A Study of the Performance of Self-$\star$ Memetic Algorithms on Heterogeneous Ephemeral Environments

Rafael Nogueras and Carlos Cotta

Dept. Lenguajes y Ciencias de la Computación, Universidad de Málaga,
ETSI Informática, Campus de Teatinos, 29071 Málaga, Spain
ccottap@lcc.uma.es

Abstract. We consider the deployment of island-based memetic algorithms (MAs) endowed with self-$\star$ properties on unstable computational environments composed of a collection of computing nodes whose availability fluctuates. In this context, these properties refer to the ability of the MA to work autonomously in order to optimize its performance and to react to the instability of computational resources. The main focus of this work is analyzing the performance of such 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. We use for this purpose a simulated environment subject to different volatility rates, whose topology is modeled as scale-free networks and whose computing power is distributed among nodes following different distributions. We observe that in general computational homogeneity is preferable in scenarios with low instability; in case of high instability, MAs without self-scaling and self-healing perform better when the computational power follows a power law, but performance seems to be less sensitive to the distribution when these self-$\star$ properties are used.

1 Introduction

Population-based optimization algorithms are very well suited to parallel environments thanks to their flexibility and decentralized nature. This has been known and exploited since the late 80s. In contrast to the dedicated networks of computational resources that were typical in the past, recent years have witnessed the emergence of other kind of environments of a much more dynamic and unsteady nature though. This is the case of peer-to-peer (P2P) networks and volunteer computing networks, composed of volatile nodes whose availability usually responds to uncontrollable external factors. Such environments are particularly interesting in light of the increasingly pervasive abundance of computational devices which are permanently networked (think for example of smartphones, wearables, and any other kind of handheld devices) and whose computing power is often unused or at least under-exploited [6]. Capitalizing on such power can be a practical solution for solving many complex computational tasks but tackling the underlying dynamic computational landscape is not exempt of difficulties.

Of course, intermediate layers can be constructed to hide the transient nature of computational nodes but it is not easy to have such an abstract layer making effective use of brief, ephemeral bursts of computing availability. While this direction is in any case interesting and valid, we consider here a much more direct approach in which the optimization algorithm is cognizant of the volatile environment.

Algorithms consciously running on computational environments with the features mentioned above must be resilient in order to withstand sudden node failures. In the case of evolutionary algorithms this resilience is partly provided by their inherent features [13, 15], and can be further boosted by exploiting their capacity for adaptiveness and self-control [8,11]. This latter feature is essential to have the algorithm readjusting its behavior in response (or even in anticipation) to the fluctuations of the environment. Indeed, much work has been done in the area of self-adaptation in evolutionary algorithms, e.g., [5, 24, 25] in general, and in connection with unstable environments in particular. In this work we build on previous research [19–22] in order to tackle the potential heterogeneity of the environment [1] 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. The underlying rationale for examining this matter lies in the potentially different impact than the failure of a node can have on the system as a whole depending on its computational power, and determining whether or in which conditions the system is sensitive to this environmental heterogeneity. To this end we consider island-based memetic algorithms (MAs) endowed with self-$\star$ properties [2] and use a simulated computational environment that allows experimenting with different scenarios both in terms of the volatility of computing nodes and the distribution of computing power of constituent nodes. A broad experimentation is done to assess the performance of the MA in these different scenarios.

2 Materials and Methods

2.1 Basic Algorithmic Setting

As stated in the introduction, the basic algorithm considered is an island-based MA. Let there be n_ι panmictic islands, each of them running a simple MA (using tournament selection, one-point crossover, bit-flip mutation, and replacement of the worst parent) on a different computing node. Such nodes are interconnected among them according to a certain topology $\mathcal{N}$ – see Sect. 2.3. In addition to standard selection, variation and local improvement, each island perform asynchronous migration: at the beginning of each cycle the island checks if migrants were received from any neighboring nodes and are stored in the input buffer. Were this the case, they would be inserted in the population 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 and sent to the neighbors. Following previous analysis of migration strategies in island-based MAs [18], we use random selection of migrants and deterministic replacement of the worst individuals in the receiving island.

2.2 Self-⋆ Properties

Self-⋆ properties [2] are those that enable a computational system to exert advanced control on its own functioning and/or structure. This goes beyond parameter control (its practical importance notwithstanding) and encompasses advanced capabilities 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-⋆ properties with which the MA considered is endowed.

