

Introduction

1.1 Overview and Background

1.1.1 Overview

Evolutionary algorithms (EAs) are heuristic, stochastic search algorithms often used for optimization of complex, multi-dimensional, multi-modal functions, where the actual functional form is not known. EAs are adaptive algorithms in the sense that they accumulate and use information to progressively improve their ability to solve some problem of interest. EAs operate by creating a population of potential solutions to a particular problem and evaluating those solutions directly. The mechanism by which an EA accumulates information regarding the problem of interest is an exact evaluation of the quality of each of the potential solutions, using a problem-specific evaluation method, referred to as the fitness function. Various iterative operations, called genetic operators, then improve the quality of the solution by: (1) successive refinement of the best solutions found, and (2) searching the unexplored solution space to identify promising areas containing solutions better than those found so far. Identification of the appropriate balance of exploitation of the best solutions and further exploration of the solution space have been the focus of much research in the EA community.

The robust capability of EAs to find solutions to difficult problems has permitted them to become the optimization and search techniques of choice by many industries. From the design of jet engines [30] to the scheduling of airline crews [39], EAs, in their various forms, are routinely solving a multitude of complex, multi-dimensional, multi-modal optimization problems.

But what happens if the information that has been provided to the EA changes? Despite the obviously successful application of evolutionary techniques to complex problems in many different environments, the resultant solutions are often fragile, and prone to failure when subjected to even minor changes in the problem. In the context of evolutionary computation, a problem with an objective function that changes over time is referred to as having

a “dynamic fitness landscape.” Since information is accumulated by the EA during successive iterations, there is an implicit assumption of consistency in the evaluation function. Unfortunately, the consistency assumption is not the case in many real problems. A variety of engineering, economic, and information technology problems require systems that adapt to changes over time. Examples of problems where environmental changes could cause the fitness landscape to be dynamic include: target recognition, where the sensor performance varies based on environmental conditions; scheduling problems, where available resources vary over time; financial trading models, where market conditions can change abruptly; investment portfolio evaluation, where the assessment of investment risk varies over time; and data mining, where the contents of the database are continuously updated. These types of problems may experience simple dynamics, where the fitness peaks that represent the optimal problem solution drift slowly from one value to the next, or complicated dynamics, where the fitness peaks change more dramatically, with current peaks being destroyed and new, remote peaks arising from valleys.

The motivation for designing EAs for solving dynamic problems is simple. With the continuing increases in available processing power, it is becoming computationally possible to assign an EA to continuously solve actual dynamic problems without the need for human intervention. This, however, requires that the EA can continuously provide a “satisfactory” level of performance when subjected to a dynamic fitness landscape and, at our current level of understanding, we are not able to ensure such performance. In this book we will discuss research that exploits insights provided by biology and control systems engineering to address issues involved in the design of EAs for dynamic fitness landscapes. The results of this research provide a basis for the improved ability to create EAs that successfully operate in dynamic environments without human intervention.

1.1.2 EAs Described

There are four major classes of EAs: genetic algorithms (GAs), evolutionary strategies (ESs), evolutionary programming (EP), and genetic programming (GP) [5], [33]. Solving a problem with EAs involves ten steps. The differences between the classes of EAs result from a difference in emphasis on, or approach to, some of the ten steps [41], [7], [34].

The ten steps are:

1. Decide on an encoding scheme for the possible solutions to the problem that will permit alternative solutions to be “evolved.” This selection of problem representation involves mapping the problem solution space (called phenotypic space) into an equivalent representation (called the genotypic space) that is amenable to application of the operators of an EA. For genetic algorithms, the encoding scheme most often used is bit-strings, whereas evolutionary strategies and evolutionary programming

most often use strings of real numbers for their representation. In genetic programming computer programs are represented as variable-sized trees consisting of arithmetic functions, conditional operators, variables, and constants.

