An Analysis of Dynamic Severity and Population Size

Karsten Weicker

University of Stuttgart, Institute of Computer Science,
Breitwiesenstr. 20–22, 70565 Stuttgart, Germany,
email: Karsten.Weicker@informatik.uni-stuttgart.de

Abstract. This work introduces a general mathematical framework for non-stationary fitness functions which enables the exact definition of certain problem properties. The properties' influence on the severity of the dynamics is analyzed and discussed. Various different classes of dynamic problems are identified based on the properties. Eventually, for an exemplary model search space and a $(1, \lambda)$-strategy, the interrelation of the offspring population size and the success rate is analyzed. Several algorithmic techniques for dynamic problems are compared for the different problem classes.

1 Introduction

Dynamic optimization problems are an area of increasing importance and increasing interest. This is reflected in the number of publications at recent conferences as well as in emerging applications in industry, e.g. control tasks and dynamic scheduling. In research, dynamic problems are primarily used to demonstrate adaptation of evolutionary algorithms and to motivate manifold different techniques to cope with dynamics. Nevertheless, we observe a significant lack of established foundations to examine and compare problems as well as techniques.

This work presents one possible approach to reach a common framework for experimental as well as theoretical examinations of different dynamic problems. The framework is used in this work for a theoretical analysis of the influence of population size and certain techniques to cope with dynamic problems on the success rate of algorithms based on local mutation operators.

2 Related Work

Optimization of dynamic problems is primarily driven by the experimental analysis of special optimization techniques on different problems. One classical benchmark function is the moving peaks problem which was recently implemented differently in various problem generators (Grefenstette, 1999; Branke, 1999b; Morrison & De Jong, 1999). Those problems are usually determined by the position of the peaks and their height and width. Nevertheless the parameterization of the dynamics differs from problem generator to problem generator. Also there

is no common basis for the comparison of different dynamic function optimization problems. An overview on dynamic problems as well as techniques is given in (Branke, 1999a).

Two standard techniques used to improve evolutionary optimization of dynamic problems are memory for previous solutions and an increase of the population diversity. Memory can be introduced either for each individual by using a diploid representation (e.g. Goldberg & Smith, 1987; Lewis, Hart, & Ritchie, 1998) or by an explicit global memory (e.g. Mori, Imanishi, Kita, & Nishikawa, 1997; Branke, 1999b; Trojanowski & Michalewicz, 1999). One possibility to increase the diversity are random immigrants within a hypermutation operator (Cobb & Grefenstette, 1993; Cobb, 1990; Grefenstette, 1999).

One focus of this work is the analysis of the population size for dynamic problems and local mutation operators. The population size was already often target of various examinations. De Jong and Spears (1991) and Deb and Agrawal (1999) analyzed the interaction of population size and genetic operators. Mahfoud (1994) derived a lower bound for the population size in order to guarantee niching in sharing techniques. Miller (1997) deduced optimal population sizes for noisy fitness functions. An overview on topics in population sizing can be found in (Smith, 1997).

Other theoretical investigations of evolutionary algorithms and dynamic problems may be found in Rowe (1999) and Ronnewinkel, Wilke, and Martinetz (2000).

3 Dynamic Problem Framework

This section introduces a general framework to describe and characterize dynamic fitness functions. The goal of this framework is to establish a basis for comparison and classification of non-stationary functions as well as for theoretical results on problem hardness and algorithms' power. Such results are possible since the framework enables the exact definition of problem properties.

The basic idea of the following definition is that each dynamic function consists of several static functions where for each static function a dynamic rule is given. The dynamics rule is defined by a sequence of isometric, e.g. distance preserving, coordinate transformations and fitness rescalings. The possible coordinate transformations are translations and rotations around a center.