Self-Generation. According to [4], 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. In the case of bioinspired optimization algorithms, this objective function does not directly refer to the fitness function to be optimized, but to the capability of the algorithm to optimize the latter. Such capability can be improved for example by tuning some parameters (self-parameterization) or even by adjusting qualitatively the way the search is done (self-generation). The latter approach amounts to have the algorithm adjusting the search strategy during runtime [12], and is related to the notion of memetic computing [23] whereby memes (understood as representations of problem solving strategies) are explicitly represented and evolved [17].

In the MA considered we follow the model by Smith [24] in which memes are attached to individuals and evolve alongside them. More precisely, these memes take the form of pattern-based rewriting rules $A \rightarrow B$, where A, B are variable-length strings taken from the same alphabet used to encode solutions plus a wildcard symbol. The action of the meme is finding an occurrence of pattern A in the solution and changing it by pattern B if it leads to a fitness improvement (otherwise the solution is left unchanged). Self-generation is attained due to the fact that memes are subject to mutation and are transferred from parent to offspring via local selection (offspring inherit the meme of the best parent).

Self-Scaling. This property involves the ability of the system to react efficiently to changes in its scale parameters, that is, changing its size or its structure in response to modifications in the size of the problem being solved, in the amount of computational resources available, or in any other circumstance of the computation, e.g., [9,28]. In this case, the main factor to be taken into account is the volatility of the environment that results in certain islands getting lost when the supporting node goes down. This implies that the overall size of the system will fluctuate, affecting genetic diversity and resulting in the loss of information. To cope with this, a self-balancing policy has been proposed [22]. This strategy is aimed to resize dynamically islands so that some of them increase their sizes when they detect a neighboring island has gone down, and analogously decrease their sizes when a new neighbor appears. This is done by communicating with neighbors at the beginning of each evolutionary cycle exchanging information on the size of their populations and the number of active neighbors, and performing a local balancing procedure [27] by transferring individuals. New islands can absorb this way a part of the existing population in neighboring nodes and, likewise, nodes detecting that a previously active neighbor is no longer available try to compensate this loss by increasing their own population sizes (using the recorded information on the size and number of active neighbors the lost island had in order to calculate the required increase). While 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. In the context of evolutionary algorithms this property is not new since the use of ad-hoc procedures for repairing infeasible solutions produced by variation operators in constrained problems [16] can be regarded as a simple form of self-healing. More particularly for the case of ephemeral computational environments, the volatility of the system can be the source of at least two issues the algorithm needs to deal with: (i) node failures disrupt the connectivity of the network, limiting the flow of information and hindering the progress of the search, and (ii) forcing an island to increase its size can perturb the convergence of the search if the new information is simply random. To tackle the first issue, a self-rewiring strategy [19] is used: whenever an island detects that its number of active neighbors has fallen below a predefined threshold, it looks for additional neighbors to reach this minimum level, hence aiming to maintain a rich connectivity at all times. As to the second issue, it is dealt with by means of self-sampling [20], that is, the island keeps a probabilistic model of the current population and samples it (much like it is done in EDAs) when new individuals are required. This way, the latter are representative of the current state of the population. We consider here a tree-based bivariate probabilistic model.

2.3 Environmental Model

This island-based model runs on a simulated distributed system composed of n_ι nodes. These nodes are interconnected following a scale-free topology. This connectivity pattern is commonly observed in many real-world systems, particularly in P2P systems. To generate this topology we consider the Barabási-Albert (BA) model [3]: starting with a clique of $m+1$ nodes (m being a parameter of the model), new nodes are added one at a time, selecting for each of them m neighbors among previous nodes; neighbor selection is driven by preferential attachment, whereby the probability of picking a certain node is proportional to the number of neighbors it already has.

To model the volatility of the nodes, we consider that failures/recoveries are Weibull distributed [14]. This distribution is described by a shape parameter η

and a scale parameter β: the probability of a node being available up to time t is $p(t, \eta, \beta) = \exp(-(t/\beta)^{\eta})$. Thus, for shape parameters larger than 1 (as we use in the experiments), failure/recovery probabilities increase with time.

