A Benchmark Generator for Dynamic Permutation-Encoded Problems

Michalis Mavrovouniotis[1], Shengxiang Yang[2], and Xin Yao[3]

[1] Department of Computer Science, University of Leicester
University Road, Leicester LE1 7RH, United Kingdom
mm251@mcs.le.ac.uk
[2] Department of Information Systems and Computing, Brunel University
Uxbridge, Middlesex UB8 3PH, United Kingdom
shengxiang.yang@brunel.ac.uk
[3] CERCIA, School of Computer Science, University of Birmingham
Birmingham B15 2TT, United Kingdom
x.yao@bham.cs.ac.uk

Abstract. Several general benchmark generators (BGs) are available for the dynamic continuous optimization domain, in which generators use functions with adjustable parameters to simulate shifting landscapes. In the combinatorial domain the work is still on early stages. Many attempts of dynamic BGs are limited to the range of algorithms and combinatorial optimization problems (COPs) they are compatible with, and usually the optimum is not known during the dynamic changes of the environment. In this paper, we propose a BG that can address the aforementioned limitations of existing BGs. The proposed generator allows full control over some important aspects of the dynamics, in which several test environments with different properties can be generated where the optimum is known, without re-optimization.

1 Introduction

Over the years, there has been a growing interest for dynamic optimization problems (DOPs). However, after years of research, the field of dynamic optimization still has many open issues. One of them is the development of suitable dynamic benchmark generators (BGs) that can be easily adapted by researchers. A benchmark can be defined as standard test problems designed for the development of new algorithms and the comparison with existing ones.

A DOP can be otherwise defined as a series of several static instances. Hence, a straightforward, but not efficient, method to construct a dynamic test problem is to switch between different static instances that will cause dynamic changes. However, several general dynamic BGs have been proposed that re-shape the fitness landscape, including: 1) Moving Peaks [1]; 2) DF1 [10]; and 3) exclusive-or (XOR) [16]. The first two benchmark problems work for the continuous domain where they use functions with adjustable parameters to simulate shifting landscapes. The continuous space can be modelled as a "field of cones" [10], where

each cone is adjusted individually to represent different dynamics. A similar approach is not feasible for the combinatorial space because the landscape is indistinct and cannot be defined without reference to the optimization algorithm.

The XOR DOP generator [16] is the only dynamic BG for the combinatorial space that can convert any static binary-encoded problem (with known optimum) to a dynamic environment without affecting the global optimum value. In this way, one can see how close to the optimum each algorithm performs. A similar approach does not exist for permutation-encoded problems, such as the travelling salesman problem (TSP) and the vehicle routing problem (VRP).

In this paper, a general dynamic BG for permutation-encoded (DBGP) problems is developed which can convert a static problem instance to a dynamic one, by modifying its encoding. The experiments on some static benchmarks with known optimum show that the dynamic changes affect the algorithm but the optimum remains the same. Furthermore, the experiments show that DBGP can be directly applied to other variants of the fundamental TSP, e.g., the capacitated VRP [9].

The rest of the paper is organized as follows. Section 2 gives a summary of dynamic optimization, including the performance measurements, algorithmic methods, and BGs currently used. Section 3 describes the proposed DBGP. Section 4 presents the experimental study, in which DBGP is used to generate several dynamic test problems from static TSPs and VRPs. Finally, Section 5 concludes this paper with concluding remarks and directions for future work.

2 Background

2.1 Dynamic Combinatorial Optimization

The objective for static optimization problems is to find the optimum solution efficiently. For DOPs, the environment changes as a function of time t, which causes the global optimum to move. Hence, the objective for DOPs is to track the moving optimum efficiently. Formally, a combinatorial DOP can be defined as $\Pi = (\boldsymbol{X}, \boldsymbol{\Omega}, f, t)$, where Π is the optimization problem, $\boldsymbol{X}$ is a set of feasible solutions, $\boldsymbol{\Omega}$ is a set of constraints, f is the objective function which assigns an objective value to a solution $\boldsymbol{x}(t)$, where $\boldsymbol{x}(t) = \{x_1, \ldots, x_n\}$ is a vector of n discrete optimization variables that satisfy the constraints $\boldsymbol{\Omega}$, and t is the time.

