Studying Self-Balancing Strategies in Island-Based Multimemetic Algorithms

Rafael Nogueras[a], Carlos Cotta[a,*]

[a]*Dept. Lenguajes y Ciencias de la Computación, Universidad de Málaga, ETSI Informática, Campus de Teatinos, 29071 Málaga, Spain*

Abstract

Multimemetic algorithms (MMAs) are memetic algorithms that explicitly exploit the evolution of memes, i.e., non-genetic expressions of problem-solving strategies. We aim to study their deployment on an unstable environment with complex topology and volatile resources. We analyze their behavior and performance on environments with different churn rates, and how they are affected by the use of self-balancing strategies aiming to compensate the loss of existing islands and react to the apparition of new ones. We investigate two such strategies, one based on quantitative balance (in which populations are resized dynamically to cope with node failure/recoveries) and another on qualitative balance (in which genetic/memetic information is actually exchanged to achieve balance). We evaluate these on scale-free network topologies and compare them to an unbalanced strategy that keeps island sizes constant. Experimentation firstly focuses on memetic takeover, carried out on an idealized selecto-Lamarckian model of MMAs (used as a surrogate of the latter) and indicating that the two balancing strategies exhibit complementary profiles in terms of diversity preservation. The results also indicate that the qualitative version is more robust to churn than both the unbalanced and the quantitatively balanced counterpart. This is subsequently confirmed with an empirical evaluation of full-fledged MMAs on a benchmark composed of four hard pseudo-Boolean problems. The qualitative version provides the best performance in global terms, significantly outperforming the remaining variants.

Keywords: Memetic algorithms, multimemetic algorithms, load balancing, self-adaptation, faulty environment

1. Introduction

Memetic algorithms (MAs) [1] are optimization techniques based on the orchestrated interplay of elements from population-based global search methods

*Corresponding author: Phone: +34 952 137158; Fax: +34 952 131397
Email address: `ccottap@lcc.uma.es` (Carlos Cotta)

and trajectory-based local search techniques [2]. A central tenet in MAs is the notion of *meme* [3]: originally defined as units of imitation, memes can be interpreted in the context of MAs as computational problem-solving procedures. While these can take different forms, they commonly represent local-search techniques, often fixed or pre-defined in advance. Hence, these MAs can be regarded as operating with static implicit memes. This is not the only possibility though. Indeed, explicitly handling (and evolving) memes is an idea that has been around for some time now –cf. [4]– and is now a core idea in the concept of memetic computing [5, 6, 7, 8]. Such an explicit treatment of memes can be found in, for example, multimemetic algorithms (MMAs) [9, 10, 11, 12, 13]. In these techniques, each solution carries memes that determine the way self-improvement is conducted. Since these memes evolve alongside solutions, the whole system constitutes a self-adaptive search approach [14, 15, 16, 17].

When analyzing the way memes propagate throughout the population in an MMA, we can observe that the propagation dynamics is more complex than that of genes, if only because memes are only indirectly evaluated according to the effect they exert on the latter. For this reason, mismatches between genes and memes may cause potentially good memes to become extinct or poor memes to proliferate [18]. These issues are particularly relevant to multi-population models of MMAs, in which, besides internal population dynamics, we have to consider the effect of the communication between populations too [19]. This is even more true in the presence of complex, dynamic computational environments such as those emerging from the use of peer-to-peer networks [20] and volunteer computing networks [21]. These are characterized by the volatility of computational resources, the term *churn* having been coined to denote the collective effect of a plethora of peers entering or leaving the system independently over time.

Focusing on the use of island-based evolutionary algorithms on these kinds of unstable computational platforms, the presence of churn can cause the best solution to be lost (if the island comprising it goes down before it has the opportunity to migrate [22]) and will, in general, have detrimental effects on the overall population diversity. This can be tackled using corrective measures –e.g., using a fault-management strategy to recover from failures [23, 24, 25, 26]– or by preventive measures, whereby the algorithm self-adapts to failures as they happen, trying to maintain a broad genetic/memetic pool at all times. The latter may have the advantage of being inherently autonomous and decentralized, not requiring the global state of the system to be monitored or for external snapshots of it to be maintained. This approach is precisely the focus of this paper: we depart from the use of fault-recovery strategies considered in previous work [27] and investigate the effect that introducing decentralized balancing strategies has on the functioning of the algorithm. To this end, we firstly use an idealized selecto-Lamarckian model of MMAs [18] which allows studying issues such as memetic diversity and convergence. This model is extended here to an island-based context, as described in Section 2.1. Subsequently, we describe a model of the computational environment (analogous to that used in [27]) in Section 2.2 and present a self-balancing algorithm in Section 2.3. Then, we report a broad experimental evaluation in Section 3. After analyzing the behavior of the surrogate model, results are reported on actual full-fledged MMAs in order to confirm the previous findings, analyzing performance and providing a sensitivity analysis of the self-balancing strategy. We close the paper with an overview of conclusions and an outline of future work in Section 4.

2. Material and Methods

2.1. Algorithmic Setting

As stated in the previous section, the first part of the experimentation has been done using an idealized selecto-Lamarckian model so as to obtain a preliminary assessment of the behavior of MMAs in terms of convergence and diversity when deployed on a dynamic computational scenario. This model is an abstract characterization of MMAs, first introduced in [18]. It consists of a population $P = [\langle g_1, m_1\rangle, \cdots, \langle g_\mu, m_\mu\rangle]$ of μ individuals, which are subject to the evolutionary operations of selection, local search and replacement as shown in Algorithm 1 (selecto-Lamarckian phase). Each individual is a tuple $\langle g_i, m_i\rangle \in D^2$, for some $D \subset \mathbb{R}$. In each tuple, g_i is the genotype (also representing fitness for simplicity) and m_i is a meme (its *potential*, to be precise). The latter is an idealized concept that tries to capture how good solutions can become by using this meme (thus constituting an abstract notion of meme fitness [28]). More precisely, this potential is expressed via a function $f : D^2 \to D$ monotonically increasing in the first parameter, which represents the application of a meme to a gene: an individual $\langle g, m\rangle$ becomes $\langle f(g, m), m\rangle$ after the application of the meme. It must hold that (i) $\lim_{n\to\infty} f^n(g, m) = m$ if $g < m$ ($f^n(g, m)$ being the n-fold application of the meme m to g) and that (ii) $f(g, m) = g$ if $g \geqslant m$. This means that the meme has no effect on solutions whose quality is higher than the meme's potential, but in the case that the quality is lower it improves the latter, reaching its potential in the limit. While this is obviously a highly idealized description of the action of memes (which in general depends on the match between the genotype and the meme on a problem-specific basis) it constitutes an initial approximation that can be used to study the generalities of meme propagation as shown in [18].

