# Sensitivity Analysis of Checkpointing Strategies for Multimemetic Algorithms on Unstable Complex Networks

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.** The use of volatile decentralized computational platforms such as, e.g., peer-to-peer networks, is becoming an increasingly popular option to gain access to vast computing resources. Making an effective use of these resources requires algorithms adapted to such a changing environment, being resilient to resource volatility. We consider the use of a variant of evolutionary algorithms endowed with a classical fault-tolerance technique, namely the creation of checkpoints in a safe external storage. We analyze the sensitivity of this approach on different kind of networks (scale-free and small-world) and under different volatility scenarios. We observe that while this strategy is robust under low volatility conditions, in cases of severe volatility performance degrades sharply unless a high checkpoint frequency is used. This suggest that other fault-tolerance strategies are required in these situations.

## 1 Introduction

Distributed computing platforms have been used for running population-based metaheuristics for decades now. This is a direct consequence of the flexibility and adaptability of these techniques whose functioning is intrinsically parallel. Hence they can be naturally deployed on networked computers, cf. [1]. Numerous research works have focused on different design aspects of these techniques and how they affect performance in distributed environments – see, e.g., [2, 6, 25]. Exploiting efficiently distributed computing resources has become one of the signature weapons of these techniques and is a major factor for boosting their performance. In this sense, it is worth noting how technological advances are reshaping both the underlying computational substrate and the very needs to be addressed in computational terms. Regarding the latter, the problems and their data are becoming increasingly larger and complex [22]. The term *Big Data* [28] is nowadays a hot buzzword used to denote such large collections of data, very much requiring vast computational power in order to harnessed.

While traditional supercomputing techniques (namely, dedicated systems hosting a large array of processors and colossal memory banks) are certainly one of the lines of attack to Big Data problems, the preponderance of computing resources permanently connected to the Internet has led to the emergence of other computational environments such as peer-to-peer (P2P) networks [14] and volunteer computing networks [23]. These are bound to play a key role in this kind of endeavors since they allow the orchestration of enormous decentralized collections of computational nodes. This comes at a cost though: these computational resources are unstable (they are typically contributed by volunteers during their idle time) and this must be taken into account when deploying applications on this kind of environments. Focusing specifically on applications (population-based metaheuristics in our case) running natively on these environment (i.e., being aware of its dynamicity and dealing with it directly), they can either use some fault-management policy for corrective purposes [12] or can self-adapt their behavior/parameterization to cope with it. We aim our attention at the former approach in this work. More precisely we analyze the performance of strategies based on creating restoration checkpoints [16]. This is done within the context of multimemetic algorithms [11], namely memetic algorithms which self-adapt the local search procedure, cf. [21]. These are described next.

## 2 Fault-Tolerant Model in an Island-based Multimemetic Algorithm

As stated before we consider the use of multimemetic algorithms (MMAs) on a unstable computational scenario. Our MMA is organized as an island-based algorithm [24, 29], that is, it has a population distributed over a collection of $n$ islands. Each of these islands comprises a panmictic (i.e., unstructured) subpopulation and runs a basic steady-state MMA procedure. This procedure follows the standard pattern of memetic algorithms, namely, selection, recombination, mutation and local search [15] but has a distinctive feature: local search is not done using a predefined strategy but using search patterns (*memes*) embedded in each individual and evolving alongside the latter (note the connection with the concept of memetic computing [20]). Inspired by the model by Smith [26, 27], these memes are expressed as variable-length pattern-based rewriting rules $A \to B$ (i.e., find $A$ in the genome and change it into $B$; both $A, B \in \Sigma \cup \{\#\}$ where $\Sigma$ is the alphabet used for encoding solutions and # is a wildcard). They evolve via mutation and are transferred from parent to offspring via local selection. We refer to [18] for further details.