The main aspects of "dynamism" are the frequency and the magnitude of environmental changes. The former corresponds to the speed and the latter to the degree of an environmental change, respectively. An environmental change may involve factors like the objective function, input variables, problem instance, constraints, and so on, that cause the optimum to change.

The environmental changes are classified into two types: dimensional and non-dimensional changes. Dimensional changes correspond to adding/removing variables from the problem. Such environmental changes affect the representation of the solutions and alter a feasible solution to an infeasible one. A repair operator may address this problem, but requires a prior knowledge of the problem

and the dynamic changes. Non-dimensional changes correspond to the change of the variables of the problem. Such environmental changes do not affect the representation of the solutions, and, thus, are easier to address.

2.2 Nature-Inspired Approaches in Dynamic Environments

There is a growing interest in the evolutionary computation (EC) community to apply nature-inspired algorithms to address combinatorial DOPs due to their inspiration from nature, which is a continuous adaptation process [5]. Popular examples of such algorithms are evolutionary algorithms (EAs) [6] and ant colony optimization (ACO) algorithms [2].

One common characteristic of these nature-inspired algorithms is that they are iterative. Hence, they are able to transfer knowledge from previous iterations and adapt to the new environment [1]. EAs have search operators to exchange information between a population of individuals. In ACO algorithms, ants share their pheromone trails with other ants to communicate.

The difference between an EA and an ACO algorithm lies in that the former maintains an actual population of μ solutions, whereas the latter consists of a "virtual" population. More precisely, ACO is a constructive heuristic in which all the ants deposit pheromone to mark their solution every iteration. Therefore, the information of the solutions is only kept to the pheromone trails. The constructive procedure of ACO is biased by the existing pheromone trails and some heuristic information available a priori [2].

2.3 Performance Measurements

For DOPs, it is difficult to analyze the adaptation and searching capabilities of an algorithm. This is because there is no agreed measurement to evaluate algorithms and researchers view their algorithms from different perspectives. Different measurements have been proposed to compare algorithms such as "accuracy, stability, and reactivity" [14], "collective mean fitness" [11], "accuracy and adaptability" [15], and "performance and robustness" [13].

A common method used to compare algorithms for DOPs is the best offline performance [5], which is defined as:

$$\bar{P}_B = \frac{1}{G}\sum_{i=1}^{G}\left(\frac{1}{R}\sum_{j=1}^{R}P_{ij}^{*}\right), \tag{1}$$

where G is the number of iterations, R is the number of independent runs, and P_{ij}^{*} is the fitness of the best solution of iteration i in run j after the last dynamic change. Apart from the performances which describe the best the system can do, other researchers are concerned for measurements which can characterize the population as a whole [13]. Traditionally, the target in DOPs is to track the moving optimum over time [5]. Recently, a new perspective on DOPs has been established, known as robust optimization over time (ROOT), where the

target is to find the sequence of solutions which are robust over time [17]. More precisely, a solution is robust over time when its quality is acceptable to the environmental changes during a given time interval.

Other researchers want to observe "how close to the moving optimum a solution, either robust or not, is?". Probably it is the best way to evaluate the effectiveness of an algorithm for DOPs, in addition to the time needed to converge to that optimum. However, the global optimum is needed for every changing environment and this is very challenging due to the $\mathcal{NP}$-Hardness of most combinatorial optimization problems (COPs). Since a DOP can be considered as several static instances, a direct way is to solve each one to optimality, which may be non-trivial or even impossible, especially for large problem instances. It may be possible on small problem instances, but then it will reduce the usefulness of benchmarking. Hence, the need for a BG to address the challenges of comparison is increased, but it is even harder to develop a BG for DOPs with known optimum in COPs, without re-optimization.

