# Self-Sampling Strategies for Multimemetic Algorithms in Unstable Computational 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.** Optimization algorithms deployed on unstable computational environments must be resilient to the volatility of computing nodes. Different fault-tolerance mechanisms have been proposed for this purpose. We focus on the use of dynamic population sizes in the context of island-based multimemetic algorithms, namely memetic algorithms which explicitly represent and evolve memes alongside solutions. These strategies require the eventual creation of new solutions in order to enlarge island populations, aiming to compensate the loss of information taking place when neighboring computing nodes go down. We study the influence that the mechanism used to create these new individuals has on the performance of the algorithm. To be precise, we consider the use of probabilistic models of the current population which are subsequently sampled in order to produce diverse solutions without distorting the convergence of the population and the progress of the search. We perform an extensive empirical assessment of those strategies on three different problems, considering a simulated computational environment featuring diverse degrees of instability. It is shown that these self-sampling strategies provide a performance improvement with respect to random reinitialization, and that a model using bivariate probabilistic dependencies is more effective in scenarios with large volatility.

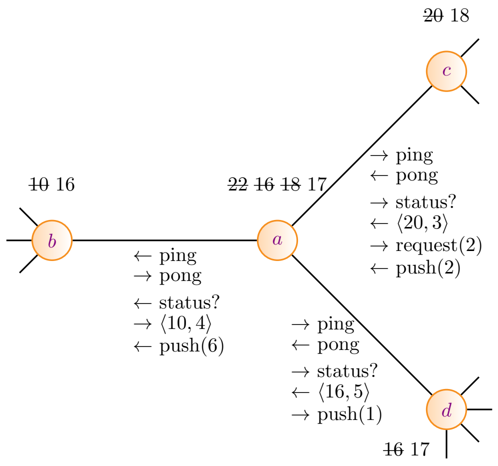
## 1 Introduction

One of the greatest advantages of metaheuristics, and in particular of population-based variants thereof, is their amenability for deployment on parallel and distributed environments [2]. Consider for example the so-called island model of evolutionary algorithms [24], whereby multiple populations evolve in parallel and occasionally exchange information. Distributing these islands among different computing nodes can lead to notably improved solutions and remarkable reductions in the computational time required to reach them [1]. Although this strategy has been successfully exploited since the late 1980s, emerging computational environments such as peer-to-peer (P2P) networks [13] and volunteer computing networks [22] are currently bringing both new opportunities and new challenges. Regarding the latter, it must be noted that the nature of these computational platforms is inherently dynamic due to the volatility of computational resources attached to them (i.e., computing nodes can enter and abandon the system in a dynamic way – the term *churn* has been coined to denote this phenomenon). For this reason, it is essential that algorithms running on this kind of environments are adequately suited to work under these conditions. For example, it has been shown that the instability of the computational environment can lead to the loss of the current incumbent solution in island-based evolutionary algorithms (EAs) [8]; it will also pose additional difficulties to the progress of the search due to the loss of good solutions, genetic diversity, etc. Although EAs are intrinsically robust at a fine-grain scale [10], algorithms must nevertheless be fault-aware in some sense in order to cope with highly volatile systems. In this line, different fault-management strategies have been proposed – see [16]. In essence, these strategies can be seen as corrective, exhorting their effect when a computing node re-enters the system after having left it previously. As an alternative to these strategies, we can think of reactive strategies whereby the algorithm self-adapts to the changing environment on the fly.