The islands are distributed over a network of nodes and perform migration asynchronously (randomly picking an individual from an island and transferring it to another one, where it replaces the worst individual [17]). Two factors define this computational scenario: the interconnection topology and the dynamic model of the network. Regarding the topology, we consider to possibilities:

– Scale-free networks (SF): these are characterized by the existence of a power-law distribution in node degrees, and are often observed in many natural processes. We use the Barabási-Albert (BA) model [3] to generate this kind of SF networks. This model uses preferential attachment [4] to grow a network

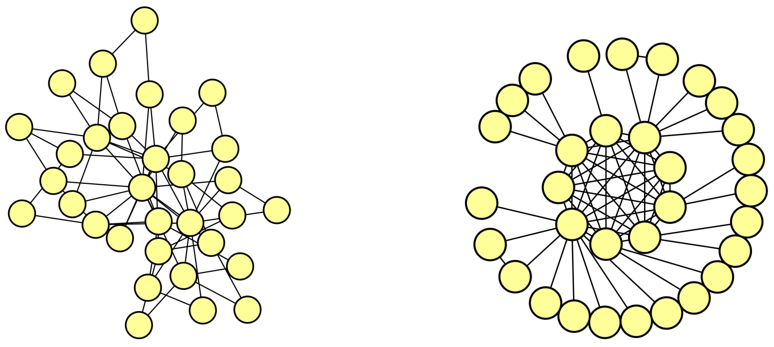


Fig. 1: Example networks for $n = 32$. The left figure is a SF network ($m = 2$) and the right one is a SW network ($M = 61$).

by adding a new node at a time. A parameter $m$ determines the number of links each new node gets.

– Small-world networks (SW): these are characterized by very small average distances between nodes (often $O(\log n)$ where $n$ is the number of nodes). We use a variant of the Barmpoutis-Murray (BM) model [5] to create ultra-SW networks. This model takes as a parameter the total number of nodes $n$ and the total number of links $M$ and uses a backtracking procedure to successively build the largest clique that leaves enough links available to connect the rest of the network. In our variant, each of these cliques are then connected using random vertices in the first clique created so as to make the resulting network more resilient.

Fig. 1 shows an example of both kinds of network with the same number of nodes and links.

As to the dynamics of the network, it is characterized by the availability patterns of computing nodes. We use the model in [16]: all $n$ nodes are initially available and then their permanence in the system follows a Weibull distribution. This distribution is characterized by a shape parameter $\eta$ that determines whether failure probability increases with time ($\eta > 1$), decreases with time ($\eta < 1$) or is time-independent ($\eta = 1$), and a scale parameter $\beta$ determining the mean lifetime for a given shape. Each node has an independent dynamics and will contribute to the so-called *churn* phenomenon, namely the collective effect on the network of computing nodes independently entering and leaving it over time. Churn can have different effects on a distributed population-based metaheuristic, the most obvious being that the current incumbent solution can be lost [9]. Needless to say, the progress of the search will be also affected by the disappearance of whole subpopulations. To tackle this in the context of corrective fault-management policies, we consider two strategies [16]:

– rand: when a node becomes available again, it is initialized from scratch much like in the initialization of the algorithm.

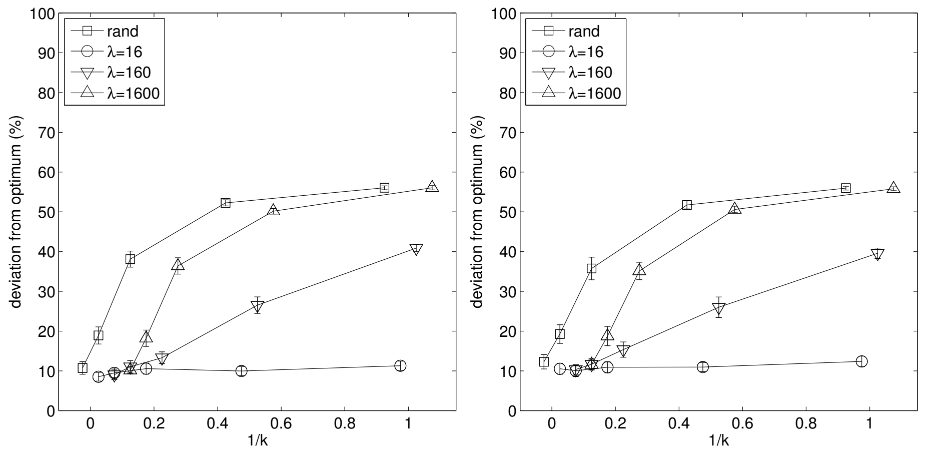
– checkpoint: the algorithm uses some external safe storage in order to create restoration checkpoints, namely periodical backups of the populate state that are used to recover the last state of the population when a node becomes available again.

