Analyzing Self-⋆ Island-based Memetic Algorithms in Heterogeneous Unstable Environments

International Journal of High Performance Computing Applications
XX(X):1–11

DOI: 10.1177/ToBeAssigned
www.sagepub.com/
SAGE

Rafael Nogueras[1] and Carlos Cotta[1]

Abstract

Computational environments emerging from the pervasiveness of networked devices offer a plethora of opportunities and challenges. The latter arise from their dynamic, inherently volatile nature that puts to the test the resilience of algorithms running on them. We here consider the deployment of population-based optimization algorithms on such environments, using the island model of memetic algorithms (MAs) for this purpose. These MAs are endowed with self-⋆ properties that provide them the ability to work autonomously in order to optimize their performance and to react to the instability of computational resources. The main focus of this work is analyzing the performance of these MAs when the underlying computational substrate is not only volatile but also heterogeneous in terms of the computational power of each of its constituent nodes. To this end, we use a simulated environment that allows experimenting with different volatility rates and heterogeneity scenarios (that is, different distributions of computational power among computing nodes), and we study different strategies for distributing the search among nodes. We observe that the addition of self-scaling and self-healing properties makes the MA very robust to both system instability and computational heterogeneity. Additionally, a strategy based on distributing single islands on each computational node is shown to perform globally better than placing many such islands on each of them (either proportionally to their computing power or subject to an intermediate compromise).

Keywords

Memetic Algorithm, Island Model, Self-⋆ Properties, Heterogeneous Environment, Unstable Environments

Introduction

The use of parallel computing has become the weapon-of-choice for the practical resolution of complex, computationally-intensive tasks. Traditionally, it has been common to approach this resolution by exploiting dedicated computational resources, whether it be a supercomputer or a local network of computational resources. These have the advantage of providing a rather stable and controlled environment for the deployment of parallel algorithms. Recent years have witnessed the emergence of other kind of environments of a much more dynamic and unsteady nature though. Consider for example the case of peer-to-peer (P2P) networks (Milojičić et al. 2002) and volunteer computing (VC) networks (Sarmenta 1998). These are two particularly important scenarios if we consider how prevalent computational devices which are permanently networked (e.g., smartphones, wearables, and any other kind of handheld devices) are nowadays, and the extent to which their computing power (individually limited but collectively formidable) is often under-exploited (Cotta et al. 2015). A common feature of this kind of environments is their volatile structure: they are composed of unstable nodes whose availability fluctuates in response to uncontrollable external factors. Thus, computing nodes can dynamically enter and abandon the system giving rise to an ever-changing computational landscape. The term *churn* has been precisely coined to denote the collective effect that this constant inflow/outflow of computational units has on the system at large.

Taking advantage of these environments can constitute a practical solution for solving many complex computational tasks, e.g., (Korpela et al. 2001; Larson et al. 2002). We are here specifically concerned about the deployment of evolutionary algorithms (EAs) (Eiben and Smith 2003) on such environments. EAs –and population-based optimization algorithms in general– are very well suited to parallel environments thanks to their flexibility and decentralized nature, a fact that has been put into work since the early times of the paradigm (Grefenstette 1981; Grosso 1985; Gorges-Schleuter 1989). Indeed the performance of population-based algorithms does improve when run in parallel (Alba and Tomassini 2002; Sudholt 2014). Having them work effectively on computational environments with the features mentioned above is not exempt of difficulties though, and puts to the test one of the salient features of these techniques, namely resilience. In the case of EAs this resilience is an inherent characteristic resulting from the use of a population of solutions (Laredo et al. 2008; Lewis et al. 2009). As a matter of fact, it has

[1] Dept. Lenguajes y Ciencias de la Computación, Universidad de Málaga, ETSI Informática, Spain

Corresponding author:
Dept. Lenguajes y Ciencias de la Computación, Universidad de Málaga, ETSI Informática, Campus de Teatinos, 29071 Málaga, Spain
Tel.: +34 952 137158
Fax: +34 952 131397
Email: ccottap@lcc.uma.es

been shown that they are resilient enough to withstand a reduction in computing resources of up to one order of magnitude (Laredo et al. 2014) in the context of master-slave models – see also (Lombraña González et al. 2010, 2012). The island-model of EAs (Tanese 1989) introduces a more coarse level of computational granularity whereby the continuous execution of evolutionary cycles on separate subpopulations is distributed (as opposed to the evaluation of single individuals as in typical master-slave models). Not surprisingly, this makes the algorithm more sensitive to environmental instability since the failure of whole subpopulations has a larger effect on the progress of the search and can lead to the loss of the current incumbent solution (Hidalgo et al. 2007). Tackling this issue put the focus on another of the distinctive features of EAs, namely their capacity for adaptiveness and self-control (Hinterding et al. 1997; Eiben 2005). This feature can be exploited in order to make the optimization algorithm cognizant of the volatile environment, having it adapting its functioning in response to computational fluctuations.

Much work has been done in the area of self-adaptation in evolutionary algorithms, e.g., (Caraffini et al. 2014; Eiben et al. 2006; Smith 2007, 2008, 2012; Zhang et al. 2014) in general, and in connection with unstable environments in particular. Such strategies are often captured under the umbrella term of self-$\star$ properties (Babaoğlu et al. 2005). In this work we build upon previous research on unstable environments (Nogueras and Cotta 2015a,b, 2016a,c) in order to tackle a new angle on them, namely their potential heterogeneity (Lastovetsky 2003; Anderson and Reed 2009) in terms of the computational power of individual nodes (which in a setting such as the one described before could range from tiny devices to desktop computers for example) and ascertain to which extent this can exert an influence in the performance of the algorithm. It must be note that heterogeneity is not an exclusive feature of the kind of volatile environments considered here. In fact, it emerged as an increasingly important issue of high-performance computing platforms in the 1990s in the context of heterogeneous networks of workstations (Boulet et al. 1999). In many computationally-intensive parallel applications, heterogeneity is dealt with by adequately balancing the load of the heterogeneous processors (Beltrán and Guzmán 2009) and minimizing the cost of moving data between the processors (Lastovetsky 2014). Several strategies can be defined for this purpose but these are mostly oriented to data parallel applications related to Big Data analytics. As such, they often rely on finding an appropriate distribution of data among processors –e.g., (Renard et al. 2006; Lastovetsky and Reddy 2007)– in order to minimize the overall time to process these data. The case of of EAs (and metaheuristic optimization techniques in general) is somewhat different though. They are anytime algorithms and –in the case of the island-model– the computation performed is influenced by the communication between processes. Thus, there is not a certain fixed computation to be performed in minimal time, but a variable computation whose outcome can have different quality (depending on the solutions found) that has to be maximized. To study this particular setting and determine the impact of environmental heterogeneity on population-based optimization algorithms, we analyze the performance of memetic algorithms (MAs) (Neri et al. 2012) when deployed on a simulated unstable computational environment. The MAs are endowed with different self-$\star$ properties stacked together in order to withstand this instability, and their performance is examined in different scenarios both in terms of the volatility of computing nodes, the distribution of computing power among these, and the strategy for allocating computation to them.

Materials and Methods

As stated in the previous section, we aim to analyze the performance of an island-based MA in an heterogeneous unstable environment. This island-based MA is composed of n_ι islands. Each of these islands is a process that runs a MA on an independent unstructured subpopulation. This involves the standard evolutionary cycle of selection, variation and replacement, augmented in this case with memetic local improvement. In addition to this, islands are interconnected among them according to a certain topology $\mathcal{N}$ as described later on. This topology is important for the purposes of intercommunication via the migration operation. This is performed in an asynchronous way by each island, i.e., at the beginning of each cycle the island checks if migrants received from any neighbors are stored in the input buffer. If so, these are inserted in the local subpopulation following a certain migrant replacement policy. Later, at the end of each cycle, each island decides stochastically whether to send individuals to neighboring islands. If done, migrants are selected using a given migrant selection policy. Following previous analysis of migration strategies in island-based MAs (Nogueras and Cotta 2014), we use random selection of migrants and deterministic replacement of the worst individuals in the receiving island.

This basic algorithm is endowed with several self-$\star$ properties as described in next section. The resulting algorithm is subsequently deployed on a simulated distributed system featuring heterogeneity and instability, as described later on.

Self-$\star$ Properties

Self-$\star$ properties (Babaoğlu et al. 2005) are those that enable a computational system to exert advanced control on its own functioning and/or structure. The rationale of self-$\star$ properties can be traced back to the notion of autonomic computing (Horn 2001), whereby the metaphor of the autonomic nervous system is exploited in order to capture essential system functions that are carried out without conscious control. Thus, it is sought to endow complex computational systems with the capacity of self-management (Sterritt and Bustard 2003; Huebscher and McCann 2008), hence relieving the human user/designer as much as possible from maintenance, repairing or optimization tasks. This has been a long-standing goal in the area of computational systems in general, and in metaheuristics in particular, in light of the growing complexity and sophistication of their design (Cotta et al. 2008). As a matter of fact, it is not uncommon to see metaheuristic techniques endowed with self-adaptive strategies for parameter control, e.g., (Schwefel 1992; Beyer 1995; Angeline 1995; Hinterding et al. 1997; Eiben et al. 2006). Self-$\star$ properties can nevertheless encompass other advanced capabilities beyond

self-parameterization such as, e.g., self-maintaining in proper state, self-healing externally infringed damage, or self-optimizing its behavior, just to cite a few. In the following we will describe the particular self-$\star$ properties with which the MA considered is endowed.

Self-Optimization. According to Berns and Ghosh (2009), a system is self-optimized if starting from an arbitrary initial configuration it is capable of improving a certain objetive function of its global state. This objective function often refers to some desired functionality or goal of the system considered, whose attainment or efficacy is sought to be improved. In the case of the MAs considered here, the intended goal is optimizing a certain target function, and therefore self-optimization refers to enhancing the capability of the system for performing this latter optimization, i.e., having it finding better solutions or doing it in less time. Much of the research in self-parameterization mentioned before goes precisely in this direction. In addition, it is also possible to improve the search efficiency by adjusting qualitatively the way the search is done (Moscato 1999), that is, by self-generating search strategies (Krasnogor and Smith 2000; Smith 2002; Krasnogor and Gustafson 2004; Ong and Keane 2004); this is related to the current notion of memetic computing (Ong et al. 2010) whereby memes (understood as representations of problem solving strategies) are explicitly represented and evolved (Neri and Cotta 2012).