2.4 Benchmark Generators for DOPs

The field of dynamic optimization is related to the applications of nature-inspired algorithms [5]. The area is rapidly growing on strategies to enhance the performance of algorithms, but still there is limited theoretical work, due to the complexity of nature-inspired algorithms and the difficulty to analyze them in the dynamic domain. Therefore, the development of BGs to evaluate the algorithms is appreciated by the EC community. Such tools are not only useful to evaluate algorithms but also essential for the development of new ones.

The XOR DOP generator [16] is the only benchmark for the combinatorial space that constructs a dynamic environment from any static binary-encoded function $f(\boldsymbol{x}(t))$, where $\boldsymbol{x}(t) \in \{0,1\}^n$, by a bitwise XOR operator. It simply shifts the population of individuals into a different location in the fitness landscape. Hence, the global optimum is known during the environmental changes.

In the case of permutation-encoded problems, where $\boldsymbol{x}(t)$ is a set of numbers that represent a position in a sequence, researchers prefer their own benchmark instances to address different real-world applications, e.g., the dynamic TSP (DTSP) with exchangeable cities [4], DTSP with traffic factors [8], and dynamic VRP (DVRP) with dynamic demands.

3 Proposed Dynamic Benchmark Generator

3.1 General Framework

Most research on dynamic optimization has been done with EAs on binary-encoded COPs. Recently, ACO has been found effective on permutation-encoded DOPs, e.g., DTSP [8]. Due to the high number of specialized BGs for permutation-encoded COPs the establishment of a generalized one that converts the base of a static COP to a dynamic one is vital. Most of the existing BGs are not easily

available and they are difficult to be adapted. Moreover, on each environmental change, the fitness landscape is modified no matter whether dimensional or non-dimensional changes are applied. Hence, it is impossible to know how close to the optimum an algorithm performs on each environmental change.

Younes *et al.* [18] introduced a general benchmark framework that applies a mapping function on each permutation-encoded individual. The mapping function swaps the labels, i.e., the city index, between two objects and all the individuals in the population are treated in the same way. This way, the individuals represent different solutions after a dynamic change but the fitness landscape of the problem instance does not change. However, this generator is restricted to the range of algorithms and COPs that they are compatible with, and it is limited to the accuracy regarding the magnitude of change.

The proposed DBGP is designed to allow full control over the important aspects of dynamics and convert the base of any benchmark static COP with known optimum to a dynamic one without causing the optimum to change. Such static instances can be obtained from the TSPLIB[4] and VRPLIB[5], where most of the instances have been solved to optimality.

The basic idea of the proposed DBGP is to modify the encoding of the problem instance, instead of the encoding of each individual, i.e., the distance matrix, without affecting its fitness landscape. To illustrate such a dynamic change, let $G = (\boldsymbol{V}, \boldsymbol{E})$ be a weighted graph where $\boldsymbol{V}$ set of n nodes and $\boldsymbol{E}$ is a set of links. Each node u_i has a location defined by (x, y) and each link (u_i, u_j) is associated with a non-negative distance d_{ij}. Usually, the distance matrix of a problem instance is defined as $\boldsymbol{D} = (d_{ij})_{n \times n}$. Then, an environmental change may occur at any time by swapping the location of some node i with the location of some node j. In this way, the values in the distance matrix are re-allocated but the optimum remains the same; see Fig. 1.

The dynamic environments constructed by DBGP[6] may not reflect a full real-world situation but achieve the main goal of a benchmark in which the optimum is known during all the environmental changes. In other words, DBGP sacrifices the realistic modelling of application problems for the sake of benchmarking. Moreover, it is simple and can be adapted to any TSP and its variants to compare algorithms in dynamic environments.