It is clear that rand is a simpler strategy that has also the potential advantage of reintroducing diversity in the search process. On the other hand, checkpoint has the advantage of not wasting the previous progress of the search, being more amenable to keep its momentum. The negative side of this latter strategy is the requirement of this external safe storage and the associated overhead (particularly if security and privacy concerns are important [13]) introduced by the periodical backups. The latter effect can be somehow ameliorated by tuning the period $\lambda$ (measured in number of iterations) between checkpoints. The effect of this parameter is studied next.

## 3 Experiments

We have done experiments using a distributed MMA with $n = 32$ islands. Each island has a population size of $\mu = 16$ individuals. Meme lengths evolve within $l_{\min} = 3$ and $l_{\max} = 9$, mutating their length with probability $p_r = 1/9$ following [18]. We use crossover probability $p_X = 1.0$ (one-point crossover), mutation probability $p_M = 1/\ell$ (bit-flip mutation), where $\ell$ is the genotype length, and migration probability $p_{mig} = 1/80$. In order to generate the network topology we use $m = 2$ in the BA model of SF networks, and the corresponding value of $M = nm - m(m+1)/2$ in the BM model of SW networks so that the number of links is the same in both cases. As to node dynamics, we use the shape parameter $\eta = 1.5$ (and hence the probability of failure increases with time), and scale parameters $\beta = -1/\log p(k)$ for $p(k) = 1-(kn)^{-1}$, $k \in \{1, 2, 5, 10, 20\}$. By doing this, the mean availability stint per node is about $90\% \cdot kn$ iterations. We therefore obtain scenarios ranging from rather low ($k = 20$) churn up to extremely high ($k = 1$) churn. To analyze the sensitivity of the checkpoint strategy we consider values $\lambda \in \{\mu, 10\mu, 100\mu\}$ where $\mu$ is the island population size. For comparison purposes we also consider in the experimentation the use of the rand strategy. We have considered four test functions, namely Deb's trap (TRAP) function [7], Watson et al.'s Hierarchical-if-and-only-if (HIFF) and Hierarchical-Exclusive-OR (HXOR) functions [30] and Goldberg et al.'s Massively Multimodal Deceptive Problem (MMDP) [8]. We perform 25 simulations running for a total number of 50 000 evaluations for each value of $\lambda$, churn scenario, problem and network topology.

Fig. 2 shows the results obtained in terms of deviation with respect to the optimal solution (averaged for the four problems) as a function of the churn rate, separately for each $\lambda$ value and for each network topology. First of all, it is clear that performance degrades for increasing churn rate. This fact notwithstanding, we can observe that variants using checkpoint reactivation perform notably better than random reactivation. This confirms previous research on this kind of strategies [16] and validates its usefulness on different network topologies. Note



(a) (b)

Fig. 2: Deviation from the optimal solution as a function of the churn rate for (a) SF and (b) SW.

