

Performance Analysis of Evolutionary Search with a Dynamic Restart Policy

Michael Solano

Oklahoma State University

700 N Greenwood, Tulsa, OK 74106

mike.solano@okstate.edu

Istvan Jonyer

Oklahoma State University

700 N Greenwood, Tulsa, OK 74106

jonyer@cs.okstate.edu

Abstract

In this work we explore how the complexity of a problem domain affects the performance of evolutionary search using a performance-based restart policy. Previous research indicates that using a restart policy to avoid premature convergence can improve the performance of an evolutionary algorithm. One method for determining when to restart the search is to track the fitness of the population and to restart when no measurable improvement has been observed over a number of generations. We investigate the correlation between a dynamic restart policy and problem complexity in the context of genetic programming. Our results indicate the emergence of a universal restart scheme as problems become increasingly complex.

Introduction

The traditional approach to genetic programming attempts to find optimal individuals in a single run. This approach can suffer from phenomena known as *premature convergence* and *program bloat*. Premature convergence occurs when the search reaches a local optimum and is unable to move to other parts of the search space (Goldberg, 1989). Bloating occurs when the size of an individual increases with no noticeable change in the fitness of that individual. With bloating, the functional component of a program typically remains small while nonfunctional components (those that never execute) grow without limit. Minimizing these pitfalls continues to be an area of active research.

One method that has shown promise in combating these phenomena is the implementation of a restart policy. A restart involves re-initializing the population based on some event. A static policy restarts the population at predetermined intervals, while a dynamic approach analyzes the performance of the run to determine whether or not a re-initialization should occur (Fukunaga, 1997; Jansen, 2002). Determining what criteria should be used to decide whether or not a restart should occur can have a dramatic effect on the performance of the algorithm. This work explores the relationship between problem complexity and the optimal performance-based restart criteria.

Methodology

Defining and quantifying the complexity of a genetic programming problem domain has, until recently, been difficult.

A domain independent methodology for determining the complexity of a problem domain was proposed by Tomassini et al (2005) which defines the complexity of any problem in terms of a single value called the *fitness distance correlation*. Structural distance (Ekart and Nemeth, 2002) provides a convenient method to calculate the distance between any two program trees. Using the structural distance, it becomes possible to calculate the fitness distance correlation (*fdc*) for a particular problem domain. The importance of the *fdc* metric is that it provides a quantifiable relationship between the fitness of an individual and its distance from the ideal individual. This value can be used to gauge problem complexity (Tomassini et al, 2005).

Previous research with genetic algorithms (Jones and Forrest, 1995) suggests that a problem may be classified into three categories based on the *fdc* coefficient. The trap function (Deb and Goldberg, 1993) makes it possible to define, using parameters, the difficulty of finding the perfect individual in this domain. We use this as our artificial domain to compare the performance of the different techniques. We also perform experiments using well-known toy problems.

For each domain, the results of 150 runs are averaged in both no-restart and performance-based restart approaches. When discussing the restart policy, we will use the term *momentum* to refer to the number of generations we allow a run to continue without any fitness improvement. Momentums from 1 to 20 with increments of 1 are used. In each domain, the population size is set to 1024 individuals and runs are allowed to continue for 100 generations. The probability of success is calculated as the number of runs for which an ideal individual was found divided by the total number of runs. We define restarts to occur within runs; therefore a restart does not start a new run.

Experimental Results

In this section we discuss the results obtained using the above described experimental procedure. First, we present the results using the artificial trap function domain in which we examine the correlation of problem complexity with our random restart scheme. We then apply our scheme to two well-known domains.

We investigate several variations of the trap function by manipulating its B and R parameters to alter problem complexity. Specifically, twenty-five instances of the trap function are produced for B and R values ranging from 0.1 to 0.9 with increments of 0.2. A perfect-D tree is chosen as the ideal individual and is used when calculating the structural distance. Such a tree has the highest arity function, in this case D, at the root with each function in the tree having only functions of one less arity as children (Punch et al, 1996).

Figure 1 includes all the plots for trap functions with $B=0.1$. (All figures show the momentum on the horizontal axis.) All these problems have an f_{dc} value (> 0.998) that places them among the hardest of problems. For functions using higher R values (0.5 through 0.9) the probability of success remains low. For functions using lower R values (0.1 and 0.3), however, the restart policy already makes great headway, compared to the no-restart approach. At the highest complexity, little success is achieved regardless of the momentum used. As problems begin to slowly decrease in complexity, an optimal momentum emerges.