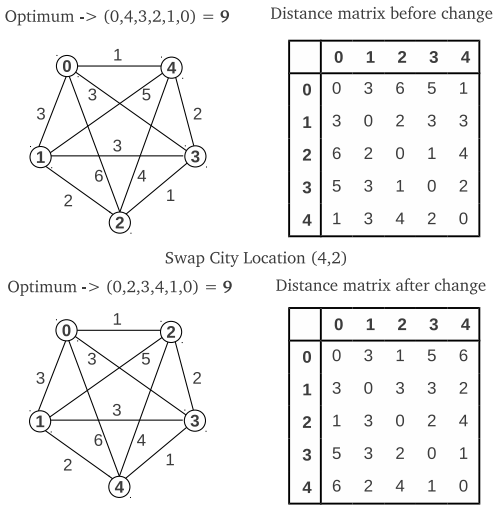
3.2 Frequency and Magnitude of Change

Every f iterations a random vector $\boldsymbol{r}(T)$ is generated that contains all the objects of a problem instance of size n, where $T = \lceil t/f \rceil$ is the index of the period of change, and t is the iteration count of the algorithm. For example, for the TSP the objects are the cities which have a location (x, y). The magnitude m of change depends on the number of swapped locations of objects.

[4] Available at `http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/`

[5] Available at: `http://neo.lcc.uma.es/radi-aeb/WebVRP/` or `http://www.or.deis.unibo.it/research_pages/ORinstances/VRPLIB/VRPLIB.html`

[6] Available at: `http://www.cs.le.ac.uk/~mm251`



	0	1	2	3	4
0	0	3	6	5	1
1	3	0	2	3	3
2	6	2	0	1	4
3	5	3	1	0	2
4	1	3	4	2	0

	0	1	2	3	4
0	0	3	1	5	6
1	3	0	3	3	2
2	1	3	0	2	4
3	5	3	2	0	1
4	6	2	4	1	0

Fig. 1. Illustration of the distance matrix with the optimum solution of the problem instance before and after a dynamic change

More precisely, $m \in [0.0, 1.0]$ defines the degree of change, in which only the first $m \times n$ of $\boldsymbol{r}(T)$ object locations are swapped. In order to restrict the swaps to the first cities, a randomly re-ordered vector $\boldsymbol{r'}(T)$ is generated that contains only the first $m \times n$ objects of $\boldsymbol{r}(T)$. Therefore, exactly $m \times n$ pairwise swaps are performed using the two random vectors starting from the first pairs. In Younes' generator [18], the magnitude of change is expressed as the number of swaps imposed on the mapping function. In this way, the objects affected from the dynamic change may not correspond to the predefined magnitude parameter. For example, if $m = 0.5$, half of the objects may be swapped with the remaining half of the objects of the optimization problem. Hence, the change affects all the objects and may be considered as $m = 1.0$.

The frequency of change is defined by the constant parameter f which is usually defined by the algorithmic iterations. However, before each environmental change, the previous pairwise swaps are reversed, starting from the end of $\boldsymbol{r}(T-1)$ and $\boldsymbol{r'}(T-1)$. In this way, the environmental changes are always applied to the encoding of the initial static problem instance.

3.3 Effect on Algorithms

DBPG can be applied to algorithms that either maintain an actual population or not, because the dynamic changes occur to the encoding of the actual problem instance. In this way, the solutions of EAs with the same encoding as before a dynamic change have a different cost after a dynamic change. On the other hand, the constructive procedure of ACO is affected since different heuristic information is generated whereas the pheromone matrix remains unchanged.

In general, DBGP shifts the population of EAs and biases the population of ACO algorithms to a new location in the fitness landscape. Younes' generator assumes that the solver has a population of solutions since the mapping function is applied on the encoding of each individual, e.g., EAs. Hence, it cannot be applied to algorithms that do not maintain an actual population, e.g., ACO.