Table 1: Results of Holm Test ($\alpha = 0.05$) using $\lambda = 16$ as control parameter.

| $i$ strategy | $z$-statistic | $p$-value | $\alpha/i$ |
|---|---|---|---|
| 1 $\lambda = 160$ | 2.598e+00 | 4.687e–03 | 5.000e–02 |
| 2 $\lambda = 1600$ | 7.015e+00 | 1.151e–12 | 2.500e–02 |
| 3 rand | 8.747e+00 | 1.097e–18 | 1.667e–02 |

however that there is a marked performance degradation when the checkpoint frequency is increased. This degradation is shown to be statistically significant according to Quade test ($p$-value $\approx 0$) both globally and when SF and SW are separately analyzed. Subsequently we used Holm test to do a post-hoc analysis. The use of checkpoint with parameter $\lambda = \mu$ is shown to be statistically superior to the remaining techniques at $\alpha = 0.05$ level – see Table 1. This result suggests that less expensive strategies (in terms of requiring less frequent state snapshots) are not capable of dealing with churn (this result also holds if a separate analysis is conducted for SF and SW topologies). A more clear depiction of the behavior of the MMAs is provided by Fig. 3 for low ($k = 20$), high ($k = 5$) and extremely high ($k = 1$) churn. Note that in the most stable scenario the algorithm performs robustly regardless of the frequency of the snapshots (although as seen in Fig. 3f there is a noticeable difference in genetic diversity when the period $\lambda$ is large). However, as churn increases the difference in fitness turns out to be remarkably higher in favor of $\lambda = \mu$. As seen in Fig. 3d and 3f, the MMA has convergence problems in these scenarios when $\lambda$ is high. The less frequent snapshots cannot keep the momentum of the search in such unstable environments.

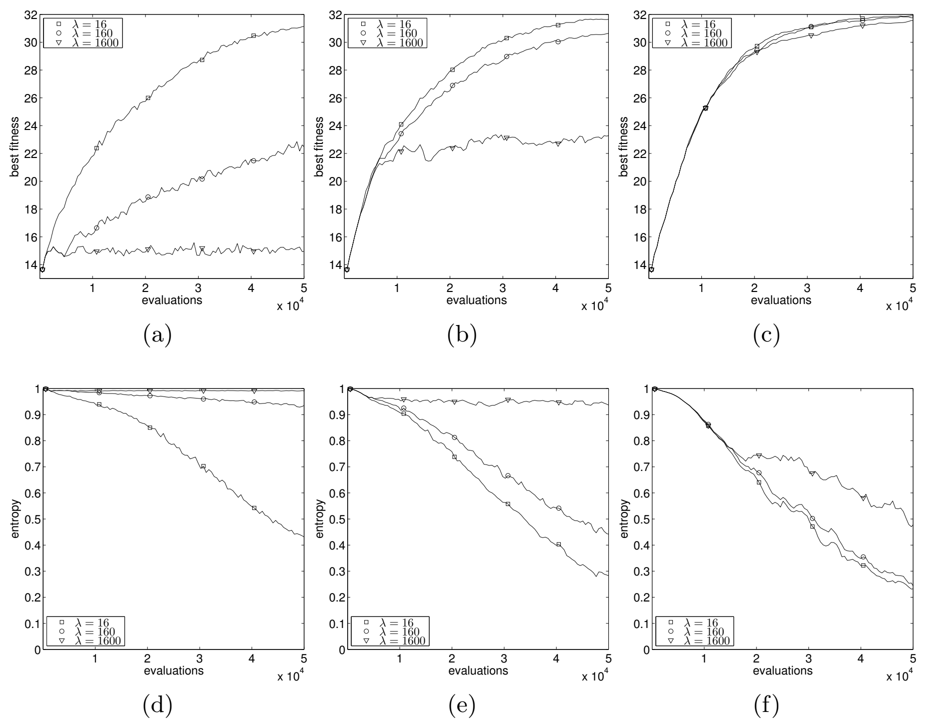


Fig. 3: Best fitness (top) and entropy (bottom) for TRAP with SF topology. From left to right: $k = 1$, $k = 5$ and $k = 20$.

