A Performance Analysis of Self-⋆ Evolutionary Algorithms on Networks with Correlated Failures

Rafael Nogueras and Carlos Cotta

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

Abstract. We consider the deployment of island-based evolutionary algorithms (EAs) on unstable networks whose nodes exhibit correlated failures. We use the sandpile model in order to induce such complex, correlated failures in the system. A performance analysis is conducted, comparing the results obtained in both correlated and non-correlated scenarios for increasingly large volatility rates. It is observed that simple island-based EAs have a significant performance degradation in the correlated scenario with respect to its uncorrelated counterpart. However, the use of self-⋆ properties (self-scaling and self-sampling in this case) allows the EA to increase its resilience in this harder scenario, leading to a much more gentle degradation profile.

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

1 Introduction

The use of parallel environments is of paramount interest for tackling intensive computational tasks. In particular, evolutionary algorithms (EAs) have a long success story in this kind of environments, dating back to the 1980s. In this sense, there has been during the last years an important focus on the use of EAs in emergent computational scenarios that depart from classical dedicated networks so common in the past. Among these we can cite cloud computing [13], P2P networks [21], or volunteer computing [5], just to name a few. The dynamic nature of the underlying computational substrate is one of the most distinguished features of some of these new scenarios –consider for example a P2P network in which nodes enter or leave the system subject to some uncontrollable dynamics caused by user interventions, network disruptions, eventual crashes, etc. The term *churn* is used to denote this phenomenon [17]. Under some circumstances, a potential solution to this issue might be to hide these computational fluctuations under an intermediate layer, providing a virtual stable environment to algorithms running on it. Nonetheless, this can constitute a formidable challenge, mainly in situations in which the underlying substrate is

composed of nodes with low computing power just providing brief, ephemeral bursts of computation (think of, e.g., a large collection of low-end networked devices –cell phones, smart wearables, etc.– contributing their idle time) [6]. The alternative is making the algorithm aware of the dynamic nature of the environment, endowing it with the means for reacting and self-adapting to the volatility of the computational substrate. EAs are in this regard well-suited to this endeavor, since they are resilient techniques that have been shown to be able to withstand –at least to some extent– the sudden loss of part of the population [11], and can be readily endowed with self-$\star$ properties [2] so as to exert self-control on their functioning.

Recent work has precisely studied the use of self-$\star$ properties such as self-scaling [15] and self-healing [14] in this context, providing some evidence on the contribution of these techniques to the robustness of the algorithm when run on unstable computational environments. Quite interestingly, these previous studies have however only considered simple network models in which the dynamics of each node is independent of the rest of the network, that is, the availability of a computing node does not depend on the availability of other nodes. A more general situation would encompass correlated availability patterns, that is, the dynamics of each node might be affected by the dynamics of other nodes, see e.g., [10]. Overall, the presence of correlated failures puts to test the robustness and resilience of the EA, and hence studying it can provide a wider perspective on the usefulness of self-$\star$ techniques to cope with computational instability.

2 Methodology

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

2.1 Network Model

Let us consider a network composed on n_ι nodes interconnected following a certain topology. More precisely, we consider a regular lattice with von Neumann connectivity (virtual topology used for the purposes of migration in the island model) overlaid on a scale-free network (underlying topology for the purposes of failure correlation) as it is often the case in P2P networks, e.g., [12]. In the latter, node degrees are distributed following a power-law and hence there will be a few hubs with large connectivity and increasingly more nodes with a smaller number of neighbors. To generate this kind of networks we use the Barabási-Albert model [1], whereby the network is grown from a clique of $m + 1$ nodes by adding a node at a time, connecting it to m of the nodes previously added (selected with

probability proportional to their degree – the so-called, preferential attatchment mechanism) where m is a parameter of the model.

As stated before, these nodes are volatile, and may abandon the system and re-enter it at a later time, eventually repeating the process over and over again. To model this instability we consider two scenarios: (i) independent or non-correlated failures and (ii) correlated failures. The first one is the simplest model. Therein, the dynamics of each node is independent of other nodes. Each of them can switch from active to inactive or vice versa independently of other nodes with some probability $p(t)$ that only depends on the time it has been in its current state. Following previous work, as well as the commonly observed behavior of e.g., P2P systems [17], $p(t)$ follows a Weibull distribution. This distribution is controlled by two parameters β and η. The first one is the scale parameter and captures the spread of the distribution. The larger this parameter, the less frequent failure events are. The second one is the shape parameter and captures the effect that time has on failure events: for $\eta > 1$ (resp. $\eta < 1$), the longer the time elapsed, the more (resp. less) likely a failure event will be. If η was exactly 1, failures would be time-independent and hence exponentially distributed.