Another advantage of the DBGP against Younes' generator [18] is that in the VRP the solutions after a dynamic change may represent an infeasible solution when EAs are used. This is because when the label of the customer changes then its demand changes, and the capacity constraint is possible to be violated. Hence, a repair operator or a penalty function has to be applied. The proposed DBGP overcomes this problem since only the location of the customer changes whereas the label and the demand remain unchanged.

3.4 Cyclic Dynamic Environments

The default dynamic environments generated by DBGP do not guarantee that any of the previously generated environment will re-appear. Such environments are called *random dynamic environments* in this paper. In fact, some algorithms that are enhanced with memory are expected to work better on dynamic environments that re-appear in the future [1]. Such environments are called *cyclic dynamic environments* in this paper, and they can be generated as follows. First, we generate K random vectors $(\boldsymbol{r}(0), \ldots, \boldsymbol{r}(K-1))$ with their corresponding re-ordered vectors as the base states in the search space. Initially, the first base state is applied. Then, every f iterations the previous dynamic changes are reversed, and then the new ones are applied from the next base state. In this way, it is guaranteed that the environments generated from the base states will re-appear.

DBGP has two options for cyclic dynamic environments regarding the way the base states are selected: 1) cyclic, where the base states are selected as in a fixed logical ring; and 2) randomly, where the base states are selected randomly.

From the above cyclic environment generator, we can further construct cyclic dynamic environments with noise as follows. Each time a new base state is to be selected, swaps are performed from the objects that are not in the $\boldsymbol{r}(T)$ with a small probability, i.e., p_{noise}. Note that the swaps occurring from the noise are reversed in the same way as with the dynamic changes above.

3.5 Varying f and m Parameters

In the random and cyclic environments, the f and m parameters remain fixed during the execution of the algorithm. An additional feature of DBGP is to vary the values of f and m with a randomly generated number with a uniform distribution in $[1, 100]$ and $[0.0, 1.0]$, respectively, for each environmental change.

4 Experimental Study

In this section, we perform some preliminary experiments based on two of the best performing ACO algorithms, i.e., $\mathcal{MAX}-\mathcal{MIN}$ Ant System ($\mathcal{MM}$AS)