2. Select an initial population size and randomly create an initial population of potential solutions to the problem. The initial population size selection involves tradeoffs between the need to adequately sample the solution space and the need to bound the computational requirements of the EA. In most EAs, the population size, once selected, remains constant during operation. Less traditional implementations involve population sizes that vary.
3. Select an evolutionary architecture. Two main architectures are used in EAs: generational and steady-state. In the generational architecture, there are distinct generations, where all population members are replaced by the succeeding generation. In a steady-state EA, population members are added at the same rate that population members are removed. The most common EA architecture is generational.
4. Evaluate the quality of the solutions in the population. This step involves converting the genotype representation for each member of the population into the phenotype representation for evaluation using an unequivocal evaluation of how “good” the solution is. This is measured by use of a fitness function. The design of an appropriate fitness function for each problem being solved by an EA can be very difficult. When it is not possible to determine an unequivocal gradation of “better” and “worse” for alternative solutions, an EA is not a good choice of method for problem solution.
5. Select the parents of the next generation. This is where the “best” members of the population are selected for the generation of offspring. In EAs, this is the step where the genetic material that will survive to the next generation is determined. In some EA implementations the entire population has offspring. Genetic algorithms usually use a probabilistic rule, based on fitness evaluations, to select parents for the offspring.
6. Mix the genes of the parents to form offspring. This step, called crossover or recombination, is a major focus in the construction of new potential problem solutions in genetic algorithms and genetic programming; it is used to a lesser extent or not at all in evolutionary strategies; and is not used at all in evolutionary programming. Genetic algorithms use a variety of crossover techniques, including 1 to N -point crossover (where N is the length of the genome). In GAs, individual genes are generally left unaltered in their recombination to a new genome. When crossover is used with real-numbered encoding schemes in evolutionary strategies, genes are sometimes recombined to an intermediate value between the values at the crossover points. Crossover techniques usually involve two parents, but schemes have been devised that involve three or more parents.

7. Apply a mutation operator to either modify the offspring just created or create new offspring from the parents. In evolutionary programming and in evolutionary strategies without recombination, this is the method for creating new offspring from the parents. Mutation operators randomly change individual genes of the members of the population. In GAs, mutation is usually fixed at some small uniform probability, whereas for both evolutionary strategies and evolutionary programming, mutation is governed by statistical operators that vary during the operation, based on genetic diversity, fitness, or both.
8. Select which of the offspring survive. This is the primary method of selection for the survival of genetic material in the case of evolutionary strategies and evolutionary programming, where the entire population generates offspring. Evolutionary strategies most commonly use one of two selection methods: a new generation is created from the μ best ($1 \leq \mu \leq \lambda$) individuals from the set of λ offspring, referred to as (μ, λ) selection; or a new generation is created from the μ best individuals from the union set of λ offspring and their immediate antecedents, referred to as $(\mu + \lambda)$ selection. Evolutionary programming normally involves a tournament selection technique for survival from the union of parents and offspring.
9. Loop through steps 4 through 8 until termination. It is necessary to have some criterion for stopping. Most often, this criterion is a measure of failure to improve over some number of iterations.
10. Repeat steps 1 through 9 a statistically significant number of times. Since the actual results of the EA vary based on the randomly selected initial population and the stochastic genetic operators, EAs are usually run repeatedly with different initial populations until their experimental performance on a problem of interest is statistically established. In actual practice, there are times where one operation will derive a satisfactory solution to a problem at hand, and there is no reason to execute the first nine steps more than once.

The application of these ten steps, in their various forms, has been a research focus for over 25 years [20], [29], [18] and, with the increased performance of commonly available computers, has been the source of solutions to practical engineering and mathematical problems for over ten years. To understand the difficulties anticipated in the analysis of the performance of EAs in dynamic fitness landscapes, we must first briefly review what is known, or at least fairly widely accepted, about the mechanisms by which EAs operate in static fitness landscapes.

When a population is initialized, it randomly samples the solution space for promising areas to pursue. A histogram of the fitness values for this population would illustrate the fitness distribution for the population. If an appropriate encoding scheme has been selected, the problem is well formed, and the population is large enough to adequately sample the solution space, this fitness distribution should have a range of fitness values that is large enough to per-

mit the EA to distinguish between promising areas of the solution space and areas that are less promising. Occasionally, when designing an EA solution for a real problem, it is found that the initial population has fitness values that are all zeros (or all some other value). This situation provides no information to the EA about where to focus the search in the solution space, so an EA in this situation becomes just an inefficient method for random search. Other difficult problems exist where the regions of the search space with increasing fitness lead the EA away from the desired solution. These problems are referred to as deceptive [24].