Interaction between individuals in a population is restricted by a spatial structure given by a $\mu \times \mu$ Boolean matrix S, where S_{ij}= `true` if, and only if, the individual in the i-th location can interact with the individual in the j-th location [29]. In this case we consider panmixia, i.e., S_{ij}= `true` for all i, j. This basic model is here extended to a multi-population setting [30, 31] as illustrated in Algorithm 1: n_ι islands are assumed to work in parallel (being interconnected according to a certain topology $\mathcal{N}$) and migration steps are added before/after the selecto-Lamarckian phase. Migration is performed asynchronously: at the beginning of each cycle the island checks whether or not migrants have been received. If this is the case, they are accepted into the population following a given migrant replacement policy. Then, at the end of each cycle, migration is stochastically performed just like any other evolutionary operator. If done,

Algorithm 1: Island-Based Selecto-Lamarckian Model

for $i \in [1 \cdots n_\iota]$ **do in parallel**
    Initialize(pop_i) ; `// initialize` i`-th population`
    $buffer_i \leftarrow \emptyset$; `// initialize` i`-th migration buffer`
end
while $\neg$ BudgetExhausted() **do**
    for $i \in [1 \cdots n_\iota]$ **do in parallel**
        CheckMigrants (pop_i, $buffer_i$) ; `// accept migrants (if any)`
        `// -----------Begin selecto-Lamarckian phase-----------`
        $k \leftarrow rand(1, \mu_i)$; `// pick random location`
        $\langle g, m\rangle \leftarrow Selection(pop_i, S_i, k)$; `// do tournament selection`
        $g' \leftarrow f(g, m)$; `// local improvement`
        $pop_i \leftarrow Replace(pop_i, S_i, k, \langle g', m\rangle)$; `// replace worst parent`
        `// -----------End selecto-Lamarckian phase-------------`
        if $rand() < p_{mig}$ **then**
            for $j \in \mathcal{N}_i$ **do**
                SendMigrants(pop_i, $buffer_j$) ; `// send migrants`
            end
        end
    end
end

some migrants are selected using a certain migrant selection policy and sent to neighboring islands. Following the results in [19], we use random selection of migrants and deterministic replacement of the worst individuals in the receiving island.

The selecto-Lamarckian model can be readily extended to a full-fledged MMA. Following previous work –e.g., [19, 32]– we have specifically considered an MMA inspired by the work of Smith [13, 33] wherein each individual in the population carries a binary genotype and a single meme representing a rewriting rule $A \rightarrow C$, where both A and C are patterns of a certain length taken from $\{0, 1, \#\}$; the symbol '#' is a wildcard interpreted as "don't care" in the antecedent A of the rule and as "don't change" in the consequent C. These memes are utilized to generate neighbors of the solutions they are attached to, by looking for instances of A and substituting them with C; for example, let a genotype be 11101100, and let a meme be 1#1 → 0#1. A possible application of the meme could be as follows:

$$11\,\overbrace{101}^{A}\,100 \quad \xrightarrow{\text{meme}} \quad 11\,\underbrace{001}_{C}\,100$$

Since a meme might be applied in different parts of the genotype, a parameter w (determining the maximal number of meme applications) is used to keep the total cost of the process under control. The best neighbor generated (if better

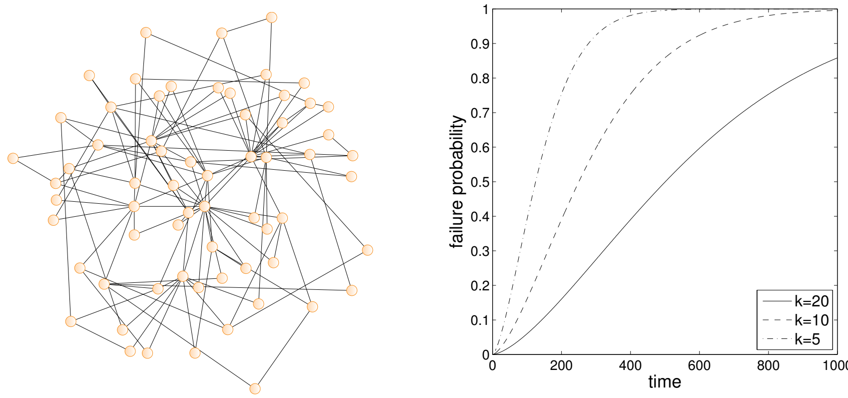

Figure 1: (Left) Example of scale-free network generated with Barabási-Albert model ($n_\iota = 64$, $m = 2$). (Right) Failure probabilities under a Weibull distribution with the parameters used in Section 3.

that the current solution) is kept. Note that the length of each meme is not fixed rather it evolves itself, increasing or decreasing by one with probability p_r within a certain length range $[l_{\min}, l_{\max}]$ – see [33]. Apart from the use of memes embedded within individuals, this MMA otherwise resembles a steady-state memetic algorithm in which parents are selected using binary tournament, and recombination (one-point crossover), mutation (bit-flip) and local-search (conducted using the meme carried out by the individual as illustrated before) are used to generate the offspring, which replaces the worst parent, following the model presented in [18].

2.2. Computational Environment Model

We assume the deployment of the aforementioned island-based model on a simulated distributed system composed of n_ι nodes whose availability changes dynamically. The interconnection network is assumed to be scale-free, a non-regular complex topology commonly observed in many natural, social and computational processes in which node degrees exhibit a power-law distribution. We generate this kind of topology using the Barabási-Albert model [34], whereby a network is grown by adding a new node at a time and in which the selection of new links is driven by preferential attachment [35] (i.e., each new node is connected to m existing nodes, selected with a probability proportional to their current degree). Fig. 1 (left) shows an example of this kind of network.

As to the dynamics of the system, we consider a setting analogous to [27]: the n_ι nodes it comprises are assumed to be initially available but they can individually abandon the system at any point, possibly becoming available again later on. To model the dynamics of each node we consider that failures/recoveries are Weibull distributed [36]. This distribution is a generalization of the exponential distribution: while the latter is memoryless (i.e., time-independent), the former supports hazard rates increasing or decreasing over time. For this reason, it

Algorithm 2: Standard Balancing Procedure

```
procedure Standard-LB (↓ 𝒩, ↕ A[], n[], W[])
// 𝒩:  list of references to neighboring islands
// A:  Boolean array to keep track of which neighbors are
//     active.
// n, W:Integer arrays with the number of active neighbors
//      and population sizes of each neighbor in 𝒩.

for v ∈ 𝒩 do
    if v.ping() then
        // The neighbor is active.  Balancing is attempted.
        (n_v, W_v, b) ← Do-LB (v);
        A_v ← true;
    else
        if A_v then
            // The neighbor was active last time.
            // Enlarging own population.
            w ← GetPopSize();
            SetPopSize(w + W_v/n_v);
            A_v ← false;
        end
    end
end
```

is often utilized in survival analysis to model time-to-failure in mechanical or biological systems [37]. Moreover, it is known that the duration of some human tasks on the computer follows a Weibull distribution, see [38, 39]. Hence, it can be used to model computing environments such as, for example, volunteering computing systems in which computer nodes are contributed when idle. Mathematically, the distribution is controlled by a shape parameter η and a scale parameter β. The probability of a node being available up to time t_1 given that it was available up to time t_0 is

$$p(t_0, t_1, \eta, \beta) = e^{-[(t_1/\beta)^\eta - (t_0/\beta)^\eta]}.$$

We use $\eta > 1$ in our experimentation –see Section 3– and hence the hazard rate increases with time. Fig. 1 (right) shows an example of failure probability as a function of time. For the sake of simplicity, when a node leaves the system we do not re-wire the interconnections between islands (as done in [22]) so as to not introduce an additional level of complexity in the algorithm, allowing a more focused study of the balancing strategies presented in next section.