As to the correlated scenario, it features node failures that will be influenced by neighboring nodes. Consider for example the case of sensor networks in which nodes with a large number of active neighbors have their energy depleted faster due to the increased energy toll for communications, or the case of networks that carry load and in which the failure of a node makes other ones absorb the load of the latter, eventually resulting in additional overload failures [10]. This can be modeled in different ways, e.g., [4, 20]. In this work we have considered the sandpile model in order to induce cascading failures [8]. Much like in the previous case, we consider micro-failure events happening on each node with a certain probability $p(t)$. Now, each node i will have an associated threshold value θ_i, indicating the number of micro-failure events required for it to go down. When the number of such micro-failures effectively equals this threshold, the node is disconnected from the system, and each of the active neighbors of this node receives an additional micro-failure event[1]. In case any of these neighbors now accumulated a number of micro-failures equal to its own threshold, it would go down as well, propagating in turn another micro-failure to its active neighbors, and so on (hence the possibility of cascading failures). Fig. 1 shows an example: after node b (which was in a critical state, i.e., one event short of going down) fails, neighboring nodes a and e (which were also in such a critical state) fail as well. We have considered for simplicity that the threshold θ_i of each node i is constant and equal to the number of neighbors (active or inactive) of the node. As to reactivation, just like in the non-correlated case a single event is required.

2.2 Algorithmic Model

Two variants of the island-based EA are considered: (i) a basic one (termed noB) in which every island has a fixed size and random reinitialization is used

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

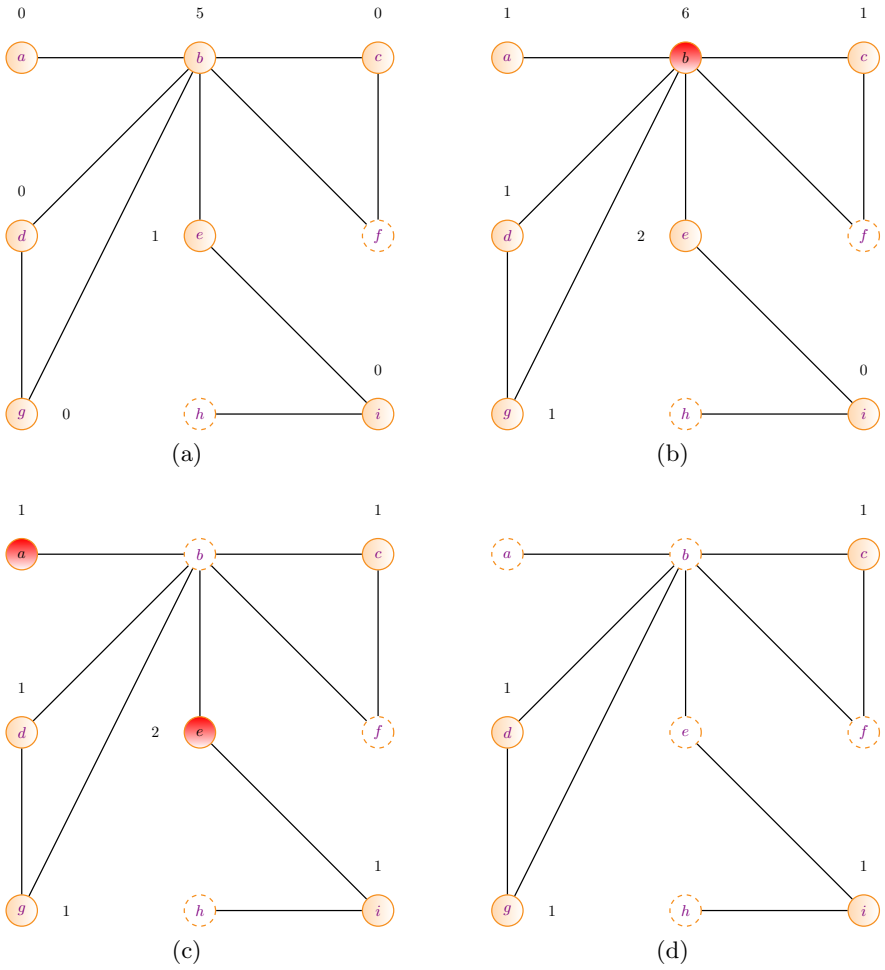


Fig. 1: Example of failure propagation in the sandpile model. Active (resp. inactive) nodes are depicted with solid (resp. dashed) borders. The numbers next to each active node indicate the cumulated number of failure events. The threshold θ_i for each node equals here its degree. (a) Initial state (b) Failure on node b (c) Failure propagation to nodes a and e (d) Final state.

