Eugenic Evolution for Combinatorial Optimization

John William Prior

Report AI98-268 May 1998

`jprior@cs.utexas.edu`
`http://www.cs.utexas.edu/users/jprior/`

Artificial Intelligence Laboratory
The University of Texas at Austin
Austin, TX 78712

Eugenic Evolution for Combinatorial Optimization

by

John William Prior, B.S. Applied Mathematics, Columbia University

Thesis

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Master of Arts

The University of Texas at Austin

May 1998

Eugenic Evolution for Combinatorial Optimization

Acknowledgments

I would like to thank Professor Risto Miikkulainen for being unbelievably patient and encouraging during the creation of this thesis. I want to thank all my friends who put up with me saying ”I'm going finish it soon ... really I am ...” thousands of times.

JOHN WILLIAM PRIOR

The University of Texas at Austin
May 1998

Eugenic Evolution for Combinatorial Optimization

John William Prior, M.A.
The University of Texas at Austin, 1998

Supervisor: Risto Miikkulainen

In the past several years, evolutionary algorithms such as simulated annealing and the genetic algorithm have received increasing recognition for their ability to optimize arbitrary functions. These algorithms rely on the process of Darwinian evolution, which promotes highly successful solutions that result from random variation. This variation is produced by the *random* operators of mutation and/or recombination. These operators make no attempt to determine which alleles or combinations of alleles are most likely to yield overall fitness improvement. This thesis will explore the benefits that can be gained by utilizing a direct analysis of the correlations between fitness and alleles or allele combinations to intelligently and purposefully design new highly-fit solutions.

An algorithm is developed in this thesis that explicitly analyzes allele-fitness distributions and then uses the information gained from this analysis to purposefully construct new individuals "bit by bit". Explicit measurements of "gene significance" (the effect of a particular gene upon fitness) allows the algorithm to adaptively decide when *conditional* allele-fitness distributions are necessary in order to correctly track important allele interactions. A new operator—the "restriction" operator—allows the algorithm to simply and quickly compute allele selection probabilities using these conditional fitness distributions. The resulting feedback from the evaluation of new individuals is used to update the statistics and therefore guide the creation of increasingly better individuals. Since explicit analysis and creation is used to guide this evolutionary process, it is not a form of Darwinian evolution. It is a pro-active, contrived process that attempts to intelligently create better individuals through the use of a detailed analysis of historical data. It is therefore a *eugenic* evolutionary process, and thus this algorithm is called the "Eugenic Algorithm" (EuA).

The EuA was tested on a number of benchmark problems (some of which are NP-complete) and compared to widely recognized evolutionary optimization techniques such as simulated annealing and genetic algorithms. The results of these tests are very promising, as the EuA optimized all the problems at a very high level of performance, and did so much more *consistently* than the other algorithms. In addition, the operation of EuA was very helpful in illustrating the structure of the test problems. The development of the EuA is a very significant step to statistically justified combinatorial optimization, paving the way to the creation of optimization algorithms that make more intelligent use of the information that is available to them. This new evolutionary paradigm, eugenic evolution will lead to faster and more accurate combinatorial optimization and to a greater understanding of the structure of combinatorial optimization problems.

Contents

Chapter 1

Introduction

The field of combinatorial optimization deals with problems whose inputs are discrete arrangements or permutations. These problems are typically NP-complete, and therefore no polynomial time algorithms exist which can solve them. However, some algorithms forsake the goal of finding *optimal* solutions and instead attempt to find *good* solutions in polynomial time. Since there is no guarantee that an optimal solution will be found, these algorithms are called "approximation" algorithms; it is hoped that solutions almost as good as the optimal solution will be found. Because most of these algorithms guide their quest for better solutions by making small modifications to existing solutions, they rely on the heuristic of "neighborhood search" (Garey and Johnson, 1979). Both genetic algorithms ("GA"s) (Goldberg, 1989; Holland, 1975) and simulated annealing ("SA") (Kirkpatrick and Sherrington, 1988) are examples of neighborhood search algorithms. In past years, both GAs and SA have received increasing recognition for their ability to efficiently approximate arbitrary functions. Both of these algorithms are "evolutionary" algorithms, as they are characterized by survival-of-the-fittest search pruning and the inheritance of information from one iteration ("generation") of solutions to the next.