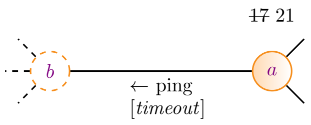
An example of such strategies can be found in the model for dynamic resizing of populations defined by Nogueras and Cotta – see Sect. 2. Unlike other fault-management strategies, this scheme is intrinsically decentralized and emergent, suiting computational scenarios without global control, and does not require external persistent storage. By using this model, a rather constant population size is kept on a global scale, having the size of individual populations fluctuate to compensate for node activations/deactivations. Notice in this sees that while populations can be easily reduced by truncation and/or distribution of individuals, enlarging them requires the creation of new solutions. In this work we focus on this aspect and study different probabilistic modeling strategies for this purpose. We apply these in the context of island-based model of multimemetic algorithms (MMAs) [11], an extension of memetic algorithms [15] in which computational representations of problem solving strategies (neighborhood definitions for a local search operator in this case) are explicitly stored and evolved as a part of solutions, much in the line of the concept of memetic computing [20]. We consider the deployment of these techniques on a simulated computational environment that allows experimenting with different churn rates. We report the results of a broad empirical assessment of the strategies considered in Sect. 4. We close the paper with conclusions and an outline of future work in Sect. 5.

## 2 Algorithmic Model

We have $n_\iota$ panmictic (i.e., unstructured) islands, each of them running a MMA in which memes are attached to individuals and evolve alongside them. These memes are represented as pattern-based rewriting rules following the model by Smith [23]. Memes are subject to mutation and are transferred from parent to offspring via local selection (offspring inherit the meme of the best parent). We refer to [18] for details. Besides the use of memes, the MMA run on each is-



(a)



(b)

Fig. 1: (a) Node $a$ attempts to balance with nodes $b$, $c$ and $d$ in that order. The numbers next to each node indicate the population sizes (crossed-out values correspond to previous sizes). (b) Node $a$ attempts to balance with node $b$ which is now inactive. The information from the previous balancing attempt, namely that node $b$ had size 16 and 4 active neighbors, is recalled in order to enlarge the population of node $a$ by 16/4 individuals.

land can be otherwise regarded as a steady-state evolutionary algorithm using tournament selection, one-point crossover, bit-flip mutation, and replacement of the worst parent. From a global point of view, the islands work in parallel and are interconnected as a scale-free network, a pattern of connectivity observed in many real-world systems, and in particular in peer-to-peer systems. We use Barabási-Albert model [3] in order to generate these networks. This network topology is used for the purposes of information exchange via migration. This operation is asynchronously performed with a certain probability at the end of each cycle. When performed, we use random selection of migrants and deterministic replacement of the worst individuals in the receiving island [17].

To model the instability of the computational environment we assume each island of the MMA runs on one out of $n_\iota$ nodes. These nodes are volatile and become inactive or active on a time-dependent basis (much like P2P networks or volunteer computing platforms). To be precise we use a Weibull distribution to model the dynamics of the system – see [12,16]. In response to such variability of computational resources, the MMA tries to dynamically resize the islands so as to counteract the loss of information taking place when a computing node goes down. To this end, each island performs a handshake with its neighbors at the beginning of each evolutionary cycle and tries to balance the size of their population by exchanging individuals (see Fig. 1a). In the event of a new computing node going up, this same process is used for it to absorb a fraction of its neighbors' population (using a default population size $C_1$ if no neighbor can donate solutions). If during this handshaking process a node detects that a neighbor that was active has just become inactive, it enlarges its own population to compensate the loss of that island (see Fig. 1b). For this purpose new individuals have to be created. Next section studies how to approach this task.

## 3 Self-Sampling Strategies

Dynamic population resizing strategies such as those described before only capture the quantitative aspect of the process when it comes to enlarge a population without having a donor to provide solutions. The simplest approach to solve the qualitative question of how to produce those required individuals is to resort to blind search: new solutions, as many as needed, are generated from scratch, essentially using the same mechanism by which the initial population was created. While this can be regarded as an appropriate method to boost diversity, introducing fresh information in the population, it certainly has the drawback of not keeping up with the momentum of the search. In fact, the new random individuals are to a certain extent dragging backwards the population in terms of global convergence to promising regions of the search space. Needless to say, this effect will be more or less strong depending upon the frequency at which the algorithm has to resort to this mechanism, which will be directly linked to the severity of churn.