We have specifically considered an MA inspired by the work of Smith (2008, 2012) wherein each individual in the population carries a meme that represents a rewriting rule $A \rightarrow C$; in this rule, both A and C are patterns of symbols taken from the same alphabet used to represent solutions, augmented with a wildcard symbol '#'. The effect of this meme is to look for some match of the antecedent A in the genotype being optimized (using the wildcard as "don't care") and substitute it by the consequent C (where the wildcard now means "don't change"); it thus follows that A and C have the same length. For example, let a genotype be 11101101, and let a meme be $1\#1 \rightarrow 0\#1$. The meme could be applied on the first position yielding 01101101, on the third position yielding 11001101, and on the sixth position yielding 11101001. The order in which these positions are examined is randomized to avoid any positional bias, and a maximal number of rewritings w is used to keep under control the total cost of the process. The best neighbor generated (if better that the current solution) is kept.

As they are attached to solutions, these memes are subject to mutation, both in the patterns of the rule and in their length (Smith 2008), and are transferred from parent to offspring via local selection (offspring inherit the meme of the best parent). As a result of this evolutionary process, self-generation is attained.

Self-Scaling. This property involves the ability of the system to react efficiently to changes in its scale parameters. Such scale parameters may refer to the size of the task being tackled, to the amount of computational resources available, or to any other circumstance of the computation, e.g., (Fernández de Vega et al. 2004; Zhao and Schulzrinne 2004). The goal of a self-scaling system is to adjust its behavior (by tuning some internal parameters or even by re-configuring itself) so that the computation can be completed in an effective way. Focusing on MAs deployed in unstable environments, they must take into account the volatility of the environment that causes the computational landscape to shrink or expand as computing nodes go down or up respectively. As a consequence, the overall size of the system will fluctuate, affecting genetic diversity and resulting in the loss of information when islands disappear, and requiring the construction of new islands when new nodes become available.

This instability has been dealt with using the self-balancing policy proposed by Nogueras and Cotta (2016c). Like other strategies oriented to handle variable-size populations, e.g., (Fernández de Vega et al. 2003; Eiben et al. 2006), this policy aims to resize dynamically islands, having them grow when they detect a neighboring island has gone down and, analogously, having them shrink when new neighbors appear. More precisely, each island keeps a local memory of the size and number of active neighbors or neighboring islands. This memory is updated at the beginning of each evolutionary cycle by performing handshakes with neighbors. In normal conditions, islands use this handshaking to perform a local balancing procedure (Zambonelli 1999) by transferring individuals from the larger island to the smaller one. Eventually, this handshake may time out due to one of the islands having disappeared. In this case, the active node expands its own population (as described in next subsubsection) by a fraction of the size of the lost island (determined using the local information on its size and number of neighbors on the last balancing attempt). Likewise, a new neighbor may appear (initially with an empty subpopulation) and absorb a part of the existing population in neighboring nodes. Of course, while inaccuracies in the local memory held by each island and simultaneous failures of neighboring nodes can still produce fluctuations, this strategy promotes the stabilization of the overall population size.

Self-Healing. This property focuses on the maintenance and restoration of system attributes that may have been affected by internal or external actions (Blair et al. 2002; Ghosh et al. 2007). In some sense, this property has a long tradition in the context of EAs. A very good example can be found in the domain of constrained optimization: among the different strategies available to deal with infeasible solutions (generated during initialization or after the application of variations operators) we can consider the use of ad-hoc repairing procedures that restore the feasibility of these solutions (Michalewicz 1997). Focusing more specifically on the deployment of MAs on unstable computational environments, it must be noted that there are at least two major issues resulting from the volatility of the system: (i) connectivity disruptions (node failures limit the flow of information and hinder the progress of the search) and (ii) convergence perturbations (simply using random information when an island needs to grow produces a mismatch with more evolved solutions in the population and is in a way tantamount to forcing the search backwards to the earlier stages – it may provide some benefits in terms of diversity but the net effect is often detrimental as shown by Nogueras and Cotta (2015b)). To deal with the first issue, a self-rewiring strategy (Nogueras and Cotta

2015a) is used. This strategy aims to keep at all times a rich connectivity by adding new neighbors to any island whose number of active neighbors is detected to have fallen below a predefined threshold (the particular procedure used to select these neighbors is described in next subsection). As to the second issue, it is tackled by means of self-sampling (Nogueras and Cotta 2015b): new individuals used for enlarging a population are created by sampling a probabilistic model of the current population, analogously to estimation of distribution algorithms (Larrañaga and Lozano 2002; Pelikan et al. 2015). The rationale of this procedure is to have new individuals be diverse but at the same time representative of the current degree of convergence of the population. The particular probabilistic model considered in this work is a bivariate tree-based model analogous to that used in COMIT (Baluja and Davies 1997).

Model of the Computational Environment

This island-based model runs on a simulated distributed system composed of n_ν nodes on which $n_\iota \geqslant n_\nu$ islands are distributed. We here assume that islands are interconnected following a scale-free topology. This connectivity pattern is characterized for having the degree distribution follow a power law, and is commonly observed in many real-world systems, particularly in P2P systems (Albert and Barabási 2002). To generate this topology we consider the Barabási-Albert (BA) model (Barabási and Albert 1999), a network construction approach driven by preferential attachment: starting with a clique of $m+1$ vertices (where m is parameter of the model), new vertices are added one at a time, selecting for each of them m neighbors among previous vertices, selected with a probability proportional to the number of neighbors these have (hence, the more neighbors a vertex has, the more likely it is it will receive new links). The resulting network is non-regular and feature *hubs* (vertices with large degree) that contribute to its resilience (Cohen et al. 2000).

Nodes in the network are volatile and can enter and abandon the system. To model this, we consider that failures/recoveries are Weibull distributed (Weibull 1951). This distribution is commonly used to model survival times in many domains (Lee and Wang 2003) and can be seen as a generalization of the exponential distribution: while the latter is memoryless and hence failures can take place at any time with the same probability (determined by a so-called scale parameter β), the former allows failure rates to depend on time by virtue of an additional shape parameter η. To be precise, the probability of a node being available up to time t is $p(t,\eta,\beta) = \exp(-(t/\beta)^\eta)$. The particular case $\eta = 1$ corresponds to the exponential distribution and for values $\eta > 1$ failure/recovery probabilities increase with time. This is precisely the setting we have used in the experiments. Note also in relation to the use of this particular distribution that the duration of many tasks performed by human users on the computer is Weibull distributed, e.g., (Liu et al. 2010), and hence it can provide a good approximation to VC systems in which nodes become available when idle – see also (Stutzbach and Rejaie 2006) for a detailed analysis of the case of P2P systems.

Modeling and Tackling Heterogeneity

Heterogeneity is modeled using the constant performance model (Dongarra and Lastovetsky 2009), a simple modeling approach in which each network node $i \in \{1,\ldots,n_\nu\}$ is characterized just by a positive coefficient $w_i \in \mathbb{N}^+$ that indicates its computing power (no communication-related parameters are included in this model). These coefficients can represent some absolute performance measure, such as for example the number of instructions computed per unit time, or some relative measure obtained from normalizing these values. In this work, we assume for the sake of simplicity that these coefficients are known in advance and indicate the number of evolutionary cycles each node can perform per unit of time. Hence they constitute a relative performance index, and the computing power of each node can be understood to be proportional to its coefficient.

We have considered three heterogeneity scenarios differing in the way that the coefficients w_i are distributed. In all of them, the overall computing power of the network $W = \sum_i w_i$ is the same so as to not introduce any bias towards any particular configuration:

- uniform: the overall computing power W is evenly distributed among nodes, meaning that $\lfloor W/n_\nu \rfloor \leqslant w_i \leqslant \lceil W/n_\nu \rceil$. This scenario actually represents an homogeneous distribution of computing resources among nodes, all of which have a similar computing power.
- random: each coefficient w_i can have a random value in $\{1,\ldots,W-n_\nu+1\}$, subject to $W = \sum_i w_i$ as mentioned before. This is accomplished by having $w_i = 1$ initially, attributing random values in $(0,1)$ to each node and then using D'Hondt's method to distribute $W - n_\nu$ additional units among nodes according to these values. This way, each node is ensured to have at least unit power, and the remaining computing power is distributed rather proportionally to these random values.
- powerlaw: coefficients are grouped in r levels, where $r \in \{0,\ldots,r_{\max}\}$ with $r_{\max} = \lfloor \log_2 n_\nu \rfloor - 1$, so that there is a single node with power $\lfloor n_\nu/2 \rfloor$ in the highest level, and in subsequent levels there are twice as many nodes, each one with half as much power as in the previous upper level (depending on the value of n_ι the lowest level can have additional nodes if these are not enough to create a new level).

The value W implied for the last configuration (powerlaw) is used for the remaining distributions as well. Notice that depending on the scenario considered a node failure can have a different impact on the overall capacity of the system. For example, under the powerlaw distribution around half of node failures will have a low impact in this overall capacity (they will affect to the least powerful nodes) but larger disruptions are possible (albeit with a increasingly lower probability). On the opposite side of the spectrum, all failures have a priori the same moderate impact under a uniform distribution, or a completely different impact under the random distribution.

In order to deal with heterogeneity, the MA considers three different strategies with regard to how the computing resources are used. These are the following:

- single-island: the MA places an island on each node (i.e., $n_\iota = n_\nu$). Given the different computing power of each node, this means that certain islands will progress faster than others.
- multi-island: the MA places on each node i as many islands as the corresponding coefficient w_i. With this setting, there will be more islands than in the previous case (now $n_\iota = W$) but all of them will progress at about the same speed. Notice that the space requirements of MAs are typically low (even large populations are very unlikely to saturate the computer memory), and therefore placing more than one island in a node does not imply any unrealistic assumption.
- hybrid: this is a combination of the previous two, whereby the number of islands placed on a node i is more than one but fewer than the node power w_i. Thus, the fewer islands placed on the node, the faster they will progress and conversely the more islands placed, the closer to unit speed they will be. Clearly, such a distribution has the previous two configurations as limiting cases. We have opted for picking a midpoint between these two extremes: the number of islands n_ι^i placed on node i is set as $n_\iota^i = \lfloor \sqrt{w_i} + r \rfloor$ where r is a random number in $(0, 1)$ – thus $\lfloor \sqrt{w_i} \rfloor \leqslant n_\iota^i \leqslant \lceil \sqrt{w_i} \rceil$. The total w_i evolutionary cycles performed by unit time in this node are evenly distributed among these islands. This means node i has approximately $\sqrt{w_i}$ islands, each performing about $\sqrt{w_i}$ cycles per unit of time.