There are many methods by which evolutionary algorithms pass information from generation to generation, but most modern algorithms rely on *random* choices to decide which information shall be inherited. For example, genetic algorithms perform a type of beam search, where a simulated process of "natural selection" prunes the event space ("population") and selects solutions from the population to be modified by the random state-space operators of "recombination" and "mutation". Standard recombination operators *randomly* combine parts of highly fit solutions to form complete new solutions. No attempt is made to determine which parts might perform best with each other. SA is a probabilistic optimization algorithm that can be viewed as a simplification of the GA; it uses a population of only a single solution, and relies solely on random mutation of this solution to create new solutions. The standard mutation operators used by both GAs and SA make small random changes to existing solutions. No attempt is made to determine what changes would be most likely to yield improvements in fitness.

What benefit might come from "smart" recombination operators that analyze interactions and dependencies among parts of solutions and solution fitness, and then *intelligently* assemble these parts into new, hopefully higher fitness, solutions? What benefit might come from "smart" mutation operators that only perform those mutations most likely to increase the fitness of selected solutions? This thesis explores these questions through the development of an algorithm that explicitly analyzes the distribution of fitness with respect to specially selected allele combinations, and *intelligently* creates solutions using the information gained from this analysis. Instead of relying on *random* recombinations or mutations of parent solutions to generate offspring, this new algorithm—the "Eugenic Algorithm" (EuA)—constructs new solutions "bit by bit", probabilistically including parts of solutions based upon statistical measures of their historical performance. The major innovation of this algorithm, the "restriction operator", will be shown to be paramount to effectively tracking and exploiting dependencies among solution parts ("epistasis"). In particular, the EuA studied in thesis will measure the benefit of genotypic parts by computing the average fitness of solutions in which these parts are found, and then selecting these parts for use in new individuals proportionately to these average fitnesses. The EuA will be compared to standard evolutionary techniques on a variety of combinatorial optimization problems, and will be demonstrated to be superior to these techniques in many situations. The EuA's success at selecting the right combinations of alleles through the use of statistical measures of performance of both single alleles and dependent groups of alleles represents a major step forward in approximating very difficult combinatorial optimization problems.

Chapter 2

Evolutionary Combinatorial Optimization

Evolution is the adaptation of a population to its environment. This adaptation causes the creation of individuals of increasingly higher "fitness"; in environments where the definition of fitness remains static, evolution drives the population to better and better individuals. This process is similar to approximation—the search for good solutions to a particular problem. The parallels between the concepts of evolution and approximation have lead to the creation of *evolutionary approximation algorithms*—algorithms that attempt to optimize a particular function through a simulated process of evolution. This section will briefly describe the particular field of optimization studied in this thesis (combinatorial optimization), and then will discuss how various paradigms of evolution (Darwinian, Lamarckian, and eugenic) relate to approximation.

2.1 Combinatorial Optimization and Evolution

Combinatorial problems (defined below) are often NP-complete. Approaches to solving NP-complete problems can be roughly divided into two categories (Garey and Johnson, 1979). The first category includes more "traditional" techniques, while the second consists of more modern (and radical) algorithms. The traditional techniques focus mainly on reducing the size of the search by eliminating as many potential solutions as possible. For example, "branch-and-bound" and "implicit-enumeration" (Garfinkel and Nemhauser, 1972) utilize a tree-structured search that generates "partial" solutions, subsequentally identifying partial solutions that could not possibly be part of a true solution and then eliminating those solutions containing these partial solutions. Other traditional techniques include "dynamic programming" and "cutting-plane" techniques (Garfinkel and Nemhauser, 1972). The second major category of algorithms used for attacking NP-complete problems pertain soley to combinatorial optimization. These algorithms do not attempt to guarantee the finding of an optimal solution, but instead just try to find good solutions as quickly as possible. They are *approximation* algorithms. This thesis focuses on this class of algorithms.