Definition 1 (Dynamic fitness function (maximization)). *Let Ω be the search space with distance metric $d : \Omega \times \Omega \to \mathbb{R}$. A* dynamic fitness function *$F \equiv \left(F^{(t)}\right)_{t\in\mathbb{N}}$ with $F^{(t)} : \Omega \to \mathbb{R}$ for $t \in \mathbb{N}$ is defined by $n \in \mathbb{N}$ components consisting of a static fitness function $f_i : \Omega \to \mathbb{R}$ $(1 \leq i \leq n)$ with optimum at $\mathbf{0} \in \Omega$, $f_i(\mathbf{0}) = 1$, and a dynamics rule with*

- *coordinate transformations $\left(c_i^{(t)}\right)_{t\in\mathbb{N}}$ with $c_i^{(t)} : \Omega \to \Omega$ where $d(c_i^{(t)}(\omega_1), c_i^{(t)}(\omega_2)) = d(\omega_1, \omega_2)$ for all $\omega_1, \omega_2 \in \Omega$ and*
- *fitness rescalings $\left(s_i^{(t)}\right)_{t\in\mathbb{N}}$ with $s_i^{(t)} \in \mathbb{R}_0^+$.*

The resulting dynamic fitness function is defined as

$$F^{(t)}(\omega) = \max\left\{S_1^{(t)} f_1(C_1^{(t)}(\omega)), \ldots, S_n^{(t)} f_n(C_n^{(t)}(\omega))\right\}$$

where $C_i^{(t)} = C_i^{(0,t)}$ *and* $C_i^{(t_1,t_2)} = c_i^{(t_2)} \circ \ldots \circ c_i^{(t_1+1)}$ *are the accumulated coordinate transformations and* $S_i^{(t)} = S_i^{(0,t)}$ *and* $S_i^{(t_1,t_2)} = \prod_{t=t_1+1}^{t_2} s_i^{(t)}$ *are the accumulated fitness rescalings.*

The placement of each optimum at position **0** and the initial optimal fitness 1 has been chosen for simplification of the analysis only. It does not put any constraint on the function since the rescaling of the fitness value and the positioning may be achieved by the first dynamics rule. Furthermore it is assumed that at each time step only one of the component fitness functions attains the optimal fitness value, i.e. there is at each time step only one global optimum.

Before problem properties are considered in detail a few examples are given how problems from recent publications may be studied in the given framework. One example are the moving peaks problems (e.g. Branke, 1999b; Morrison & De Jong, 1999) which can be realized by one component function for each peak and an additional function $f_0 \equiv 0$. Then, the motion of the peaks can be given by the coordinate transformations and the peak heights may be changed by fitness rescaling. Note, that this framework allows no random influences, i.e. each fitness function produced by above paper's problem generators defines a new dynamic fitness function within the framework. The fitness function by Trojanowski and Michalewicz (1999) divides the search space in different segments which hold each a peak where the height of the peaks is changing according to a schedule. In this framework the static functions may be defined on the according segment only. Where most other dynamic problems exhibit only coordinate translations, Weicker and Weicker (1999) presented a fitness function with rotation as coordinate transformations which may be reproduced easily within the framework.

Using Definition 1 several problem properties may be identified which influence the hardness of a dynamic problem. The following definition formalizes a few basic problem properties inherent in the dynamics of the problem.

Definition 2 (Basic problem properties). *Let* $F_i = \left(f_i, (c_i^{(t)})_{t\in\mathbb{N}}, (s_i^{(t)})_{t\in\mathbb{N}}\right)$ *and* $F_j = \left(f_j, (c_j^{(t)})_{t\in\mathbb{N}}, (s_j^{(t)})_{t\in\mathbb{N}}\right)$ *be two components of a dynamic fitness function* F*. Then the following properties are defined with respect to the coordinate transitions and a set of time steps* $T \subseteq \mathbb{N}$*.*

- *with constant dynamics:* $const_c(F_i)$ *iff* $\forall_{t\in T}\ c_i^{(t)} = c_i^{(t+1)}$
- *stationary:* $stat_c(F_i)$ *iff* $\forall_{t\in T}\ c_i^{(t)} = id$
- *weak periodic:* $wperiod_c(F_i)$ *iff* $\forall_{t_1,t_2\in T, t_1<t_2}\ C_i^{(t_1,t_2)} = id$
- *homogeneous:* $homo_c(F_i, F_j)$ *iff* $\forall_{t\in T}\ c_i^{(t)} = c_j^{(t)}$