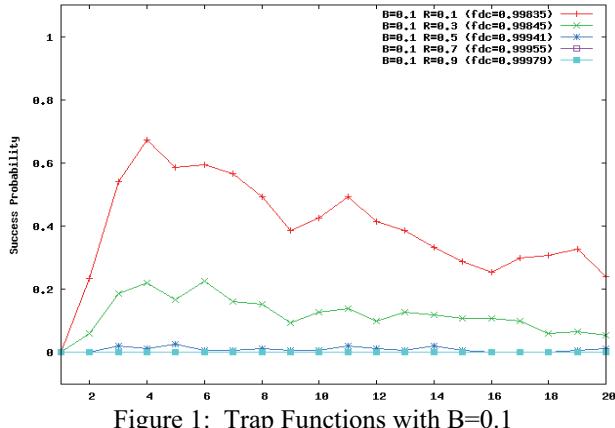


Figure 1: Trap Functions with $B=0.1$

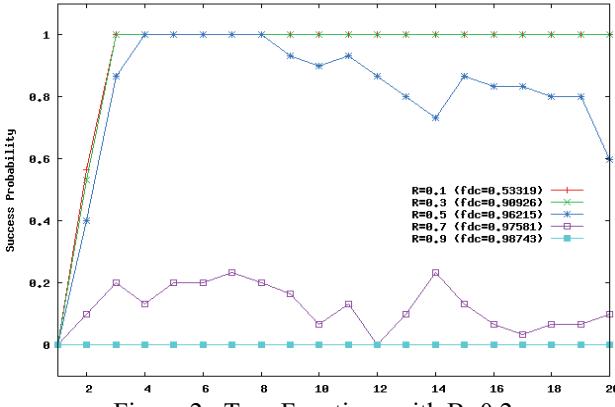


Figure 2: Trap Functions with $B=0.2$

Figure 2 shows the plots of trap functions for which $B=0.2$. The functions using the lowest R values (0.1 & 0.3) achieve 100% success for all momentums greater than or equal to 4, with no decrease in performance as the momentum increases. With $R = 0.9$, there is a 0 probability of success regardless of the momentum. When $R=0.5$, however, a curve emerges for which the best performance occurs at a momentum of 4. As in the previous results, with a high R value (and thus a higher complexity), the momentum has a minimal effect on the performance. This is the case at low complexities as well where 100% success is achieved regardless of the momentum. In between the two extremes an optimal momentum emerges.

As the evidence shows the momentum has little effect on problems with low and very high complexities. There is, however, a complexity range in which the momentum greatly influences the performance. The reason for this is

that more complex problems have larger, more feature-rich search spaces, and the search may get stuck in a remote part of the search space where.

We performed experiments using popular toy problems of varying difficulty. The plots in figures 3 show the success probability for these problems, which, in increasing order of complexity, are even-4-parity, Boolean multiplexer (11-bit), artificial ant, symbolic regression of a sextic function and two-box. This plot clearly confirms our findings on the artificial domain: the more complex the problem, the shorter the momentum with the highest success probability.

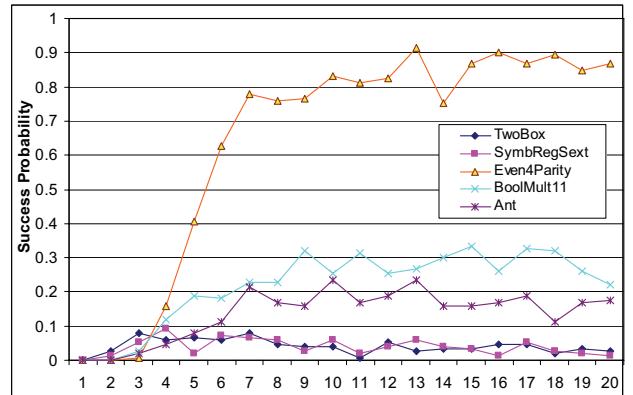


Figure 3: Toy Problems with Dynamic Restart

References

- Deb, K. and Goldberg, D. E. 1993. Analyzing deception in trap functions. In Foundations of Genetic Algorithms, 2. pp. 93-108. Morgan Kaufmann.
- Ekart, A. and Nemeth, S. Z. 2002. Maintaining the diversity of genetic programs. In Genetic Programming, Proceedings of the 5th European Conference. Vol. 2278 of LNCS. pp. 162-171. Springer-Verlag.
- Fukunaga, A. 1997. Restart scheduling for genetic algorithms. In Genetic Algorithms: Proceedings of the Seventh International Conference.
- Goldberg, David E. 1989. Genetic Algorithms in Search, Optimization, & Machine Learning. Boston, Massachusetts: Addison-Wesley.
- Jansen, Thomas. 2002. On the analysis of dynamic restart strategies for evolutionary algorithms. In Proceedings of the 7th International Conference on Parallel Problem Solving From Nature (PPSN VII). Berlin, Germany: Springer.
- Jones, T. and Forrest, S. 1995. Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In Proceedings of the Sixth International Conference on Genetic Algorithms. pp. 184-192. Morgan Kaufmann.
- Punch, B., Zongker, D., and Goodman, E. 1996. The royal tree problem, a benchmark for single and multiple population genetic programming. In Advances in Genetic Programming 2. pp. 299-316. Cambridge, Mass.: The MIT Press.
- Tomassini, M., Vanneschi, L., Collard, P., Clergue, M. 2005. A Study of Fitness Distance Correlation as a Difficulty Measure in Genetic Programming. Evolutionary Computation. Vol. 13. Issue 2. pp. 13-239. Cambridge, Mass.: The MIT Press.