Investigating Restricted Tournament Replacement in ECGA for Non-Stationary Environments

Claudio F. Lima
DEEI-FCT
University of Algarve
Campus de Gambelas
8005-139, Faro, Portugal
clima.research@gmail.com

Carlos Fernandes[1,2]
[1]LaSEEB
Technical Univ. of Lisbon
[2]Departamento ATC
University of Granada
cfernandes@laseeb.org

Fernando G. Lobo
DEEI-FCT
University of Algarve
Campus de Gambelas
8005-139, Faro, Portugal
fernando.lobo@gmail.com

ABSTRACT

This paper investigates the incorporation of restricted tournament replacement (RTR) in the extended compact genetic algorithm (ECGA) for solving problems with non-stationary optima. RTR is a simple yet efficient niching method used to maintain diversity in a population of individuals. While the original version of RTR uses Hamming distance to quantify similarity between individuals, we propose an alternative substructural distance to enforce the niches. The ECGA that restarts the search after a change of environment is compared with the approach of maintaining diversity, using both versions of RTR. Results on several dynamic decomposable test problems demonstrate the usefulness of maintaining diversity throughout the run over the approach of restarting the search from scratch at each change. Furthermore, by maintaining diversity no additional mechanisms are required to detect the change of environment, which is typically a problem-dependent and non-trivial task.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search

General Terms

Algorithms, Experimentation, Performance

Keywords

Genetic algorithms, estimation of distribution algorithms, restricted tournament replacement, niching, diversity preservation, non-stationary environments.

1. INTRODUCTION

The extended compact genetic algorithm (ECGA) [9, 7] replaces the standard crossover and mutation operators by building a probabilistic model of promising solutions and sampling from the corresponding probability distribution. This feature allows ECGA and other advanced estimation of distribution algorithms (EDAs) [11, 14] to automatically identify the problem decomposition and important problem substructures, leading to superior performance for many problems when compared with EAs that use fixed, problem-independent variation operators.

Despite the growing interest in EDAs, the application of these algorithms to dynamic optimization problems (DOPs) has been scarce [5, 1, 16, 23, 22, 21]. Moreover, most of these works have been limited to univariate EDAs [5, 23, 22, 21], that use a probabilistic model of simple and fixed structure which assumes each variable to be independent from every other. The relevance of applying more powerful EDAs has been highlighted in recent work [1, 16], where the ECGA was modified in order to tackle problems with dynamic environments. While the previous approach is based on increasing diversity after a change, in this paper we focus on maintaining diversity throughout the run, removing the need of predicting the changes of environment.

More precisely, we investigate restricted tournament replacement (RTR) [8] as the source of continuous diversity in the ECGA. The original version of RTR uses genotypic distance to promote diversity in the population. Besides testing the traditional RTR, we propose a substructural distance that exploits model structural information. We compare the ECGA that restarts the population with the approach of maintaining diversity, testing both versions of RTR, when solving several dynamic decomposable problems of bounded difficulty [1, 20].

The paper is structured as follows. The next section provides a brief introduction to the ECGA, while Section 3 analyzes the application of ECGA to DOPs. Section 4 presents the results obtained and corresponding discussion, and Section 5 focuses on the generality of application of the proposed approach to DOPs. The paper ends with major conclusions.

2. EXTENDED COMPACT GENETIC ALGORITHM

Estimation of distribution algorithms [11, 14, 13] replace traditional variation operators of genetic algorithms (GAs) by building a probabilistic model of promising solutions (that survive selection) and sampling the corresponding probability distribution to generate the offspring population. The extended compact genetic algorithm (eCGA) [9, 7] uses a product of marginal distributions on a disjoint partition

GECCO'08, July 12–16, 2008, Atlanta, Georgia, USA.

of variables of the problem to sample new solutions,

$$p(X) = \prod_{i=1}^{m} p(X_{I_i}), \quad (1)$$

where $X = (X_1, X_2, \ldots, X_\ell)$ is a vector that contains all the variables of the problem and I_i is the index set that contains the index of the variables that belong to the i^{th} marginal distribution.

This kind of probability distribution belongs to a class of probabilistic models known as marginal product models (MPMs). For example, the following MPM, `[1,3][2][4]`, for a 4-bit problem represents that the 1^{st} and 3^{rd} variables are linked, and the 2^{nd} and 4^{th} variables are independent.

In ECGA, both the structure and the parameters of the model are searched and optimized to best fit the data (promising solutions). A greedy MPM search is performed at each generation. The greedy search algorithm starts with the simplest possible model, assuming that all variables are independent, and then keeps merging groups of variables whenever a certain score metric is improved. This process goes on until no further improvement is possible.

The score metric used in ECGA is the minimum description length (MDL) metric [15], that penalizes both inaccurate and complex models, thereby leading to a (locally) optimal distribution. More precisely, the MPM complexity is given by the sum of model complexity, C_m, and compressed population complexity, C_p. The model complexity, C_m, quantifies the model representation in terms of the number of bits required to store all the marginal probabilities,

$$C_m = \log_2(n+1) \sum_{i=1}^{m} (2^{k_i} - 1), \quad (2)$$

where n is the population size, m is the number of linkage groups, and k_i is the size of the i^{th} group.

