International Journal of Pure and Applied Mathematics
Volume 46 No. 2 2008, 211-218

Invited Lecture Delivered at
Forth International Conference of Applied Mathematics
and Computing (Plovdiv, Bulgaria, August 12–18, 2007)

ADVANCEMENTS IN EVOLUTIONARY ALGORITHMS FOR DYNAMIC PROBLEMS

Ronald W. Morrison
Noblis, Inc.
3150 Fairveiw Park Drive South
Falls Church, Virginia, 22042, USA
e-mail: ronald.morrison@noblis.org

Abstract: Evolutionary algorithms (EAs) are iterative, heuristic search methods that have found broad application in a variety of industries. They are often applied to very complex multi-modal, multi-dimensional optimization problems where the exact functional form is unknown. EAs operate by accumulating information at each iteration, so there is an assumption of consistency in the evaluation function in traditional EA application. Many real-world optimization problems in finance and engineering involve dynamic environments where this consistency assumption does not hold. This paper discusses recent advancements in the field of evolutionary computation which have enhanced the ability of EAs to perform in dynamic environments.

AMS Subject Classification: 65K10, 90C59, 68T20
Key Words: optimization, search, evolutionary algorithm

1. Introduction and Overview of EAs

Many practical optimization problems in finance and engineering that are otherwise amenable to solution by EAs are dynamic on a time scale within the range of an EA execution. EAs operate by creating a population of potential solutions to a particular problem and evaluating those solutions using a fitness function (also called a fitness landscape). Biologically-inspired iterative techniques are then applied to discover successively improved solutions through refinement of

Received: August 17, 2007

the best discovered solutions and additional searching for promising regions in the unexplored solution space. The construction of an EA involves the following steps:

(1) Select an encoding scheme (genetic representation) and evolutionary architecture.

(2) Select an initial population size and randomly create an initial population of potential problem solutions.

(3) Evaluate the solutions in the population using a fitness function.

(4) Select the parents of the next generation.

(5) Mix the genes of the parents to form offspring.

(6) Apply a mutation operator to modify the offspring created by the previous step or to create new offspring from the parents.

(7) Select which of the offspring survive. This step is often used for terminating any offspring that have violated any problem constraints.

(8) Loop through steps (3) through (7) until termination, usually based on some measure of failure to improve over a number of iterations.

Various EAs differ principally in their approach to some of the above steps. For additional detail, see [2].

2. EAs in Dynamic Environments

Actual applications of EAs are often very computationally expensive in the fitness evaluation step. Because of this, we do not want to "waste" previous fitness evaluations or restart simultaneous EA applications when the problem changes. Instead, we want to apply techniques that preserve progress towards solutions or speed the identification of new solutions after changes. Figure 1 provides an example of normal EA behavior that can be expected if the underlying problem changes during the EA execution [5].

As Figure 1 shows, the EA initially locates successively better solutions. This continues until the fitness landscape moves, at point A, and the fitness suddenly drops, as shown at point B. The EA will then start to identify improved solutions as shown in labeled area C. This pattern will continue with each landscape change, but with each change the EA tends to "lose ground" as is seen in the performance gap labeled D. Eventually, the EA may be unable to find any areas of improved fitness to exploit.

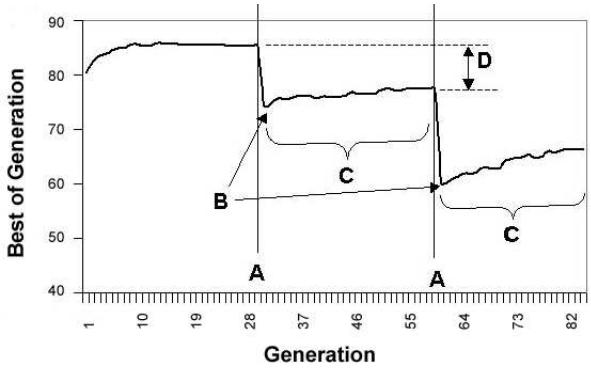

Figure 1: Canonical EA performance in a dynamic environment

To improve the performance of the EA in the dynamic environments, we are looking for techniques that reduce the depth of the fitness reduction at points B, and techniques that increase the slope in regions labeled C.

3. Current Issues and Current Research

3.1. Diversity

Mathematical analysis, experimental evidence, and intuition indicate that a loss of diversity in the population contributes to poor EA performance in dynamic environments. As an EA population converges on a solution, the population loses diversity and is less able to detect and respond to changes in other regions of the solution space. In this context, the term "diversity" refers to the distribution of the population throughout the solution space.

3.1.1. Diversity Measurement

As diversity appears to be important for EA performance in dynamic environments, we must first address the method of measurement. Most traditional EA implementations use bit string representations and population Hamming distance as a diversity measure. The obvious problem in the practical use of pair-wise population diversity measures is that the computation of the measure may be quadratic with the size of the population P.