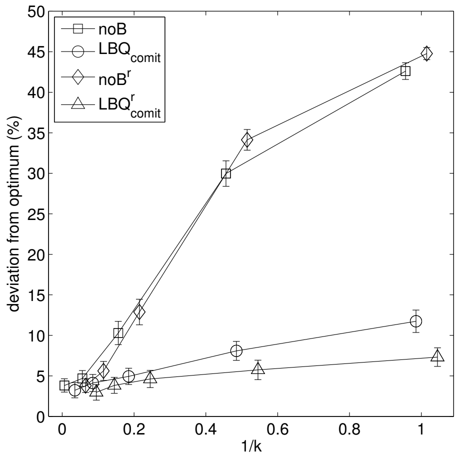
Network heterogeneity is modeled by assuming each node i has a certain computing power $w_i \in \mathbb{N}^+$. These coefficients represent for simplicity a relative performance index and hence each node's power can be understood to be proportional to its coefficient. From the point of view of the MA, this computational power determines the number of evolutionary cycles (and hence the number of fitness function evaluations) each node can perform per unit of time. We have considered several scenarios regarding the distribution of values for these coefficients but in all cases, 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 $\lfloor W/n_\iota \rfloor \leqslant w_i \leqslant \lceil W/n_\iota \rceil$.
- random: each coefficient w_i can have a uniformly random value in $\{1, \ldots, W - n_\iota + 1\}$, 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 $W - n_\iota$ additional units among nodes according to these values.
- binomial: coefficients can take values in $\{1, \ldots, W - n_\iota + 1\}$ and the probability of a certain value w is $p(w) = C(W - n_\iota, w - 1)q^{w-1}(1 - q)^{W - n_\iota - w + 1}$ where $q = 1/n_\iota$. As in the previous case, the boundary conditions ensure that each node has at least unit power.
- power law: coefficients are grouped in r levels, where $r \in \{0, \ldots, r_{\max}\}$ with $r_{\max} = \lfloor \log_2 n_\iota \rfloor - 1$, so that there is a single node with power $\lfloor n_\iota/2 \rfloor$ in the highest level, and in subsequent levels there are twice as many nodes, each with half as much power as in the previous upper level (although depending on the value of n_ι the lowest level can have additional nodes if these are not enough to create a new level).

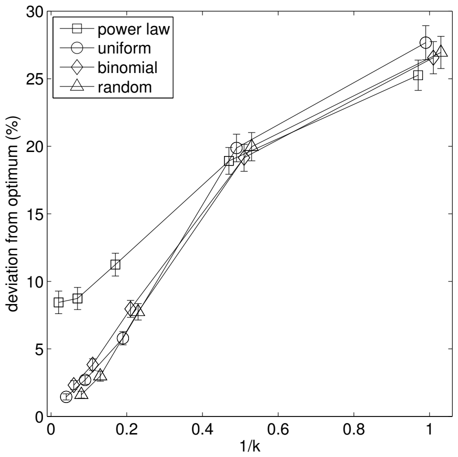
The value W implied for the last configuration (power law) is used for the remaining distributions. Notice that depending on the configuration a node failure will have a different impact on the overall capacity of the system. For example, under the power law distribution around half of node failures will have a low impact in this overall capacity 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. Next section describes the experimentation conducted with these distributions to determine more quantitatively the effect they exert on performance.

3 Experimental Analysis

We consider $n_\iota = 32$ islands whose initial size is $\mu = 16$ individuals and a total number of evaluations $maxevals = 50000$. Meme lengths evolve within $l_{\min} = 3$ and $l_{\max} = 9$, mutating their length with probability $p_r = 1/9$. We



(a)



(b)

Fig. 1: Average deviation from the optimal solution across all problems. (a) According to algorithmic variant. (b) According to power distribution.

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

i	strategy	z-statistic	p-value	α/i
1	LBQ$_{\text{comit}}$	2.934e+00	1.670e−03	5.000e−02
2	noB	6.293e+00	1.554e−10	2.500e−02
3	noB^r	9.298e+00	7.125e−21	1.667e−02

use crossover probability $p_X = 1.0$, mutation probability $p_M = 1/\ell$, where ℓ is the genotype length, and migration probability $p_{mig} = 1/80$. Regarding network topology, we use $m = 2$ in the Barabási-Albert model. This model is also used for self-rewiring when a node 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 $p = 1 - (kn_\iota)^{-1}$, $k \in \{1, 2, 5, 10, 20\}$. These parameters can be interpreted as corresponding to an average of one island going down/up every k cycles if the failure rate was constant (it is not since $\eta > 1$ but this serves as a first approximation). This provides different scenarios ranging from low volatility ($k = 20$) to very high volatility ($k = 1$). We perform 25 simulations for each algorithm and volatility scenario. We consider four algorithmic variants (in parentheses the self-$\star$ properties involved – all variants use self-generation): LBQ$^r_{\text{comit}}$ (self-rewiring, self-sampling, self-scaling), 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 [7] (concatenating 32 four-bit traps), Watson et al.'s Hierarchical-if-and-only-if function [26] (using 128 bits) and Goldberg et al.'s Massively Multimodal Deceptive Problem [10] (using 24 six-bit blocks).