Notice how the single-island and multi-island scenarios represent two different approaches for taking advantage of the heterogeneity of resources: the former is oriented to maximize the depth of the search in some islands, exploiting the larger speed of some nodes, and the latter focuses on the breadth of the search, using this power to diversify the search in new islands. The hybrid would fall in between these two strategies. Obviously, in all cases the overall computational cost (measured in terms of the number of fitness evaluations) in the experiments has to be the same, in order to establish a fair comparison. Next section describes the experimentation conducted with these distributions to determine more quantitatively the effect they exert on performance.

Results

Experimental Setting

We consider $n_\nu = 32$ nodes in the network. The topology is determined in each run of the algorithm using the Barabási-Albert model with parameter $m = 2$. This model is also used for self-rewiring when an island has less than m active neighbors. Regarding node deactivation/reactivation, we use the shape parameter $\eta = 1.5$ to have an increasing hazard rate, and scale parameters $\beta = -1/\log(p)$ for $p = 1 - (kn_\nu)^{-1}$, $k \in \{1, 2, 5, 10, 20\}$. To grasp the meaning of these parameters, consider that they would correspond to an average of one node going down/up every k cycles were the failure rate constant (that is, if $\eta = 1$; we use a larger value and therefore the failure rate is somewhat smaller at the beginning of a node's stint and gets larger as time passes). This provides different scenarios ranging from low volatility ($k = 20$) to very high volatility ($k = 1$). As to heterogeneity, the total computational power of the network is $W = 81$, a value dictated by the powerlaw distribution described in the previous section. The other two heterogeneity scenarios consider this same total computational power.

The MA is composed of islands whose initial size is $\mu = 16$ individuals. We use tournament selection, one-point crossover (with probability $p_X = 1.0$), bit-flip mutation (with probability $p_M = 1/\ell$, where ℓ is the genotype length), replacement of the worst parent and migration probability $p_{mig} = 1/80$. Memes are applied using $w = 1$ and their lengths evolve within $l_{\min} = 3$ and $l_{\max} = 9$ (this length mutates with probability $p_r = 1/9$). We consider the following four algorithmic variants (the self-$\star$ properties involved are shown in parentheses – note that all variants use self-generation): $\text{LBQ}^r_{\text{comit}}$ (self-rewiring, self-sampling, self-scaling), $\text{LBQ}_{\text{comit}}$ (self-sampling, self-scaling), noB^r (self-rewiring) and noB. The experimental benchmark comprises three test functions, namely Deb's trap function (Deb and Goldberg 1993) (concatenating 32 four-bit traps), Watson et al.'s Hierarchical-if-and-only-if function (Watson et al. 1998) (using 128 bits) and Goldberg et al.'s Massively Multimodal Deceptive Problem (Goldberg et al. 1992) (using 24 six-bit blocks). We perform 25 simulations for each algorithm, problem, volatility value, heterogeneity scenario and island distribution strategy. Each run consider a total of 50 000 fitness evaluations.

Experimental Results

Figure 1 provides a global view of the results (full numerical data is provided in Tables 3-11 in the appendix). First of all, Figure 1a shows the average deviation from the optimal for each of the four algorithmic variants across all problems, heterogeneity scenarios and island distribution strategies. The variants with self-scaling and self-sampling outperform variants without them, even in the presence of self-rewiring, particularly as volatility increases. This confirms the robustness of these strategies across the different scenarios considered. In fact, $\text{LBQ}^r_{\text{comit}}$ is significantly better than the remaining algorithms (Quade test p-value ≈ 0, Holm test passed at $\alpha = 0.05$ as shown in Table 1). Thus, from a global point of view endowing the MA with self-$\star$ properties appears to be an advantageous option to cope with instability and heterogeneity, behaving in a relatively robust way with respect to the island distribution strategy chosen (we shall return to this point later on). Now, if we turn our attention to the overall results obtained under each island distribution strategy –Figure 1b– there seems to a trend of superiority of the single-island strategy (again Quade test p-value ≈ 0, and Holm test passed at $\alpha = 0.05$ as shown in Table 2). The results are actually consistent in highlighting the global appropriateness of the depth-based strategy as opposed to the breadth-based (multi-island) one, since the hybrid also falls in between these two in terms of performance. Clearly, the larger diversification of multi-island in low-volatility scenarios does not seem to pay-off with respect to the more intensive behavior of single-island. The latter also appears to be more sensitive to churn in high-volatility scenarios, in which single node failures can lead to the loss of many islands. Another interesting reading of

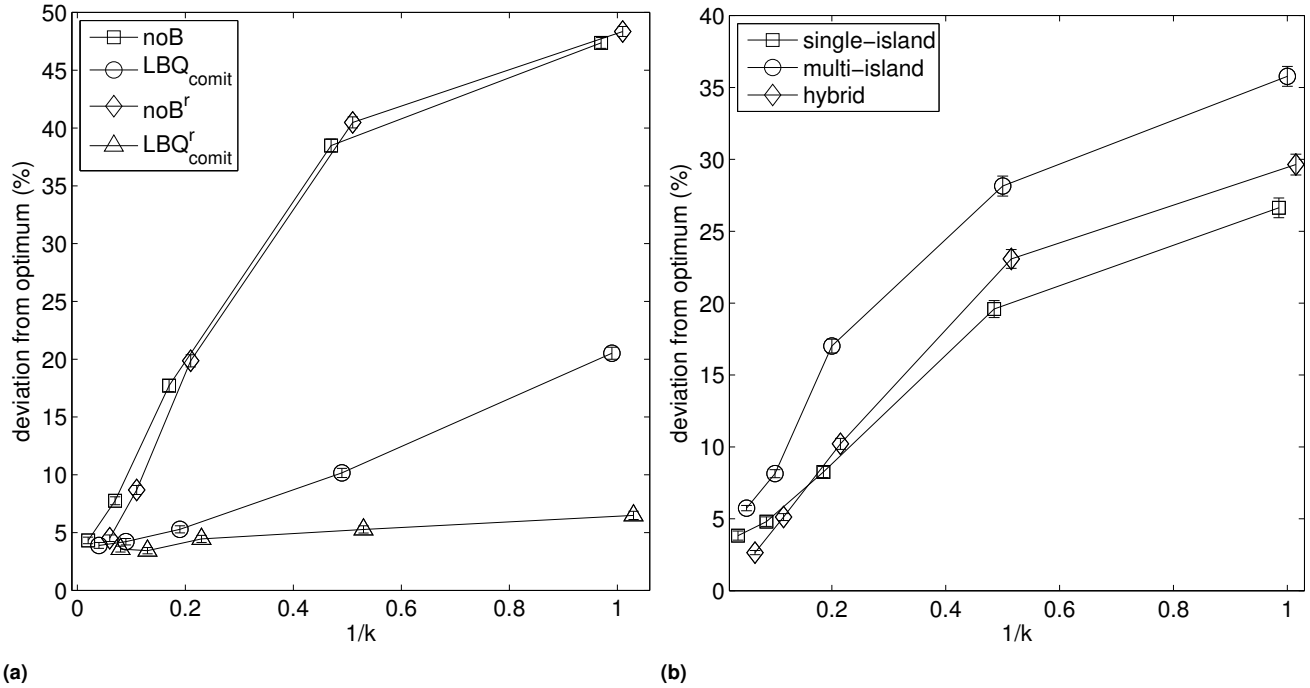

Figure 1. Average deviation from the optimal solution across all problems and heterogeneity scenarios. (a) According to algorithmic variant for all island distribution strategies. (b) According to island distribution strategy for all algorithmic variants. Note that volatility is larger as we move away from 0 in the X axis.

Table 1. Results of Holm Test ($\alpha = 0.05$) using $\text{LBQ}^r_{\text{comit}}$ as control algorithm.

i	strategy	z-statistic	p-value	α/i
1	$\text{LBQ}_{\text{comit}}$	3.840e+00	6.154e−05	5.000e−02
2	noB	9.351e+00	4.329e−21	2.500e−02
3	noB^r	1.211e+01	4.848e−34	1.667e−02

Table 2. Results of Holm Test ($\alpha = 0.05$) using single-island as control algorithm.

i	strategy	z-statistic	p-value	α/i
1	hybrid	4.427e+00	4.773e−06	5.000e−02
2	multi-island	1.146e+01	1.009e−30	2.500e−02

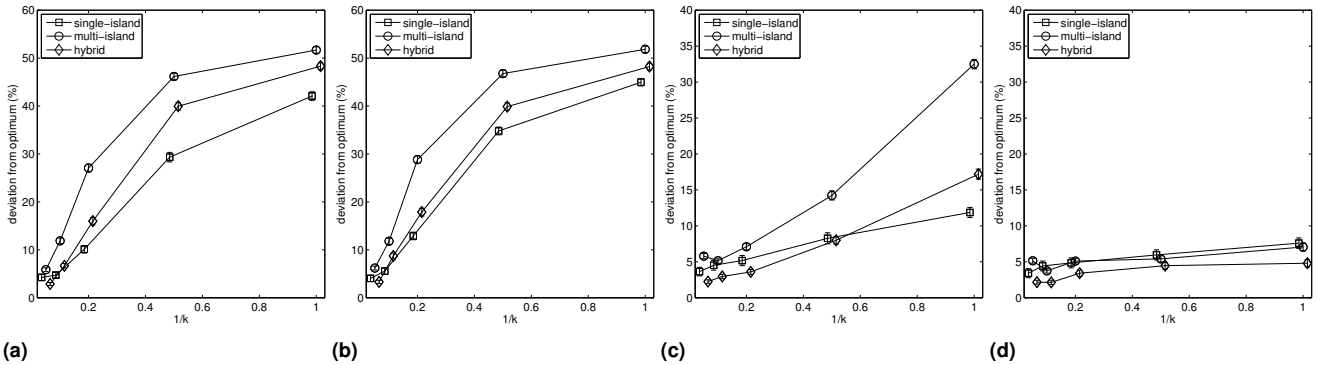

Figure 2. Average deviation from the optimal solution across all problems and heterogeneity scenarios for each algorithmic variant. (a) noB (b) noB^r (c) $\text{LBQ}_{\text{comit}}$ (d) $\text{LBQ}^r_{\text{comit}}$. Notice the different range of the Y scale in the figures.

this multi-island scenario emerges from its interpretation as a single-island situation with a larger number of computing nodes and in which node failures are not independent. It thus turns out that the single-island model would be sensitive to catastrophic simultaneous failures (albeit probably less than multi-island, if the trends observed here were extrapolated).