The compressed population complexity, C_p, which quantifies the data compression in terms of the entropy of the marginal distribution over all partitions, is given by

$$C_p = n \sum_{i=1}^{m} \sum_{j=1}^{2^{k_i}} -p_{ij} \log_2(p_{ij}), \quad (3)$$

where p_{ij} is the frequency of the j^{th} gene sequence of the genes belonging to the i^{th} partition.

After learning the probabilistic model, the offspring population is generated by randomly sampling substructures, according to the probabilities stored in the MPM. More details about the pseudo-code of ECGA are given in the next section.

3. ECGA FOR NON-STATIONARY ENVIRONMENTS

In this section we review previous efforts on using ECGA to solve DOPs and introduce a new methodology to tackle these class of problems by maintaining diversity along the run.

3.1 Previous Work

Dynamic optimization problems have attracted significant interest from the evolutionary computation community in the last two decades [3, 10]. More recently, EDAs have been also applied to DOPs, although a significant part of the work has been focused in simpler univariate models [5, 23, 22, 21].

Abbass, Sastry, and Goldberg [1] were the first to introduce ECGA [9, 7] to problems with dynamic environments. Their approach is based on random restarts of the population at each change so that diversity in the population can be increased at the beginning of each new environment. This work demonstrated the utility of learning possible structural decompositions in changing environments.

ECGA with random restart

(1) Create a random population `P` of n individuals.

(2) Evaluate population `P`.

(3) If an environment change is detected:

  (3.1) Re-initialize population `P` at random.

  (3.2) Evaluate population `P`.

(4) Select `P'` individuals from `P` by tournament selection.

(5) Model the selected individuals `P'` using the greedy MPM search procedure.

(6) Sample a new population `O` according to the MPM learned in step 4.

(7) Replace all individuals in population `P` by those from `O`.

(8) If stopping criteria is not satisfied, return to step 2.

Figure 1: Steps of the extended compact genetic algorithm (ECGA) with random restart of the population for non-stationary environments.

The outline of the algorithm can be viewed in Figure 1. The algorithm is similar to the original ECGA, but in order to deal with DOPs, diversity is increased by resetting the population when a change of environment is detected. The change of environment was assumed to be known, so that the authors could focus on the ECGA response behavior to new environments when the population is restarted. Additionally, Abbass et al. proposed a slightly different approach that used the MPM from the previous generation when the population was to be restarted. Although for problems where the structure does not change with the environment this procedure is advantageous, when the environment change is related to structural changes this procedure can be harmful. Since we are interested in studying both fitness and structural changes in DOPs, we use the more general version for comparison with our approach.

The ECGA with random restart was later extended [16] to include substructural niching [17]. In substructural niching, niches are defined within the linkage groups rather than at the individual level. After the corresponding schema average fitness is calculated for each substructure [18], the sampling probabilities are changed according with their associated fitness. While this methodology can be used to maintain diversity in the population, Abbass et al. used as a way to accelerate the growth ratio of highly-fit substructures, using the random restart of the population as the main source of

diversity. Our approach is to maintain diversity throughout the run so that ECGA can be able to respond to a change without explicitly needing additional methods to detect new environments.

3.2 Restricted Tournament Replacement

When handling DOPs with evolutionary algorithms (EAs), the main approaches to deal with premature convergence can be roughly grouped into four categories [10]: increase diversity after an environment change, maintain diversity throughout the run, memory-based approaches, and multi-population approaches. While previous efforts with ECGA [1, 16] have been done in the first category, this paper follows the second approach by maintaining diversity in the population with RTR.

The restricted tournament replacement (RTR) [8] is a niching method that has been used successfully in the hierarchical Bayesian optimization algorithm (hBOA) [13]. In RTR, each new solution X is incorporated in the original population using the following procedure:

1. Select a random subset of individuals W with size w from the original population.
2. Let Y be the solution from W that is most similar to X, in terms of genotypic distance.
3. Replace Y with X, if X is better, otherwise discard X.

The window size w is set to the problem size, as suggested elsewhere [13]. The ECGA that uses RTR to address DOPs is outlined in Figure 2. After sampling the offspring population, the new individuals are incorporated one by one back into the original population using RTR. To make a fair comparison with the restart approach, we first assume that we know when the change of environment occurs, but later on (Section 5) we remove this assumption to further demonstrate the usefulness of maintaining diversity in the population. At each change, the original population is reevaluated before RTR takes place, so that fitness values are updated and subsequent comparisons are made correctly.

As mentioned before, RTR uses genotypic distance to measure similarity between individuals, which for the case of binary-coded strings is simply the Hamming distance. We propose an alternative similarity metric based on substructural distance. The substructural distance between two individuals is defined as the number of substructures in which they differ, according to the MPM structure.

Consider the following example, with the MPM `[1,2,3,4][5,6,7,8][9,10,11,12]`:

`Offspring`	`1111 1111 1111`		
`Parent1`	`0111 1011 1110`	$d_1 = 3$	$d_2 = 3$
`Parent2`	`0000 1111 1111`	$d_1 = 4$	$d_2 = 1$

The individual `Offspring` is compared with two parents ($w = 2$) to decide to which to compete for a place in the population. Using the traditional similarity measure of RTR, the Hamming distance (d_1), the closest parent is `Parent1`. However, if we use substructural distance (d_2), `Parent2` is the closest individual to `Offspring`. While `Parent2` is $d_1 = 4$ bits away from `Offspring`, the different bits belong all to the same substructure, meaning that these individuals only differ in one substructure. Therefore, and substructural-wise, these two individuals belong to the same niche.