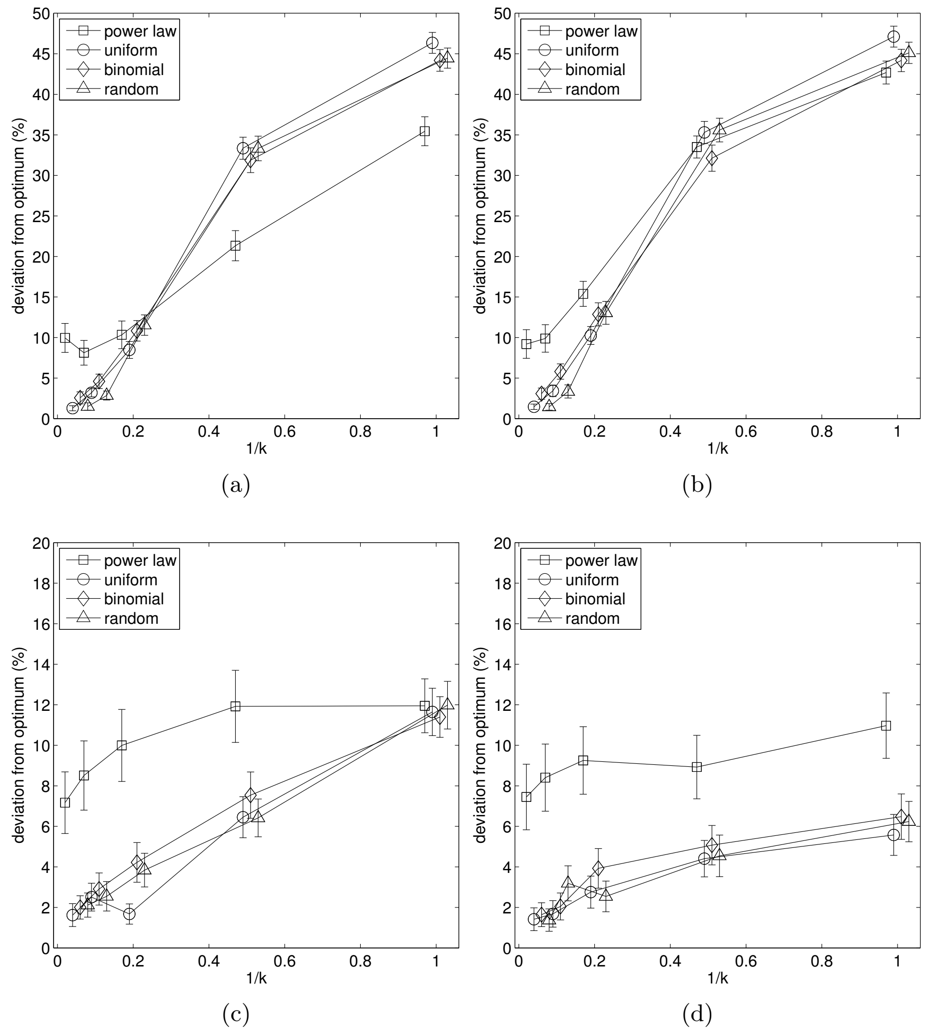


Fig. 2: Average deviation from the optimal solution across all problems and power distributions for each algorithmic variant. (a) noB (b) noB^r (c) LBQ$_{\text{comit}}$ (d) LBQ$^r_{\text{comit}}$.

Fig. 1a shows the average deviation from the optimal for each of the four algorithmic variants across all problems and power distributions. The variants with self-scaling and self-sampling outperform variants without them, even in the presence of self-rewiring. This confirms the robustness of the former across different the scenarios considered. In fact, 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 a advantageous option. If we now turn our attention to the overall results obtained under each different configuration by all four variants, the differences are not always large (Fig. 1b), although this is understandable in light of the great performance diversity of the algorithms involved that diminishes and conceals dissimilitudes among configurations. For this reason, it is more convenient to factorize the analysis and observe the impact that configurations have on each algorithm separately. This is shown in Fig. 2.