2.3. Self-Balancing Strategies

The instability of the system makes some islands disappear when a node goes down and likewise, new islands must be (re-)created when a node goes up again.

Algorithm 3: Basic Balancing Routine

```
function Do-LB (↕ v) returns (ℕ, ℕ, 𝔹)
// v:  neighbor to do balancing with.
w ← GetPopSize();
w′ ← v.GetPopSize();
Δ ← w − w′;
if |Δ| > δ then
|   SetPopSize(w − Δ/2);
|   v.SetPopSize(w′ + Δ/2);
|   b ← true;
else
|   b ← false;
end
return (v.GetActiveNeighbors(), v.GetPopSize(), b)
```

Algorithm 4: Balancing Procedure upon Reactivation

```
procedure Reactivate-LB (↓ 𝒩, ↑ A[], n[], W[])
// Parameters with the same meaning as in algorithm 2.
balanced ← GetPopSize() > 0;
for v ∈ 𝒩 do
|   if v.ping() then
|   |   // The neighbor is active.  Balancing is attempted.
|   |   (n_v, W_v, b) ← Do-LB (v);
|   |   A_v ← true;
|   |   balanced ← balanced ∧ b;
|   else
|   |   A_v ← false;
|   end
end
if ¬balanced then
|   // No balancing done.  Reinitializing from scratch.
|   SetPopSize(C_1);
end
```

This means that in the absence of any strategy to deal with this phenomenon, the global population size will fluctuate (possibly wildly, depending on the churn rate) and so will genetic/memetic diversity. To cope with this, we can introduce a balancing strategy. Given the inherently decentralized focus of this work, we consider local strategies, both in the decisional and the migrational sense, i.e., both the decision making and the information exchange are done locally between neighboring islands, without having global information or central control [40]. More precisely, we use a variation of a direct-neighbor policy [41] as illustrated in Algorithms 2–4.

The core of this policy is captured by Algorithm 3: therein, two nodes communicate and try to achieve a locally-balanced status between them; if the difference between their population sizes is above a certain threshold δ, they resize their populations accordingly to meet at the middle point. This basic routine is used within the standard balancing procedure performed at each node (see Algorithm 2), whereby the neighbors are pinged to determine whether or not they are active and if so, balancing is attempted with them. In the case a neighbor has just gone down (i.e., it was active in the previous balancing attempt but it is no longer active), the island enlarges its own population by a fraction (proportional to the number of active neighbors of the node that went down) of the population size of the former node. The situation is slightly different when a node goes up: it attempts to balance with neighboring islands and in case it cannot do this (because no neighbor is active or their population differences are below the balancing threshold), the node resorts to self-reinitializing using a fixed population size C_1 – see Algorithm 4. Note that a reactivated node may have been at the passive end of a balancing attempt before entering its own balancing procedure, and hence it may have a non-zero population at the start of this procedure. No reinitialization is required in this case. As a final caveat, it is possible that the network disconnects at some point and hence a node may go down without active neighbors to absorb a part of its population size. In this situation, the total population size can eventually decrease; in the long run this can be alleviated by picking a large enough value of C_1.

We have approached the balancing procedure described above in two ways: quantitative and qualitative. In the quantitative approach resizing is done by truncation (removing the worst individuals in the population, as many as required) for reduction and addition of random immigrants [42] (as many as needed) for enlarging. Thus, balancing is done just on numerical terms. In the qualitative approach, balancing involves the actual exchange of genetic/memetic information: a packet of individuals of the required size is randomly selected in (and removed from) the donating island and transferred to the receiving island. Note that island reinitialization from scratch and population enlarging when a neighbor goes down are always quantitative procedures in either case.

3. Experimental Results

The experimentation with the selecto-Lamarckian model has been done using the model described in Section 2 with $n_\iota = 64$ islands of $\mu = 50$ individuals initially. Each of these individuals $\langle g_i, m_i \rangle$ is initialized by picking $g_i \sim U(0, 1/2)$ and $m_i \sim U(0, 1)$, thus giving genotypes room for improvement with high potential memes and minimizing the chances of low-quality memes thriving by attaching them to high-quality genotypes [18]. The meme is applied, using a linear combination $f(g, m) = \gamma g + (1 - \gamma)m$ (for $m > g$). We use $\gamma = 0.9$ (i.e., the gap between the gene and the meme is decreased by 10% in each meme application) to have a gentle improving curve. We denote as LB and LBQ, respectively, the algorithmic variants with quantitative and qualitative balancing. We also use a variant without balancing –noB– in which reactivated nodes are

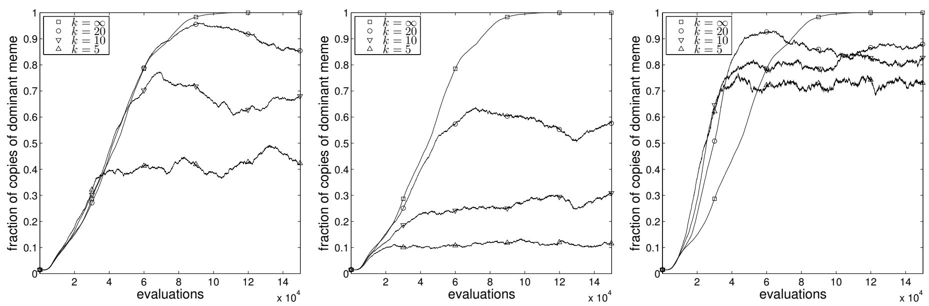

Figure 2: Fraction of copies of dominant meme. From left to right: no balancing, quantitative balancing, qualitative balancing. The curve for $k = \infty$ is common to all algorithms.

reinitialized from scratch. Parameter m in the Barabási-Albert model is set to $m = 2$, and we let $p_{mig} = 1/250$. 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 $p = 1 - (kn_\iota)^{-1}$, $k \in \{5, 10, 20, \infty\}$. Intuitively, these settings correspond to an average of one island going down/up every k iterations if the hazard rate is constant (it is not since $\eta > 1$, but this gives a mental anchor to interpret these values – numerically, the resulting scale parameter can be approximated as $\beta \simeq kn_\iota - 1/2$). This provides different scenarios ranging from low ($k = 20$) to high ($k = 5$) churn rates (the case $k = \infty$ corresponds to a static network without churn). The balancing threshold is set to $\delta = 1$ and the parameter C_1 used during eventual island reinitialization from scratch is set to $2\mu = 100$ individuals to account for the fact that the average asymptotic number of active islands with the parameters used is $n_\iota/2$. We perform 25 simulations for each algorithm and churn scenario.