(Garey and Johnson, 1979) provides a concise discussion of combinatorial optimization. Combinatorial optimization problems consist of three components: (1) a set D_π of problem instances, (2) for each problem instance $I \in D_\pi$ a set $S_\pi(I)$ of possible solutions for I, and (3) an *objective function* f_π that assigns a positive rational number $f_\pi(I, x)$ (a *solution value*) to each problem instance I and solution $x \in S_\pi(I)$. Combinatorial optimization problems can either be *maximization* or *minimization* problems. For a maximization problem instance I, an *optimal* solution x^* is one such that $f_\pi(I, x^*) \geq f_\pi(I, x)$ for all $x \neq x^*$. For a minimization problem, the solution value of x^* must be less than or equal to the solution value of any other solution. A *combinatorial approximation algorithm* is an algorithm that finds a solution x for any given problem instance $I \in D_\pi$. An *combinatorial optimization algorithm* is an algorithm that finds x^* for *all* $I \in D_\pi$. The performance of these algorithms is judged upon how quickly they find good or optimal solutions, respectively.

Combinatorial approximation is similar to the search for "better" organisms that is performed during biological evolution. This similarity is so great that highly successful combinatorial approximation algorithms have been designed that mimic the natural forces that occur during evolution. In this thesis, these algorithms shall be referred to as *evolutionary combinatorial approximation algorithms*, or simply *evolutionary algorithms.* Just as combinatorial approximation algorithms search for better solutions, biological evolution drives systems of populations to generate individuals of increasingly higher "fitness", where fitness is implicitly *defined* as successful survival and propagation[1]. In evolutionary combinatorial approximation algorithms, the blurred mixing of the concepts of reproductive success and fitness that occurs in biological evolution is replaced by explicit measurements of fitness and explicit laws of reproduction based upon fitness measurements. This allows evolutionary algorithms to produce solutions that are "better" in any sense a practitioner defines.

The biological terminology of natural evolution transfers directly to the mathematical terminology of combinatorial approximation. Solutions are "individuals", and an individual's solution value—its "goodness" measurement—is referred to as its "fitness". In evolutionary combinatorial approximation, solution components get a new name: "genes", and each different solution component value is termed an "allele". The aggregate of a solution's components is called its "genotype", while a solution's specific form is called its

[1]Propagation is more important than simple reproduction, since production of offspring that do not reproduce themselves would not be "fit".

"phenotype". It is commonly necessary to "decode" individuals' genotypes in order to determine or create their phenotypes. Often, individuals are encoded as binary vectors, although other representations (such as real vectors) are also commonly used. When individuals are represented as vectors, each vector element is consider to be a gene and each vector element value an allele. This thesis will focus on problems that have been encoded in binary vectors, since most GA research deals with such problems.

Biological evolution itself does not necessarily drive the population to generate individuals of higher ability, unless this ability is strictly defined as propagative success. In biological evolution, better individuals are simply those that produce the most reproducing offspring under a variety of environmental conditions. No decisions are made that *directly* select and promote individuals on the basis of their ability to run faster, jump higher, or go farther with less food. In contrast, for an evolutionary algorithm to efficiently approximate or optimize a function, fitness must be clearly defined, and higher fitness individuals must be explicitly promoted. As a result, if any other ability besides propagative success is desired, an evolutionary algorithm must *directly* encourage the formation of individuals with the desired ability.

In addition to these philosophical similarities between biological evolution and combinatorial approximation, there is a very distinct similarity in the *mechanism* of improvement that is utilized by both processes. A heuristic common among many combinatorial approximation algorithms is "neighborhood search", in which the search proceeds by making small modifications to existing solutions (Garey and Johnson, 1979). This closely parallels the random genotypic mutations that occur during biological reproduction. Some algorithms even go so far as to mimic sexual reproduction in order to share information among solutions. One such algorithm, the genetic algorithm (GA), is probably the most widely recognized evolutionary combinatorial approximation algorithm. Genetic algorithms rely upon fitness-biased simulated sexual reproduction and random mutation of offspring to generate variation. The fitness-biased reproduction ensures that lower fitness solutions have a smaller chance of contributing their genetic material to the next generation (Goldberg, 1989). This process is very similar to that used of Darwinian evolution. Darwinian evolution relies upon random variation to expand its search into unexplored regions of the genotypic search space, and natural selection to discourage exploration into lower fitness regions. Darwinian evolution is not the only type of evolution possible for evolutionary algorithms. In particular, the Eugenic Algorithm (EuA) developed in this paper does not rely on random mutations to create variation; nor does it rely on simple sexual reproduction to combine information from differing previous solutions. Instead, new individuals are purposefully constructed one part at a time, using any genetic material previously encountered.