Recently two methods of computing population Hamming distance have

been published in EA literature that are linear in P [5], [6]. The first does a little unusual mixing of discrete and continuous mathematics:

For $y_{ij} \in \{0, 1\}$:

$$\sum_{i=1}^{i=L} \sum_{j=1}^{j=P-1} \sum_{j'=j+1}^{j'=P} |y_{ij} - y_{ij'}| = P \left[\sum_{i=1}^{i=L} \sum_{j=1}^{j=P} (y_{ij} - c_i)^2 \right] , \tag{1}$$

where: $c_i = \frac{\sum_{j=1}^{j=P} x_{ij}}{P}$.

The second, and slightly more efficient method can be found in [5].

In addition to Hamming distance, population entropy is commonly used in EA studies as a diversity measure. Like Hamming distance measures, entropy is maximized if all the strings are different, but does not measure the distribution of the population throughout the solution space.

One recently developed measurement method for efficiently measuring population dispersion throughout a search space is called the dispersion index (Δ) [5]. This index evaluates the diversity of a population in real-numbered phenotypic space instead of genotypic space. There are two components of Δ. The first compares the moment of inertia along each axis to that for a uniform distribution. The second component is based on discrepancy theory concepts [3], but reduces the computational complexity by checking only a few spatial partitions. The reduction in spatial partition measurements is justified by the normal patterns of EA convergence (see [5] for details).

3.1.2. Population Diversity Maintenance

Maintaining diversity to improve EA performance redirects some of the computational effort normally associated with the refinement of already-discovered promising solutions to the exploration of the search space. The following sections discuss diversity maintenance methods and address the advantages and limitations of the techniques. Neither the list of techniques nor the list of researchers cited is exhaustive, but both are representative of current advancements.

Some of the first approaches used to improve EA performance in dynamic environments involved increasing diversity through manipulation of the mutation operator. Mutation variation methods are still being developed [8] and are used in practice. A traditional drawback with these methods is that selecting the time to change the mutation operator often requires knowledge of when the environment has changed, and this can be difficult in real applications. Instead

of using mutation rates to randomize the genetic material, a similar technique adds random members to the population. A recent enhancement that combines addition of random population with a similarity measure for choosing members to replace was devised by Tinós and Yang [10].

Memory techniques have received attention in recent years. They retain information about previously discovered good solutions and have been shown to be quite effective in cases, where the changing fitness landscape is recurrent. Some of these methods for retention of previously-found, high-quality solutions use biologically inspired methods such as diploid representations and polygenic inheritance. Recent variations in the application of these techniques for specific problem types can be found in [7], [11], [13]. Other memory techniques remember entire solutions with good fitness for re-introduction into the population at a later time [14].

A similar category of techniques uses subpopulation or speciation schemes to retain good-performing solutions for later use. These techniques may or may not continue to evolve the sub-population after separation from the main population. Examples of these techniques include Bränke's "scouts" [1], Klinkmeijer, de Jong, and Weiring's "serial population" [4], and "variable-sized memory" by Simões and Costa [9].

The above memory-based and subpopulation techniques have been shown to have good performance when the environment is recurrent, but may also provide sufficient diversity so that they are also effective in some environments that are not recurrent, as in [1]. The disadvantages of both memory and subpopulation techniques are that it can be difficult to determine what to keep in memory and difficult to determine how big to make the memory. Many of these techniques also need to explicitly detect changes in the environment.

The last technique that we will consider for diversity maintenance distributes a small portion of the population, called "sentinels" throughout the solution space, and then leaves these individuals unaffected by the algorithm operators (i.e., no crossover or mutation) [5]. The sentinels maintain the diversity necessary to respond to changes. The method, however, requires an increased number of fitness evaluations and also generally requires a problem-specific method of distributing the sentinels throughout the solution space.

Finally, hybrid approaches involving combinations of techniques to enhance performance are becoming more common. Recent examples include the combination of an EA with a hill-climbing algorithm [15] and the combination of an EA with incremental learning [12]. The choice of techniques to combine is often based on knowledge of the dynamics of the specific problem being solved.

3.1.3. Change Detection

Several of the diversity maintenance techniques discussed above require some action when a change is detected in the problem. If a change is inside the boundaries of a converging EA population it is likely that the EA will respond to it. The situation is much more problematic if the change is outside of the population's current search space coverage. When identifying a landscape change, it is important that neither false positives nor false negatives occur for computation efficiency. Similarly, it is important not to add a much computational complexity in order to identify a change.

Commonly applied techniques to attempt to identify a landscape change use declining population fitness as an indicator. However, the stochastic nature of EAs make these methods subject to both false positives and false negatives. More reliable methods re-evaluate the same solution (such as individuals in memory or sentinels) across evolutionary iterations. These later methods are only susceptible to false negatives in the identification of change, but they require additional fitness evaluations.