Let us firstly focus on memetic takeover. Fig. 2 shows the fraction of copies in the whole population corresponding to the most spread meme. As expected, while the whole population eventually converges to a homogeneous state for $k = \infty$, values $k < \infty$ lead to a semi-stable state in which only a fraction of the population is taken-over by a dominant meme. We can see in Fig. 2 (left) that this semi-stable fraction becomes increasingly lower with the increasing churn rate, which is explained by the continuous loss of islands in advanced state of convergence and reintroduction of new fresh islands. A more qualitative view of this situation is provided in Fig. 3 (top row). Therein, memes are represented by gray shades[1] (the darker the color the worse the meme), and each vertical slice of the figure represents the distribution of memes at a certain moment. The white area representing high-quality memes starts to grow and stabilizes at a certain level under the pressure of low-quality memes being reintroduced into the population (bottom half of each plot). The situation is more marked in

[1]For better visualization, a color version of this figure is available online at `http://figshare.com/s/c050e114635011e4939706ec4bbcf141`.

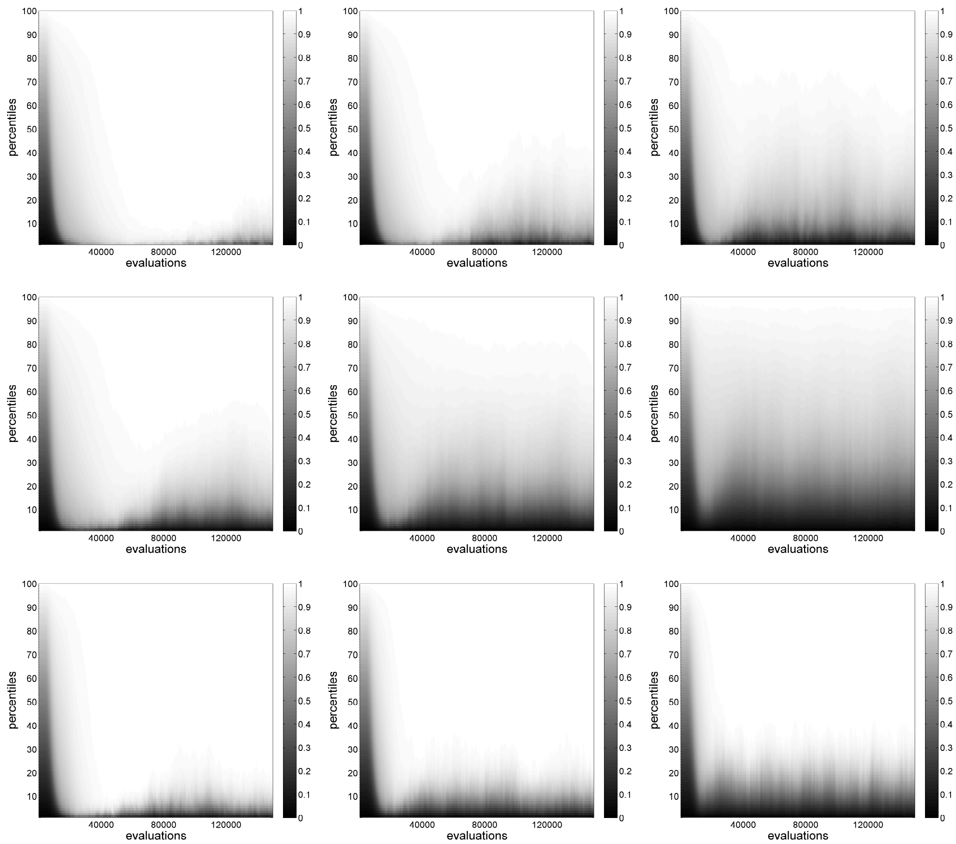

Figure 3: Meme maps for the different strategies. (Top) No balancing (Middle) Quantitative balancing (Bottom) Qualitative balancing. In each row, from left to right: $k = 20$, $k = 10$ and $k = 5$.

the case of quantitative balance, due to the additional diversity provided by the introduction of random migrants for population enlargement, as shown in Fig. 3 (middle row). This results in the dominant meme taking over only a small fraction of the whole population – Fig. 2 (middle). This is completely different to the behavior of qualitative balancing: it looks to be a more robust strategy, providing a similar level of convergence regardless of the churn rate – see Fig. 2 (right). Indeed, using local balancing to reconstruct islands upon reactivation allows keeping the momentum of the search, redistributing the existing population among the new nodes without having to resort to random reinitialization so frequently as noB and LB; hence it can cope better with churn.

In order to confirm the behavioral patterns observed, we now turn our attention to a full-fledged MMA as described in Section 2.1. We consider $n_\iota = 32$ islands whose initial size is $\mu = 16$ individuals and use $p_{mig} = 1/80$ and *maxevals* = 50000. Meme evolution and application are controlled by parameters $w = 1$,

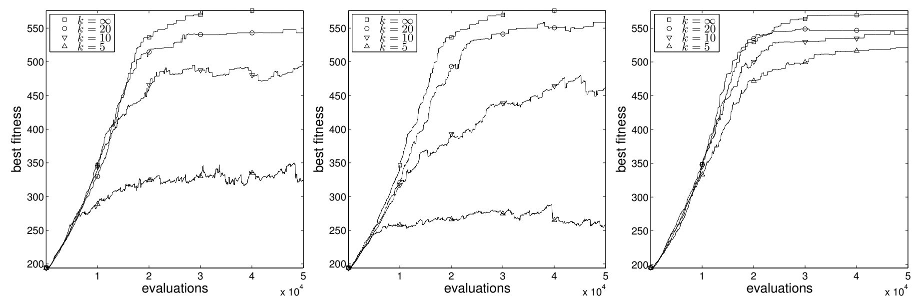

Figure 4: Best fitness for the HIFF function depending on parameter k. From left to right: no balancing, quantitative balancing, qualitative balancing. The curve for $k = \infty$ is common to all algorithms.

Table 1: Results (25 runs) of the different MMAs on the four problems considered. The median ($\tilde{x}$), mean ($\bar{x}$) and standard error of the mean (σ_x) are indicated. The symbols $\bullet$ and $\circ$ indicate whether numerical differences are significant or not according to a Wilcoxon ranksum test ($\alpha = 0.05$). The first symbol in the pair corresponds to the comparison with $k = \infty$, and the second one to the comparison with the best algorithm (marked with $\star$) for the corresponding problem and k.