To better understand the macro-processes that evolutionary approximation algorithms implicitly utilize, and to understand the process that the EuA will employ, the traditional paradigms of Darwinian and Lamarckian evolution must be understood. The next section will describe these two paradigms, their differences, and how the EuA creates a new paradigm of optimization—eugenic evolution.

2.2 Evolution

Evolution is the adaptation of an entire population of replicating individuals to their environment. Two major forms of evolution—Darwinian and Lamarckian—dominate our understanding of adaptation in natural environments and our research in artificial environments. A third form of evolution—eugenic evolution—will be introduced in this thesis. Eugenic evolution differs from Darwinian and Lamarckian evolution by how the genotypes of new individuals are constructed. While both Darwinian and Lamarckian evolution rely on random genetic inheritances, new individuals in a eugenically evolving population are purposefully created using genetic parts that have been carefully selected based on their historical performance. When contrasting these forms of evolution, the specific mechanism of replication is not important, nor are the particular genotypes, phenotypes, or environments present in the evolutionary systems. What is important is how feedback from individuals' performance is used to guide the construction of future genotypes.

2.2.1 Darwinian Evolution

In Darwinian evolution (Darwin, 1859), the genetic operators of random mutation and crossover cause genotypic variation in offspring, resulting in variations in phenotype. The phenotypic variations that produce advantages in survival and reproduction, relative to other variations, become increasingly predominant in the population from generation to generation. There is no *direct* feedback from the environment to an individual's genotype; each individual's genotype remains constant, unaffected by the environment. New genomes arise only through the random actions of mutation and crossover. However, the *frequencies* of genotypes adapt in response to the environment's effect on their corresponding phenotypes, and, as a result, higher "fitness" genotypes come to dominate the evolving population. Thus, in Darwinian evolution, entire *populations* adapt to their environment through biased sampling of higher "fitness" genotypes, even though the genotypes themselves do not directly respond or adapt to the environment. Darwinian evolution is the form of evolution prevalent in such well-known evolutionary approximation algorithms as genetic algorithms and simulated annealing, which both rely on random modifications of existing individuals to create new individuals.

2.2.2 Lamarckian Evolution

In Lamarckian evolution (Lamarck, 1914), adaptation occurs in not only the genotypic frequencies of the population, but also in the genotypes themselves. In Lamarckian evolution, it is possible for the phenotypic state of an individual to directly modify its current genotypic state. As environmental interaction shapes and molds an individual's phenotypic state, the environmentally specific "adaptation information" encoded in the resultant state (*learned* information) can directly modify and be absorbed by the individual's genotype(Luria et al., 1981; Whitley et al., 1994). These modifications to offsprings' genotypes are a form of "smart" mutation, guided by the parent's environmental encounters, and this smart mutation can supplant or augment the random genetic operations that cause individual variation in Darwinian evolution. Just as with Darwinian evolution, variations that confer higher reproductive success upon individuals increase in frequency in Lamarckian populations, and therefore populations (in addition to genotypes) will adapt to the environment.

2.2.3 Eugenic Evolution

A third evolutionary method is proposed in this paper—eugenic evolution. In eugenic evolution, as with both Darwinian and Lamarckian evolution, genotypic frequencies adapt in response to feedback from environmental trials. However, unlike the more "standard" versions of Darwinian and Lamarckian evolution, where an unsupervised (random), trial and error process of reproduction is biased by natural selection, and where the genotypes of new individuals are limited to slightly modified parental inheritances, eugenic evolution is a *supervised* (directed) adaptive process, where the genotypes of newly created individuals are virtually unlimited.

In eugenic evolution, the correlation[2] between occurrences of genotypic "parts" and individual fitness is explicitly analyzed, and new individuals are carefully constructed part by part, by allowing these correlations to bias the probability of choosing each part for inclusion in these new individuals. Environmental feedback is not limited to only *complete* genotypes and phenotypes—environmental feedback can bypass genotypes and be transmitted directly to genotypic parts. In addition, the genotype of a new individual is not constrained to be similar to the genotypes of a handful of parents—the new genotype can be constructed from any alleles found in the entire population. By piecing together the best "parts" of the most elite individuals, the entire population becomes the parent of the new individual.