The properties $const_s(F_i)$*,* $stat_s(F_i)$*,* $wperiod_s(F_i)$*, and* $homo_s(F_i, F_j)$ *are analogously defined with respect to the fitness rescalings where* c *is replaced by* s *and* id *by* 1*. Additionally the following property is defined using the fitness rescalings.*

– *alternating:* $alter_s(F_i)$ *iff* $\forall_{t \in T}\ 1 < \frac{S_j^{(t)}}{S_i^{(t)}} < \frac{s_i^{(t+1)}}{s_j^{(t+1)}}$

In general, a dynamic fitness function is denoted to have constant dynamics if it holds that $\forall_{1 \leq i \leq n}\ (const_c(F_i) \wedge const_s(F_i))$. This is also true for the other defined properties.

The next section defines certain severity measures and analyzes how the severity is affected by the problem properties.

4 Problem Properties and Severity

This section presents in the following definition four different severity measures and discusses the influence of the basic problem properties on the severity in the remainder of the section.

Definition 3 (Severity of dynamics). *Let Ω be a search space, $dia \in \mathbb{R}$ the maximal distance within Ω, and F a dynamic fitness function consisting of components $\left(f_i, (c_i^{(t)})_{t \in \mathbb{N}}, (s_i^{(t)})_{t \in \mathbb{N}}\right)$ where $1 \leq i \leq n$. Then the following severity measures are defined*

– *minimal general severity:* $minsev(F^{(t)}) = \min_{\omega \in \Omega} maxcomp(\omega)$
– *maximal general severity:* $maxsev(F^{(t)}) = \max_{\omega \in \Omega} maxcomp(\omega)$
– *average general severity:* $avgsev(F^{(t)}) = \frac{1}{|\Omega|} \sum_{\omega \in \Omega} maxcomp(\omega)$

where $maxcomp(\omega) = \max_{1 \leq i \leq n} \frac{d(c_i^{(t-1)}(\omega), c_i^{(t)}(\omega))}{dia}$. *Moreover, the severity of the optimum is defined as* $optsev(F^{(t)}) = \frac{d(opt^{(t-1)}, opt^{(t)})}{dia}$ *where for each* $opt^{(t)}$ *there exists a* i $(1 \leq i \leq n)$ *such that* $opt^{(t)} = C_i^{(t)}(\mathbf{0})$ *and for all* $j \in \{1, \ldots, n\} \setminus \{i\}$ *it holds that* $S_i^{(t)} f_i(C_i^{(t)}(\mathbf{0})) > S_j^{(t)} f_j(C_j^{(t)}(\mathbf{0}))$.

First, the influence of the problem properties on the general severity measures is analyzed. The general severity is determined by the transformations in the coordinate space only. As long as there are only linear translations in a homogeneous problem from one generation to the next, the average severity equals both the minimal and the maximal severity for each component function. In an inhomogeneous problem the differences are determined by the differences in component functions. As soon as a rotating transformation is introduced the minimal and the maximal severity differ within one component function since the minimal severity can be zero (in case of rotation only) and the maximal severity can be rather large depending on the rotation angle and the size of the search space.

Nevertheless the severity of the optimum is not only determined by the general severity but also by the property "alternating" of the fitness rescalings. If a problem is alternating at a time step t the optimum is changing from one component function to another component function, i.e. the severity is not predictable from the coordinate transformations only. Note that the

alternating property is strongly correlated to the stationarity and homogeneity ($stat_s \vee homo_s \Rightarrow \neg alternate$)

The following problem classes are identified for complete problems as well as periods of problems. The ordering does not imply a strict increase in problem difficulty.

Class 0 (both parts stationary): static optimization task (severity 0)

Class 1 (constant and homogeneous coordinate translations, not alternating): pure tracking task with constant severity, i.e. the static fitness landscape is moving as a whole

Class 2 (constant and inhomogeneous coordinate translations, not alternating): pure tracking task with constant severity but in a changing environment

Class 3 (inconstant coordinate translations, no jumping): either not so easily predictable tracking task or rather chaotic or random problem

Class 4 (rotating coordinate translations, not alternating): tracking task with different inherent degrees of severity

Class 5 (alternating, stationary coordinate translations): oscillation between several static optima, the severity depends on the position of the component functions' optima

Class 6 (alternating, non-stationary coordinate translation): tracking and determination of moving and oscillating optima, the severity depends on the component functions' dynamics

For each problem class different optimization techniques are useful. In the next section the interdependencies are analyzed between the population size and some memorizing rsp. diversity preserving techniques. The impact of these results on the optimization of the problem classes is discussed at the end of the next section.