3.2. Experimental Studies

As is the case with most complex adaptive systems, our theoretical understanding of EA performance in dynamic environments remains rudimentary, so most published work is based on experimental results. There are difficulties in advancing a field based entirely on experimental studies, mostly related to questions about whether specific experimental observations are generalizable.

In the reporting of experimental results, most studies use a controlled test problem. These studies, therefore, need to address the choice of test problem: Is it representative of some problem of interest? Are the dynamics of the test problem understood? Can observed results be generalized?

EA researchers generally use a test problem generator for their experiments. These start with a static test problem, and then allow the researcher to create very complex dynamic environments through modifications to one or more of the problem's characteristics. One common type is a "moving peaks" fitness landscape with a choice of dynamics and dimensionality. Test problem generators of this type were independently developed by both Branke and Morrison, although Branke's version in [1] has recently become more popular. Additional test problems have used a dynamic knapsack problem, coordinate transformations of a static landscape, or dynamic bit-pattern matching problems.

Each of these test problems provides an environment for the examination of EA variants on different classes of problems, but in all cases the extension of the experimental results to actual problems of interest remains a subject of debate.

4. Summary and Conclusion

There is a continued interest in the practical use of EAs for dynamic environments. A variety of promising techniques have been developed for use in these environments. Future research is necessary to improve our understanding of the theoretical foundations of EA performance in dynamic environments in order to improve our ability to identify the circumstances when specific experimental observations are generalizable.

References

[1] J. Branke, *Evolutionary Optimization in Dynamic Environments*, Kluwer Academic Publishers (2002).

[2] Kenneth A. De Jong, *Evolutionary Computation: A Unified Approach*, MIT Press (2006).

[3] Michael Drmota, Robert F. Tichy, *Sequences, Discrepancies and Applications*, Lecture Notes in Mathematics, **1651**, Springer-Verlag (1997).

[4] Lars Zwanepol Klinkmeijer, Edwin de Jong, Marco Wiering, A serial population genetic algorithm for dynamic optimization problems, In: *Proceedings of the 15-th Belgian-Dutch Conference on Machine Learning* (2006), 41-48.

[5] Ronald W. Morrison, *Designing Evolutionary Algorithms for Dynamic Environments*, Natural Computing Series, Springer-Verlag (2004).

[6] Ronald W. Morrison, Kenneth A. De Jong, Measurement of population diversity, In: *Artificial Evolution, Lecture Notes in Computer Science*, **2310**, Springer-Verlag (2002), 31-41.

[7] Connor Ryan, J.J. Collins, David Wallin, Non-stationary function optimization using polygenic inheritance, In: *Genetic and Evolutionary Computation - GECCO (2003), Lecture Notes in Computer Science*, **2724**, Springer-Verlag (2003), 1320-1331.

[8] Lutz Schönemann, Optimal number of evolution strategies mutation step sizes in dynamic environments, In: *GECCO (2005): Proceedings of the (2005) conference on Genetic and evolutionary computation*, **1**, ACM Press (2005), 923-924.

[9] A. Simöes, E. Costa, Improving memory's usage in evolutionary algorithms for changing environments, In: *Proceedings of Congress on Evolutionary Computation, CEC07*, ACM Press (2007), 1530-1537.

[10] Renato Tinós, Shengxiang Yang, A self-organizing random immigrants genetic algorithm for dynamic optimization problems, *Genetic Programming and Evolvable Machines* (2007).

[11] A. Sima Uyar, An adaptive diploid evolutionary algorithm for floating-point representations in dynamic environments, In: *Late Breaking Papers at the (2004) Genetic and Evolutionary Computation Conference*, Seattle, Washington, USA (2004).

[12] Shengxiang Yang, Population-based incremental learning with memory scheme for changing environments, In: *GECCO (2005): Proceedings of the (2005) Conference on Genetic and Evolutionary Computation*, **1**, ACM Press (2005), 711-718.

[13] Shengxiang Yang, Dominance learning in diploid genetic algorithms for dynamic optimization problems, In: *GECCO (2006): Proceedings of the 8-th Annual Conference on Genetic and Evolutionary Computation*, **2**, ACM Press (2006), 1435-1436.

[14] Shengxiang Yang, Explicit memory schemes for evolutionary algorithms in dynamic environments, In: *Evolutionary Computation in Dynamic and Uncertain Environments*, Studies in Computational Intelligence, Springer-Verlag (2007).

[15] Sabyou Zeng, Hui Shi, Guang Chen, Hugo deGaris, Lishan Kang, Lixin Ding, Orthogonal dynamic hill-climbing algorithm for dynamic optimization problems, In: *Proceedings of Congress on Evolutionary Computation, IEEE* (2006), 1331-1338.