		TRAP			HIFF		
strategy	k	$\tilde{x}$	$\bar{x} \pm \sigma_{\bar{x}}$		$\tilde{x}$	$\bar{x} \pm \sigma_{\bar{x}}$	
–	∞	32.0	31.8 ± 0.1		576.0	570.2 ± 5.8	
no balancing	20	31.6	31.3 ± 0.2	●●	576.0	549.6 ± 11.3	○○
	10	30.0	29.6 ± 0.4	●●	576.0	496.7 ± 19.7	●○
	5	22.4	22.4 ± 0.4	●●	308.0	332.1 ± 18.6	●●
quantitative	20	30.8	30.5 ± 0.3	●●	576.0	557.1 ± 8.8	○⋆
	10	27.4	27.4 ± 0.5	●●	450.0	464.2 ± 17.9	●●
	5	21.2	20.6 ± 0.4	●●	266.0	266.6 ± 7.1	●●
qualitative	20	32.0	31.9 ± 0.1	○⋆	576.0	546.9 ± 12.1	○○
	10	32.0	31.7 ± 0.1	○⋆	576.0	540.3 ± 12.2	●⋆
	5	30.6	30.5 ± 0.2	●⋆	576.0	521.2 ± 15.7	●⋆

		HXOR			MMDP		
strategy	k	$\tilde{x}$	$\bar{x} \pm \sigma_{\bar{x}}$		$\tilde{x}$	$\bar{x} \pm \sigma_{\bar{x}}$	
–	∞	408.0	418.9 ± 8.4		23.6	23.5 ± 0.1	
no balancing	20	372.0	383.4 ± 7.1	●●	23.3	22.9 ± 0.2	●●
	10	343.0	348.8 ± 6.4	●●	20.8	20.7 ± 0.2	●●
	5	263.0	267.4 ± 5.3	●●	17.5	17.4 ± 0.2	●●
quantitative	20	376.0	370.6 ± 5.8	●●	21.8	21.7 ± 0.2	●●
	10	317.0	319.7 ± 6.2	●●	19.8	19.9 ± 0.2	●●
	5	253.0	254.1 ± 4.3	●●	16.5	16.7 ± 0.2	●●
qualitative	20	400.0	407.3 ± 6.8	○⋆	23.6	23.5 ± 0.1	○⋆
	10	384.0	389.7 ± 7.1	●⋆	23.3	23.2 ± 0.2	○⋆
	5	371.0	373.2 ± 4.6	●⋆	22.8	22.0 ± 0.2	●⋆

Table 2: Results of Holm Test ($\alpha = 0.05$) using LBQ as the control algorithm.

i	strategy	z-statistic	p-value	α/i
1	noB	2.04124	0.02061	0.05
2	LB	4.08248	0.00002	0.025

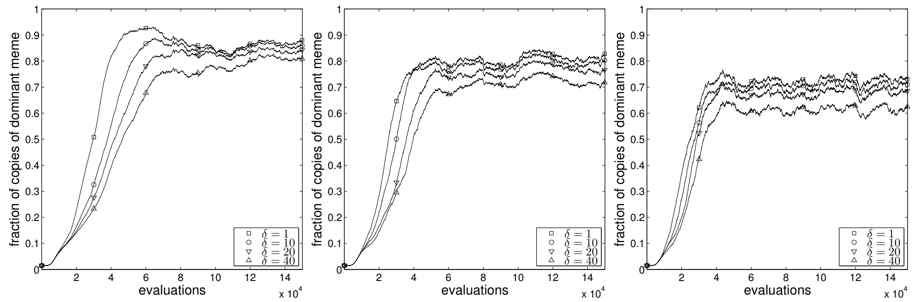

Figure 5: Fraction of copies of dominant meme in the selecto-Lamarckian model using the LBQ strategy with different values of the balancing threshold parameter δ (from left to right: $k = 20, 10$ and 5).

$p_r = 1/9$, $l_{\min} = 3$ and $l_{\max} = 9$ analogously to [32]. We also use crossover probability $p_X = 1.0$, and mutation probability $p_M = 1/\ell$, where ℓ is the genotype length. We have considered four test functions, namely Deb's trap (TRAP) function [43] (concatenating 32 four-bit traps), Watson et al.'s Hierarchical-if-and-only-if (HIFF) and Hierarchical-Exclusive-OR (HXOR) functions [44] (using 128 bits) and Goldberg et al.'s Massively Multimodal Deceptive Problem (MMDP) [45] (using 24 six-bit blocks) – see Appendix A for a description of these functions.

Table 1 shows the results. As seen, there is a marked performance degradation in both noB and LB for decreasing k (that is, increasing churn rates) – the differences with the faultless ($k = \infty$) scenario are statistically significant ($\alpha = 0.05$) in all cases except for the HIFF function with $k = 20$. However, LBQ is much more robust, with a much less noticeable performance loss for increasing volatility, in accordance with the behavior observed in the surrogate model. Indeed, the differences between the faultless algorithm and LBQ are not significant for $k = 20$ in either problem and in problems such as TRAP or MMDP they only become statistically significant in the most volatile scenario ($k = 5$). Another perspective on this is provided in Fig. 4 (for the HIFF function). Note how the convergence of LBQ is affected by increasing volatility in a much gentler way than noB and LB (the situation is analogous or even more marked in favor of LBQ in the other problems). Moreover, as shown in Table 1 the superiority of LBQ over noB and LB for a given value of k is almost always statistically significant. From a global point of view, if we consider the results of each strategy for each pair $\langle k$, problem$\rangle$ Quade test [46] indicates that at

Table 3: Results (25 runs) of the LBQ strategy for different values of the balancing threshold parameter δ on the four problems considered. The median ($\tilde{x}$), mean ($\bar{x}$) and standard error of the mean (σ_x) are indicated. The symbols $\bullet$ and $\circ$ indicate whether numerical differences are significant or not according to a Wilcoxon ranksum test ($\alpha = 0.05$) with respect to the best value of δ (marked with $\star$) for the corresponding problem and k. Results for $\delta = 1$ are taken from Table 1 and included for the convenience of the reader.