ECGA with RTR

(1) Create a random population `P` of n individuals.

(2) Evaluate population `P`.

(3) Select `P'` individuals from `P` by tournament selection.

(4) Model the selected individuals `P'` using the greedy MPM search procedure.

(5) Sample a new population `O` according to the MPM learned in step 4.

(6) If an environment change is detected, reevaluate population `P`.

(7) Insert individuals from population `O` back into `P` using the restricted tournament replacement.

(8) If stopping criteria is not satisfied, return to step 3.

Figure 2: Steps of the extended compact genetic algorithm (ECGA) with restricted tournament replacement (RTR) for non-stationary environments.

By using substructural distance in RTR, the niches are maintained at the substructural level rather than at the individual level. In this way, the problem decomposition capability of EDAs is also exploited for diversity maintenance. This method is based on the same principle of substructural niching [17]. However, in that work explicit substructural fitness information is used, while in our case the choice between good and poor substructures is implicitly made through the individual's overall fitness.

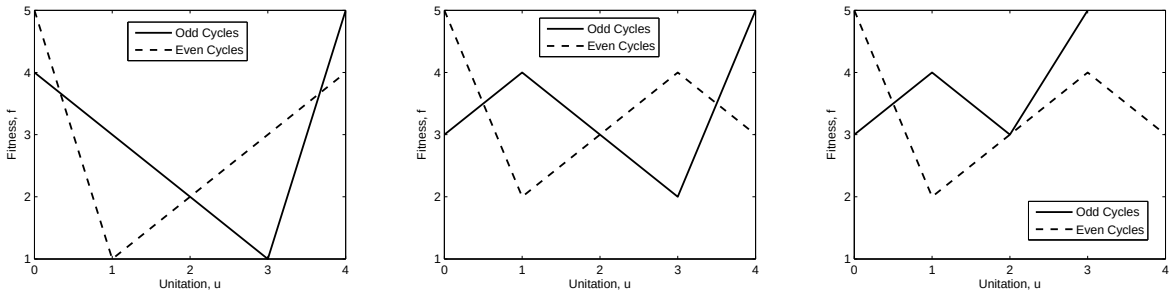
In the next section, we present the results obtained for several test functions with both versions of RTR in ECGA and compare them to the restart approach [1].

4. EXPERIMENTS

4.1 Experimental Setup

In this paper, we use the same experimental setup as used elsewhere [1] to be able to compare the different approaches. Abbass et al. [1] used dynamic versions of adversarial problems with bounded difficulty [6, 20], where the identification of the underlying problem substructure is crucial for GA success. Obviously, the solver should not have any previous knowledge about the problem structure. One such class of problems are the m-k additively decomposable problems [6], which consists of m additively separable functions of size k each, resulting in a total problem size of $\ell = m \cdot k$. The fitness is simply given by the sum of the fitness contribution of all subfunctions. Like in [1], experiments are performed for three different subfunctions. The subfunctions are defined over unitation u (number of ones) and can be visualized in Figure 3. The changes in the environment are assumed to be cyclic, where two different cycles of length 5 and 10 generations are tested.

The first function is the well-know deceptive trap function [2, 4] adapted to dynamic environments by switching between the two versions plotted in Figure 3 (a). At the beginning of the run and for odd cycles, the fitness is calculated according to the standard trap function, with the optimal



(a) Trap $k = 4$ (b) Modified trap $k = 4$ (c) Switching trap $k = 3 \leftrightarrow 4$

Figure 3: (a) Trap, (b) modified trap, and (c) switching trap subfunctions.

solution at 1111 and the local optimum at 0000. For even cycles, the trap is inverted so that the optimal and local optima switch their location. While the function plotted is for $k = 4$, we test this function with different subfunction sizes, $k = 3, 4, 5$, to analyze the behavior of the algorithms for increasing problem difficulty.

The second test function is a modified version of the previous trap function to break the symmetry in the attractors [1]. This function can be visualized in Figure 3 (b). In this case, the location of both local and optimal solutions for odd cycles does not tell anything about the optima location for the even cycles and vice-versa, therefore the algorithms cannot take advantage from previous local optima information.

The third function, called switching trap [1], promotes a more severe change between environments by changing both optima location and subfunction size. As can be seen in Figure 3 (c), the environment changes between a modified trap $k = 3$ and an inverted modified trap with $k = 4$.

For all experiments, the population size is set to $n = 5000$ and the tournament selection to $s = 16$ [1]. Each run terminates when the algorithms reach $t = 100$ generations. All the results presented through the paper are averaged over 30 independent runs.

4.2 Results and Discussion

Figures 4 and 5 present the results for the dynamic version of the trap function (Figure 3(a)). Clearly, ECGA that maintains diversity performs much better than the restart approach. Although for $k = 3, 4$, the restart approach is able to find the optimum for all problem sizes, with $k = 5$ it cannot reach the global solution for the larger problem. Even for $t = 10$, where enough time is given, the algorithm gets stuck in local optima. Eventually, ECGA with the restart approach could have solved this problem, but would require a larger population size to do so. On the contrary, ECGA with RTR benefits from diversity to require smaller population sizes for correct model building.