As an alternative to such a random reinitialization from scratch, some smarter strategies based on probabilistic modeling could be used. The underlying idea would be to estimate the joint probability distribution $p(\boldsymbol{x})$ representing the population of the island to be enlarged; subsequently, that distribution would be sampled as many times as necessary in order to produce the new individuals. By doing so, two objectives are pursued: (1) the momentum of the search is kept since the new individuals would be representative of the current state of the population (as indicated by the probabilistic model) and (2) diversity would be still introduced since the new individuals can be different to individuals already in the population.

To approach the probabilistic modeling we assume the population is a matrix $[pop_{ij}]$ where each row $pop_{i\cdot}$ represents an individual and each column $pop_{\cdot j}$ represent a variable (a genetic/memetic symbol). We consider two alternatives to model this population: univariate and bivariate models. In univariate models, variables are assumed to be independent and therefore the joint probability distribution $p(\boldsymbol{x})$ is factorized as

$$p(\boldsymbol{x}) = \prod_{j=1}^{n} p(x_j).$$

This is the model used by simple estimation of distribution algorithms (EDAs) such as UMDA [14], in which $p(x_j)$ is estimated as

$$p(x_j = v) = \frac{1}{\mu}\sum_{i=1}^{\mu}\delta(pop_{ij}, v),$$

where $\mu = |pop|$ is the size of the population, $pop_{ij}$ is the value of the $j$-th variable of the $i$-th individual in *pop*, and $\delta(\cdot,\cdot)$ is Kronecker delta ($\delta(a,b) = 1$ if $a = b$ and $\delta(a,b) = 0$ otherwise).

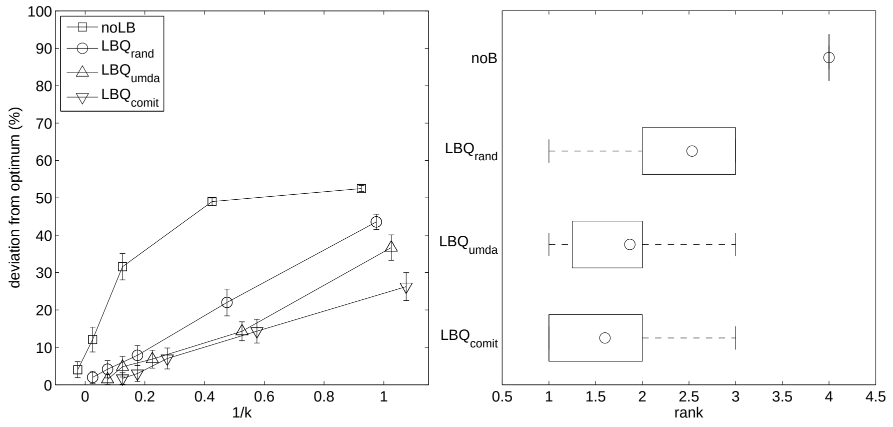
On the other hand, bivariate models assume relations between pairs of variables. Thus, $p(\boldsymbol{x})$ is factorized as

$$p(\boldsymbol{x}) = p(x_{j_1})\prod_{i=2}^{n} p(x_{j_i}|x_{j_{a(i)}}),$$

where $j_1 \cdots j_n$ is a permutation of the indices $1 \cdots n$, and $a(i) < i$ is the permutation index of the variable which $x_{j_i}$ depends on. This model is used by EDAs such as MIMIC [5] and COMIT [4]. We focus on a mechanism analogous to the latter EDA and hence we pick $j_1$ as the variable with the lowest entropy $H(X_k)$ in the population, and then we pick $j_i$ ($i > 1$) as the variable (among those not yet selected) that minimizes $H(X_k|X_{j_s}, s < i)$. Each variable will therefore depend on a variable previously selected hence leading to a tree-like dependence structure. As a final detail, it must be noted that we compute separate models for both genotypes and memes.