		TRAP			HIFF		
δ	k	$\tilde{x}$	$\bar{x} \pm \sigma_{\bar{x}}$		$\tilde{x}$	$\bar{x} \pm \sigma_{\bar{x}}$	
1	20	32.0	31.9 ± 0.1	$\star$	576.0	546.9 ± 12.1	$\circ$
	10	32.0	31.7 ± 0.1	$\star$	576.0	540.3 ± 12.2	$\circ$
	5	30.6	30.5 ± 0.2	$\star$	576.0	521.2 ± 15.7	$\circ$
10	20	32.0	31.8 ± 0.1	$\circ$	576.0	562.6 ± 7.4	$\star$
	10	31.2	31.2 ± 0.2	$\bullet$	576.0	537.8 ± 13.1	$\circ$
	5	29.8	29.8 ± 0.3	$\circ$	576.0	529.6 ± 15.0	$\star$
20	20	32.0	31.7 ± 0.1	$\circ$	576.0	562.3 ± 8.3	$\circ$
	10	31.2	30.8 ± 0.2	$\bullet$	576.0	535.6 ± 13.9	$\circ$
	5	28.8	28.2 ± 0.4	$\bullet$	576.0	502.5 ± 20.0	$\circ$
40	20	31.6	31.0 ± 0.2	$\bullet$	576.0	543.6 ± 13.7	$\bullet$
	10	29.0	28.8 ± 0.3	$\bullet$	576.0	550.2 ± 12.3	$\star$
	5	23.6	23.6 ± 0.4	$\bullet$	418.0	426.6 ± 23.5	$\bullet$

		HXOR			MMDP		
δ	k	$\tilde{x}$	$\bar{x} \pm \sigma_{\bar{x}}$		$\tilde{x}$	$\bar{x} \pm \sigma_{\bar{x}}$	
1	20	400.0	407.3 ± 6.8	$\circ$	23.6	23.5 ± 0.1	$\star$
	10	384.0	389.7 ± 7.1	$\star$	23.3	23.2 ± 0.1	$\star$
	5	371.0	373.2 ± 4.6	$\star$	22.8	22.0 ± 0.2	$\star$
10	20	403.0	411.4 ± 7.0	$\star$	23.3	23.0 ± 0.1	$\bullet$
	10	385.0	381.4 ± 4.7	$\circ$	22.2	22.4 ± 0.2	$\bullet$
	5	356.0	358.4 ± 3.8	$\bullet$	21.2	21.4 ± 0.2	$\bullet$
20	20	400.0	406.2 ± 6.3	$\circ$	23.3	22.9 ± 0.1	$\bullet$
	10	370.0	368.8 ± 4.6	$\bullet$	21.8	21.8 ± 0.2	$\bullet$
	5	329.0	329.2 ± 4.8	$\bullet$	20.0	19.9 ± 0.1	$\bullet$
40	20	383.0	384.7 ± 5.4	$\bullet$	22.2	22.3 ± 0.1	$\bullet$
	10	348.0	352.8 ± 5.4	$\bullet$	20.8	20.6 ± 0.2	$\bullet$
	5	286.0	285.0 ± 2.6	$\bullet$	17.8	17.9 ± 0.1	$\bullet$

least one of the strategies performs significantly differently (p-value $\simeq 0$) so we perform a post-hoc test [47], namely a Holm test [48] using LBQ as the control strategy. As illustrated in Table 2, both noB and LB pass the test thus confirming that LBQ performs significantly better than the other two strategies.

Let us take a closer look at LBQ and at the effect that the threshold parameter δ has on its behavior. To this end, we have repeated the experiments for this strategy using values $\delta \in \{10, 20, 40\}$. As shown in Fig. 5 for the selecto-Lamarckian model, by increasing the threshold δ, takeover is slower and the dominating meme stabilizes around a lower fraction of the population. This can be explained by the information spread pattern of LBQ. The information

Table 4: Results of Holm Test ($\alpha = 0.05$) using $\delta = 1$ as the control algorithm.

i	value of δ	z-statistic	p-value	α/i
1	10	0.79057	0.21460	0.05
2	20	3.00416	0.00133	0.025
3	40	4.42719	< 0.00001	0.017

diffusion process works in bursts triggered by the deactivation/reactivation of islands: a node going down makes neighboring islands enlarge, causing a flow of information in the opposite direction; to the contrary, a new node becoming available causes neighboring nodes to donate part of their populations to it, triggering in turn, a flow of information in its direction. Low values of the threshold parameter causes the effect of these bursts last longer and reversely, high values of δ introduce a damping effect in the propagation of balancing waves.

We have also conducted an analogous experimentation with the full MMA on the problems in the test bed. The results are given in Table 3. As seen, the results of LBQ markedly degrade for the largest values of the balancing threshold, in particular for the most volatile scenarios. While it is clear that by tuning this parameter the LBQ strategy will asymptotically reduce to noB, it is also interesting to note that for a moderately small value of δ, namely $\delta = 10$, the results are comparable to the lower limit $\delta = 1$. In fact, performing a multiple-comparison statistical test on the different values of δ indicates that no statistically significant difference can be established between $\delta = 1$ and $\delta = 10$ – see Table 4 (Quade test p-value $\simeq 0$). This suggests that the LBQ is somewhat robust to small variations of this parameter and that there may be room for fine-tuning it in specific situations.

4. Conclusions

This paper has focused on the study of self-balancing strategies in multimemetic algorithms. The deployment of these techniques on volatile environments such as those arising in peer-to-peer networks and volunteer computing networks requires them to be able to cope with the instability of computing nodes, being resilient to the churn phenomenon. In this sense, the use of self-balancing techniques is appealing since they provide a means for correcting (or at least alleviating) in a decentralized way, the perturbation exerted by the unstable environment on the search dynamics of the algorithm. The results obtained have been promising in this sense, since they suggest that a qualitative balancing strategy can provide resilience to the algorithm. Further research conducted on the balancing threshold parameter δ indicates that the performance of the qualitative balancing strategy degrades for large values of δ but is robust for values around the lower end of parameter values. Looking beyond, this opens up new avenues for developing balancing strategies, such as fine-tuning the balancing threshold, ideally adaptively. Current work is directed towards the use of dynamic topologies, re-wiring connections so as to keep all nodes with a certain minimum number of active neighbors at all times, as well as using other network topologies. Also of interest for future developments is the use of detached models in which memes and genes co-evolve in separate populations [49].

Acknowledgements

This work is partially funded by the MINECO project ANYSELF (TIN2011-28627-C04-01) and EphemeCH (TIN2014-56494-C4-1-P), by Junta de Andalucía project DNEMESIS (P10-TIC-6083) and by Universidad de Málaga, Campus de Excelencia Internacional Andalucía Tech.

References

[1] F. Neri, C. Cotta, P. Moscato, Handbook of Memetic Algorithms, Vol. 379 of Studies in Computational Intelligence, Springer, Berlin Heidelberg, 2012.

[2] P. Moscato, On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms, Tech. Rep. Caltech Concurrent Computation Program, Report. 826, California Institute of Technology, Pasadena, California, USA (1989).

[3] R. Dawkins, The Selfish Gene, Clarendon Press, Oxford, 1976.

[4] P. Moscato, Memetic algorithms: A short introduction, in: D. Corne, M. Dorigo, F. Glover (Eds.), New Ideas in Optimization. Mcgraw-Hill's Advanced Topics In Computer Science Series, McGraw-Hill, London UK, 1999, pp. 219–234.

[5] Y.-S. Ong, M.-H. Lim, X. Chen, Memetic computation-past, present and future, IEEE Computational Intelligence Magazine 5 (2) (2010) 24–31.