It should be noted that eugenic and Darwinian evolution do not necessarily require

[2]In this paper, "correlation" refers to the qualitative correspondence between two variables, and not necessarily simple linear correlation.

the same level of technological sophistication as Lamarckian evolution, and therefore it is easier to implement eugenic or Darwinian evolutionary combinatorial approximation algorithms. Darwinian evolution requires only random variation and selection, and eugenic evolution requires only fairly simple correlations between alleles or allele combinations and fitness. For example, calculating the average fitness of individuals containing particular alleles or allele combinations can be sufficient to identify patterns of high average fitness. In contrast, Lamarckian evolution requires much more technological sophistication. In order to ensure that offspring share the adaptations of the parent, it must be possible for transients in the phenotypic state of the parent to be encoded in the genotype of the offspring. Therefore, some mechanism for performing specialized modifications to the genotype must exist, and the mapping from phenotypic state to genotype must somehow be available. Such a mechanism is not necessary for eugenic or Darwinian evolution. Furthermore, even if it were possible for a Lamarckian algorithm to *encode* phenotypic adaptations into a genotype, some mechanism for *identifying* the necessary modifications to the genotype to cause the desired pre-adaptations must be available. In addition, the genotype-phenotype mapping may not rich enough to encode these transients. For example, it may be practically impossible to extend a genotype in order to encode an individual's acquired memories or learned concepts. Because of these additional complications, Lamarckian evolutionary algorithms are rare and, for many problems, not even possible, and therefore eugenic evolutionary algorithms are much more promising for practical implementations.

2.3 Current Methods of Evolutionary Combinatorial Approximation

This section will discuss current methods of evolutionary combinatorial approximation. It will introduce a new way of categorizing evolutionary approximation algorithms into two groups—"mutation-based" and "pattern-based" algorithms, and discuss some of the best known representatives from each group. Afterward, two major characteristics of evolutionary algorithms—the use of a population for search focus and learning, and the processing of schemata, will be discussed.

2.3.1 Mutation-based and Pattern-based Evolutionary Algorithms

Modern evolutionary approximation algorithms rely primarily on Darwinian evolution. Depending on how many individuals contribute to the creation of a new individual, they can be roughly divided into two categories: mutation-based methods and pattern-based methods[3]. Mutation-based methods are those that modify a single existing individual to produce a single new individual. Simulated annealing is probably the best known mutation-based method. In contrast to mutation-based methods, pattern-based methods create new individuals by combining parts of *multiple* existing individuals to produce a new individual. Genetic algorithms are pattern-based algorithms. Current approaches to both mutation- and pattern-based evolutionary combinatorial approximation will now be discussed.

Mutation-Based Methods

Mutation-based algorithms repeatedly modify a single individual. These modifications ("mutations") produce new solutions. In all mutation-based methods, tournaments between the original and mutated solution are performed. The loser of each competition is discarded, and a new solution is then created by modifying the winner. The winner and the new solution then compete in a new tournament, and the process is repeated. In simulated annealing (Kirkpatrick and Sherrington, 1988), the tournaments are stochastic—i.e. the winner of each tournament is not necessarily the highest fitness individual. This allows for "stochastic backtracking", where algorithm can extract itself from dead-ends. The probability of accepting a lower fitness individual over a higher fitness individual (the "acceptance probability") can be made to decrease over time. Other mutation-based algorithms, including the "two-membered evolution strategy" (1+1)-ES (Bäck et al., 1993) and the roughly equivalent MBSH algorithm (Baluja, 1995), utilize deterministic competitions that are always won by the competitor of higher fitness.

A secondary characteristic of mutation-based methods are their mutation "temperatures"—the average size of modifications made to competition winners to create new individuals. Lower temperatures correspond to fewer mutations, while higher temperatures correspond to more mutations. This temperature can vary over the course of an algorithm's search. "Cooling" or "heating" schedules control the rate at which temperature decreases or increases. The cooling schedule of simulated annealing can be set in such a way that the entire search space is eventually explored, and can therefore *guarantee* the discovery of an optimal solution. When a search is guaranteed to visit all solutions in a search space, it is called "ergodic search". However, any algorithm can be modified to simply enumerate the entire search space and therefore be ergodic; therefore ergodicity by itself is a rather uninteresting property. What is more important is *efficiency*—how quickly an algorithm can find good or even optimal solutions. A simulated annealing process can be made more efficient at finding good (but not necessarily optimal) solutions by increasing its cooling rate. When this is done, the process is called "simulated quenching" (SQ). Simulated quenching

[3]The methods could also be termed "asexual" and "sexual" methods, respectively.

does not have the same ergodic search property as simulated annealing, and therefore can not guarantee an optimal solution. However, it has been demonstrated previously (Ingber, 1993) and will be shown in this thesis that simulated quenching can compare favorably in efficiency to genetic algorithms (and even the EuA) on some practical combinatorial optimization problems, but suffer from the same premature convergence problems that GAs encounter as a consequence of too rapid a cooling schedule.