However, different behavior can be observed for the two RTR approaches. While the standard RTR responds quite fast to the environment change, keeping most of the times the global solution for both environments in the population, the substructural RTR takes slightly more time to respond. Nevertheless, the standard RTR does not scale as well as its substructural counterpart regarding problem difficulty, adjusted through the subfunction size k. This can be seen for $k = 5$ and $m = 20$, where it fails to obtain the best solution for the first environment. Indeed, allowing more time between environment changes ($t = 10$) deteriorate the performance.

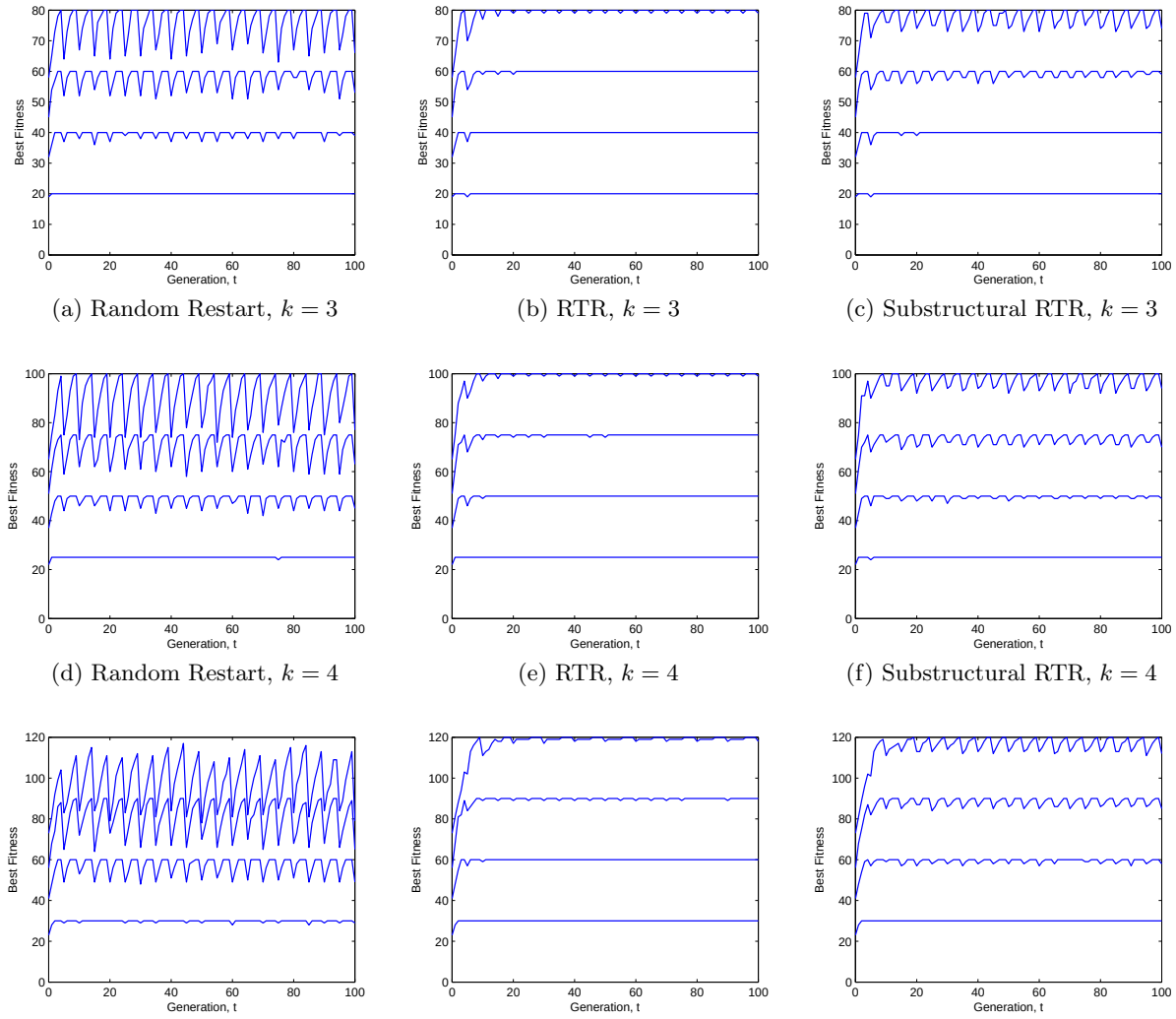
While convergence is delayed by using standard RTR, not all local optima can be preserved. Note that for a problem with m subfunctions there are a total of 2^m local optima. On the contrary, when using substructural RTR the maintenance of substructures is more tractable since only $2 \cdot m$ different locally optimal substructures need to be preserved.

The results for the modified trap with $k = 4$ are shown in Figures 6 and 7. The function setup is now adequate to break the symmetry of the attractors [1]. Again, ECGAs that use RTR outperform the restart approach. However, for $t = 10$, the standard RTR fails to obtain the optimal solution for the second environment. Contrary to the previous problem, substructural RTR is now able to maintain the global optima after reaching it, but does not suffer from the difficulties of the standard RTR, when the time between changes increases to $t = 10$.