5 Analysis of Techniques and Population Size

This section analyzes one generation of an optimizer in a dynamic environment. In particular, the influence of the population size on the probability to get close to the next position of the optimum is of interest.

The analysis uses an abstract problem as well as an abstract optimizer. The problem is only considered as far as the problem properties above are concerned. For the algorithm only some underlying working principles are assumed instead of analyzing a concrete instance of evolutionary computation.

The search space is a two-dimensional raster 1000×1000. We assume the current optimum to be placed close to the center such that no newly created individual by a local mutation lies outside the search space. Furthermore we consider a distance metric which is defined by vertical and horizontal crossings of raster boundaries (cf. Figure 1). Simplifying it is assumed that the optimum is moved horizontally.

In the analysis the following numbers are of interest. $4i$ points have exactly distance i to a given point and the number of points which are at most i steps off is $2i(i+1)+1$.

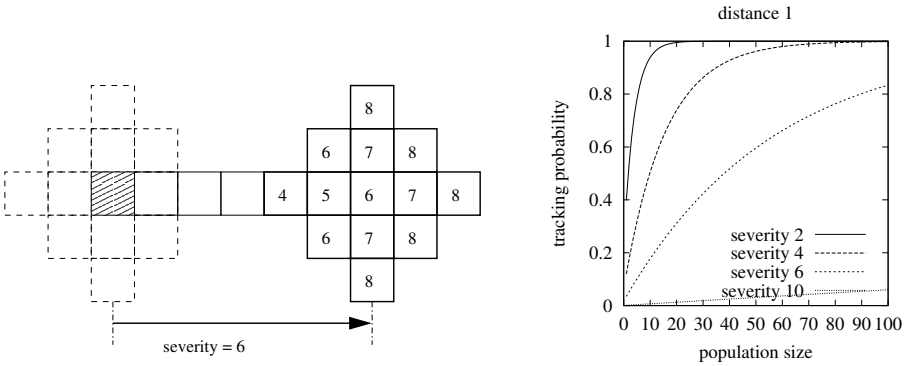

Fig. 1. Left: The hatched point marks the position of the optimum in the last generation which was now moved horizontally six steps. The region on the right denotes those points which are within a distance 2 of the new optimum's position. The numbers in the squares denote the distance to the previous optimum. Right: Probability to get close to the optimum (within distance 1) for an algorithm with local mutation only.

First, a pure mutation based local search algorithm is analyzed. It is assumed that the algorithm performs a mutation with step size $1 \leq i \leq 20$ with a probability according to the binomial distribution.

$$\Pr[i] = \frac{1}{2^{40}-1}\binom{41}{20+i} \tag{1}$$

The maximal step size is 20. Moreover, each mutation reaches a new point in the search space ($\Pr[0] = 0$). Given that the parent individual is situated at last generations optimum which moves by severity s, the number of points within a distance d from the new optimum and an exact distance $s - d \leq d' \leq s + d$ equals

$$\mathrm{N}_{d,s}(d') = 2d + 1 - 2\left\lfloor \frac{s+d+1-d'}{2} \right\rfloor$$

(cf. left part of Figure 1). Then, the probability to hit a point within distance d from the new optimum by creating one offspring equals

$$\Pr[< d(s)] = \sum_{s-d \leq d' \leq s+d} \mathrm{N}_{d,s}(d')\Pr[d']$$

Note, that no recombination is considered in the scope of this work. Assumed that the algorithm uses a $(1, \lambda)$ strategy, the probability that at least one offspring is within a distance d of the optimum results as

$$\Pr[< d(s) \text{ of } \lambda] = 1 - (1 - \Pr[< d(s)])^{\lambda}$$

The results in Figure 1 (right) show the dependence of the tracking probability of the population size. With increasing severity of the optimum a large