whenever a new node enters the system, and (ii) a self-$\star$ EA (termed LBQ) that uses self-scaling and self-sampling to re-size each island individually in response to fluctuations in the number of active neighbors and in the population sizes of these. In both cases, the islands run a basic steady-state EA and stochastically perform migration (with probability p_{mig}) of a single individual to neighboring islands. In each migration event the migrant is randomly selected from the cur-

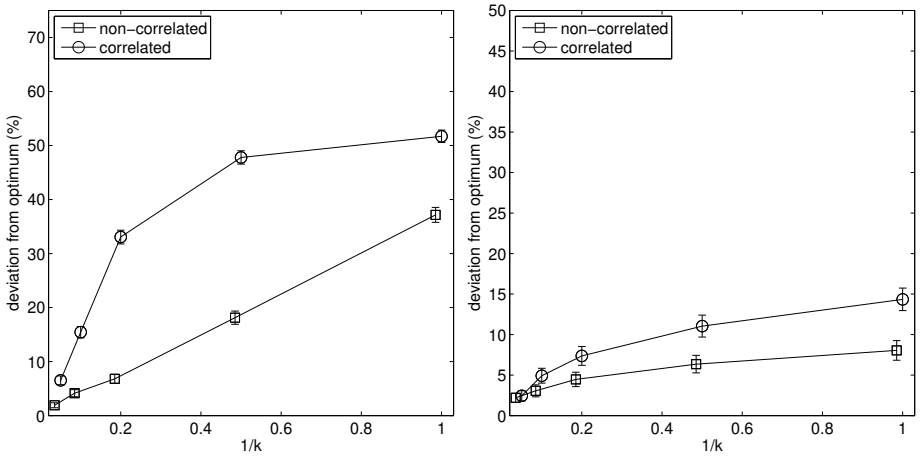
rent population and the receiving island inserts it in its population by replacing the worst individual.

Regarding the self-$\star$ properties considered, self-scaling attempts to attain a rather stable global population size across the islands active in each moment. To this end, each island periodically monitors the state of its neighbors to determine: whether they are active or not, their population sizes and the number of active neighbors they have in turn. When a neighboring island is detected to have just gone down, the island increases its own population size in order to compensate the loss of the former. This is done by calculating the fraction of the population size of the deactivated island corresponding to the number of active neighbors it had. On the other hand, if all neighboring islands are active then they exchange individuals in order to balance their population sizes. See [15] for details. Note that this is a completely autonomous and decentralized policy and therefore each node cannot comprehend the global state of the network, for instance, a node does not account for the simultaneous failure of nodes that are themselves neighbors, hence the interest of studying the robustness of the EA in the correlated scenario.

As to self-sampling, it amounts to maintaining within each island a probabilistic model of its current population in order to sample it whenever the population needs to grow. This has the advantage of introducing diversity due to the stochastic sampling, keeping as well the momentum of the search since the newly created individuals are coherent with the current state of the population (unlike the case of using random individuals to this end). In this work we have considered the use of a tree-like bivariate probabilistic model such as that used in the COMIT estimation of distribution algorithm [3] – see also [14] for details.

3 Experimental Results

We consider $n_\iota = 64$ islands whose initial size is $\mu = 32$ individuals and a total number of evaluations $maxevals = 250\ 000$. Each island runs a steady-state EA with one-point crossover, bit-flip mutation, binary tournament selection and replacement of the worst parent. We use crossover probability $p_X = 1.0$, mutation probability $p_M = 1/\ell$, where ℓ is the genotype length, and migration probability $p_{mig} = 1/160$. Regarding the network parameters, we use $m = 2$ in the Barabási-Albert model in order to define the topology; as for node deactivation/reactivation, we use the shape parameter $\eta = 1.5$ (larger than 1 and hence implying an increasing hazard rate with time), and scale parameters $\beta = -1/\log(p)$ for $p = 1 - (kn_\iota)^{-1}$, $k \in \{1, 2, 5, 10, 20\}$. To interpret these parameters, note that they would correspond to an average of one micro-failure event every k cycles if the failure rate was constant. This provides different scenarios ranging from low volatility ($k = 20$) to very high volatility ($k = 1$). To gauge the results, we also perform experiments with $k = \infty$ (situation corresponding to a stable network without failures). Also, in order to have a more meaningful comparison between both scenarios (accommodating the fact that several micro-failure events are required in order to take down a node in the