The last function considered switches between two modified trap functions of different size, the first with $k = 3$ and the second with $k = 4$. The problem sizes $\ell = 24, 48, 72, 96$ are chosen to be compatible with both subfunction sizes. Given that for the same total problem size, a switch in the environment means a change in the number of subfunctions, the fitness of the optimal solution for each environment differs, as can be easily verified in Figures 8 and 9. For the ECGA with restart, the outcome is similar to previous problems, the algorithm being slower than diversity-based approaches, and failing to get the optimal solution for $t = 5$ and $\ell = 96$. The behavior of both RTR approaches is somewhat similar, responding relatively fast to the change of environment.

5. BLACK-BOX OPTIMIZATION IN NON-STATIONARY ENVIRONMENTS

Black-box optimization algorithms need little or no information about a problem when solving it. These type of algorithms has often been regarded as incompatible with no free lunch (NFL) theorems [19]. However, in practice we are not interest in solving all possible problems, but a much narrow subset that focus on a particular domain of interest (or problems under certain bounds of difficulty). In fact, evolutionary algorithms (EAs) themselves are more adequate for problems where some sort of structure can be exploited, whether by recombining different good solutions or through small variations of good solutions (mutation). EDAs are no



(a) Random Restart, $k = 3$ (b) RTR, $k = 3$ (c) Substructural RTR, $k = 3$

(d) Random Restart, $k = 4$ (e) RTR, $k = 4$ (f) Substructural RTR, $k = 4$

(g) Random Restart, $k = 5$ (h) RTR, $k = 5$ (i) Substructural RTR, $k = 5$

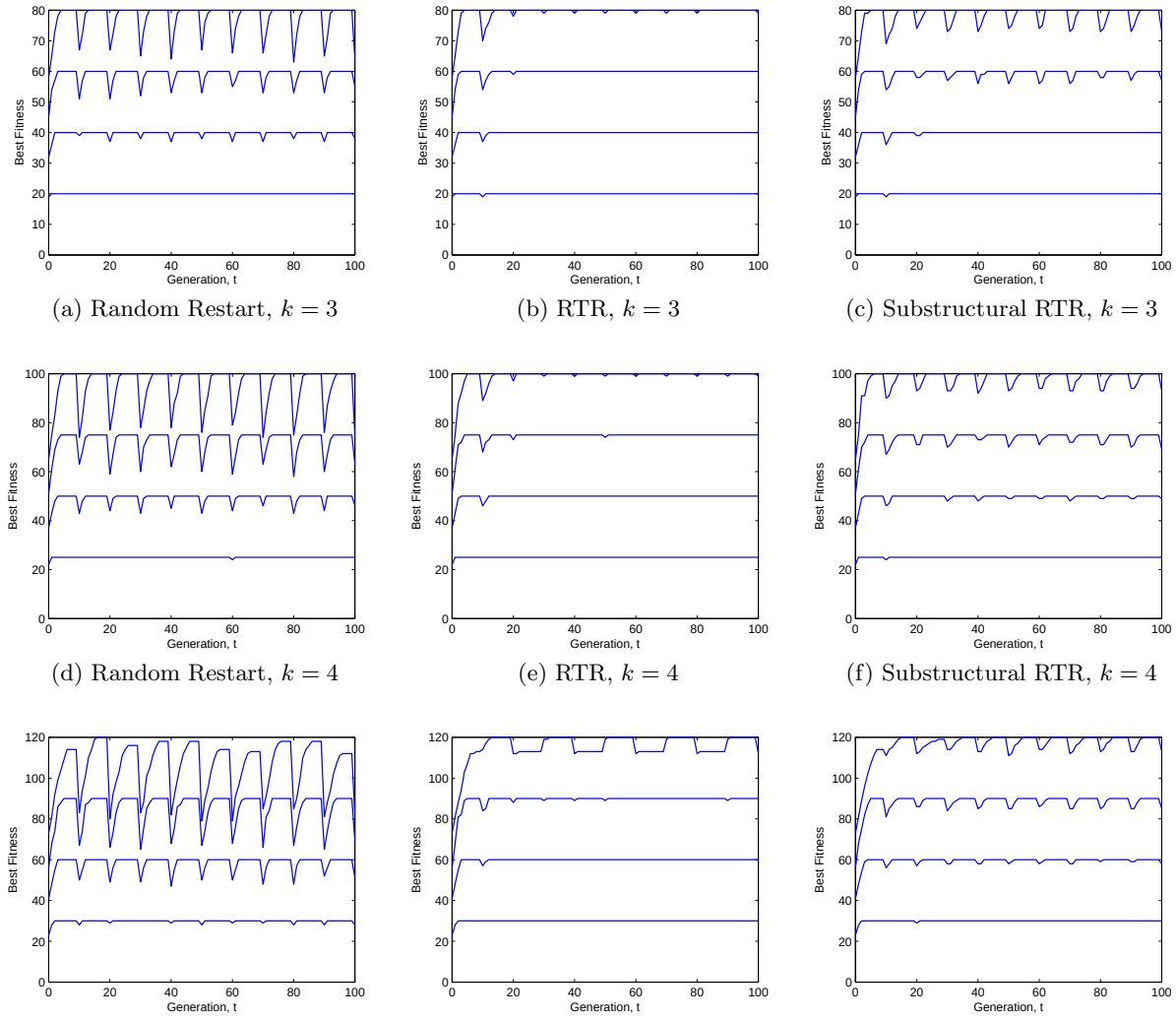
Figure 4: Results for the trap problem with $k = 3, 4, 5$, $m = 5, 10, 15, 20$, and $t = 5$. Comparison between ECGA with random restart (left), RTR (middle), and substructural RTR (right).

exception and take advantage of their mechanisms in problems where identifying the problem decomposition and important problem substructures is crucial to solve them efficiently. For example, consider a problem with size $\ell = 100$ solely composed by a trap function with $k = 100$. For this problem, random search will be on average as good as any EDA or EA in finding the global optimum (obviously without previous knowledge on the problem), simply because there is no decomposition or substructure that can be exploited.