Comparing the behavior of the different algorithms, there is a common feature: in scenarios of low to moderate volatility ($k \geqslant 5$) the uniform distribution (corresponding to near-homogeneous nodes) provides better results. The differences are not always significant considering individual algorithms, but the trend is clear and yields a global significant difference (Quade test p-value $\approx 7.022\text{e}{-4}$, Holm test passed for all distributions except random at $\alpha = 0.05$ and for the latter as well at $\alpha = 0.1$). We interpret this result as indicating that in scenarios in which the environmental instability is not strong enough to pose a great handicap to the search process, device homogeneity contributes to balance the search, having all islands progressing at about the same rate. This is further vindicated by the comparatively worse results of the power law distribution in this range, suggesting that the decompensation of computational power (and thus the unbalance of search progress among islands) is not a cost-effective solution. It is also interesting to note the different behavior of noB and $\text{LBQ}^r_{\text{comit}}$ on the other part of the spectrum, that is, for moderate to high volatility ($k \leqslant 5$). In this case, it seems that noB benefits from heterogeneity. This can be due to the fact that having nodes with high computational power can push forward the search significantly during their short availability stint in a much more cost-effective way than less powerful nodes. The situation is different in the presence of self-healing and/or self-scaling, particularly for $\text{LBQ}^r_{\text{comit}}$. These self-$\star$ properties help to absorb the impact of the environment instability and hence $\text{LBQ}^r_{\text{comit}}$ is not in a so-markedly different scenario as before. In fact, power law is significantly worse than uniform for $\text{LBQ}^r_{\text{comit}}$ globally considering all values of k (although the performance is in that case still superior to the remaining algorithms).

4 Conclusions

Deploying population-based optimization algorithms on unstable environments require resilience to deal with the fluctuating computational landscape. Such fluctuations respond to the volatility of computing nodes but can also encompass the heterogeneity of the system and the corresponding variations in the computational power of nodes available at a certain moment. In this sense, endowing MAs with self-$\star$ properties has been shown as an effective solution. The combined use of self-scaling and self-healing seems robust under different configurations of the system, even in scenarios with extreme heterogeneity in which its performance is comparatively less favorable. 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.

Acknowledgements. We acknowledge support from Spanish MinEco and FEDER under project EphemeCH (TIN2014-56494-C4-1-P), from Junta de Andalucía under project DNEMESIS (P10-TIC-6083), and from Universidad de Málaga, Campus de Excelencia Internacional Andalucía Tech.