In addition to the previous global view of the results, it is interesting to take a more detailed look at the behavior of each algorithmic variant separately. This is shown in Figure 2. The behavior of noB and noB^r is similar and in both cases single-island provides the best results (Quade test p-value ≈ 0 and $3.519\text{e}{-14}$ respectively, and Holm test passed at $\alpha =$

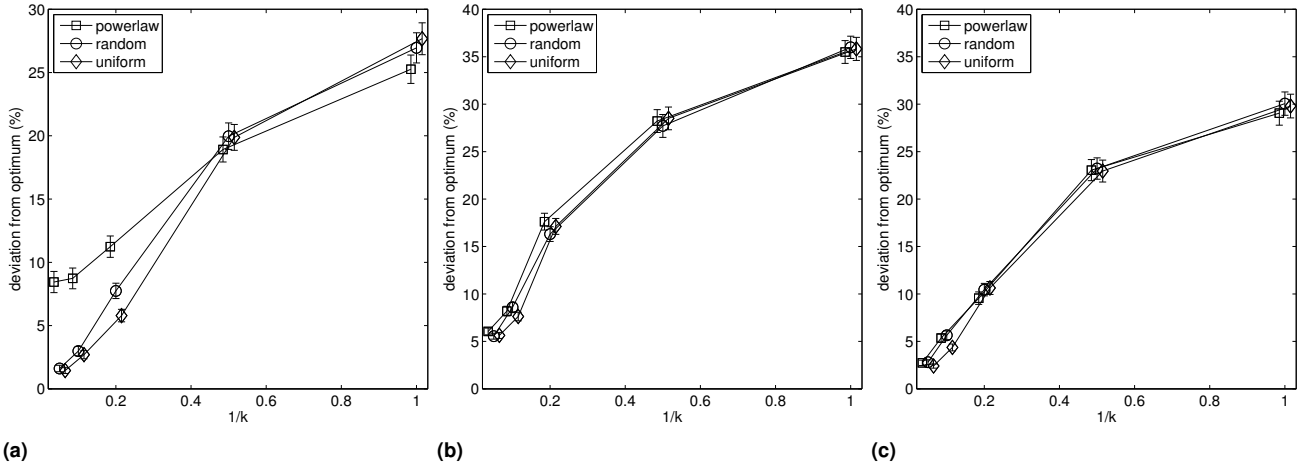

Figure 3. Average deviation from the optimal solution across all problems and algorithmic variants depending on the heterogeneity scenario for a certain island distribution strategy. (a) single-island (b) multi-island (c) hybrid. Notice the different range of the Y scale in the figures.

0.05). The qualitative difference with LBQ variants is clear (and in turn, these two also have distinctive features from each other). $\text{LBQ}_{\text{comit}}$ exhibits more graceful degradation thanks to the use of self-scaling and self-sampling. In this case, multi-island and hybrid also behave significantly worse than singe-island with statistical significance at a global level (Quade test p-value $= 1.788\text{e}{-5}$, Holm test passed at $\alpha = 0.05$). The addition of these self-$\star$ properties has made the algorithm somewhat more robust and less sensitive to the actual island distribution strategy, at least on low-volatility scenarios since single-island is not significantly better than hybrid for $k \geqslant 5$ (low to moderate volatility). This is further vindicated by the results of $\text{LBQ}^{r}_{\text{comit}}$, whose behavior is much more robust and constant, to the point that the difference between the island distribution strategies is not statistically significant (Quade test p-value $= 0.2816$).

Consider now the complementary perspective of the heterogeneity of the environment, and how it affects performance in light of the distribution strategy used. This is shown in Figure 3. Generally speaking, all strategies seem to be robust to this factor, in the sense that for a given strategy there does not seem to be statistically significant differences in performance depending on the heterogeneity scenario when all volatility values are considered. It is however interesting to note that for low volatility, the performance of single-island significantly degrades under the powerlaw scenario (Quade test p-value $= 0.0011$, Holm test passed at $\alpha = 0.05$ against powerlaw using uniform as control strategy, $k \geqslant 5$) – see Figure 3a. This can be due to the unbalance of search progress among islands caused by the decompensation of computational power of nodes in this moderate-high stability situation. On the contrary, when larger volatility values are considered, all strategies perform better in the powerlaw scenario even if not always with statistical significance (for the hybrid strategy and $k \leqslant 5$, Quade test p-value $= 8.410\text{e}{-03}$, Holm test passed at $\alpha = 0.05$ against uniform and random). This can interpreted in light of nodes with high computational power pushing forward the search during their short availability stint in a much more cost-effective way than less powerful nodes. Be it as it may, from a global point of view the single-island strategy provides the best results in each of the heterogeneity scenarios, outperforming multi-island and hybrid in all of them (the superiority of uniform is always statistically significant at $\alpha = 0.05$ according to Quade and Holm tests, except against hybrid under the powerlaw scenario). This single-island strategy thus seems to be the most robust approach to tackle heterogeneity in the scenarios considered here.

Conclusions

Resilience is an essential property that algorithms deployed on unstable environments must feature. Population-based optimization algorithms in general and memetic algorithms in particular are no exception and, although their structure provides some inherent resilience to a certain degree, they require being augmented with adequate mechanisms to counteract fluctuations in the computational landscape. In this sense, endowing them with self-$\star$ properties has been shown as an effective solution to respond to both system instability and computational heterogeneity. These properties generally capture the capability of a system for self-management in any aspect, and are reflected in this case by the capability of the EA for reacting to the instability of the system, reconfiguring itself to counteract churn. The combined use of self-scaling (whereby islands are dynamically resized and balanced in order to spread the search effort among the nodes available at each moment) and self-healing (to thwart the damage infringed by churn, either in terms of island connectivity or in island contents) seems robust under different configurations of the system, even in scenarios with large heterogeneity and volatility. Furthermore, they seem to perform synergistically as indicated by the results provided by $\text{LBQ}^{r}_{\text{comit}}$, where these properties are stacked together. Another important factor to be taken into account is the utilization pattern of the heterogeneous resources. With regard to this, we have shown that a simple strategy based on placing single islands on each computational node ultimately offers the best results when compared to placing many of them (either proportionally

to the power of the node or looking for an intermediate compromise). This is particularly interesting due to the fact that such a strategy (one computational node, one island) does not require a-priori knowledge of the computational power of nodes in the system. This strategy is less sensitive to single-node failures and seems to behave more robustly with increasing churn. Future work will be directed to confirm these findings extending the range of scenarios considered both in terms of heterogeneity and of the volatility patterns of the system. In particular, we plan to study scenarios featuring correlated node failures to analyze more in depth how they affect performance and which counter-measures could be taken to deal with it.

Acknowledgements

This work is a substantially extended version of (Nogueras and Cotta 2016b). We have expanded the background and description of methods, and completely refurbished the experimentation and the analysis by adding two new island distribution strategies.

Funding

We acknowledge support from Spanish Ministry of Economy and Competitiveness (MinEco) and European Regional Development Fund (FEDER) under project EphemeCH (TIN2014-56494-C4-1-P).

References

Alba E and Tomassini M (2002) Parallelism and evolutionary algorithms. *IEEE Transactions on Evolutionary Computation* 6(5): 443–462.

Albert R and Barabási AL (2002) Statistical mechanics of complex networks. *Review of Modern Physics* 74(1): 47–97.

Anderson DP and Reed K (2009) Celebrating diversity in volunteer computing. In: *Proceedings of the 42nd Hawaii International Conference on System Sciences*, HICSS '09. Washington, DC, USA: IEEE Computer Society, pp. 1–8.

Angeline P (1995) Morphogenic evolutionary computations: Introduction, issues and example. In: et al JM (ed.) *Fourth Annual Conference on Evolutionary Programming*. Cambridge, Massachusetts: MIT Press, pp. 387–402.

Babaoğlu Ö, Jelasity M, Montresor A, Fetzer C, Leonardi S, van Moorsel A and van Steen M (eds.) (2005) *Self-star Properties in Complex Information Systems*, *Lecture Notes in Computer Science*, volume 3460. Berlin Heidelberg: Springer-Verlag.

Baluja S and Davies S (1997) Using optimal dependency-trees for combinatorial optimization: Learning the structure of the search space. In: *14th International Conference on Machine Learning*. Morgan Kaufmann Publishers, pp. 30–38.

Barabási AL and Albert R (1999) Emergence of scaling in random networks. *Science* 286(5439): 509–512.

Beltrán M and Guzmán A (2009) How to balance the load on heterogeneous clusters. *International Journal of High Performance Computing Applications February* 23: 99–118.

Berns A and Ghosh S (2009) Dissecting self-⋆ properties. In: *Third IEEE International Conference on Self-Adaptive and Self-Organizing Systems - SASO 2009*. San Francisco, CA: IEEE Press, pp. 10–19.

Beyer H (1995) Toward a theory of evolution strategies: Self-adaptation. *Evolutionary Computation* 3(3): 311–348.

Blair G, Coulson G, Blair L, Duran-Limon H, Grace P, Moreira R and Parlavantzas N (2002) Reflection, self-awareness and self-healing in openORB. In: *First Workshop on Self-healing Systems*. New York, NY, USA: ACM Press, pp. 9–14.

Boulet P, Dongarra J, Rastello F, Robert Y and Vivien F (1999) Algorithmic issues on heterogeneous computing platforms. *Parallel Processing Letters* 9(2): 197–213.

Caraffini F, Neri F and Picinali L (2014) An analysis on separability for memetic computing automatic design. *Information Sciences* 265: 1–22.

Cohen R, Erez K, ben Avraham D and Havlin S (2000) Resilience of the internet to random breakdowns. *Physical Review Letters* 85: 4626–4628.

Cotta C, Fernández-Leiva AJ, Fernández de Vega F, Chávez F, Merelo JJ, Castillo PA, Bello G and Camacho D (2015) Ephemeral computing and bioinspired optimization - challenges and opportunities. In: *7th International Joint Conference on Evolutionary Computation Theory and Applications*. Lisboa, Portugal, pp. 319–324.