[6] X. Chen, Y.-S. Ong, M.-H. Lim, K. C. Tan, A multi-facet survey on memetic computation, IEEE Transactions on Evolutionary Computation 15 (5) (2011) 591–607.

[7] X. Chen, Y.-S. Ong, A conceptual modeling of meme complexes in stochastic search, IEEE Transactions on Systems, Man, and Cybernetics, Part C 42 (5) (2012) 612–625.

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

[9] N. Krasnogor, B. Blackburne, E. Burke, J. Hirst, Multimeme algorithms for protein structure prediction, in: J. Merelo, et al. (Eds.), Parallel Problem Solving From Nature VII, Vol. 2439 of Lecture Notes in Computer Science, Springer, Berlin, 2002, pp. 769–778.

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

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

[12] F. Neri, V. Tirronen, T. Kärkkäinen, T. Rossi, Fitness diversity based adaptation in multimeme algorithms: A comparative study, in: IEEE Congress on Evolutionary Computation – CEC 2007, IEEE Press, Singapore, 2007, pp. 2374–2381. doi:10.1109/CEC.2007.4424768.

[13] J. E. Smith, Self-adaptative and coevolving memetic algorithms, in: F. Neri, C. Cotta, P. Moscato (Eds.), Handbook of Memetic Algorithms, Vol. 379 of Studies in Computational Intelligence, Springer-Verlag, Berlin Heidelberg, 2012, pp. 167–188. doi:10.1007/978-3-642-23247-3_11.

[14] R. Hinterding, Z. Michalewicz, A. E. Eiben, Adaptation in evolutionary computation: A survey, in: Fourth International Conference on Evolutionary Computation – ICEC 97, IEEE Press, 1997, pp. 65–69.

[15] W. Jakob, Towards an adaptive multimeme algorithm for parameter optimisation suiting the engineers' needs, in: T. P. Runarsson, et al. (Eds.), Parallel Problem Solving from Nature - PPSN IX, Vol. 4193 of Lecture Notes in Computer Science, Springer-Verlag, Berlin Heidelberg, 2006, pp. 132–141. doi:10.1007/11844297_14.

[16] A. Berns, S. Ghosh, Dissecting self-⋆ properties, in: Third IEEE International Conference on Self-Adaptive and Self-Organizing Systems, IEEE Press, San Francisco CA, 2009, pp. 10–19. doi:10.1109/SASO.2009.25.

[17] Y.-S. Ong, M.-H. Lim, N. Zhu, K.-W. Wong, Classification of adaptive memetic algorithms: a comparative study, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 36 (1) (2006) 141–152. doi:10.1109/TSMCB.2005.856143.

[18] R. Nogueras, C. Cotta, Analyzing meme propagation in multimemetic algorithms: Initial investigations, in: 2013 Federated Conference on Computer Science and Information Systems, IEEE Press, Cracow (Poland), 2013, pp. 1013–1019.

[19] R. Nogueras, C. Cotta, An analysis of migration strategies in island-based multimemetic algorithms, in: T. Bartz-Beielstein, J. Branke, B. Filipič, J. Smith (Eds.), Parallel Problem Solving From Nature – PPSN XIII, Vol. 8672 of Lecture Notes in Computer Science, Springer, Berlin Heidelberg, 2014, pp. 731–740.

[20] D. S. Milojičić, V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, B. Richard, S. Rollins, Z. Xu, Peer-to-peer computing, Tech. Rep. HPL-2002-57, Hewlett-Packard Labs (2002).

[21] L. F. Sarmenta, Bayanihan: Web-based volunteer computing using java, in: Y. Masunaga, T. Katayama, M. Tsukamoto (Eds.), Worldwide Computing and Its Applications – WWCA'98, Vol. 1368 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 1998, pp. 444–461. doi:10.1007/3-540-64216-1_67.

[22] J. I. Hidalgo, J. Lanchares, F. Fernández de Vega, D. Lombraña, Is the island model fault tolerant?, in: Proceedings of the 9th Annual Conference Companion on Genetic and Evolutionary Computation, GECCO '07, ACM, New York, NY, USA, 2007, pp. 2737–2744. doi:10.1145/1274000.1274085.

[23] D. Lombraña González, J. L. Jiménez Laredo, F. Fernández de Vega, J. J. Merelo Guervós, Characterizing fault-tolerance of genetic algorithms in desktop grid systems, in: P. Cowling, P. Merz (Eds.), Evolutionary Computation in Combinatorial Optimization, Vol. 6022 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2010, pp. 131–142.

[24] D. Lombraña González, J. L. J. Laredo, F. F. de Vega, J. J. M. Guervós, Characterizing fault-tolerance in evolutionary algorithms, in: F. Fernández de Vega, J. I. H. Pérez, J. Lanchares (Eds.), Parallel Architectures and Bioinspired Algorithms, Vol. 415 of Studies in Computational Intelligence, Springer, 2012, pp. 77–99.

[25] Y. Sato, M. Sato, Parallelization and fault-tolerance of evolutionary computation on many-core processors, in: IEEE Congress on Evolutionary Computation, IEEE, 2013, pp. 2602–2609.

[26] J. L. Jiménez Laredo, P. Bouvry, D. Lombraña González, F. Fernández de Vega, M. García Arenas, J. J. Merelo Guervós, C. M. Fernandes, Designing robust volunteer-based evolutionary algorithms, Genetic Programming and Evolvable Machines 15 (3) (2014) 221–244. doi:10.1007/s10710-014-9213-5.

[27] R. Nogueras, C. Cotta, Studying fault-tolerance in island-based evolutionary and multimemetic algorithms, Journal of Grid Computing (2015). doi:10.1007/s10723-014-9315-6.

[28] J. E. Smith, Meme fitness and memepool sizes in coevolutionary memetic algorithms, in: 2010 IEEE Congress on Evolutionary Computation, IEEE Press, Barcelona, Spain, 2010, pp. 1–8. doi:10.1109/CEC.2010.5586401.

[29] M. Tomassini, Spatially Structured Evolutionary Algorithms, Natural Computing Series, Springer-Verlag, 2005.

[30] E. Cantu-Paz, Efficient and Accurate Parallel Genetic Algorithms, Kluwer Academic Publishers, Norwell, MA, USA, 2000.

[31] R. Schaefer, A. Byrski, M. Smołka, The island model as a Markov dynamic system, International Journal of Applied Mathematics and Computer Science 22 (4) (2012) 971–984.

[32] R. Nogueras, C. Cotta, On meme self-adaptation in spatially-structured multimemetic algorithms, in: I. Dimov, S. Fidanova, I. Lirkov (Eds.), Numerical Methods and Applications, Vol. 8962 of Lecture Notes in Computer Science, Springer, Berlin-Heidelberg, 2015, pp. 70–77.