References

1. Anderson, D.P., Reed, K.: Celebrating diversity in volunteer computing. In: Proceedings of the 42nd Hawaii International Conference on System Sciences. pp. 1–8. HICSS '09, IEEE Computer Society, Washington, DC, USA (2009)
2. Babaoğlu, Ö., Jelasity, M., Montresor, A., Fetzer, C., Leonardi, S., van Moorsel, A., van Steen, M. (eds.): Self-star Properties in Complex Information Systems, Lecture Notes in Computer Science, vol. 3460. Springer-Verlag, Berlin Heidelberg (2005)
3. Barabási, A.L., Albert, R.: Emergence of scaling in random networks. Science 286(5439), 509–512 (1999)
4. Berns, A., Ghosh, S.: Dissecting self-⋆ properties. In: Third IEEE International Conference on Self-Adaptive and Self-Organizing Systems - SASO 2009. pp. 10–19. IEEE Press, San Francisco, CA (2009)
5. Caraffini, F., Neri, F., Picinali, L.: An analysis on separability for memetic computing automatic design. Information Sciences 265, 1–22 (2014)
6. Cotta, C., Fernández-Leiva, A., de Vega, F.F., Chávez, F., Merelo, J., Castillo, P., Bello, G., Camacho, D.: Ephemeral computing and bioinspired optimization - challenges and opportunities. In: 7th International Joint Conference on Evolutionary Computation Theory and Applications. pp. 319–324. Lisboa, Portugal (2015)
7. Deb, K., Goldberg, D.: Analyzing deception in trap functions. In: Whitley, L. (ed.) Second Workshop on Foundations of Genetic Algorithms. pp. 93–108. Morgan Kaufmann Publishers, Vail, Colorado, USA (1993)
8. Eiben, A.E.: 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, vol. 3460, pp. 36–48. Springer (2005)
9. Fernández, F., Vanneschi, L., Tomassini, M.: 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, vol. 2610, pp. 317–326. Springer-Verlag, Berlin Heidelberg (2003)
10. Goldberg, D., Deb, K., Horn, J.: Massive multimodality, deception and genetic algorithms. In: Männer, R., Manderick, B. (eds.) Parallel Problem Solving from Nature - PPSN II. pp. 37–48. Elsevier Science Inc., New York, NY, USA (1992)
11. Hinterding, R., Michalewicz, Z., Eiben, A.: Adaptation in evolutionary computation: A survey. In: Fourth IEEE Conference on Evolutionary Computation. pp. 65–69. IEEE Press, Piscataway, New Jersey (1997)
12. Krasnogor, N., Gustafson, S.: A study on the use of "self-generation" in memetic algorithms. Natural Computing 3(1), 53–76 (2004)
13. Laredo, J., Castillo, P., Mora, A., Merelo, J., Fernandes, C.: Resilience to churn of a peer-to-peer evolutionary algorithm. International Journal of High Performance Systems Architecture 1(4), 260–268 (2008)
14. Liu, C., White, R., Dumais, S.: 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. pp. 379–386. ACM Press, New York, NY, USA (2010)
15. Lombraña González, D., Jiménez Laredo, J., Fernández de Vega, F., Guervós, J.M.: Characterizing fault-tolerance in evolutionary algorithms. In: Fernández de Vega, F., et al. (eds.) Parallel Architectures and Bioinspired Algorithms, Studies in Computational Intelligence, vol. 415, pp. 77–99. Springer-Verlag, Berlin Heidelberg (2012)
16. Michalewicz, Z.: Repair algorithms. In: Bäck, T., et al. (eds.) Handbook of Evolutionary Computation, pp. C5.4:1–5. Institute of Physics Publishing and Oxford University Press, Bristol, New York (1997)
17. Neri, F., Cotta, C.: Memetic algorithms and memetic computing optimization: A literature review. Swarm and Evolutionary Computation 2, 1–14 (2012)
18. Nogueras, R., Cotta, C.: An analysis of migration strategies in island-based multimemetic algorithms. In: Bartz-Beielstein, T., et al. (eds.) Parallel Problem Solving from Nature - PPSN XIII. Lecture Notes in Computer Science, vol. 8672, pp. 731–740. Springer Verlag, Berlin Heidelberg (2014)
19. Nogueras, R., Cotta, C.: Self-balancing multimemetic algorithms in dynamic scale-free networks. In: Mora, A., Squillero, G. (eds.) Applications of Evolutionary Computing. Lecture Notes in Computer Science, vol. 9028, pp. 177–188. Springer Verlag, Berlin Heidelberg (2015)
20. Nogueras, R., Cotta, C.: Self-sampling strategies for multimemetic algorithms in unstable computational environments. In: Ferrández Vicente, J., et al. (eds.) Bioinspired Computation in Artificial Systems. Lecture Notes in Computer Science, vol. 9108, pp. 69–78. Springer Verlag, Berlin Heidelberg (2015)
21. Nogueras, R., Cotta, C.: Self-healing strategies for memetic algorithms in unstable and ephemeral computational environments. Natural Computing [In press] (2016)
22. Nogueras, R., Cotta, C.: Studying self-balancing strategies in island-based multimemetic algorithms. Journal of Computational and Applied Mathematics 293, 180–191 (2016)
23. Ong, Y., Lim, M., Chen, X.: Memetic computation –past, present and future. IEEE Computational Intelligence Magazine 5(2), 24–31 (2010)
24. Smith, J.E.: Self-adaptation in evolutionary algorithms for combinatorial optimisation. In: Cotta, C., Sevaux, M., Sörensen, K. (eds.) Adaptive and Multilevel Metaheuristics, Studies in Computational Intelligence, vol. 136, pp. 31–57. Springer-Verlag, Berlin Heidelberg (2008)
25. Smith, J.E.: Self-adaptative and coevolving memetic algorithms. In: Neri, F., Cotta, C., Moscato, P. (eds.) Handbook of Memetic Algorithms, Studies in Computational Intelligence, vol. 379, pp. 167–188. Springer-Verlag, Berlin Heidelberg (2012)
26. Watson, R., Hornby, G., Pollack, J.: Modeling building-block interdependency. In: Eiben, A., et al. (eds.) Parallel Problem Solving from Nature - PPSN V. Lecture Notes in Computer Science, vol. 1498, pp. 97–106. Springer Verlag, Berlin Heidelberg (1998)
27. Zambonelli, F.: Exploiting biased load information in direct-neighbour load balancing policies. Parallel Computing 25(6), 745–766 (1999)
28. Zhao, W., Schulzrinne, H.: 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 (2004)