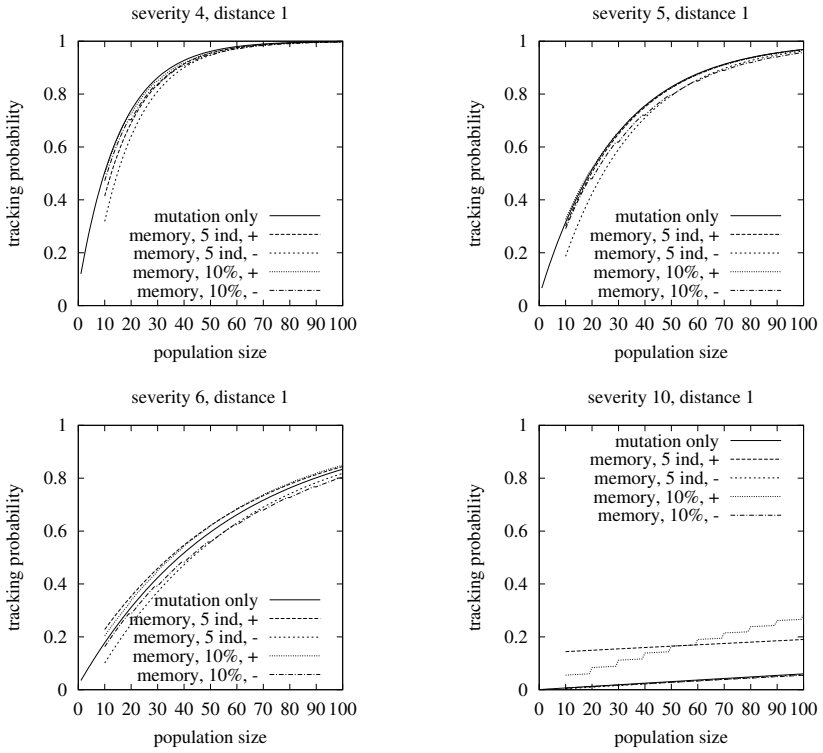

Fig. 2. Results for using an external memory for previous good solutions. Individuals from the memory are inserted into the offspring population. "+" denotes that the optimum is contained in the memory and "−" the contrary.

population size becomes more important. This result is especially of interest for pure tracking tasks like in class 1 and class 2 problems. It gives a guideline for choosing an appropriate population size. In some algorithms with local mutation an adaptation of the step size is used (e.g. in evolution strategies). However, this is not considered in this analysis. Note, that the results do not scale with increasing step size since the basic probability for hitting a point in the search space decreases. For problems of class 3 or 4 the determination of the population size is difficult since the severity is varying. For the same reason (self-)adaptive techniques must be questioned critically for class 3 and 4.

Second, an external memory is added to the algorithm in which certain good solutions from previous generations are stored. A memory size of 40 individuals is assumed. In order to analyze the behavior with memory two cases are distinguished: the successful case where the optimum (or a point close to the optimum) is contained in the memory and the unsuccessful case. Furthermore, two replacement strategies are considered. On the one hand, each generation 5 individuals are chosen randomly from the memory and inserted into the off-

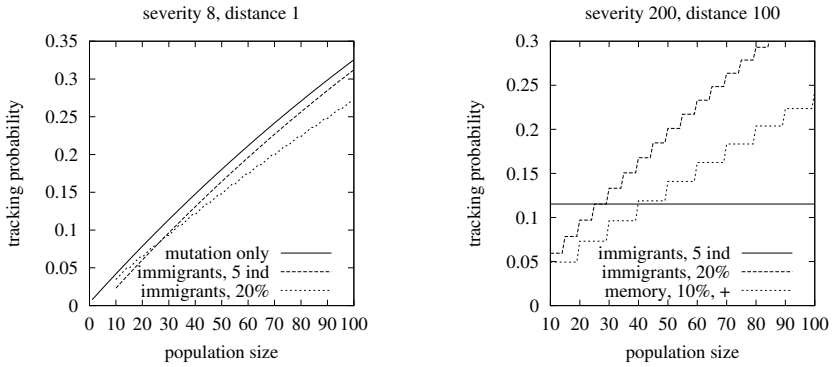

Fig. 3. Results for introducing random immigrants into the population: small severity and required minimal distance 1 to the optimum (left) and high severity and required minimal distance 100 to the optimum (right)