Cotta C, Sevaux M and Sörensen K (eds.) (2008) *Adaptive and Multilevel Metaheuristics*, *Studies in Computational Intelligence*, volume 136. Berlin Heidelberg: Springer-Verlag.

Deb K and Goldberg D (1993) Analyzing deception in trap functions. In: Whitley L (ed.) *Second Workshop on Foundations of Genetic Algorithms*. Vail, Colorado, USA: Morgan Kaufmann Publishers, pp. 93–108.

Dongarra J and Lastovetsky AL (2009) *High Performance Heterogeneous Computing*. John Wiley & Sons.

Eiben AE (2005) Evolutionary computing and autonomic computing: Shared problems, shared solutions? In: Babaoğlu Ö et al. (eds.) *Self-star Properties in Complex Information Systems*, *Lecture Notes in Computer Science*, volume 3460. Berlin Heidelberg: Springer-Verlag, pp. 36–48.

Eiben AE, Schut MC and de Wilde AR (2006) Is self-adaptation of selection pressure and population size possible? - a case study. In: Runarsson T et al. (eds.) *Parallel Problem Solving from Nature - PPSN IX*, *Lecture Notes in Computer Science*, volume 4193. Berlin Heidelberg: Springer-Verlag, pp. 900–909.

Eiben AE and Smith JE (2003) *Introduction to Evolutionary Computation*. Natural Computing Series. Berlin Heidelberg: Springer-Verlag.

Fernández de Vega F, Cantú-Paz E, López J and Manzano T (2004) Saving resources with plagues in genetic algorithms. In: Yao X et al. (eds.) *Parallel Problem Solving from Nature - PPSN VIII*, *Lecture Notes in Computer Science*, volume 3242. Berlin Heidelberg: Springer-Verlag, pp. 272–281.

Fernández de Vega F, Vanneschi L and Tomassini M (2003) The effect of plagues in genetic programming: A study of variable-size populations. In: Ryan C et al. (eds.) *Genetic Programming*, *Lecture Notes in Computer Science*, volume 2610. Berlin Heidelberg: Springer-Verlag, pp. 317–326.

Ghosh D, Sharman R, Rao H and Upadhyaya S (2007) Self-healing systems - survey and synthesis. *Decision Support Systems* 42(4): 2164–2185.

Goldberg D, Deb K and Horn J (1992) Massive multimodality, deception and genetic algorithms. In: Männer R and Manderick B (eds.) *Parallel Problem Solving from Nature - PPSN II*. New York, NY, USA: Elsevier Science Inc., pp. 37–48.

Gorges-Schleuter M (1989) ASPARAGOS: an asynchronous parallel genetic optimization strategy. In: Schaffer J (ed.)

Third International Conference on Genetic Algorithms. San Francisco, CA: Morgan Kaufmann Publishers, pp. 422–427.

Grefenstette JJ (1981) Parallel adaptive algorithms for function optimization. Technical Report CS-81-19, Vanderbilt University, Nashville, TN.

Grosso P (1985) *Computer simulation of genetic adaptation: Parallel subcomponent interaction in a multilocus model*. PhD Thesis, University of Michigan, Ann Arbor.

Hidalgo J, Lanchares J, Fernández de Vega F and Lombraña D (2007) Is the island model fault tolerant? In: Thierens D et al. (eds.) *Genetic and Evolutionary Computation - GECCO 2007*. New York, NY, USA: ACM Press, pp. 2737 – 2744.

Hinterding R, Michalewicz Z and Eiben A (1997) Adaptation in evolutionary computation: A survey. In: *Fourth IEEE Conference on Evolutionary Computation*. Piscataway, New Jersey: IEEE Press, pp. 65–69.

Horn P (2001) Autonomic computing: IBM's perspective on the state of information technology. Technical report, IBM Research. URL http://people.scs.carleton.ca/~soma/biosec/readings/autonomic_computing.pdf. Accessed 14/05/2015.

Huebscher M and McCann J (2008) A survey of autonomic computing -degrees, models and applications. *ACM Computing Surveys* 40(3). Article 7.

Korpela E, Werthimer D, Anderson D, Cobb J and Leboisky M (2001) Seti@home-massively distributed computing for seti. *Computing in Science Engineering* 3(1): 78–83.

Krasnogor N and Gustafson S (2004) A study on the use of "self-generation" in memetic algorithms. *Natural Computing* 3(1): 53–76.

Krasnogor N and Smith JE (2000) A memetic algorithm with self-adaptive local search: TSP as a case study. In: Whitley D et al. (eds.) *Genetic and Evolutionary Computation - GECCO 2000*. San Francisco, CA, USA: Morgan Kaufmann Publishers, pp. 987–994.

Laredo J, Bouvry P, González D, Fernández de Vega F, Arenas M, Merelo JJ and Fernandes C (2014) Designing robust volunteer-based evolutionary algorithms. *Genetic Programming and Evolvable Machines* 15(3): 221–244.

Laredo J, Castillo P, Mora A, Merelo JJ and Fernandes C (2008) Resilience to churn of a peer-to-peer evolutionary algorithm. *International Journal of High Performance Systems Architecture* 1(4): 260–268.

Larrañaga P and Lozano J (eds.) (2002) *Estimation of Distribution Algorithms*, *Genetic Algorithms and Evolutionary Computation*, volume 2. Berlin Heidelberg: Springer Verlag.

Larson SM, Snow CD, Shirts MR, and Pande VS (2002) Folding@home and genome@home: Using distributed computing to tackle previously intractable problems in computational biology. arXiv:0901.0866.

Lastovetsky A (2014) Heterogeneous parallel computing: from clusters of workstations to hierarchical hybrid platforms. *Supercomputing Frontiers and Innovations* 1(3): 70–87.

Lastovetsky A and Reddy R (2007) Data partitioning with a functional performance model of heterogeneous processors. *International Journal of High Performance Computing Applications* 21: 76–90.

Lastovetsky AL (2003) *Parallel Computing on Heterogeneous Networks*. New York, NY, USA: John Wiley & Sons, Inc.

Lee E and Wang J (eds.) (2003) *Statistical Methods for Survival Data Analysis*. Hoboken, NJ, USA: John Wiley & Sons.

Lewis A, Mostaghim S and Scriven I (2009) Asynchronous multi-objective optimisation in unreliable distributed environments. In: Lewis A, Mostaghim S and Randall M (eds.) *Biologically-Inspired Optimisation Methods: Parallel Algorithms, Systems and Applications*, *Studies in Computational Intelligence*, volume 210. Berlin Heidelberg: Springe-Verlag, pp. 51–78.

Liu C, White R and Dumais S (2010) Understanding web browsing behaviors through Weibull analysis of dwell time. In: *33rd International ACM SIGIR Conference on Research and Development in Information Retrieval - SIGIR 2010*. New York, NY, USA: ACM Press, pp. 379–386.

Lombraña González D, Jiménez Laredo J, Fernández de Vega F and Merelo Guervós JJ (2010) Characterizing fault-tolerance of genetic algorithms in desktop grid systems. In: Cowling P and Merz P (eds.) *Evolutionary Computation in Combinatorial Optimization*, *Lecture Notes in Computer Science*, volume 6022. Berlin Heidelberg: Springer-Verlag, pp. 131–142.

Lombraña González D, Jiménez Laredo J, Fernández de Vega F and Merelo Guervós JJ (2012) Characterizing fault-tolerance in evolutionary algorithms. In: Fernández de Vega F et al. (eds.) *Parallel Architectures and Bioinspired Algorithms*, *Studies in Computational Intelligence*, volume 415. Berlin Heidelberg: Springer-Verlag, pp. 77–99.

Michalewicz Z (1997) Repair algorithms. In: Bäck T et al. (eds.) *Handbook of Evolutionary Computation*. Bristol, New York: Institute of Physics Publishing and Oxford University Press, pp. C5.4:1–5.

Milojičić D, Kalogeraki V, Lukose R, Nagaraja K, Pruyne J, Richard B, Rollins S and Xu Z (2002) Peer-to-peer computing. Technical Report HPL-2002-57, Hewlett-Packard Labs.

Moscato P (1999) Memetic algorithms: A short introduction. In: Corne D, Dorigo M and Glover F (eds.) *New Ideas in Optimization*. Maidenhead, Berkshire, England, UK: McGraw-Hill, pp. 219–234.

Neri F and Cotta C (2012) Memetic algorithms and memetic computing optimization: A literature review. *Swarm and Evolutionary Computation* 2: 1–14.

Neri F, Cotta C and Moscato P (eds.) (2012) *Handbook of Memetic Algorithms*, *Studies in Computational Intelligence*, volume 379. Berlin Heidelberg: Springer-Verlag.

Nogueras R and Cotta C (2014) 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*, volume 8672. Berlin Heidelberg: Springer Verlag, pp. 731–740.

Nogueras R and Cotta C (2015a) Self-balancing multimemetic algorithms in dynamic scale-free networks. In: Mora A and Squillero G (eds.) *Applications of Evolutionary Computing*, *Lecture Notes in Computer Science*, volume 9028. Berlin Heidelberg: Springer Verlag, pp. 177–188.

Nogueras R and Cotta C (2015b) 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*, volume 9108. Berlin Heidelberg: Springer Verlag, pp. 69–78.

Nogueras R and Cotta C (2016a) Self-healing strategies for memetic algorithms in unstable and ephemeral computational environments. *Natural Computing* DOI:10.1007/s11047-016-9560-7.

Nogueras R and Cotta C (2016b) A study of the performance of self-⋆ memetic algorithms on heterogeneous ephemeral environments. In: Handl J et al. (eds.) *Parallel Problem Solving from Nature – PPSN XIV*, *Lecture Notes in Computer Science*, volume 9921. Berlin Heidelberg, pp. 91–100.

Nogueras R and Cotta C (2016c) Studying self-balancing strategies in island-based multimemetic algorithms. *Journal of Computational and Applied Mathematics* 293: 180–191.

Ong Y and Keane A (2004) Meta-lamarckian learning in memetic algorithm. *IEEE Transactions on Evolutionary Computation* 8(2): 99–110.

Ong Y, Lim M and Chen X (2010) Memetic computation –past, present and future. *IEEE Computational Intelligence Magazine* 5(2): 24–31.

Pelikan M, Hauschild M and Lobo F (2015) Estimation of distribution algorithms. In: Kacprzyk J and Pedrycz W (eds.) *Handbook of Computational Intelligence*. Berlin Heidelberg: Springer Verlag, pp. 899–928.