## 4 Experimental Analysis

The experiments have been done using an MMA with $n_\iota = 32$ islands whose initial size is $\mu = 16$ individuals, interconnected utilizing a scale-free topology generated with Barabási-Albert model using parameter $m = 2$. We consider crossover probability $p_X = 1.0$, mutation probability $p_M = 1/\ell$ –where $\ell$ is the genotype length– and migration probability $p_{\text{mig}} = 1/80$. As indicated in Sect. 2, memes are rewriting rules. Following [18], their length varies from $l_{\min} = 3$ up to $l_{\max} = 9$ and is mutated with probability $p_r = 1/l_{\max}$. To control churn, we consider that the Weibull distribution describing node availability is defined by the shape parameter $\eta = 1.5$ (implying an increasing hazard rate since it is larger than 1) and by six scale parameters $\beta = -1/\log(p)$ for $p = 1 - (Kn_\iota)^{-1}$, $K \in \{1, 2, 5, 10, 20, \infty\}$ describing scenarios of different volatility: from no churn ($K = \infty$) to very high churn ($K = 1$). As an approximate interpretation, notice that under an hypothetically constant hazard rate these scale parameters would correspond to an average of one island going down/up every $K$ cycles. Parameter $C_1$ used during eventual island reinitialization from scratch is set to $2\mu = 32$. We perform 25 simulations for each algorithm and churn scenario, each of them comprising *maxevals* $= 50000$ evaluations. We denote by LBQ$_{\text{rand}}$, LBQ$_{\text{umda}}$ and LBQ$_{\text{comit}}$ the MMAs using random reinitialization of



(a) (b)

Fig. 2: (a) Mean deviation from the optimal solution across all three problems for each algorithm as a function of the churn rate. (b) Distribution of ranks for each algorithm. As usual the boxes comprise the 2nd and 3rd quartiles of the distribution, the whiskers extend to the most extreme data points not considered outliers and the circle marks the mean rank.

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

| $i$ | algorithm | $z$-statistic | $p$-value | $\alpha/i$ |
|---|---|---|---|---|
| 1 | LBQ$_{\text{umda}}$ | 5.657e–01 | 2.858e–01 | 5.000e–02 |
| 2 | LBQ$_{\text{rand}}$ | 1.980e+00 | 2.386e–02 | 2.500e–02 |
| 3 | noB | 5.091e+00 | 1.779e–07 | 1.667e–02 |

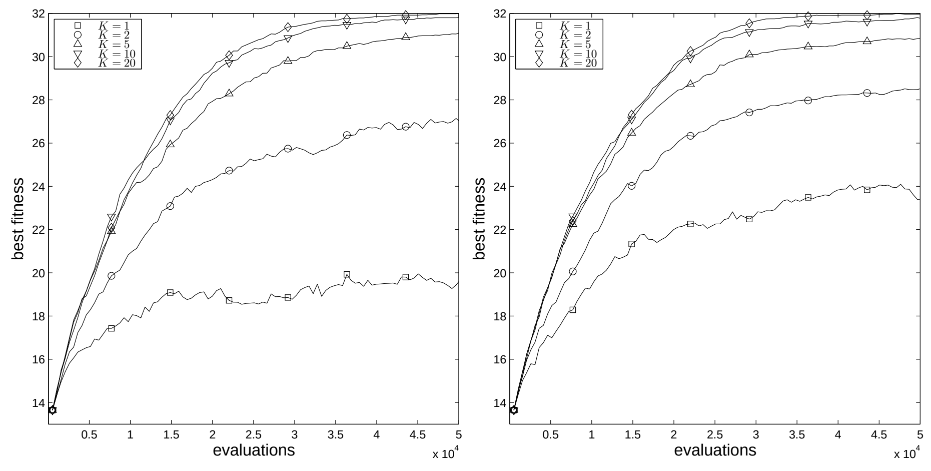
individuals, or probabilistic modeling with univariate (UMDA-like) or bivariate (COMIT-like) models respectively. As a reference we also consider an MMA without any balancing at all which we denote as noB. We have considered three test functions, namely Deb's trap (TRAP) function [6] (concatenating 32 four-bit traps), Watson et al.'s Hierarchical-if-and-only-if (HIFF) function [25] (using 128 bits) and Goldberg et al.'s Massively Multimodal Deceptive Problem (MMDP) [7] (using 24 six-bit blocks).