The parent-selection step determines which genetic material will survive to future generations. Different selection strategies place different emphasis on retention of the genetic material in the best solutions found so far. When using an elitist strategy, for example, the best solutions found are retained in the population at the expense of losing the genetic material present in the lower-fitness population members [40]. Other selection strategies may not exclusively retain the highest fitness members, but merely bias their selection towards the members of higher fitness. It should be noted that important genetic material can be lost during selection, because it was only present in members with relatively low overall fitness.

The two operators which are responsible for the creation of offspring are crossover and mutation. Because its effects are easier to describe, mutation will be discussed first, although, in practice, it is often applied second.

Mutation makes small random changes to the genetic material that has made it through the selection stage. Since mutation rates are generally small and mutation is applied to the genetic material of population members that have already been determined to be relatively highly fit, minor random changes can be visualized as small local searches of a promising area of the solution space. This search, however, is “local” in genotypic space, whereas fitness is measured in phenotypic space. It is easy to envision encoding schemes where minor genetic material changes cause jumps to vastly different areas of the phenotypic solution space. It is also important to notice that many of the studies of the performance of EAs ignore the additional dynamics caused by the genotypic to phenotypic mappings, and focus on problems where this mapping is trivial (such as the ONEMAX problem described in the next paragraph). The point to remember here is that mutation is a local search operator as viewed in genotypic space.

Crossover is the mixing of the genes of the selected parents and is a much more complicated operator. An excellent analysis of the effects of crossover can be found in [62] but we will adopt a simplified explanation here to facilitate conceptual understanding. There are many techniques for mixing the genetic material of the selected parents to form offspring, but the reason that we mix the genetic material is to examine the ability of the genetic material from one highly fit individual to improve the overall fitness of another highly fit individual. This can be illustrated with a trivial, binary-encoded GA at-

tempting to find a string of all ones (known as the ONEMAX problem), where the fitness evaluation is a simple count of the number of ones in the string:

Parent 1: 11110000 Fitness = 4 Parent 2: 00001111 Fitness = 4.

With simple one-point crossover, it is possible for the GA to supplement the genetic material of Parent 1 with that of Parent 2 and create an offspring:

11111111 Fitness = 8.

Of course, in this case it is also possible to create an offspring of fitness zero. It is also possible in other cases for the crossover operator to disrupt good sequences of alleles that existed in one of the parents and are necessary for the solution. In the case of multi-modal fitness landscapes, when mixing the genes of relatively highly fit individuals from the same peak, crossover is essentially a local search operator that is strongly examining the solution space near that peak. If the highly fit parents are from different peaks in a multi-modal landscape, crossover becomes a global search operator. Unfortunately, it is usually not possible to easily identify whether population members are on the same peak in a multi-modal problem due to lack of detailed knowledge of the functional form of the fitness landscape.

Although our theoretical understanding of EA behavior in static fitness landscapes is far from complete (see, for example, [63], [6]), the situation gets much worse when dynamic fitness landscapes are introduced. Dynamic fitness landscapes present the additional problems of detecting and responding to changes in information that the EA has used.

1.2 Previous Research

Early research into EA performance in dynamic fitness landscapes was conducted over ten years ago [25], [14], [27]. Recent years have seen a significant increase in interest in this subject [42], [35], [4], [69], [64], [67], [38], [72], [55], [12]. Despite this increased interest in the performance of EAs in changing environments, the “successes and failures” of EAs in dynamic fitness landscapes that have been reported to date have mostly just measured the speed of the adaptation of some specific EA implementation. This measurement is usually made using a simple example problem and often reported without analysis of the dynamics of the sample problem selected or the generality of any results. As has been shown in the study of EAs in static environments, comparative studies of the effectiveness of various EAs in dynamic environments will require rigorous and standardized test functions. Furthermore, since it is likely that techniques that are successful in improving the performance of EAs in adapting to some types of fitness landscape changes are likely to be less effective in other types of landscape changes, standardized test functions will be required to cover a variety of different types of dynamic behavior.