One weakness of mutation-based based methods is the paucity of information preservation from one iteration to the next. Pattern-based methods attack this weakness by employing a "memory" more sophisticated than just a single stored individual. Pattern based methods will be discussed next.

Pattern-Based Methods

Pattern-based methods can be considered to be more sophisticated than mutation-based methods, because they utilize *groups* of solutions to generate new individuals. Information is garnered from individuals in the "parent" group and used in the creation of offspring. The parent group, just like mutation-based methods' "current individual", serves as a storehouse of knowledge about the search space.

The genetic algorithm (Goldberg, 1989; Holland, 1975) is probably the best known pattern-based evolutionary algorithm. They begin with a (typically random) population of n solutions. Two parent solutions are chosen and the "recombination" operator is used to combine parts of the parents to form offspring individuals. This process is repeated n times. The n generated can either replace individuals in the current population or be added to a completely new parent population. If offspring replace individuals in the current population, the algorithm is called a "steady state" genetic algorithm (SSGA) (Syswerda, 1991). There are a great many variations of the crossover operator, but two forms of recombination—"M-point crossover" and "uniform crossover" are probably the most common (Goldberg, 1989). In M-point crossover, each parent is divided at M locations into M+1 contiguous sections, numbered 1 through M+1. Each parent is divided at the same M locations. Two offspring are created by exchanging every odd section between the two parents. One-point crossover is probably the most commonly used type of M-point crossover. Uniform crossover is very similar to M-point crossover. Uniform crossover can be thought of M-point crossover, where M+1 is the number of genes in each parent. Therefore each gene is a section, and every section is *probabilistically* interchanged between the two parents. Other recombination operators take many shapes and forms, including crossover operators that use more than two parents to generate a single offspring (Eiben et al., 1994). However, these operators typically give no consideration to the *functional interdependence of alleles* ("epistasis") when deciding how to section the parents. Alleles are functionally interdependent when their "best" settings (resulting in individuals of highest fitness) depend on the settings of each other—when found in specific combinations, they are "more than the sum of their parts". Epistasis is considered to be the most significant factor in a problem's difficulty. Chapter 3 will discuss epistasis in depth.

The genetic algorithm also employs mutation; however, the probability of mutation is usually very low. Just as with mutation-based methods, standard GA mutation operators "perturb" offspring by changing a small number of alleles. Some GAs use only the mutation operator—they perform no recombination. These GAs are roughly equivalent to running many simulated annealing algorithms in parallel, and are therefore mutation-based methods. Genetic algorithm use mutation operators for two main purposes: to maintain allele diversity and to "tune" individuals by allowing genetic search to explore those areas of the sample space that have a very small Hamming distance (usually 1) from these individuals (neighborhood search). No attempt is made to determine which genes of selected individuals would be most likely to yield overall fitness improvement through mutation. The use of standard mutation operators for local tuning assumes that the difference in fitness is small among individuals separated by small Hamming distances—that *genotypically* similar individuals are *phenotypically* similar.

As mentioned above, the offspring created by the recombination and mutation operators replace lower fitness individuals in the population or are used to create entirely new populations. By repeatedly creating new offspring more often from the parts of highly fit solutions than from low fitness solutions, a simulated process of natural selection and evolution occurs. The process is usually continued until the chance of generating an offspring that is more fit than the best individual in the current population becomes very small. At this point, the search is terminated, and the best individual in the final population is chosen as the solution to the problem at hand.