A global view of the results is provided by Fig. 2a. This figure shows the relationship between the mean distance to the optimum attained by each algorithm –averaged for the three problems– as a function of the churn rate (complete numerical data is provided in Table 2). Not surprisingly the distance to the optimum increases for increasing churn for all algorithms. It is however interesting to note how the different algorithms do this at a different rate. In the case of noB there is an abrupt performance degradation as we move to the right of the X axis (increasing churn) but this degradation is much more gently (rather linear actually) for variants with balancing. Among these, variants with prob-

Table 2: Results (25 runs) in terms of deviation from the optimal solution of the different MMAs on the three problems considered. The median ($\tilde{x}$), mean ($\bar{x}$) and standard error of the mean ($\sigma_{\bar{x}}$) are indicated. The three symbols next to each entry indicate whether differences are statistically significant (●) or not (○). The first symbol correspond to a comparison between the corresponding algorithm and the fault-less ($k = \infty$) version; the second one reflects a comparison with LBQ$_{\text{rand}}$ and the third one is a comparison with the algorithm that provides the best results for the corresponding problem and value of $k$ (marked with $\star$).

| | | TRAP | | | HIFF | | | MMDP | | |
|---|---|---|---|---|---|---|---|---|---|---|
| strategy | $K$ | $\tilde{x}$ | $\bar{x} \pm \sigma_{\bar{x}}$ | | $\tilde{x}$ | $\bar{x} \pm \sigma_{\bar{x}}$ | | $\tilde{x}$ | $\bar{x} \pm \sigma_{\bar{x}}$ | |
| – | $\infty$ | 0.00 | 0.55 ± 0.18 | | 0.00 | 1.00 ± 1.00 | | 1.50 | 2.08 ± 0.33 | |
| noB | 20 | 1.25 | 1.65 ± 0.39 | ●●● | 0.00 | 4.88 ± 2.05 | ○○○ | 5.99 | 5.51 ± 0.77 | ●●● |
| | 10 | 8.75 | 8.72 ± 1.09 | ●●● | 0.00 | 12.30 ± 3.11 | ●○○ | 13.48 | 15.25 ± 1.03 | ●●● |
| | 5 | 27.50 | 28.59 ± 1.49 | ●●● | 44.44 | 39.61 ± 3.28 | ●●● | 25.13 | 26.55 ± 0.69 | ●●● |
| | 2 | 48.12 | 47.49 ± 0.71 | ●●● | 61.98 | 61.51 ± 0.43 | ●●● | 38.45 | 38.02 ± 0.51 | ●●● |
| | 1 | 51.88 | 52.35 ± 0.57 | ●●● | 64.76 | 64.17 ± 0.27 | ●●● | 41.12 | 40.93 ± 0.54 | ●●● |
| LBQ$_{\text{rand}}$ | 20 | 0.00 | 0.50 ± 0.26 | ○○● | 0.00 | 2.83 ± 1.64 | ○○○ | 3.00 | 2.56 ± 0.38 | ○○○ |
| | 10 | 0.00 | 0.45 ± 0.19 | ○○$\star$ | 0.00 | 7.28 ± 2.32 | ●○○ | 4.49 | 4.71 ± 0.55 | ●○● |
| | 5 | 5.00 | 5.22 ± 0.75 | ●○● | 0.00 | 9.67 ± 2.55 | ●○$\star$ | 8.99 | 8.72 ± 0.66 | ●○● |