Until recently, most of the studies of dynamic fitness landscape performance have involved randomly or gradually changing the location of the fitness peak during the progress of a GA and examining the resulting performance of the algorithm. This type of study only answers the question, “How fast can a particular EA solve a particular problem, starting with the population that remains after solving (or partially solving) a different problem with a similar structure?” The answer, not surprisingly, is often “not very well,” since finding the new peak location in a fitness landscape involves solution-space exploration, and a partially converged population has already lost some of its genetic diversity and may no longer adequately cover the solution space. When the portion of the search space that is changing is not covered by the population, an EA will not detect that a change in the fitness landscape has occurred, and therefore not alter its behavior to accommodate the change.

Interesting results upon which to initiate this research have been reported in several studies of EAs in dynamic fitness landscapes and in the literature regarding the on-line adaptation of various genetic operators. The previous research potentially applicable to the study of EAs in dynamic fitness landscapes is categorized below in terms of the EA processes.

1.2.1 Diversity Introduction and Maintenance

If a dynamic fitness landscape changes, modifications to EAs that encourage re-exploration, such as increased mutation, should improve performance. These would help re-explore the fitness landscape in the event of a change, so that the new solution, wherever it resides in the fitness landscape, can be discovered.

Some of the more extensive studies of EAs in dynamic fitness landscapes have focused on the introduction of mutation mechanisms for increased solution-space exploration when changes in the solution space are detected. Cobb devised an adaptive mutation operator, called triggered hypermutation, for increasing the genetic search when changes in the fitness landscape are detected [14]. The triggered hypermutation algorithm, as originally published, is based on a standard generational EA, with a fixed population size, proportional selection, N -point crossover, and a small mutation rate (0.001) that is applied uniformly to the population. Where the algorithm differs from standard EAs is that the small mutation rate (called the “base” mutation rate) is not always the mutation rate that is applied to the population. The algorithm is adaptive in that the mutation rate does not remain constant over time. When a change in the fitness landscape is detected, the mutation rate is multiplied by a hypermutation factor before it is applied. If the hypermutation factor is very large (~ 1000), the effect of hypermutation is equivalent to re-initializing the population and starting the EA over. Smaller hypermutation factors introduce less diversity into the population.

Grefenstette continued this research [27] and proposed two diversity introduction processes intended to maintain a continuous exploration of the

search space while disrupting ongoing search as little as possible: (1) a partial hypermutation step, and (2) the introduction of random immigrants into the population. Both Cobb and Grefenstette examined the performance of an adaptive mutation operator in the context of several different kinds of fitness landscape changes [15].

Another strategy for increasing search after detecting a change in the environment was suggested by Varak [65]. They developed a variable, local-search operator to enable GA-based control systems to track slowly varying solutions to a specific control system problem. As suggested by Cobb, the new search operator is triggered only when the time-averaged best performance of the population deteriorates.

Other EAs depend on adaptive mutation operators, even in static fitness landscapes. Some of these techniques were pioneered by Schwefel and, in their most general form, use a mutation mechanism that enables the algorithm to evolve its own mutation strategy parameters during the search, creating an explicit link between the amount and type of mutation and the fitness of the population [5], [28]. Bäck has also performed an initial evaluation of the performance of evolutionary strategies in dynamic landscapes and illustrated that the performance of evolutionary strategy self-adaptation methods is sensitive to the type of dynamic problem [4].

Finally, an interesting recent study examined the use of variable-length EAs in dynamic environments [73]. The results showed promise on the problems examined, but whether the improved EA performance is due to additional genetic material availability substituting for enhanced diversity or some other effect of the variable-length EA dynamics is uncertain at this time.

1.2.2 Addition of Memory

An early study that combines EAs and case-based reasoning in dynamic environments was conducted by Ramsey and Grefenstette. They applied case-based reasoning to create a history of good solutions for previously encountered cases and, when a change was detected in the environment, the EA was re-started, using these previous cases to seed the initialized population [54].