(a) (b)

Fig. 2: Average deviation from the optimal solution across all problems for each algorithmic variant and network failure model. (a) noB (b) LBQ

correlated case but only one is needed in the non-correlated case), in the non-correlated scenario we adjust k values as $k' = k\tilde{\theta}$, where $\tilde{\theta}$ is the average of all θ_i values in the correlated scenario (which in this case is also the average degree of the network). Note at any rate that the main focus of the experimentation is the relative behavior of the algorithms considered in either scenario rather than a comparative between scenarios in absolute terms.

As stated in Sect. 2.2, we consider two algorithmic variants: noB (a standard island-based EA with fixed island sizes and random reinitialization of islands upon reactivation) and LBQ (the island-based EA endowed with self-sampling and self-scaling). The experimental benchmark comprises three test functions, namely Deb's trap function [7] (TRAP, concatenating 32 four-bit traps), Watson et al.'s Hierarchical-if-and-only-if function [19] (HIFF, using 128 bits) and Goldberg et al.'s Massively Multimodal Deceptive Problem [9] (MMDP, using 24 six-bit blocks). We perform 25 simulations for each algorithm, problem, volatility scenario and failure model.

Fig. 2 shows a summary of the results (detailed numerical data for each problem, algorithm, and network model are provided in Tables 1–2). Let us firstly focus on noB (Fig. 2a). As expected, the performance of the algorithm degrades as node volatility increases (that is, as we move to the right along the X axis). It is nevertheless interesting to note how the degradation profile of noB is more marked in the correlated scenario. More frequent and simultaneous node failures have a clear toll on performance. If we now consider the case of LBQ, two major observations stand out: on one hand, the performance of LBQ is notably better than that of noB for the same volatility rate. This had been already

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

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

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

		TRAP		H-IFF		MMDP	
strategy	k	$\tilde{x}$	$\bar{x} \pm \sigma_{\bar{x}}$	$\tilde{x}$	$\bar{x} \pm \sigma_{\bar{x}}$	$\tilde{x}$	$\bar{x} \pm \sigma_{\bar{x}}$
–	∞	0.00	0.00 ± 0.00	0.00	5.33 ± 1.49	1.50	1.50 ± 0.17
noB	20	1.25	1.47 ± 0.21	16.67	13.18 ± 1.68	4.49	4.93 ± 0.41
	10	6.87	7.15 ± 0.41	25.87	26.97 ± 1.05	11.98	12.19 ± 0.43
	5	26.25	26.15 ± 0.85	47.40	47.67 ± 0.63	25.97	25.35 ± 0.48
	2	46.88	46.33 ± 0.58	61.46	61.19 ± 0.29	35.46	35.87 ± 0.40
	1	51.25	51.27 ± 0.50	63.72	63.80 ± 0.21	39.95	40.08 ± 0.36
LBQ	20	0.00	0.05 ± 0.05	11.11	7.22 ± 1.49	0.00	0.06 ± 0.06
	10	0.00	0.10 ± 0.07	16.67	14.06 ± 1.57	0.00	0.60 ± 0.23
	5	0.00	0.70 ± 0.18	19.44	20.61 ± 1.19	0.00	0.78 ± 0.21
	2	2.50	2.10 ± 0.21	27.78	27.08 ± 0.83	4.49	3.95 ± 0.37
	1	5.00	5.68 ± 0.41	31.94	30.53 ± 1.04	7.49	6.83 ± 0.42

observed in the non-correlated case (albeit for multimemetic algorithms – this behavior is hence extended for plain EAs as well) and is now confirmed in the correlated scenario, indicating than the self-$\star$ properties seem to keep providing robustness to the algorithm in this case too. As a matter of fact –and this leads to the second observation– the degradation of performance in the correlated case is much less marked for LBQ than it was for noB. More precisely, if we conduct a ranksum test on the results obtained by each algorithm on each problem and network scenario we observe that the performance of noB significantly (at level $\alpha = 0.01$) degrades in the correlated scenario with respect to the non-correlated one for all churn rates, whereas LBQ is only significantly degraded for moderate and high churn rates ($k \leqslant 5$ for TRAP and HIFF and $k \leqslant 2$ for MMDP). This