| | 2 | 21.88 | 22.83 ± 1.25 | ●○● | 21.88 | 21.71 ± 3.39 | ●○● | 21.80 | 21.48 ± 0.87 | ●○● |
| | 1 | 44.38 | 44.15 ± 1.04 | ●○● | 51.39 | 50.17 ± 1.79 | ●○● | 36.78 | 36.38 ± 0.57 | ●○● |
| LBQ$_{\text{umda}}$ | 20 | 0.00 | 0.00 ± 0.00 | ●●$\star$ | 0.00 | 2.44 ± 1.35 | ○○$\star$ | 1.50 | 1.89 ± 0.45 | ○○○ |
| | 10 | 0.00 | 0.60 ± 0.24 | ○○○ | 0.00 | 9.94 ± 2.81 | ●○○ | 4.49 | 3.93 ± 0.45 | ●○● |
| | 5 | 1.88 | 2.82 ± 0.57 | ●●$\star$ | 0.00 | 10.44 ± 2.36 | ●○○ | 7.49 | 7.25 ± 0.57 | ●○● |
| | 2 | 17.50 | 15.55 ± 1.02 | ●●● | 0.00 | 6.77 ± 2.27 | ●●$\star$ | 21.14 | 20.56 ± 0.83 | ●○● |
| | 1 | 38.75 | 38.76 ± 1.22 | ●●● | 35.07 | 34.01 ± 3.20 | ●●● | 37.80 | 37.28 ± 0.88 | ●○● |
| LBQ$_{\text{comit}}$ | 20 | 0.00 | 0.10 ± 0.07 | ●○○ | 0.00 | 3.28 ± 1.54 | ○○○ | 1.50 | 1.66 ± 0.37 | ○○$\star$ |
| | 10 | 0.00 | 0.70 ± 0.28 | ○○○ | 0.00 | 5.83 ± 2.17 | ●○$\star$ | 3.00 | 2.47 ± 0.40 | ○●$\star$ |
| | 5 | 3.75 | 3.60 ± 0.62 | ●○○ | 0.00 | 12.28 ± 2.80 | ●○○ | 5.99 | 5.20 ± 0.52 | ●●$\star$ |
| | 2 | 10.62 | 10.88 ± 0.82 | ●●$\star$ | 19.44 | 17.89 ± 3.08 | ●○● | 14.98 | 14.23 ± 0.83 | ●●$\star$ |
| | 1 | 27.50 | 26.92 ± 1.42 | ●●$\star$ | 21.53 | 18.95 ± 3.54 | ●●$\star$ | 32.14 | 32.88 ± 0.75 | ●●$\star$ |

abilistic modeling perform distinctly better than LBQ$_{\text{rand}}$, exhibiting a lower slope, that is, a smaller degradation for increasing churn. This is more clearly seen in Fig. 2b in which a box plot of the distribution of relative ranks (1 for the best, 4 the worst) of each algorithm on each problem and churn scenario is shown. The advantage of LBQ variants over noB as well as the advantage of LBQ$_{\text{[umda|comit]}}$ over LBQ$_{\text{rand}}$ is clearly depicted. To ascertain the significance of these differences, we have conducted the Quade test [21], obtaining that at least one algorithm is significantly different to the rest (p-value $\approx 0$). Subsequently we have conduct a post-hoc test, namely Holm test [9] (see Table 1) using LBQ$_{\text{comit}}$ as control algorithm. It is shown to be significantly different to both noB and LBQ$_{\text{rand}}$ at $\alpha = 0.05$ level. The difference between LBQ$_{\text{umda}}$ and LBQ$_{\text{comit}}$ is not statistically significant at this level on a global scale. A more fine-grained perspective is shown in Table 2. In most cases LBQ$_{\text{comit}}$ provides the best results and it is better (with statistical significance) than LBQ$_{\text{umda}}$ for the most extreme churn scenarios. This is further illustrated in Fig. 3 in which