The same argument can be made for environment changes in DOPs. If the number of environments or the period between changes varies unboundedly, on average no method (without previous knowledge) will outperform the random restart of the population at each change. Therefore, we recognize that diversity maintenance approaches are more suitable if these changes are bounded in a certain way. On the other hand, using a restart approach to ensure diversity requires the solver to have an efficient method to detect changes, which often depends on the problem itself and might not be a trivial task. On one way or the other, there is some cost associated with each approach, and in some sense a particular domain of application.

In the previous section we have assumed to know the occurrence of environment changes, as was done in previous related work [1, 16]. Here, we remove this assumption to further demonstrate the utility of using RTR to maintain diversity. Specifically, we make a simple change to the ECGA described in Figure 2. Step (6) is removed, and a new step is performed between steps (7) and (8). The step consists simply in evaluating again all individuals in population `P` that were not replaced by offspring at step (7). By doing this we keep the fitness of the preserved solutions updated and only need to reevaluate a proportion of the population.

We have performed experiments for the same problems and both versions of RTR, which demonstrated similar performance to the case where the change was assumed to be known. For example, Figure 10 shows the behavior of



(a) Random Restart, $k = 3$ (b) RTR, $k = 3$ (c) Substructural RTR, $k = 3$

(d) Random Restart, $k = 4$ (e) RTR, $k = 4$ (f) Substructural RTR, $k = 4$

(g) Random Restart, $k = 5$ (h) RTR, $k = 5$ (i) Substructural RTR, $k = 5$

Figure 5: Results for the trap problem with $k = 3, 4, 5$, $m = 5, 10, 15, 20$, and $t = 10$. Comparison between ECGA with random restart (left), RTR (middle), and substructural RTR (right).

ECGA with substructural RTR for the switching modified trap problem with $t = 5$. The corresponding proportion of reevaluated individuals in the population is also plotted. As can be observed, the algorithm behaves very similar to the case where the changes were assumed to be known. The proportion of reevaluations increases when the niches stabilized around the optima for both environments. Despite the extra cost of reevaluating a proportion of the population, the diversity maintenance reduces the population size required to solve the problems when compared with the ECGA without niching.

For a complete presentation of the results obtained under this setting the reader is referred elsewhere [12].

6. CONCLUSIONS

This paper addressed restricted tournament replacement (RTR) in the extended compact genetic algorithm (ECGA) to tackle dynamic optimization problems. Apart from the original version of RTR that uses Hamming distance to quantify similarity between individuals, we have proposed an alternative metric based on substructural niching [17]. These approaches were compared with the ECGA that randomly restarts the population after an environment change. Results on several dynamic decomposable test problems demonstrated the usefulness of maintaining diversity in the population over the approach of restarting the search from the beginning at each change. Additionally, substructural RTR was shown to be more robust than the original version. Finally, by maintaining diversity along the run, no additional mechanisms are required to detect a change of environment, which is known to be a demanding task.

Although results are encouraging, further investigation needs to be done on the scalability of the RTR approach for increasing number of different environments. A combination of substructural niching with a restart mechanism based on model information may achieve a more robust performance. This is currently topic of ongoing work.

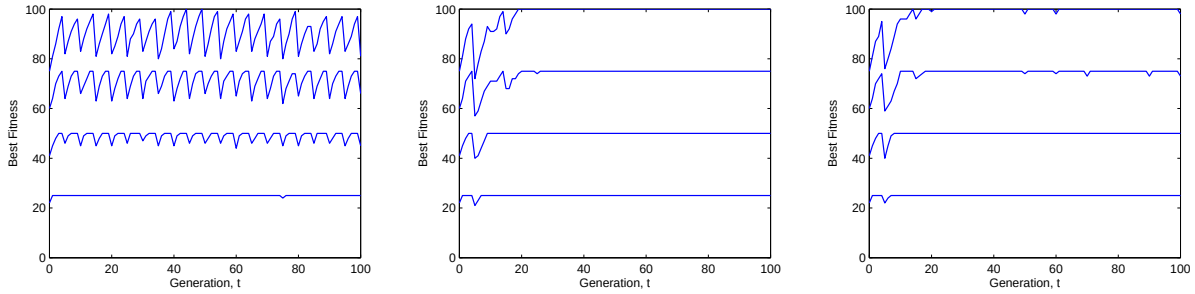


Figure 6: Results for the modified trap problem with $k = 4$, $m = 5, 10, 15, 20$, and $t = 5$. Comparison between ECGA with random restart (left), RTR (middle), and substructural RTR (right).