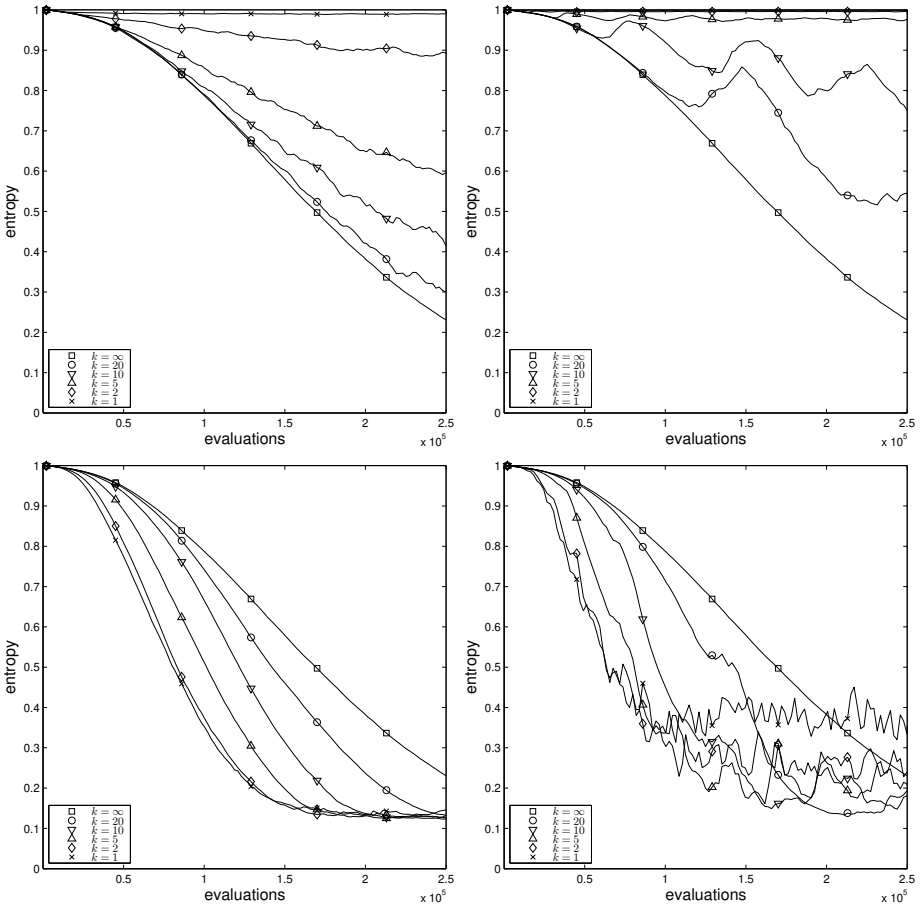


Fig. 3: Genetic diversity for the TRAP function. The top row corresponds to noB and the bottom row to LBQ; the left column corresponds to non-correlated failures, and the right row to correlated failures.

is not to say that LBQ is not adversely affected by the new scenario (in the non-correlated case the performance of LBQ was only significantly degraded with respect to the stable $k = \infty$ case for $k \leqslant 5$ in HIFF and $k \leqslant 2$ in TRAP, whereas in the correlated scenario there are statistically significant differences for $k \leqslant 2$ in MMDP, $k \leqslant 5$ in TRAP and $k \leqslant 10$ in HIFF) but this degradation is mostly in the most volatile cases (unlike noB, whose performance is degraded with respect to $k = \infty$ in the correlated case for all churn rates in all three problems) and not so large in magnitude as for noB. A result consistent with this can also be seen in Fig. 3, in which the genetic diversity of the population (measured using Shannon's entropy) is depicted for each algorithm and scenario (the data corresponds to the TRAP function, but the behavior is qualitatively similar in

the remaining problems). Notice how noB faces increasingly large difficulties to converge as the volatility goes up, and how these difficulties are noticeable even for low-volatility settings in the correlated scenario. LBQ can however maintain a better focus on the search, and seems mostly affected in the most volatile settings of the correlated scenario.

4 Conclusions

The presence of correlated failures constitutes a major threat to the robustness of computing networks. We have analyzed in this work how this phenomenon may affect the performance of an island-based EA, and observed a marked degradation of the results in absence of appropriate policies to deal with this harder scenario. Endowing the EA with self-$\star$ properties can however increase its resilience and make it able to withstand from low up to moderately high volatility.