[33] J. E. Smith, Self-adaptation in evolutionary algorithms for combinatorial optimisation, in: C. Cotta, M. Sevaux, K. Sörensen (Eds.), Adaptive and Multilevel Metaheuristics, Vol. 136 of Studies in Computational Intelligence, Springer Berlin Heidelberg, 2008, pp. 31–57. doi:10.1007/978-3-540-79438-7_2.

[34] R. Albert, A.-L. Barabási, Statistical mechanics of complex networks, Review of Modern Physics 74 (1) (2002) 47–97. doi:10.1103/RevModPhys.74.47.

[35] A.-L. Barabási, R. Albert, Emergence of Scaling in Random Networks, Science 286 (5439) (1999) 509–512.

[36] W. Weibull, A statistical distribution function of wide applicability, Journal of Applied Mechanics 18 (3) (1951) 293–297.

[37] E. T. Lee, J. W. Wang, Statistical Methods for Survival Data Analysis, John Wiley & Sons, Inc., Hoboken, NJ, 2003.

[38] C. Liu, R. W. White, S. Dumais, Understanding web browsing behaviors through weibull analysis of dwell time, in: Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '10, ACM, New York, NY, USA, 2010, pp. 379–386. doi:10.1145/1835449.1835513.

[39] D. Stutzbach, R. Rejaie, Understanding churn in peer-to-peer networks, in: Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement, IMC '06, ACM, New York, NY, USA, 2006, pp. 189–202.

[40] R. Lüling, B. Monien, F. Ramme, Load balancing in large networks: a comparative study, in: Third IEEE Symposium on Parallel and Distributed Processing, 1991, IEEE, 1991, pp. 686–689. doi:10.1109/SPDP.1991.218196.

[41] F. Zambonelli, Exploiting biased load information in direct-neighbour load balancing policies, Parallel Computing 25 (6) (1999) 745 – 766. doi:http://dx.doi.org/10.1016/S0167-8191(99)00030-7.

[42] J. Grefenstette, Genetic algorithms for changing environments, in: R. Männer, B. Manderick (Eds.), Parallel Problem Solving from Nature II, Elsevier, Brussels, Belgium, 1992, pp. 137–144.

[43] K. Deb, D. E. Goldberg, Analyzing deception in trap functions., in: L. D. Whitley (Ed.), Second Workshop on Foundations of Genetic Algorithms, Morgan Kaufmann, Vail, Colorado, USA, 1993, pp. 93–108.

[44] R. A. Watson, G. S. Hornby, J. B. Pollack, Modeling building-block interdependency, in: A. Eiben, T. Bck, M. Schoenauer, H.-P. Schwefel (Eds.), Parallel Problem Solving from Nature – PPSN V, Vol. 1498 of Lecture Notes in Computer Science, Springer-Verlag, Berlin Heidelberg, 1998, pp. 97–106. doi:10.1007/BFb0056853.

[45] D. E. Goldberg, K. Deb, J. Horn, Massive multimodality, deception, and genetic algorithms, in: Parallel Problem Solving from Nature – PPSN II, Elsevier, Brussels, Belgium, 1992, pp. 37–48.

[46] D. Quade, Using weighted rankings in the analysis of complete blocks with additive block effects, Journal of the American Statistical Association 74 (1979) 680–683.

[47] J. Derrac, S. García, D. Molina, F. Herrera, A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms, Swarm and Evolutionary Computation 1 (1) (2011) 3–18.

[48] S. Holm, A simple sequentially rejective multiple test procedure, Scandinavian Journal of Statistics 6 (1979) 65–70.

[49] J. E. Smith, Coevolving memetic algorithms: A review and progress report, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 37 (1) (2007) 6–17. doi:10.1109/TSMCB.2006.883273.

Appendix A. Description of the Test Suite

Deb's 4-bit fully deceptive function (TRAP) has a single global optimum surrounded by low-fitness solutions and a local optimum surrounded by increasingly good solutions. Hence, gradient-based methods are deceived to follow the path towards this local optimum. In mathematical terms, TRAP is defined as:

$$f(b_1 \cdots b_4) = \begin{cases} 0.6 - 0.2 \cdot u(b_1 \cdots b_4) & \text{if } u(b_1 \cdots b_4) < 4 \\ 1 & \text{if } u(b_1 \cdots b_4) = 4 \end{cases} \tag{A.1}$$

where $u(s_1 \cdots s_i) = \sum_j s_j$ is the unitation (number of 1s) of the binary string. A higher-order problem is built by concatenating k 4-bits blocks, and defining the fitness of this $4k$-bit string as the sum of the function value for all blocks/subproblems. In our experiments we have considered $k = 32$ subproblems (and hence $opt = 32$).

As to the hierarchically consistent test problems, these are recursive epistatic functions defined for 2^k-bit strings. They use two auxiliary functions, namely $f : \{0, 1, \times\} \to \{0, 1\}$ and $t : \{0, 1, \times\} \to \{0, 1, \bullet\}$, the first one being used to score the contribution of building blocks, and the second one to capture their interaction. In the case of the Hierarchical if-and-only-if (HIFF) function f and t are defined as:

$$f(a,b) = \begin{cases} 1 & a = b \neq \bullet \\ 0 & \text{otherwise} \end{cases} \tag{A.2}$$

$$t(a,b) = \begin{cases} a & a = b \\ \bullet & \text{otherwise} \end{cases} \tag{A.3}$$

These two functions are used as follows:

$$\text{HIFF}_k(b_1 \cdots b_n) = \sum_{i=1}^{n/2} f(b_{2i-1}, b_{2i}) + 2 \cdot \text{HIFF}_{k-1}(b'_1, \cdots, b'_{n/2}) \tag{A.4}$$

where $b'_i = t(b_{2i-1}, b_{2i})$ and $\text{HIFF}_0(\cdot) = 1$. The Hierarchical-XOR (HXOR) works similarly but changing f so as to provide a fitness contribution of 1 when $a = 1$ and $b = 0$ or vice versa, and having in that case $t(a,b) = a$ (and $t(a,b) = \bullet$ otherwise). We have considered $k = 7$ (i.e., 128-bit strings, $opt = 576$).

Finally, the Massively Multimodal Deceptive Problem (MMDP) is a bipolar deceptive function with two global optima located at extreme unitation values (and hence far apart from each other), and with a local deceptive attractor halfway between them. This location of the deceptive attractor results in massively more local optima than global optima (i.e., $\binom{L}{L/2}$ local vs 2 global, where L is the number of bits in each block). The basic MMDP is defined for 6-bit blocks as follows:

$$f(b_1 \cdots b_6) = \begin{cases} 1 & u(b_1 \cdots b_6) \in \{0, 6\} \\ 0 & u(b_1 \cdots b_6) \in \{1, 5\} \\ 0.360384 & u(b_1 \cdots b_6) \in \{2, 4\} \\ 0.640576 & u(b_1 \cdots b_6) = 3 \end{cases} \tag{A.5}$$

We concatenate k copies of this basic block to create a harder problem. We have considered $k = 24$ (thus, $opt = 24$).