(a) $\mathrm{LBQ}_{\mathrm{umda}}$ (b) $\mathrm{LBQ}_{\mathrm{comit}}$

Fig. 3: Evolution of best fitness in TRAP for $\mathrm{LBQ}_{\mathrm{umda}}$ and $\mathrm{LBQ}_{\mathrm{commit}}$ depending on the churn rate.

the evolution of best fitness (for the TRAP function) is shown for variants using probabilistic modeling. The performance is statistically indistinguishable for low and moderate churn (up to $K = 5$) but there is a clear superiority (statistically significant) of $\mathrm{LBQ}_{\mathrm{comit}}$ in the two most severe scenarios. This can be explained by the fact that (1) population enlarging is less frequently demanded in scenarios with low churn, and hence the particular strategy chosen bears a smaller effect on performance and (2) the better accuracy of the bivariate model for modeling the population seems to be decisive to keep the progress of the search in scenarios with severe churn.

## 5 Conclusions

Resilience is a key feature optimization techniques must exhibit in order to be successfully deployed on unstable computational environments. To this end, we have studied in this work the effect that the use of self-sampling strategies have on the performance of island-based multimemetic algorithms endowed with dynamic population balancing mechanisms. These self-sampling procedures are aimed to produce new solutions (used to enlarge the population of an island when needed) similar but not identical to other solutions in a certain population, so that the distortion on the search caused by churn is kept at a minimum. This is done by building a probabilistic model of the current population and subsequently sampling it. We have shown that the use of these strategies can effectively improve the performance of the base algorithm, maintaining better the momentum of the search particularly in situations of severe churn. We have considered two different approaches for building the probabilistic model, based on a univariate and bivariate dependencies respectively. While both approaches provide globally comparable results, the latter seems again superior when churn is high, a fact which is interpreted in light of the need of more accurate population models in such highly unstable scenarios. There are many directions for future work. In the short term we plan to extend the collection of environmental scenarios considered by including other network parameters as well as dynamically-rewired network topologies [19]. Considering more complex probabilistic models (i.e., multivariate models) is another objective which we plan to approach in the mid-term.

**Acknowledgements** This work is partially supported by MICINN 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. Alba, E.: Parallel evolutionary algorithms can achieve super-linear performance. Information Processing Letters 82, 7–13 (2002)
2. Alba, E.: Parallel Metaheuristics: A New Class of Algorithms. Wiley-Interscience (2005)
3. Albert, R., Barabási, A.L.: Statistical mechanics of complex networks. Review of Modern Physics 74(1), 47–97 (Jan 2002)
4. 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 (1997)
5. Bonet, J.S.D., Isbell, C.L., Jr., Viola, P.: MIMIC: Finding optima by estimating probability densities. In: Mozer, M., Jordan, M., Petsche, T. (eds.) Advances in Neural Information Processing Systems. vol. 9, pp. 424–430. The MIT Press (1996)
6. 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)
7. 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)
8. 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)
9. Holm, S.: A simple sequentially rejective multiple test procedure. Scandinavian Journal of Statistics 6, 65–70 (1979)
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. Liu, C., White, R.W., Dumais, S.: 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. pp. 379–386. SIGIR '10, ACM, New York, NY, USA (2010)
13. 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)
14. Mühlenbein, H., Paaß, G.: From recombination of genes to the estimation of distributions I. Binary parameters. In: Parallel Problem Solving from Nature – PPSN IV. Lecture Notes in Computer Science, vol. 1141, pp. 178–187. Springer Berlin Heidelberg (1996)
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.: 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. Springer, Berlin-Heidelberg (2015)
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. Quade, D.: Using weighted rankings in the analysis of complete blocks with additive block effects. Journal of the American Statistical Association 74, 680–683 (1979)
22. 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)
23. 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)
24. Tanese, R.: Distributed genetic algorithms. In: 3rd International Conference on Genetic Algorithms. pp. 434–439. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1989)
25. 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)