More recently, several researchers have examined whether the addition of a memory component to an EA is useful in dynamic fitness landscapes of a recurrent nature. A novel modification to the standard (μ, λ) evolutionary strategy was evaluated in [50]. This modification adds adaptive predictors that are based on the parents' memories to the normal adaptive parameters of the ES. Alternative memory addition was suggested by Mori, who combined a thermodynamical genetic algorithm (TDGA), which maintains population diversity by an entropy measure, with a memory-based feature, inspired by natural immune systems, to address changing fitness landscapes with a recurrently varying nature [42]. Diploid representation and polygenic inheritance have been examined as methods for retaining genetic information about previously found, high-quality solutions in dynamic environments by Connor Ryan

[57], [58]. Additional memory mechanisms for use in fitness landscapes where the global optimum frequently revisits the same regions were also examined in [12].

1.2.3 Importance of the Characteristics of the Landscape

While the characteristics of the landscape dynamics would be expected to affect the performance of an EA, few studies have examined this subject in any detail. Wilke examined some of the effects of landscape ruggedness in dynamic environments [69], and Karsten and Nichole Weicker have done some research into varying how much information is available to an EA in a dynamic environment and how the EA exploits that information [67].

With each type of landscape characteristic, there may be a variety of dynamic properties. For example, the magnitude of the change can be large or small, and the speed of the changes can be rapid or slow relative to EA time. Each of these dynamic properties can, in turn, be uniform, periodic, recurrent but aperiodic, random, or chaotic.

The speed of the fitness landscape changes (rapid or slow) relative to EA “time” (measured in generations) is one of the dynamic properties that has only recently received initial systematic study. This dynamic property is referred to as the “landscape change period” (or just “period”), and is defined as the number of EA generations between fitness landscape changes. We have previously performed an initial study of the effects of changing this dynamic characteristic on the performance of Cobb and Grefenstette’s triggered hypermutation technique [46].

It should also be noted that for most experiments with dynamic fitness landscapes, dynamics are overlaid onto an underlying static landscape. These static landscapes usually have their own set of attributes that can facilitate or inhibit the EA performance, and the effects of these static attributes are also often poorly understood.

1.3 Open Research Issues

As can be seen in the research described above, the techniques that have been added to EAs to improve their performance in dynamic fitness landscapes have considered only limited types of EA modifications that may be required to achieve success in dynamic landscapes. There has been little analysis of the different types of dynamic fitness landscape problems that may be solved by EAs and also little comparative analysis of the types of EA extensions that may be effective in solving the different types of problems.

There are many open research issues regarding the design of EAs for dynamic fitness landscapes. Among the most fundamental and interesting of these issues are:

1. Improved theoretical understanding of the performance of EAs in dynamic fitness landscapes. The theoretical understanding of EAs in static fitness landscapes is far from mature, but very little is understood about the behavior of EAs in dynamic fitness landscapes.
2. Diversity quantification. While a number of researchers have intuitively focused on increases in diversity to improve EA performance in dynamic fitness landscapes, there has been little research into quantitative identification of the necessary amount of diversity for different types of problems.
3. Diversity measurement for dynamic problems. Although diversity has been identified as an important aspect of EA performance in dynamic landscapes, the concept of diversity for dynamic landscapes has been carried over from studies of static EAs. However, common usage of diversity measures suggests some misunderstanding regarding what aspects of diversity are important to EA performance in dynamic fitness landscapes.
4. Performance measurement. The measurement of the relevant aspects of EA performance in dynamic landscapes is a complex issue and is largely unaddressed in the literature. Results are most often reported in simple graphs over time without any method to determine the overall EA performance, nor the statistical significance of differing results.
5. Exploration of new methods to improve the performance of EAs in practical dynamic fitness landscapes. Biological systems suggest many potential enhancements to EAs that may improve performance in dynamic fitness landscapes that have not been evaluated.
6. Comparative studies. There has not been any comprehensive comparative study of the performance of EA techniques in a controlled, representative suite of dynamic fitness landscapes. For over 20 years, the De Jong test suite [18] has been used to measure the performance of various GAs in function optimization. Additional test functions have been added over time; some of the more frequently encountered ones are known as the Rastrigin, Schwefel, and Griewangk functions. Recently, test generators suitable for use in dynamic environments have started to become available [45], [10]. These test generators are starting to be used to evaluate EA extensions in a variety of dynamic environments.