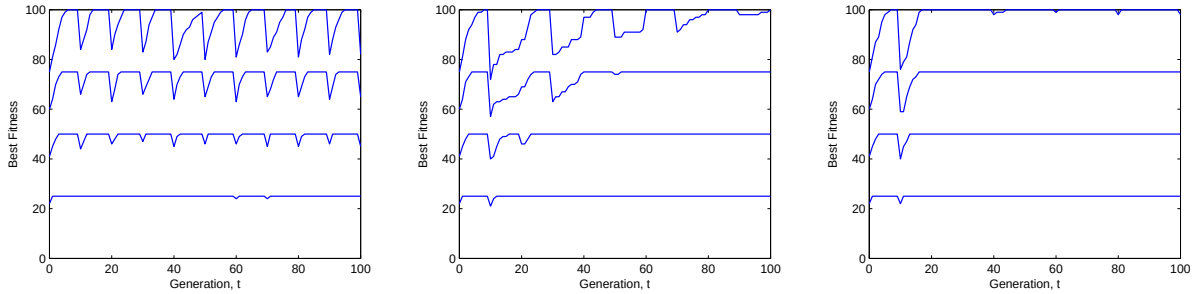


Figure 7: Results for the modified trap problem with $k = 4$, $m = 5, 10, 15, 20$, and $t = 10$. Comparison between ECGA with random restart (left), RTR (middle), and substructural RTR (right).

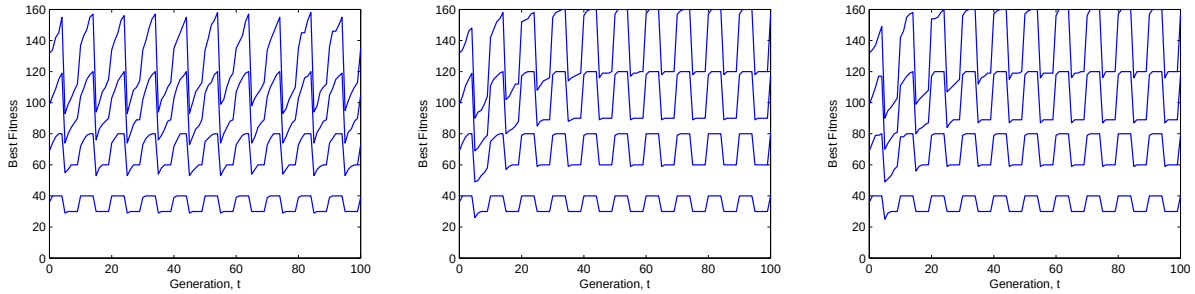


Figure 8: Results for the switching trap problem with $k = 3 \leftrightarrow 4$, $\ell = 24, 48, 72, 96$, and $t = 5$. Comparison between ECGA with random restart (left), RTR (middle), and substructural RTR (right).

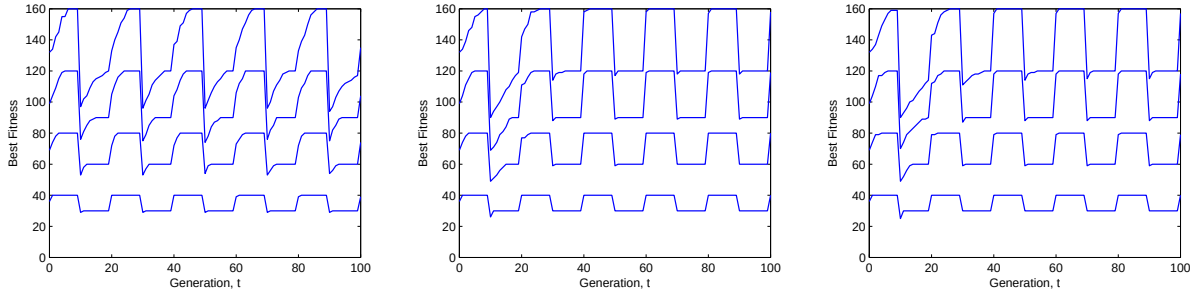


Figure 9: Results for the switching trap problem with $k = 3 \leftrightarrow 4$, $\ell = 24, 48, 72, 96$, and $t = 10$. Comparison between ECGA with random restart (left), RTR (middle), and substructural RTR (right).

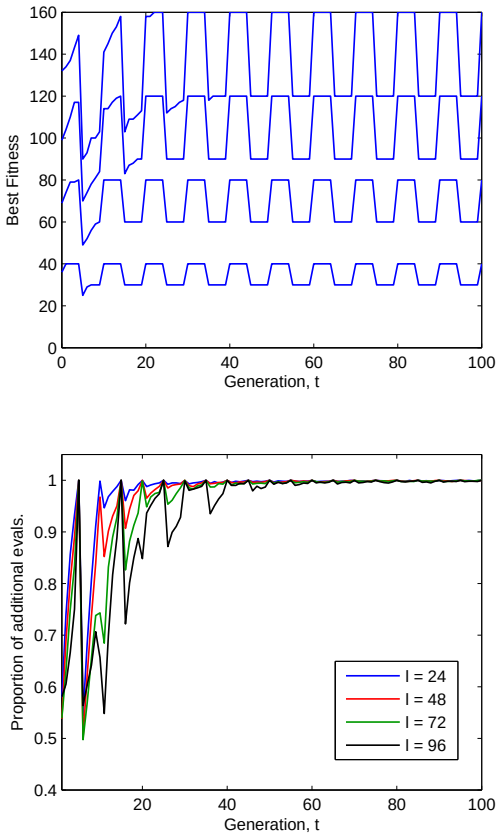


Figure 10: ECGA with substructural RTR on the switching trap problem with $k = 3 \leftrightarrow 4$, $\ell = 24, 48, 72, 96$, and $t = 5$, when the change of environment is unknown. The proportion of reevaluated individuals in the population is also shown.