## 4 Conclusion

Any algorithm directly deployed on a unstable computational environment must be resilient to the volatility of its substrate. Metaheuristics are no exception and, while they are intrinsically resilient to some extent [10], they must be augmented with adequate policies in order to cope with the loss of information associated to computing nodes that become inactive. A classical fault-tolerance technique for this purpose is the creation of periodical backups of the state of these nodes in order to recover from failures. We have performed a sensitivity analysis of this strategy in the context of island-based multimemetic algorithms. It turns out that this approach can be affordable in scenarios with low churn rates. In such a situation, checkpoints need not be frequent for the algorithm to perform adequately. However, scenarios featuring large churn rates require much more frequent backups in order to cope with node volatility. The additional overhead of such backups together with the need for having access to persistent external storage makes this approach less appealing in such situations, suggesting other approaches –autonomous, self-adaptive and purely local– can be more appropriate. Work is already in progress in this direction [19].

**Acknowledgements** This work is partially supported by the MINECO project EphemeCH (TIN2014-56494-C4-1-P), by the Junta de Andalucía project DNEMESIS (P10-TIC-6083) and by the Universidad de Málaga, Campus de Excelencia Internacional Andalucía Tech.

## References

1. Alba, E.: Parallel Metaheuristics: A New Class of Algorithms. Wiley-Interscience (2005)
2. Alba, E., Troya, J.M.: Influence of the migration policy in parallel distributed GAs with structured and panmictic populations. Appl. Intell. 12(3), 163–181 (2000)
3. Albert, R., Barabási, A.L.: Statistical mechanics of complex networks. Review of Modern Physics 74(1), 47–97 (Jan 2002)
4. Barabási, A.L., Albert, R.: Emergence of Scaling in Random Networks. Science 286(5439), 509–512 (1999)
5. Barmpoutis, D., Murray, R.M.: Networks with the smallest average distance and the largest average clustering. arXiv 1007.4031 [q-bio] (2010)
6. Cantu-Paz, E.: Efficient and Accurate Parallel Genetic Algorithms. Kluwer Academic Publishers, Norwell, MA, USA (2000)
7. Deb, K., Goldberg, D.E.: Analyzing deception in trap functions. In: Whitley, L.D. (ed.) Second Workshop on Foundations of Genetic Algorithms. pp. 93–108. Morgan Kaufmann, Vail, Colorado, USA (1993)
8. Goldberg, D.E., Deb, K., Horn, J.: Massive multimodality, deception, and genetic algorithms. In: Parallel Problem Solving from Nature – PPSN II. pp. 37–48. Elsevier, Brussels, Belgium (1992)
9. Hidalgo, J.I., Lanchares, J., Fernández de Vega, F., Lombraña, D.: Is the island model fault tolerant? In: Proceedings of the 9th Annual Conference Companion on Genetic and Evolutionary Computation. pp. 2737–2744. GECCO '07, ACM, New York, NY, USA (2007)
10. Jiménez Laredo, J.L., Bouvry, P., Lombraña González, D., Fernández de Vega, F., García Arenas, M., Merelo Guervós, J.J., Fernandes, C.M.: Designing robust volunteer-based evolutionary algorithms. Genetic Programming and Evolvable Machines 15(3), 221–244 (2014)
11. Krasnogor, N., Blackburne, B., Burke, E., Hirst, J.: Multimeme algorithms for protein structure prediction. In: Merelo, J., et al. (eds.) Parallel Problem Solving From Nature VII, Lecture Notes in Computer Science, vol. 2439, pp. 769–778. Springer, Berlin (2002)
12. Lombraña González, D., Jiménez Laredo, J.L., Fernández de Vega, F., Merelo Guervós, J.J.: Characterizing fault-tolerance of genetic algorithms in desktop grid systems. In: Cowling, P., Merz, P. (eds.) Evolutionary Computation in Combinatorial Optimization, Lecture Notes in Computer Science, vol. 6022, pp. 131–142. Springer Berlin Heidelberg (2010)
13. Mihaljević, M.J., Imai, H.: Security issues of cloud computing and an encryption approach. In: Despotović-Zrakić, M., Milutinović, V., Belić, A. (eds.) Handbook of Research on High Performance and Cloud Computing in Scientific Research and Education, pp. 388–408. IGI Global, Hershey PA (2014)
14. Milojičić, D.S., Kalogeraki, V., Lukose, R., Nagaraja, K., Pruyne, J., Richard, B., Rollins, S., Xu, Z.: Peer-to-peer computing. Tech. Rep. HPL-2002-57, Hewlett-Packard Labs (2002)