Renard H, Robert Y, and Vivien F (2006) Data redistribution algorithms for heterogeneous processor rings. *International Journal of High Performance Computing Applications February* 20: 31–43.

Sarmenta L (1998) Bayanihan: Web-based volunteer computing using java. In: Masunaga Y, Katayama T and Tsukamoto M (eds.) *Worldwide Computing and Its Applications - WWCA 1998*, *Lecture Notes in Computer Science*, volume 1368. Berlin Heidelberg: Springer-Verlag, pp. 444–461.

Schwefel H (1992) Imitating evolution: Collective, two-level learning processes. In: *Explaining Process and Change - Approaches to Evolutionary Economics*. Ann Arbor, Michigan: University of Michigan Press, pp. 49–63.

Smith JE (2002) Co-evolution of memetic algorithms: Initial investigations. In: Merelo Guervós JJ et al. (eds.) *Parallel Problem Solving from Nature - PPSN VII*, *Lecture Notes in Computer Science*, volume 2439. Berlin Heidelberg: Springer Verlag, pp. 537–548.

Smith JE (2007) Coevolving memetic algorithms: A review and progress report. *IEEE Transactions on Systems, Man and Cybernetics, Part B* 37(1): 6–17.

Smith JE (2008) Self-adaptation in evolutionary algorithms for combinatorial optimisation. In: Cotta C, Sevaux M and Sörensen K (eds.) *Adaptive and Multilevel Metaheuristics*, *Studies in Computational Intelligence*, volume 136. Berlin Heidelberg: Springer-Verlag, pp. 31–57.

Smith JE (2012) Self-adaptative and coevolving memetic algorithms. In: Neri F, Cotta C and Moscato P (eds.) *Handbook of Memetic Algorithms*, *Studies in Computational Intelligence*, volume 379. Berlin Heidelberg: Springer-Verlag, pp. 167–188.

Sterritt R and Bustard D (2003) Towards an autonomic computing environment. In: *14th International Workshop on Database and Expert Systems Applications*. pp. 694–698.

Stutzbach D and Rejaie R (2006) Understanding churn in peer-to-peer networks. In: *6th ACM SIGCOMM Conference on Internet Measurement - IMC 2006*. New York, NY, USA: ACM Press, pp. 189–202.

Sudholt D (2014) Parallel evolutionary algorithms. In: Kacprzyk J and Pedrycz W (eds.) *Handbook of Computational Intelligence*. Berlin Heidelberg: Springer-Verlag, pp. 929–959.

Tanese R (1989) Distributed genetic algorithms. In: *3rd International Conference on Genetic Algorithms*. San Francisco, CA, USA: Morgan Kaufmann Publishers, pp. 434–439.

Watson R, Hornby G and Pollack J (1998) Modeling building-block interdependency. In: Eiben A et al. (eds.) *Parallel Problem Solving from Nature - PPSN V*, *Lecture Notes in Computer Science*, volume 1498. Berlin Heidelberg: Springer Verlag, pp. 97–106.

Weibull W (1951) A statistical distribution function of wide applicability. *Journal of Applied Mechanics* 18(3): 293–297.

Zambonelli F (1999) Exploiting biased load information in direct-neighbour load balancing policies. *Parallel Computing* 25(6): 745–766.

Zhang G, Rong H, Neri F and Pérez-Jiménez M (2014) An optimization spiking neural p system for approximately solving combinatorial optimization problems. *International Journal of Neural Systems* 24(5): 1440006.

Zhao W and Schulzrinne H (2004) Dotslash: A self-configuring and scalable rescue system for handling web hotspots effectively. In: *International Workshop on Web Caching and Content Distribution - WCW 2004*. pp. 1–18.

Numerical Results

Table 3. Results (averaged for 25 runs) of the different MMAs on the three problems considered using the single-island distribution strategy in uniform environment The median ($\tilde{x}$), mean ($\bar{x}$) and standard error of the mean ($\sigma_{\bar{x}}$) are indicated.

		TRAP		H-IFF		MMDP	
strategy	k	$\tilde{x}$	$\bar{x} \pm \sigma_{\bar{x}}$	$\tilde{x}$	$\bar{x} \pm \sigma_{\bar{x}}$	$\tilde{x}$	$\bar{x} \pm \sigma_{\bar{x}}$
noB	20	0.00	0.25 ± 0.20	0.00	0.78 ± 0.78	3.00	2.82 ± 0.35
	10	0.00	1.65 ± 0.48	0.00	1.61 ± 1.12	5.99	6.29 ± 0.56
	5	5.63	6.33 ± 0.99	0.00	7.13 ± 2.74	10.48	11.96 ± 0.91
	2	28.75	28.88 ± 1.16	50.52	44.24 ± 2.80	27.46	26.93 ± 0.60
	1	43.13	43.21 ± 0.89	60.90	60.61 ± 0.40	35.13	35.23 ± 0.59
$\mathrm{LBQ}_{\mathrm{comit}}$	20	0.00	0.25 ± 0.20	0.00	2.81 ± 1.62	1.50	1.81 ± 0.37
	10	0.00	0.38 ± 0.15	0.00	5.00 ± 1.93	1.50	2.14 ± 0.40
	5	0.00	0.10 ± 0.10	0.00	2.50 ± 1.41	3.00	2.43 ± 0.39
	2	1.25	2.03 ± 0.46	0.00	11.49 ± 2.62	5.99	5.83 ± 0.73
	1	8.13	8.71 ± 0.94	0.00	12.42 ± 3.24	13.48	13.81 ± 0.80
noB^r	20	0.00	0.85 ± 0.44	0.00	0.67 ± 0.67	3.00	2.86 ± 0.49
	10	0.00	1.00 ± 0.45	0.00	4.39 ± 1.87	4.49	4.89 ± 0.78
	5	6.25	7.10 ± 1.30	20.83	13.53 ± 2.84	10.48	10.12 ± 0.82
	2	29.37	28.91 ± 1.28	51.52	49.60 ± 1.35	29.62	27.38 ± 1.01
	1	46.25	45.33 ± 0.98	61.81	60.65 ± 0.77	35.13	35.38 ± 0.60
LBQ^r_{comit}	20	0.00	0.00 ± 0.00	0.00	3.67 ± 1.60	0.00	0.59 ± 0.20
	10	0.00	0.00 ± 0.00	0.00	3.92 ± 1.88	0.00	1.12 ± 0.27
	5	0.00	0.50 ± 0.33	0.00	5.92 ± 2.21	1.50	1.86 ± 0.34
	2	0.00	0.75 ± 0.38	0.00	8.61 ± 2.42	4.49	3.87 ± 0.42
	1	2.50	3.12 ± 0.60	0.00	10.22 ± 2.72	3.00	3.40 ± 0.56

Table 4. Results (averaged for 25 runs) of the different MMAs on the three problems considered using the hybrid distribution strategy in uniform environment The median ($\tilde{x}$), mean ($\bar{x}$) and standard error of the mean ($\sigma_{\bar{x}}$) are indicated.

		TRAP		H-IFF		MMDP	
strategy	k	$\tilde{x}$	$\bar{x} \pm \sigma_{\bar{x}}$	$\tilde{x}$	$\bar{x} \pm \sigma_{\bar{x}}$	$\tilde{x}$	$\bar{x} \pm \sigma_{\bar{x}}$
noB	20	0.00	1.33 ± 0.44	0.00	0.83 ± 0.83	5.99	5.35 ± 0.64
	10	5.00	5.03 ± 0.83	0.00	2.39 ± 1.32	8.99	9.76 ± 1.00
	5	13.75	14.17 ± 1.34	21.88	21.73 ± 3.82	14.98	15.86 ± 0.88
	2	38.75	36.67 ± 0.96	56.94	54.97 ± 1.54	31.30	30.91 ± 0.64
	1	46.25	47.14 ± 0.79	61.98	61.26 ± 0.48	35.79	36.30 ± 0.56
$\mathrm{LBQ}_{\mathrm{comit}}$	20	0.00	0.55 ± 0.25	0.00	0.78 ± 0.78	4.49	5.13 ± 0.55
	10	0.00	0.62 ± 0.26	0.00	1.56 ± 1.08	4.49	4.73 ± 0.45
	5	0.00	1.05 ± 0.41	0.00	2.89 ± 1.41	7.16	7.02 ± 0.65
	2	5.00	5.92 ± 0.87	0.00	2.82 ± 1.59	11.65	11.55 ± 0.42
	1	17.50	17.44 ± 1.23	0.00	13.12 ± 3.14	22.46	21.44 ± 0.94
noB^r	20	0.00	1.50 ± 0.47	0.00	1.67 ± 1.15	5.99	5.62 ± 0.63
	10	2.50	4.12 ± 0.89	0.00	8.00 ± 2.49	8.99	9.80 ± 0.62
	5	16.25	16.83 ± 1.33	21.53	20.61 ± 3.84	17.97	17.35 ± 0.77
	2	37.50	35.36 ± 1.24	56.77	54.99 ± 1.56	31.11	30.94 ± 0.54
	1	46.88	46.43 ± 0.73	62.16	62.35 ± 0.39	38.02	37.65 ± 0.41
LBQ^r_{comit}	20	0.00	0.25 ± 0.16	0.00	1.33 ± 0.92	4.49	4.53 ± 0.52
	10	0.00	0.35 ± 0.15	0.00	2.56 ± 1.45	3.00	3.47 ± 0.53
	5	0.00	0.25 ± 0.10	0.00	6.50 ± 1.98	3.00	3.33 ± 0.54
	2	0.00	1.20 ± 0.40	0.00	4.56 ± 1.66	5.66	5.53 ± 0.50
	1	1.25	2.45 ± 0.61	0.00	7.44 ± 2.35	4.49	4.65 ± 0.61

Table 5. Results (averaged for 25 runs) of the different MMAs on the three problems considered using the multi-island distribution strategy in uniform environment. The median ($\tilde{x}$), mean ($\bar{x}$) and standard error of the mean ($\sigma_{\bar{x}}$) are indicated.