Acknowledgments. This work was sponsored by the Portuguese Foundation for Science and Technology under grants SFRH-BD-16980-2004, SFRH-BD-18868-2004, and PTDC-EIA-67776-2006.

7. REFERENCES

[1] H. A. Abbass, K. Sastry, and D. E. Goldberg. Oiling the wheels of change: The role of adaptive automatic problem decomposition in non-stationary environments. IlliGAL Report No. 2004029, University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, Urbana, IL, 2004.

[2] D. H. Ackley. *A connectionist machine for genetic hill climbing*. Kluwer Academic, Boston, 1987.

[3] J. Branke. *Evolutionary Optimization in Dynamic Environments*. Morgan Kaufmann, Norwell, MA, 2001.

[4] K. Deb and D. E. Goldberg. Analyzing deception in trap functions. *Foundations of Genetic Algorithms 2*, pages 93–108, 1993.

[5] A. Ghosh and H. Muehlenbein. Univariate marginal distribution algorithms for non-stationary optimization problems. *Int. J. Know.-Based Intell. Eng. Syst.*, 8(3):129–138, 2004.

[6] D. E. Goldberg. *The Design of Innovation - Lessons from and for Competent Genetic Algorithms*. Kluwer Academic Publishers, Norwell, MA, 2002.

[7] G. Harik, F. G. Lobo, and K. Sastry. Linkage learning via probabilistic modeling in the ECGA. In M. Pelikan, K. Sastry, and E. Cantu-Paz, editors, *Scalable Optimization via Probabilistic Modeling: From Algorithms to Applications*, pages 39–61. Springer, 2006.

[8] G. R. Harik. Finding multimodal solutions using restricted tournament selection. *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 24–31, 1995.

[9] G. R. Harik. Linkage learning via probabilistic modeling in the ECGA. IlliGAL Report No. 99010, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, Urbana, IL, 1999.

[10] Y. Jin and J. Branke. Evolutionary optimization in uncertain environments—a survey. *IEEE Transactions on Evolutionary Computation*, 9(3):303–317, 2005.

[11] P. Larrañaga and J. A. Lozano, editors. *Estimation of distribution algorithms: a new tool for Evolutionary Computation*. Kluwer Academic Publishers, Boston, MA, 2002.

[12] C. F. Lima, C. Fernandes, and F. G. Lobo. Investigating restricted tournament replacement in ECGA for non-stationary environments. IlliGAL Report No. 2008007, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, Urbana, IL, 2008.

[13] M. Pelikan. *Hierarchical Bayesian Optimization Algorithm: Toward a New Generation of Evolutionary Algorithms*. Springer, 2005.

[14] M. Pelikan, D. E. Goldberg, and F. Lobo. A survey of optimization by building and using probabilistic models. *Computational Optimization and Applications*, 21(1):5–20, 2002. Also IlliGAL Report No. 99018.

[15] J. J. Rissanen. Modelling by shortest data description. *Automatica*, 14:465–471, 1978.

[16] K. Sastry, H. A. Abbass, and D. E. Goldberg. Sub-structural niching in non-stationary environments. In *Proceedings of the 17th Australian Joint Conference on Artificial Intelligence, LNAI 3339*, pages 873–885. Springer, 2004.

[17] K. Sastry, H. A. Abbass, D. E. Goldberg, and D. D. Johnson. Sub-structural niching in estimation distribution algorithms. In H. Beyer et al., editors, *Proceedings of the ACM SIGEVO Genetic and Evolutionary Computation Conference (GECCO-2005)*. ACM Press, 2005.

[18] K. Sastry, M. Pelikan, and D. E. Goldberg. Efficiency enhancement of genetic algorithms via building-block-wise fitness estimation. In *Proceedings of the IEEE International Conference on Evolutionary Computation*, pages 720–727, 2004.

[19] D. H. Wolpert and W. G. Macready. No Free Lunch Theorems for Optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997.

[20] S. Yang. Constructing dynamic test environments for genetic algorithms based on problem difficulty. In *Proceedings of the IEEE International Conference on Evolutionary Computation*, pages 1262–1269. IEEE Press, 2004.

[21] S. Yang. Experimental study on population-based incremental learning algorithms for dynamic optimization problems. *Soft Computing*, 9(11):815–834, 2005.

[22] S. Yang. Memory-enhanced univariate marginal distribution algorithms for dynamic optimization problems. In *Proceedings of the IEEE International Conference on Evolutionary Computation*, pages 2560–2567. IEEE Press, 2005.

[23] S. Yang. Population-based incremental learning with memory scheme for changing environments. In H. Beyer et al., editors, *Proceedings of the ACM SIGEVO Genetic and Evolutionary Computation Conference (GECCO-2005)*, pages 711–718. ACM Press, 2005.