spring population and, on the other hand, 10% of the offspring population are chosen from the memory. Now the probability from above can be modified for the successful case with the replacement number $k \in \left\{5, \lfloor \frac{\lambda}{10} \rfloor\right\}$

$$\Pr[< d(s) \text{ of } \lambda(\text{memory})] = 1 - (1 - \Pr[< d(s)])^{\lambda - k} \left(\frac{39}{40}\right)^k$$

and for the unsuccessful case

$$\Pr[< d(s) \text{ of } \lambda(\text{memory})] = 1 - (1 - \Pr[< d(s)])^{\lambda - k}.$$

Figure 2 shows the results for the algorithm with memory. Obviously with low severity the tracking probability worsens even in the successful cases with small population sizes. But already with severity 5 the algorithm replacing 10% of the population shows small advantages. Nevertheless, the worsening in case of memory without the optimum are still bigger as the gains in the successful case. If only small severity values occur, memory seems only to be useful for problems of class 5 since there is a high chance that the optimum is contained in the memory. However, as soon as bigger severity values occur the diagrams show almost negligible worsenings in the unsuccessful case and significant improvements in the successful case. As a consequence memory is always useful in problems of classes 5 and 6 with high severity even if the memory of problems in class 6 might have a rather small chance of containing the optimum.

Eventually, the method of random immigrants is analyzed in this theoretical setup on its direct influence in finding the optimum or getting close to the optimum. We consider again two scenarios: on the one hand 5 random individuals are injected and on the other hand 20% of the population are replaced randomly. The tracking probability can be derived similarly to the previous case where the basic probability depends on the size of the search space and the minimal allowed distance to the optimum. The results are shown in Figure 3. As expected

this method shows a significant drawback with low severity and small expected distance to the optimum. Nevertheless, with high severity and a large allowed distance to the optimum this method can help to increase the probability considerably. This shows that the method is indeed able to find the correct region with a certain probability. If we still assume in the memory model that only one individual out of 40 is a successful individual, the random immigrants are able to beat the memory model in the successful case. The assumption on the memory is not unlikely since the tracking region covers only approx. 2% of the search space. This result can explain the good performance of the mechanism in certain problems – usually with recombination which helps to combine several individuals getting anywhere close to the optimum. With respect to the problem classes, this method should be considered for class 6 with a rather high severity since it has a guaranteed improvement in contrary to the memory approach.

6 Conclusion

This work presents a framework for the classification and comparison of dynamic problems. This framework is used for an analysis how the offspring population size and two special techniques for dynamic problems affect the tracking probability of a search algorithm based on a local mutation operator. Within the context of a $(1, \lambda)$-strategy, the analysis gives concrete information for which problem class which algorithmic technique and which population size should be at least considered. Since especially in dynamic problems the populations size can be a critical parameter – more evaluations can impose higher dynamics – more investigations are necessary in that direction.

Besides the concrete results in this work, the framework for dynamic problems enables more theoretical investigations of correlations between problem properties, parameter settings, algorithmic techniques, and adaptation measures. In particular, future work will consider more complex models of algorithms incorporating recombination operators and other performance measures covering different aspects of adaptation.

References

Branke, J. (1999a). Evolutionary approaches to dynamic optimization problems: A survey. In J. Branke & T. Bäck (Eds.), *Evolutionary algorithms for dynamic optimization problems* (pp. 134–137). (part of GECCO Workshops, A. Wu (ed.))

Branke, J. (1999b). Memory enhanced evolutionary algorithms for changing optimization problems. In *1999 Congress on Evolutionary Computation* (pp. 1875–1882). Piscataway, NJ: IEEE Service Center.

Cobb, H. G. (1990). *An investigation into the use of hypermutation as an adaptive operator in genetic algorithms having continuous, time-dependent nonstationary environments* (Tech. Rep. No. 6760 (NLR Memorandum)). Washington, D.C.: Navy Center for Applied Research in Artificial Intelligence.