		TRAP		H-IFF		MMDP	
strategy	k	$\tilde{x}$	$\bar{x} \pm \sigma_{\bar{x}}$	$\tilde{x}$	$\bar{x} \pm \sigma_{\bar{x}}$	$\tilde{x}$	$\bar{x} \pm \sigma_{\bar{x}}$
noB	20	4.37	4.12 ± 0.56	0.00	1.56 ± 1.08	11.65	11.01 ± 0.59
	10	11.25	10.38 ± 1.00	0.00	6.88 ± 2.65	16.14	15.64 ± 0.65
	5	25.00	25.20 ± 1.29	41.67	36.68 ± 3.36	24.47	24.00 ± 0.70
	2	45.63	45.17 ± 0.69	59.38	59.31 ± 0.63	35.79	35.29 ± 0.52
	1	51.25	51.00 ± 0.52	64.12	64.09 ± 0.16	40.46	40.51 ± 0.35
$\mathrm{LBQ}_{\mathrm{comit}}$	20	3.75	5.11 ± 0.70	0.00	0.67 ± 0.67	10.48	10.95 ± 0.56
	10	2.50	3.00 ± 0.41	0.00	0.00 ± 0.00	11.65	11.61 ± 0.46
	5	5.00	5.10 ± 0.67	0.00	2.56 ± 1.44	12.82	12.90 ± 0.61
	2	16.86	16.14 ± 1.12	0.00	7.84 ± 2.52	21.47	21.50 ± 0.73
	1	35.00	34.25 ± 0.95	35.59	30.32 ± 3.90	32.80	32.87 ± 0.70
noB^r	20	6.25	5.78 ± 0.80	0.00	0.78 ± 0.78	11.65	11.32 ± 0.53
	10	11.88	10.42 ± 1.00	0.00	6.98 ± 2.43	16.14	15.36 ± 0.65
	5	25.62	26.67 ± 1.31	39.06	32.90 ± 3.41	23.63	24.11 ± 0.82
	2	46.25	45.60 ± 0.70	60.76	59.40 ± 0.68	35.46	35.46 ± 0.51
	1	53.13	52.86 ± 0.39	64.06	63.83 ± 0.29	40.46	40.34 ± 0.45
LBQ^r_{comit}	20	4.37	4.42 ± 0.62	0.00	0.00 ± 0.00	11.65	11.86 ± 0.40
	10	1.25	1.75 ± 0.30	0.00	0.00 ± 0.00	9.82	9.45 ± 0.49
	5	2.50	2.50 ± 0.40	0.00	1.44 ± 1.00	11.32	11.32 ± 0.44
	2	3.13	3.48 ± 0.45	0.00	2.31 ± 1.28	10.15	10.55 ± 0.56
	1	5.00	5.80 ± 0.71	0.00	5.81 ± 2.25	7.49	8.20 ± 0.44

Table 6. Results (averaged for 25 runs) of the different MMAs on the three problems considered using the single-island distribution strategy in random environment. The median ($\tilde{x}$), mean ($\bar{x}$) and standard error of the mean ($\sigma_{\bar{x}}$) are indicated.

		TRAP		H-IFF		MMDP	
strategy	k	$\tilde{x}$	$\bar{x} \pm \sigma_{\bar{x}}$	$\tilde{x}$	$\bar{x} \pm \sigma_{\bar{x}}$	$\tilde{x}$	$\bar{x} \pm \sigma_{\bar{x}}$
noB	20	0.00	0.45 ± 0.24	0.00	2.39 ± 1.33	1.50	1.62 ± 0.31
	10	0.00	1.65 ± 0.55	0.00	3.00 ± 1.45	3.00	3.88 ± 0.59
	5	5.00	7.02 ± 1.33	19.44	17.36 ± 3.04	10.15	10.22 ± 1.11
	2	26.25	26.83 ± 1.50	51.91	46.55 ± 2.73	26.63	26.59 ± 0.81
	1	40.63	40.88 ± 0.92	58.85	58.11 ± 0.77	34.95	34.41 ± 0.64
$\mathrm{LBQ}_{\mathrm{comit}}$	20	0.00	0.50 ± 0.29	0.00	4.00 ± 1.65	1.50	1.86 ± 0.47
	10	0.00	0.40 ± 0.28	0.00	5.33 ± 2.03	1.50	1.92 ± 0.37
	5	0.00	1.42 ± 0.44	0.00	7.94 ± 2.22	1.50	2.16 ± 0.42
	2	1.25	2.70 ± 0.64	0.00	9.97 ± 2.48	7.16	6.58 ± 0.60
	1	8.13	8.64 ± 0.82	0.00	14.50 ± 3.28	13.48	12.81 ± 0.79
noB^r	20	0.00	0.75 ± 0.41	0.00	1.56 ± 1.09	1.50	2.04 ± 0.38
	10	0.00	1.97 ± 0.55	0.00	3.99 ± 2.28	3.00	4.07 ± 0.67
	5	8.75	8.70 ± 1.37	20.83	19.62 ± 3.54	10.15	10.82 ± 1.09
	2	28.75	29.21 ± 1.44	51.74	49.35 ± 2.34	28.12	28.22 ± 0.58
	1	41.87	41.54 ± 1.03	60.24	59.61 ± 0.61	34.29	34.20 ± 0.59
LBQ^r_{comit}	20	0.00	0.05 ± 0.05	0.00	2.83 ± 1.61	1.50	1.24 ± 0.27
	10	0.00	0.20 ± 0.14	0.00	7.94 ± 2.31	1.50	1.42 ± 0.32
	5	0.00	0.50 ± 0.20	0.00	6.17 ± 2.08	0.00	0.96 ± 0.32
	2	0.00	1.50 ± 0.49	0.00	9.06 ± 2.84	3.00	3.08 ± 0.40
	1	0.00	1.90 ± 0.54	19.44	13.33 ± 2.36	3.00	3.48 ± 0.36

Table 7. Results (averaged for 25 runs) of the different MMAs on the three problems considered using the hybrid distribution strategy in random environment. The median ($\tilde{x}$), mean ($\bar{x}$) and standard error of the mean ($\sigma_{\bar{x}}$) are indicated.

		TRAP		H-IFF		MMDP	
strategy	k	$\tilde{x}$	$\bar{x} \pm \sigma_{\bar{x}}$	$\tilde{x}$	$\bar{x} \pm \sigma_{\bar{x}}$	$\tilde{x}$	$\bar{x} \pm \sigma_{\bar{x}}$
noB	20	1.25	1.58 ± 0.39	0.00	2.33 ± 1.29	4.49	4.79 ± 0.45
	10	3.13	3.63 ± 0.60	0.00	10.58 ± 2.48	8.99	9.24 ± 0.74
	5	11.88	12.55 ± 1.50	21.53	16.85 ± 3.17	13.48	15.42 ± 1.23
	2	37.50	36.85 ± 1.02	54.86	52.29 ± 2.37	30.13	30.21 ± 0.56
	1	47.54	46.48 ± 0.84	61.46	61.09 ± 0.73	38.46	38.00 ± 0.54
$\mathrm{LBQ}_{\mathrm{comit}}$	20	0.00	1.42 ± 0.44	0.00	0.67 ± 0.67	4.49	4.80 ± 0.57
	10	0.00	0.95 ± 0.29	0.00	3.78 ± 1.55	5.99	6.12 ± 0.49
	5	0.00	1.42 ± 0.44	0.00	4.11 ± 1.70	5.99	6.21 ± 0.58
	2	6.25	6.39 ± 0.85	0.00	6.28 ± 2.11	11.65	11.57 ± 0.72
	1	20.00	19.83 ± 1.02	0.00	13.44 ± 3.09	23.48	22.24 ± 1.00
noB^r	20	0.00	1.20 ± 0.32	0.00	4.83 ± 1.77	4.49	4.97 ± 0.48
	10	2.50	3.97 ± 0.85	19.44	16.35 ± 3.05	7.49	8.46 ± 0.74
	5	14.38	14.88 ± 1.25	21.88	26.09 ± 3.34	16.47	17.42 ± 0.96
	2	33.75	34.36 ± 1.02	56.94	55.06 ± 1.17	31.12	30.87 ± 0.67
	1	46.88	46.64 ± 0.93	61.63	61.37 ± 0.41	36.63	36.24 ± 0.67
LBQ^r_{comit}	20	0.00	0.57 ± 0.32	0.00	2.56 ± 1.28	4.49	3.91 ± 0.43
	10	0.00	0.15 ± 0.11	0.00	1.11 ± 0.79	3.00	3.39 ± 0.35
	5	0.00	0.50 ± 0.24	0.00	6.19 ± 2.15	4.49	3.79 ± 0.46
	2	2.50	2.25 ± 0.44	0.00	7.94 ± 2.45	4.49	4.54 ± 0.45
	1	1.25	2.50 ± 0.74	0.00	9.72 ± 2.09	3.00	3.16 ± 0.55

Table 8. Results (averaged for 25 runs) of the different MMAs on the three problems considered using the multi-island distribution strategy in random environment. The median ($\tilde{x}$), mean ($\bar{x}$) and standard error of the mean ($\sigma_{\bar{x}}$) are indicated.

		TRAP		H-IFF		MMDP	
strategy	k	$\tilde{x}$	$\bar{x} \pm \sigma_{\bar{x}}$	$\tilde{x}$	$\bar{x} \pm \sigma_{\bar{x}}$	$\tilde{x}$	$\bar{x} \pm \sigma_{\bar{x}}$
noB	20	3.75	4.03 ± 0.62	0.00	0.83 ± 0.83	11.65	11.06 ± 0.53
	10	11.25	11.25 ± 1.00	0.00	11.89 ± 2.89	14.98	15.10 ± 0.55
	5	23.12	22.63 ± 1.25	31.42	24.85 ± 3.52	23.30	22.92 ± 0.87
	2	43.75	43.38 ± 0.81	59.72	58.66 ± 0.78	34.29	34.36 ± 0.61
	1	52.50	51.96 ± 0.55	63.89	63.49 ± 0.36	39.63	39.93 ± 0.57
$\mathrm{LBQ}_{\mathrm{comit}}$	20	5.00	4.76 ± 0.51	0.00	0.78 ± 0.78	13.15	11.92 ± 0.45
	10	3.75	4.41 ± 0.56	0.00	0.67 ± 0.67	10.48	10.96 ± 0.47
	5	5.63	5.88 ± 0.72	0.00	3.97 ± 1.51	13.15	13.21 ± 0.58
	2	15.00	14.66 ± 1.29	0.00	3.72 ± 1.84	20.31	20.50 ± 0.80
	1	35.62	35.15 ± 0.89	27.02	30.97 ± 2.41	33.96	33.66 ± 0.64
noB^r	20	5.00	4.72 ± 0.60	0.00	0.78 ± 0.78	11.98	12.09 ± 0.63
	10	10.62	10.05 ± 0.97	0.00	10.49 ± 2.99	17.97	16.96 ± 0.61
	5	25.00	24.66 ± 1.29	46.01	34.94 ± 3.46	25.13	26.16 ± 0.68
	2	43.75	43.90 ± 0.82	60.76	60.31 ± 0.66	36.96	36.45 ± 0.48
	1	51.88	51.52 ± 0.43	64.06	63.63 ± 0.48	39.29	39.38 ± 0.42
LBQ^r_{comit}	20	3.75	4.05 ± 0.43	0.00	0.67 ± 0.67	11.32	10.93 ± 0.49
	10	1.25	1.52 ± 0.36	0.00	0.00 ± 0.00	10.15	9.88 ± 0.43
	5	1.25	1.62 ± 0.37	0.00	3.72 ± 1.53	11.65	11.17 ± 0.41
	2	1.88	2.28 ± 0.42	0.00	4.28 ± 1.78	9.82	10.02 ± 0.59
	1	3.75	4.65 ± 0.62	0.00	11.06 ± 2.74	7.49	6.54 ± 0.61

Table 9. Results (averaged for 25 runs) of the different MMAs on the three problems considered using the single-island distribution strategy in powerlaw environment. The median ($\tilde{x}$), mean ($\bar{x}$) and standard error of the mean ($\sigma_{\bar{x}}$) are indicated.