15. Neri, F., Cotta, C., Moscato, P.: Handbook of Memetic Algorithms, Studies in Computational Intelligence, vol. 379. Springer, Berlin Heidelberg (2012)
16. Nogueras, R., Cotta, C.: Studying fault-tolerance in island-based evolutionary and multimemetic algorithms. Journal of Grid Computing (2015), DOI:10.1007/s10723-014-9315-6
17. Nogueras, R., Cotta, C.: An analysis of migration strategies in island-based multimemetic algorithms. In: Bartz-Beielstein, T., Branke, J., Filipič, B., Smith, J. (eds.) Parallel Problem Solving From Nature – PPSN XIII. Lecture Notes in Computer Science, vol. 8672, pp. 731–740. Springer, Berlin Heidelberg (2014)
18. Nogueras, R., Cotta, C.: On meme self-adaptation in spatially-structured multimemetic algorithms. In: Dimov, I., Fidanova, S., Lirkov, I. (eds.) Numerical Methods and Applications. Lecture Notes in Computer Science, vol. 8962, pp. 70–77. Springer, Berlin-Heidelberg (2015)
19. Nogueras, R., Cotta, C.: Studying self-balancing strategies in island-based multimemetic algorithms. Journal of Computational and Applied Mathematics (2015), DOI:10.1016/j.cam.2015.03.047
20. Ong, Y.S., Lim, M.H., Chen, X.: Memetic computation-past, present and future. IEEE Computational Intelligence Magazine 5(2), 24–31 (2010)
21. Ong, Y.S., Lim, M.H., Zhu, N., Wong, K.W.: Classification of adaptive memetic algorithms: a comparative study. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 36(1), 141–152 (2006)
22. Reichhardt, T.: It's sink or swim as a tidal wave of data approaches. Nature 399(6736), 517–520 (1999)
23. Sarmenta, L.F.: Bayanihan: Web-based volunteer computing using java. In: Masunaga, Y., Katayama, T., Tsukamoto, M. (eds.) Worldwide Computing and Its Applications – WWCA'98, Lecture Notes in Computer Science, vol. 1368, pp. 444–461. Springer Berlin Heidelberg (1998)
24. Schaefer, R., Byrski, A., Smołka, M.: The island model as a Markov dynamic system. International Journal of Applied Mathematics and Computer Science 22(4), 971–984 (2012)
25. Skolicki, Z., Jong, K.D.: The influence of migration sizes and intervals on island models. In: Genetic and Evolutionary Computation Conference 2005. pp. 1295–1302. ACM, New York, NY (2005)
26. 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 Berlin Heidelberg (2008)
27. 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)
28. Snijders, C., Matzat, U., Reips, U.D.: 'Big Data': Big gaps of knowledge in the field of internet. International Journal of Internet Science 7, 1–5 (2012)
29. Tanese, R.: Distributed genetic algorithms. In: 3rd International Conference on Genetic Algorithms. pp. 434–439. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1989)
30. Watson, R.A., Hornby, G.S., Pollack, J.B.: Modeling building-block interdependency. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.P. (eds.) Parallel Problem Solving from Nature – PPSN V, Lecture Notes in Computer Science, vol. 1498, pp. 97–106. Springer-Verlag, Berlin Heidelberg (1998)