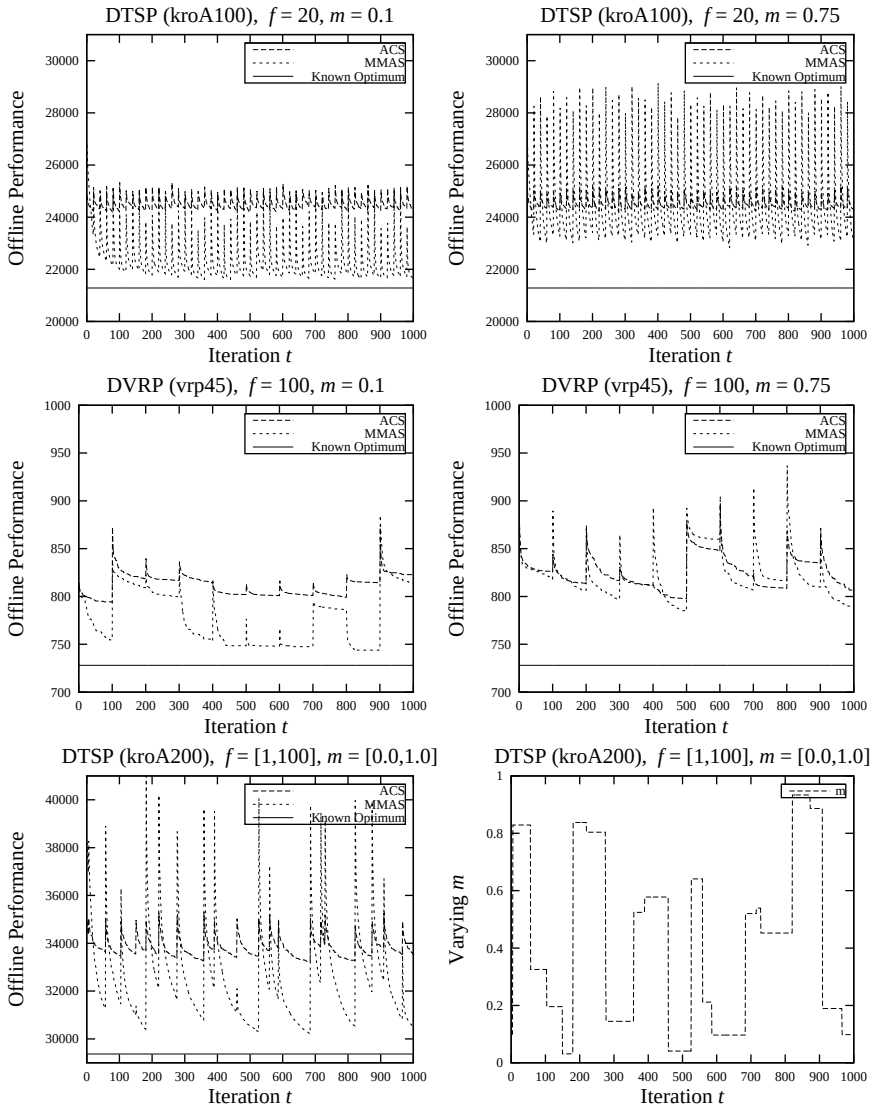


Fig. 2. Overall offline performance of ACO algorithms for different problems with different dynamic properties against the known global optimum

and Ant Colony System (ACS) [2], on two well-known COPs, i.e., TSP and VRP. From some static benchmark instances of these problems a set of dynamic test cases is generated using the proposed DBGP to test if the parameter m corresponds to the degree of a dynamic change.

In Fig. 2 we illustrate our preliminary results on dynamic test environments with different properties. The algorithms perform 1000 iterations for 30 runs and

$\boldsymbol{P}_B$, defined in Eq. (1), is chosen. On each iteration the algorithms perform the same number of function evaluations for a fair comparison. The first two graphs represent the performance of the aforementioned algorithms to a DTSP with a cyclic environment (of four base states), the middle graphs represent a DVRP with a random environment and the last ones a DTSP with varying m and f (left) and the corresponding values of m for each iteration (right).

From the experiments we can observe that $\mathcal{MM}$AS performs much closer to the optimum when $m = 0.1$, since the changing environments are similar and the knowledge transferred is useful. On the other hand, the performance of ACS is inferior to $\mathcal{MM}$AS in all dynamic test cases. The most important observation from the experiment is the reaction of algorithms to different values of m. In Fig. 2, the algorithms have a small drop in the offline performance when a change occurs with $m = 0.1$, and a large one when $m = 0.75$. This shows that DBGP defines and controls the degree of change with parameter m appropriately.

5 Conclusions and Future Work

The construction of benchmark DOP generators is important for the empirical comparison of algorithms in the EC community, due to the limited theoretical work available. This paper proposes a general BG for dynamic permutation-encoded COPs that modifies the encoding of the problem instances, to introduce dynamic changes. The proposed DBGP converts any static benchmark instance to a dynamic test environment with different properties.

In order to test DBGP, some preliminary experiments with dynamic test environments generated with DBGP for the DTSP and DVRP are carried out using ACO algorithms. From the experiments, it can be investigated how close to the optimum each algorithm converges on each environmental change. Although DBGP lacks a real-world application model, it is a simple method to empirically analyze algorithms on permutation-encoded DOPs, and can be easily adapted.

Therefore, an interesting future work is to add to the DBGP more real-world related models, such as the time-linkage property where the future behaviour of the problem depends on the current or a previous solution found by the algorithm [12]. Furthermore, it will be interesting to integrate DBGP with the ROOT framework [3, 17]. Another future work, is to test DBGP on more permutation-encoded problems, such as the capacitated arc routing problem, which is the arc counterpart of the VRP [9].