		TRAP		H-IFF		MMDP	
strategy	k	$\tilde{x}$	$\bar{x} \pm \sigma_{\bar{x}}$	$\tilde{x}$	$\bar{x} \pm \sigma_{\bar{x}}$	$\tilde{x}$	$\bar{x} \pm \sigma_{\bar{x}}$
noB	20	0.00	1.02 ± 0.38	32.64	28.30 ± 2.87	0.00	0.54 ± 0.17
	10	0.00	2.00 ± 0.72	25.00	21.10 ± 3.22	0.00	1.26 ± 0.42
	5	0.00	2.75 ± 0.79	34.72	25.08 ± 3.44	1.50	3.15 ± 0.87
	2	14.38	14.79 ± 1.69	44.10	34.74 ± 3.94	11.98	14.45 ± 1.58
	1	31.25	29.98 ± 1.76	54.86	48.27 ± 3.82	28.20	28.09 ± 1.22
$\mathrm{LBQ}_{\mathrm{comit}}$	20	0.00	1.40 ± 0.68	23.61	19.69 ± 3.33	0.00	0.42 ± 0.18
	10	0.00	1.10 ± 0.37	25.00	23.89 ± 3.46	0.00	0.54 ± 0.21
	5	0.00	1.50 ± 0.57	33.33	28.00 ± 2.92	0.00	0.48 ± 0.17
	2	2.50	3.50 ± 0.84	33.33	30.69 ± 2.49	0.00	1.57 ± 0.53
	1	5.63	6.70 ± 1.03	21.53	21.28 ± 3.01	7.49	7.87 ± 0.88
noB^r	20	0.00	0.40 ± 0.31	29.86	26.60 ± 3.14	0.00	0.60 ± 0.21
	10	2.50	3.07 ± 0.71	30.56	23.85 ± 3.64	1.50	2.73 ± 0.80
	5	10.00	9.89 ± 1.24	28.13	25.98 ± 3.49	8.99	10.28 ± 1.15
	2	29.37	28.57 ± 1.27	47.92	46.17 ± 2.00	26.96	25.78 ± 1.14
	1	39.38	39.29 ± 1.52	57.24	56.90 ± 1.17	31.12	31.87 ± 1.01
LBQ^r_{comit}	20	0.00	1.25 ± 0.48	27.78	20.75 ± 3.59	0.00	0.35 ± 0.15
	10	0.00	0.75 ± 0.43	27.78	24.11 ± 3.12	0.00	0.36 ± 0.16
	5	0.00	1.22 ± 0.50	23.61	25.22 ± 3.07	1.50	1.32 ± 0.26
	2	0.00	1.70 ± 0.50	26.39	21.96 ± 3.39	3.00	3.12 ± 0.51
	1	2.50	3.65 ± 0.74	30.56	26.55 ± 2.86	3.00	2.71 ± 0.48

Table 10. Results (averaged for 25 runs) of the different MMAs on the three problems considered using the hybrid distribution strategy in powerlaw environment. The median ($\tilde{x}$), mean ($\bar{x}$) and standard error of the mean ($\sigma_{\bar{x}}$) are indicated.

		TRAP		H-IFF		MMDP	
strategy	k	$\tilde{x}$	$\bar{x} \pm \sigma_{\bar{x}}$	$\tilde{x}$	$\bar{x} \pm \sigma_{\bar{x}}$	$\tilde{x}$	$\bar{x} \pm \sigma_{\bar{x}}$
noB	20	0.00	1.22 ± 0.37	0.00	2.89 ± 1.62	5.99	5.53 ± 0.72
	10	5.00	4.92 ± 0.82	0.00	5.86 ± 2.19	8.99	8.56 ± 0.82
	5	13.12	13.77 ± 1.77	19.44	18.02 ± 3.49	14.98	15.51 ± 1.22
	2	36.97	36.21 ± 1.24	55.03	51.43 ± 2.11	29.95	30.14 ± 0.96
	1	48.75	46.87 ± 1.36	61.98	61.05 ± 0.69	36.12	36.69 ± 0.66
$\mathrm{LBQ}_{\mathrm{comit}}$	20	0.00	0.80 ± 0.36	0.00	2.33 ± 1.29	3.00	3.73 ± 0.60
	10	0.00	1.15 ± 0.34	0.00	3.89 ± 1.65	4.49	3.87 ± 0.47
	5	0.00	1.70 ± 0.45	0.00	3.78 ± 1.55	4.49	4.09 ± 0.51
	2	3.75	4.26 ± 0.67	0.00	11.73 ± 2.90	11.65	11.40 ± 0.66
	1	15.42	14.62 ± 1.30	0.00	12.25 ± 3.07	20.31	20.35 ± 0.78
noB^r	20	1.25	2.37 ± 0.60	0.00	2.19 ± 1.57	4.49	5.38 ± 0.61
	10	8.13	8.08 ± 1.09	0.00	10.90 ± 2.73	8.99	8.80 ± 0.80
	5	11.88	11.07 ± 1.18	21.88	20.38 ± 3.83	14.98	16.21 ± 1.07
	2	35.62	34.42 ± 1.23	53.30	52.08 ± 1.67	31.30	30.85 ± 0.77
	1	46.88	47.04 ± 1.08	61.98	60.58 ± 0.86	36.12	35.91 ± 0.77
LBQ^r_{comit}	20	0.00	0.65 ± 0.27	0.00	3.11 ± 1.46	3.00	2.61 ± 0.39
	10	0.00	0.15 ± 0.11	0.00	4.78 ± 1.80	3.00	3.26 ± 0.45
	5	0.00	0.30 ± 0.13	0.00	6.25 ± 2.09	4.49	3.58 ± 0.54
	2	0.00	0.65 ± 0.33	0.00	9.61 ± 2.54	3.00	3.84 ± 0.61
	1	2.50	3.00 ± 0.65	0.00	7.94 ± 2.71	1.50	2.32 ± 0.33

Table 11. Results (averaged for 25 runs) of the different MMAs on the three problems considered using the multi-island distribution strategy under powerlaw environment. The median ($\tilde{x}$), mean ($\bar{x}$) and standard error of the mean ($\sigma_{\bar{x}}$) are indicated.

		TRAP		H-IFF		MMDP	
strategy	k	$\tilde{x}$	$\bar{x} \pm \sigma_{\bar{x}}$	$\tilde{x}$	$\bar{x} \pm \sigma_{\bar{x}}$	$\tilde{x}$	$\bar{x} \pm \sigma_{\bar{x}}$
noB	20	3.75	3.80 ± 0.66	0.00	4.61 ± 1.94	11.98	11.66 ± 0.55
	10	10.62	11.54 ± 1.19	0.00	8.67 ± 2.68	15.81	15.52 ± 0.75
	5	24.38	25.10 ± 1.33	43.06	37.79 ± 3.31	23.63	24.34 ± 0.93
	2	45.00	45.15 ± 0.59	59.20	58.32 ± 0.72	36.12	35.83 ± 0.48
	1	51.25	51.00 ± 0.61	63.84	63.49 ± 0.33	39.96	39.74 ± 0.60
$\mathrm{LBQ}_{\mathrm{comit}}$	20	5.00	5.10 ± 0.53	0.00	0.83 ± 0.83	11.98	11.96 ± 0.49
	10	2.50	3.25 ± 0.56	0.00	1.64 ± 1.16	11.65	10.81 ± 0.55
	5	5.00	5.50 ± 0.75	0.00	2.89 ± 1.36	11.65	11.83 ± 0.63
	2	13.75	13.87 ± 0.81	0.00	11.36 ± 2.85	19.98	18.50 ± 0.84
	1	35.00	33.53 ± 1.03	31.42	29.53 ± 3.16	32.29	32.22 ± 0.75
noB^r	20	4.37	3.80 ± 0.57	0.00	4.97 ± 1.88	11.65	11.50 ± 0.65
	10	11.88	11.15 ± 1.09	0.00	8.05 ± 3.44	16.92	16.63 ± 0.73
	5	28.13	27.06 ± 1.08	42.19	38.11 ± 3.53	24.47	24.75 ± 0.85
	2	44.38	43.70 ± 0.89	61.28	60.96 ± 0.31	34.62	35.24 ± 0.58
	1	51.88	51.70 ± 0.50	64.06	63.71 ± 0.32	39.62	39.64 ± 0.43
LBQ^r_{comit}	20	5.00	4.45 ± 0.55	0.00	0.00 ± 0.00	8.99	9.89 ± 0.50
	10	1.25	1.50 ± 0.42	0.00	0.67 ± 0.67	8.99	8.90 ± 0.48
	5	0.00	0.43 ± 0.16	0.00	5.39 ± 1.85	8.66	8.24 ± 0.50
	2	0.00	1.50 ± 0.41	0.00	8.94 ± 2.14	5.66	5.33 ± 0.50
	1	3.75	3.99 ± 0.64	0.00	12.17 ± 3.04	4.49	5.30 ± 0.59