There are several avenues for further work. First of all, other topologies could be tried. Work is under way here. Also, different models of correlated failures could be tried, using either dynamic thresholds (work is in progress in this area [16]) or other alternative models [4, 20]. In the longer term, a related problem is the optimization of the network itself to cope with this kind of failures. Some recent work has tackled this issue [18], paving the way for other developments in this direction.

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

References

1. Albert, R., Barabási, A.L.: Statistical mechanics of complex networks. Review of Modern Physics 74(1), 47–97 (2002)
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. Baluja, S., Davies, S.: Using optimal dependency-trees for combinatorial optimization: Learning the structure of the search space. In: 14th International Conference on Machine Learning. pp. 30–38. Morgan Kaufmann Publishers (1997)
4. Böttcher, L., Luković, M., Nagler, J., Havlin, S., Herrmann, H.J.: Failure and recovery in dynamical networks. Scientific Reports 7, 41729 EP – (2 2017)
5. Cole, N., Desell, T., González, D.L., Fernández de Vega, F., Magdon-Ismail, M., Newberg, H., Szymanski, B., Varela, C.: Evolutionary algorithms on volunteer computing platforms: The milkyway@home project. In: Fernández de Vega, F., Cantú-Paz, E. (eds.) Parallel and Distributed Computational Intelligence, Studies in Computational Intelligence, vol. 269, pp. 63–90. Springer-Verlag, Berlin Heidelberg (2010)

6. Cotta, C., Fernández-Leiva, A.J., Fernández de Vega, F., Chávez, F., Merelo, J.J., Castillo, P.A., Bello, G., Camacho, D.: Ephemeral computing and bioinspired optimization - challenges and opportunities. In: 7th International Joint Conference on Evolutionary Computation Theory and Applications. pp. 319–324. SCITEPRESS, Lisboa, Portugal (2015)
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. Dorogovtsev, S.N., Goltsev, A.V., Mendes, J.F.F.: Critical phenomena in complex networks. Rev. Mod. Phys. 80, 1275–1335 (Oct 2008)
9. 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)
10. Kong, Z., Yeh, E.M.: Correlated and cascading node failures in random geometric networks: A percolation view. In: 2012 Fourth International Conference on Ubiquitous and Future Networks (ICUFN). pp. 520–525. IEEE, Phuket, Thailand (July 2012)
11. Lombraña González, D., Jiménez Laredo, J., Fernández de Vega, F., Merelo Guervós, J.J.: Characterizing fault-tolerance in evolutionary algorithms. In: Fernández de Vega, F., et al. (eds.) Parallel Architectures and Bioinspired Algorithms, Studies in Computational Intelligence, vol. 415, pp. 77–99. Springer-Verlag, Berlin Heidelberg (2012)
12. Matei, R., Iamnitchi, A., Foster, P.: Mapping the Gnutella network. IEEE Internet Computing 6(1), 50–57 (Jan 2002)
13. Meri, K., Arenas, M., Mora, A., Merelo, J.J., Castillo, P., García-Sánchez, P., Laredo, J.: Cloud-based evolutionary algorithms: An algorithmic study. Natural Computing 12(2), 135–147 (2013)
14. Nogueras, R., Cotta, C.: Self-healing strategies for memetic algorithms in unstable and ephemeral computational environments. Natural Computing 16(2), 189–200 (2016)
15. Nogueras, R., Cotta, C.: Studying self-balancing strategies in island-based multimemetic algorithms. Journal of Computational and Applied Mathematics 293, 180–191 (2016)
16. Nogueras, R., Cotta, C.: Evaluating island-based EAs on unstable networks with complex failure patterns. In: Proceedings of GECCO' 17 Companion (late breaking abstract). Berlin, Germany (2017), 2 pages
17. Stutzbach, D., Rejaie, R.: Understanding churn in peer-to-peer networks. In: 6th ACM SIGCOMM Conference on Internet Measurement - IMC 2006. pp. 189–202. ACM Press, New York, NY, USA (2006)
18. Tang, X., Liu, J., Hao, X.: Mitigate cascading failures on networks using a memetic algorithm. Scientific Reports 6, 38713 EP – (12 2016)
19. 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)
20. Watts, D.J.: A simple model of global cascades on random networks. Proceedings of the National Academy of Sciences 99(9), 5766–5771 (2002)
21. Wickramasinghe, W., Steen, M.V., Eiben, A.E.: Peer-to-peer evolutionary algorithms with adaptive autonomous selection. In: Thierens, D., et al. (eds.) Genetic and Evolutionary Computation - GECCO 2007. pp. 1460–1467. ACM Press, New York, NY, USA (2007)