1.4 Importance and Relevance

Dynamic fitness landscapes are common in a wide variety of problems where EAs are currently being applied. For example, in financial applications, EAs are being used for portfolio balancing, risk analysis, and the identification of trading strategy parameters. Difficulties arise because the underlying behavior of the financial markets and the risk analysis basis change over time. In another example, EAs are being used for data mining in large databases [22]. As these databases change, however, new relationships in the data go unnoticed until the EA-based data-mining effort is performed again. Without effective

methods for dealing with dynamic fitness landscapes in these applications and many others, an often-applied current practice is to “periodically” re-run the EA to see if anything has changed. If circumstances have not changed, the effort to re-run the EA was wasted. If circumstances change before the anticipated need to re-run the EA, whatever information the EA was providing is likely to be wrong, and could be very wrong. Awaiting notice that the underlying problem environment has changed through the observation that the system you are using for making business decisions no longer works is often expensive, usually time consuming, and can be disastrous. The ability to allow an EA to continuously provide appropriate solutions to these (and other) changing problems without the need for discontinuous operation or human intervention would improve efficiency in these complex domains.

The research described herein will contribute to the understanding of the performance of EAs in dynamic fitness landscapes. It will examine the applicability and effectiveness of promising EA improvements that were inspired by biological and engineering systems performing in a variety of complex dynamic environments. Along the way, we will address a number of important design issues that will facilitate further research in this area. Several of these issues are related to the methods for determining population diversity and the methods for reporting EA performance. We will also provide a problem generator for dynamic fitness landscape problems that easily delivers a wide range of reproducible dynamic behaviors for EA researchers.

The combination of the resolution of some fundamental design issues with comparative experiments in a variety of dynamic fitness landscapes will provide a step towards the goal of designing EAs to continuously solve important, but changing, problems without human intervention.

1.5 Book Structure

The remainder of this book is structured as follows:

- Chapter 2 will discuss dynamic problems and the characteristics of EAs that are required for effective performance in dynamic environments.
- Chapter 3 will provide some insights from biology and control system engineering regarding methods for exploiting diversity-creating mechanisms to deal with the complexities of dynamic fitness landscapes.
- Chapter 4 will discuss measurement of population diversity and introduce new and efficient methods for computing the traditional measures of population diversity. This chapter will then go on to address significant shortcomings in the use of traditional population diversity measures and provide a new and efficient population diversity measure that corrects these shortcomings.
- Chapter 5 will present the architecture of an extended EA which has been designed to exploit our new understanding of population diversity measurement to improve the performance of the EA in dynamic environments.

Additionally, techniques developed for the creation of this new EA architecture will be addressed in the context of their potential applicability to population initialization techniques for static EAs.

- Chapter 6 will describe the experimental framework for testing this new EA. A new test problem generator will be presented to provide a standardized capability for testing EAs in dynamic landscapes and comparing results. Additionally, this chapter will address shortcomings in common performance measures for comparing results in examining EAs in dynamic fitness landscapes, and provide a performance measure that addresses the shortcomings.
- Chapter 7 will address the issue of performance measurement of EAs in dynamic fitness landscapes and derive appropriate performance reporting measures that cover an EA's exposure to a range of fitness landscape dynamics with results that can be checked for statistical significance.
- Chapter 8 will provide an analysis of the results of the performance of the extended EA, described in Chap. 5, in the dynamic fitness landscape problems described in Chap. 6. This chapter will also identify and analyze discovered relationships between important EA parameters and the resulting EA performance.
- Chapter 9 will return to the techniques developed in Chap. 5 to demonstrate their usefulness in the problem of static EA population initialization.
- Chapter 10 will provide a summary of this research and provide suggestions for future research.

Designing Evolutionary Algorithms for Dynamic
Environments

Morrison, R.W.

2004, XII, 149 p. 82 illus., Hardcover

ISBN: 978-3-540-21231-7