castillo, P., García-Sánchez, P., Laredo, J.: Cloud-based evolutionary algorithms: An algorithmic study. Natural Computing 12(2), 135–147 (2013)
36. Michalewicz, Z.: Repair algorithms. In: Bäck, T., et al. (eds.) Handbook of Evolutionary Computation, pp. C5.4:1–5. Institute of Physics Publishing and Oxford University Press, Bristol, New York (1997)
37. Milojičić, D., Kalogeraki, V., Lukose, R., Nagaraja, K., Pruyne, J., Richard, B., Rollins, S., Xu, Z.: Peer-to-peer computing. Tech. Rep. HPL-2002-57, Hewlett-Packard Labs (2002)
38. Muszyński, J., Varrette, S., Bouvry, P., Seredyński, F., Khan, S.U.: Convergence analysis of evolutionary algorithms in the presence of crash-faults and cheaters. Computers & Mathematics with Applications 64(12), 3805 – 3819 (2012)
39. Neri, F., Cotta, C.: Memetic algorithms and memetic computing optimization: A literature review. Swarm and Evolutionary Computation 2, 1–14 (2012)
40. Neri, F., Cotta, C., Moscato, P. (eds.): Handbook of Memetic Algorithms, Studies in Computational Intelligence, vol. 379. Springer-Verlag, Berlin Heidelberg (2012)
41. Nogueras, R., Cotta, C.: An analysis of migration strategies in island-based multimemetic algorithms. In: Bartz-Beielstein, T., et al. (eds.) Parallel Problem Solving from Nature - PPSN XIII. Lecture Notes in Computer Science, vol. 8672, pp. 731–740. Springer Verlag, Berlin Heidelberg (2014)
42. Nogueras, R., Cotta, C.: Self-sampling strategies for multimemetic algorithms in unstable computational environments. In: Ferrández Vicente, J., et al. (eds.) Bioinspired Computation in Artificial Systems. Lecture Notes in Computer Science, vol. 9108, pp. 69–78. Springer Verlag, Berlin Heidelberg (2015)
43. Nogueras, R., Cotta, C.: Sensitivity analysis of checkpointing strategies for multimemetic algorithms on dynamic complex networks. In: 10th International Conference on Large Scale Scientific Computations. Lecture Notes in Computer Science, vol. 9374, pp. 233–240. Springer Verlag, Berlin Heidelberg (2015)
44. Nogueras, R., Cotta, C.: A study on meme propagation in multimemetic algorithms. International Journal of Applied Mathematics and Computer Science 25(3), 499–512 (2015)
45. Nogueras, R., Cotta, C.: Studying fault-tolerance in island-based evolutionary and multimemetic algorithms. Journal of Grid Computing 13(3), 351–374 (2015)
46. Nogueras, R., Cotta, C.: Studying self-balancing strategies in island-based multimemetic algorithms. Journal of Computational and Applied Mathematics 293, 180–191 (2016)
47. Nogueras, R., Cotta, C.: Self-healing strategies for memetic algorithms in unstable and ephemeral computational environments. Natural Computing 16(2), 189–200 (2017)

48. Nogueras, R., Cotta, C.: Evaluating island-based EAs on unstable networks with complex failure patterns. In: Proceedings of GECCO' 17 Companion (late breaking abstract). Berlin, Germany (2017), 2 pages
49. Nogueras, R., Cotta, C.: A performance analysis of self-⋆ evolutionary algorithms on networks with correlated failures. In: Ivanović, M., Bădică, C., Dix, J., Jovanović, Z., Malgeri, M., Savić, M. (eds.) Intelligent Distributed Computing XI – IDC 2017, Studies in Computational Intelligence, vol. 737, pp. 3–13. Springer, Cham (2018)
50. Ong, Y., Lim, M., Chen, X.: Memetic computation –past, present and future. IEEE Computational Intelligence Magazine 5(2), 24–31 (2010)
51. Sarmenta, L.: Bayanihan: Web-based volunteer computing using java. In: Masunaga, Y., Katayama, T., Tsukamoto, M. (eds.) Worldwide Computing and Its Applications - WWCA 1998. Lecture Notes in Computer Science, vol. 1368, pp. 444–461. Springer-Verlag, Berlin Heidelberg (1998)
52. Smith, J.E.: Self-adaptation in evolutionary algorithms for combinatorial optimisation. In: Cotta, C., Sevaux, M., Sörensen, K. (eds.) Adaptive and Multilevel Metaheuristics, Studies in Computational Intelligence, vol. 136, pp. 31–57. Springer-Verlag, Berlin Heidelberg (2008)
53. Smith, J.E.: Self-adaptative and coevolving memetic algorithms. In: Neri et al. [40], pp. 167–188
54. Stutzbach, D., Rejaie, R.: Understanding churn in peer-to-peer networks. In: 6th ACM SIGCOMM Conference on Internet Measurement - IMC 2006. pp. 189–202. ACM Press, New York, NY, USA (2006)
55. Tanese, R.: Distributed genetic algorithms. In: 3rd International Conference on Genetic Algorithms. pp. 434–439. Morgan Kaufmann Publishers, San Francisco, CA, USA (1989)
56. Tang, X., Liu, J., Hao, X.: Mitigate cascading failures on networks using a memetic algorithm. Scientific Reports 6, 38713 EP – (12 2016)
57. Thierens, D., et al. (eds.): Genetic and Evolutionary Computation - GECCO 2007. ACM Press, New York, NY, USA (2007)
58. Watson, R., Hornby, G., Pollack, J.: Modeling building-block interdependency. In: Eiben, A., et al. (eds.) Parallel Problem Solving from Nature - PPSN V. Lecture Notes in Computer Science, vol. 1498, pp. 97–106. Springer Verlag, Berlin Heidelberg (1998)
59. Watts, D.J.: A simple model of global cascades on random networks. Proceedings of the National Academy of Sciences 99(9), 5766–5771 (2002)
60. Wickramasinghe, W., Steen, M.V., Eiben, A.E.: Peer-to-peer evolutionary algorithms with adaptive autonomous selection. In: Thierens et al. [57], pp. 1460–1467

Rafael Nogueras obtained his MSc and PhD in Computer Science from the University of Málaga in 1998 and 2015 respectively and his MSc in Electronic Engineering from the University of Granada in 2012, all in Spain. He worked in industry for more than ten years before returning to Public Administration in 2012. He has interests in the field of evolutionary computation, primarily in memetic algorithms in distributed systems and ephemeral computing systems with self-* properties.

Carlos Cotta obtained his MSc and PhD in Computer Science from the University of Málaga, Spain, in 1994 and 1998, respectively. He holds a full professorship at this university since 2017. His main research areas involve metaheuristic optimization, in particular hybrid and memetic approaches with the focus on both algorithmic and applied aspects (particularly combinatorial optimization) and complex systems.

Received: January 15, 2018; Accepted: September 4, 2018.