Acknowledgement

This work was partially supported by three EPSRC grants, EP/E058884/1 and EP/E060722/2 on "Evolutionary Algorithms for Dynamic Optimisation Problems: Design, Analysis and Applications", and EP/I010297/1 on "Evolutionary Approximation Algorithms for Optimisation: Algorithm Design and Complexity Analysis"

References

1. Branke, J.: Memory enhanced evolutionary algorithms for changing optimization problems. In: Proc. of the 1999 IEEE Congr. on Evol. Comput., pp. 1875–1882 (1999)
2. Dorigo, M., Stützle, T.: Ant Colony Optimization. The MIT Press, London (2004)
3. Fu, H., Sendhoff, B., Tang, K., Yao, X.: Characterising environmental changes in robust optimisation over time. In: Proc. of the 2012 IEEE Congr. on Evol. Comput., pp. 551–558 (2012)
4. Guntsch, M., Middendorf, M.: Applying population based ACO to dynamic optimization problems. In: Proc. of the 3rd Int. Workshop on Ant Algorithms, LNCS 2463, pp. 111–122 (2002)
5. Jin, Y., Branke, J.: Evolutionary optimization in uncertain environments - a survey. IEEE Trans. Evol. Comput. 9(3), 303–317 (2005)
6. Holland, J.: Adaption in Natural and artificial systems. University of Michigan Press, Ann Arbor (1975)
7. Kilby, P., Prosser, P., Shaw, P.: Dynamic VRPs: A study of scenarios, Tech. Rep. APES-06-1998, University of Strathclyde, U.K. (1998)
8. Mavrovouniotis, M., Yang, S.: Memory-based immigrants for ant colony optimization in changing environments. In: EvoApplications 2011, LNCS 6624, pp. 324–333 (2011)
9. Mei, Y., Tang. K., Yao, X.: A memetic algorithm for periodic capacitated arc routing problem. IEEE Trans. on Syst. Man and Cybern., Part B: Cybern. 41(6), 1654–1667 (2011)
10. Morrison, R.W., De Jong, K.A.: A test problem generator for non-stationary environments. In: Proc. of the 1999 IEEE Congr. on Evol. Comput., pp. 2047–2053 (1999)
11. Morrison, R.W.: Performance measurement in dynamic environments. In: Proc. of the 2003 Genetic and Evol. Comput. Conf., pp. 5–8 (2003)
12. Nguyen, T., Yao, X.: Dynamic Time-Linkage Problems Revisited. In: EvoApplications 2009. LNCS, vol. 5484, pp. 735–744 (2009)
13. Rand, W., Riolo, R.:Measurements for understanding the behavior of the genetic algorithm in dynamic environments: A case study using the shaky ladder hyperplane-defined functions. In: Proc. of the 2005 Genetic and Evol. Comput. Conf., pp. 32–38 (2005)
14. Weicker, K.:Performance measures for dynamic environments. In: Proc. of the 7th Int. Conf. on Parallel Problem Solving from Nature, LNCS 2439 pp. 64–73 (2002)
15. Trojanowski, K., Michalewicz, Z.: Evolutionary algorithms for non-stationary environments. In: Proc. of 8th Workshop on Intelligent Information Systems, pp. 229–240 (1999)
16. Yang, S.: Non-stationary problem optimization using the primal-dual genetic algorithm. In: Proc. of the 2003 IEEE Congr. on Evol. Comput., pp. 2246–2253 (2003)
17. Yu, X., Jin, Y., Tang, K., Yao, X.: Robust optimization over Time – A new perspective on dynamic optimization problems. In: Proc. of the 2010 IEEE Congr. on Evol Comput., pp. 3998–4003, 2010.
18. Younes, A.,Calamai, P., Basir, O.: Generalized benchmark generation for dynamic combinatorial problems, In: Proc. of the 2005 Genetic and Evol. Comput. Conf., pp. 25–31 (2005)