Cobb, H. G., & Grefenstette, J. J. (1993). Genetic algorithms for tracking changing environments. In S. Forrest (Ed.), *Proc. of the Fifth Int. Conf. on Genetic Algorithms* (pp. 523–530). San Mateo, CA: Morgan Kaufmann.

De Jong, K. A., & Spears, W. M. (1991). An analysis of the interacting roles of population size and crossover in genetic algorithms. In H.-P. Schwefel & R. Männer (Eds.), *Parallel Problem Solving from Nature: 1st Workshop, PPSN I* (pp. 38–47). Berlin: Springer.

Deb, K., & Agrawal, S. (1999). Understanding interactions among genetic algorithm parameters. In W. Banzhaf & C. Reeves (Eds.), *Foundations of Genetic Algorithms 5* (pp. 265–286). San Francisco, CA: Morgan Kaufmann.

Goldberg, D. E., & Smith, R. E. (1987). Nonstationary function optimization using genetic algorithms with dominance and diploidy. In J. J. Grefenstette (Ed.), *Proc. of the Second Int. Conf. on Genetic Algorithms* (pp. 59–68). Hillsdale, NJ: Lawrence Erlbaum Associates.

Grefenstette, J. J. (1999). Evolvability in dynamic fitness landscapes: a genetic algorithm approach. In *1999 Congress on Evolutionary Computation* (pp. 2031–2038). Piscataway, NJ: IEEE Service Center.

Lewis, J., Hart, E., & Ritchie, G. (1998). A comparison of dominance mechanisms and simple mutation on non-stationary problems. In A. E. Eiben, T. Bäck, M. Schoenauer, & H.-P. Schwefel (Eds.), *Parallel Problem Solving from Nature – PPSN V* (pp. 139–148). Berlin: Springer. (Lecture Notes in Computer Science 1498)

Mahfoud, S. W. (1994). *Population sizing for sharing methods* (Tech. Rep. No. IllIGAL 94005). Urbana, IL: Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign.

Miller, B. L. (1997). *Noise, sampling, and efficient genetic algorithms.* Unpublished doctoral dissertation, University of Illinois at Urbana-Champaign, Urbana, IL. (IllIGAL Report No. 97001)

Mori, N., Imanishi, S., Kita, H., & Nishikawa, Y. (1997). Adaptation to changing environments by means of the memory based thermodynamical genetic algorithm. In T. Bäck (Ed.), *Proc. of the Seventh Int. Conf. on Genetic Algorithms* (pp. 299–306). San Francisco, CA: Morgan Kaufmann.

Morrison, R. W., & De Jong, K. A. (1999). A test problem generator for non-stationary environments. In *1999 Congress on Evolutionary Computation* (pp. 2047–2053). Piscataway, NJ: IEEE Service Center.

Ronnewinkel, C., Wilke, C. O., & Martinetz, T. (2000). Genetic algorithms in time-dependent environments. In L. Kallel, B. Naudts, & A. Rogers (Eds.), *Theoretical aspects of evolutionary computing* (pp. 263–288). Berlin: Springer.

Rowe, J. E. (1999). Finding attractors for periodic fitness functions. In W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, & R. E. Smith (Eds.), *Proc. of the Genetic and Evolutionary Computation Conf. GECCO-99* (pp. 557–563). San Francisco, CA: Morgan Kaufmann.

Smith, R. E. (1997). Population size. In T. Bäck, D. B. Fogel, & Z. Michalewicz (Eds.), *Handbook of Evolutionary Computation* (pp. E1.1:1–5). Bristol, New York: Institute of Physics Publishing and Oxford University Press.

Trojanowski, K., & Michalewicz, Z. (1999). Searching for optima in non-stationary environments. In *1999 Congress on Evolutionary Computation* (pp. 1843–1850). Piscataway, NJ: IEEE Service Center.

Weicker, K., & Weicker, N. (1999). On evolution strategy optimization in dynamic environments. In *1999 Congress on Evolutionary Computation* (pp. 2039–2046). Piscataway, NJ: IEEE Service Center.