2.3.2 The Population and Search Focus

Evolutionary algorithms are "weak" search methods; they require no domain knowledge to perform their tasks. When very good heuristics are available to guide the optimization of a given objective function ("knowledge-rich domains"), weak search methods will very often not perform as well or as quickly as traditional algorithms (like branch-and-bound) employing these known heuristics, although there are counterexamples in which GAs performed better than the best known conventional algorithms (H-L. Fang, 1993). However, there exist a great number of practical problems for which few heuristics exist ("knowledge-poor domains"). For such domains, trial and error is the only possibility. But it might be hoped that, at the least, weak search methods learn something from their experiences that may help them guide future search. The populations (whether composed of single or multiple individuals) utilized by evolutionary algorithms allow these algorithms to accomplish this task.

All evolutionary search algorithms, whether mutation-based or pattern-based, maintain a search focus through the use of populations of sample points. Mutation-based methods (simulated annealing, for instance) typically use populations with only one member, while pattern-based methods typically have populations consisting of many individuals. The population can be viewed as a very basic form of learning; the algorithms are storing explicit representations of "good" parts of the target space and unrepresented regions are implicitly avoided and therefore considered "bad". The population plays a fundamental role in the search strategy, by biasing the selection probabilities of new samples. As discussed by Syswerda (Syswerda, 1993), it is not easily possible to eliminate the population completely and substitute a simply-updated set of allele statistics, as this strategy would lead to repeatedly generating the same alleles with the same probabilities. A *dynamic* population is needed, as it will adaptively change the search focus and force exploration of the search space, instead of repeated exploitation of a static region.

Although the population serves to focus the search in promising regions of the search space, these algorithms make no explicit attempts to categorize interesting or beneficial properties of particular genotypes; they notice nothing and target nothing. However, it is hypothesized that *implicitly*, important genotypic properties are exploited. This is accomplished through *schemata processing.*

2.3.3 Schemata Processing

Pattern-based methods' primary operator is the recombination operator; it chooses genotypic parts or patterns from parent individuals and combines them to form new individuals. These parts are called "schemata", as they are patterned constituents of a greater whole. Some of these schemata may be more important than others in forming higher fitness solutions, and so, intuitively, recombination operators that more often choose higher "fitness" schemata are more likely to create higher fitness solutions. This idea is called the "building block hypothesis". This section will discuss how both mutation-based and pattern-based evolutionary search use schemata, and how schemata should be exploited most effectively.

Schemata

The concepts of "schemata" and "schemata processing" have been used to quantify the structure of binary combinatorial problems, explain how evolutionary algorithms process the information contained in their populations, and how the evolutionary algorithms should exploit this information (Holland, 1975). Schemata are generalized descriptors of properties found in an individual's genotype; several individuals may share the same property, and it is hoped that an evolutionary algorithm will promote the spread of beneficial properties and curtail the propagation of detrimental properties. A single schema describes a set of individuals by enumerating a group of gene values (alleles); all the individuals that contain these alleles belong to the schema. A standard way to represent schemata is to use vectors whose nth element either specifies a particular allele for the nth gene or allows the nth gene to remain unspecified. In this way, schemata are simple, fixed-length patterns containing symbols specific to certain alleles and possibly wildcard characters. For example, a schema describing binary genotypes could be expressed as a sequence from the set {0,1,#}—the schema '0#0' would "match" the genotypes '000' and '010'. Occurrences of either of these genotypes would provide fitness information on this schema, as well as any other schemata that matched the genotypes. The "order" K of a schema is the number of fixed genes in the genotype; our example schema '0#0' is order 2. We denote "order K" as $o(K)$. A schema's order can range from 0 to l, where l is the number of genes in (fixed length) genotype.

Schemata processing occurs in both mutation- and pattern-based algorithms; schemata processing occurs when schemata are compared and combined in order to form new solutions. Mutation-based evolutionary algorithms take a single genotype and randomly change parts of it; they never deal with parts of multiple genotypes simultaneously. As a result, these algorithms perform the most basic type of schemata processing since they evaluate and compare only $o(l)$ schemata—the highest order schema that matches the current individual. Since pattern-based methods combine genotypic parts of various size from multiple individuals, they process all schemata from $o(1)$ to $o(l)$. This allows pattern-based algorithms to be more "conservatively explorative" than mutation-based algorithms, since recombination of *previously tested* genotypic parts is a more cautious strategy than randomly mutating existing parts into *completely new* forms (Syswerda, 1993). But how do pattern-based algorithms choose the "best" parts to recombine? Somehow, the "fitness" of schemata must be gauged. The next section will describe how this can be accomplished.

Schemata and Fitness Information

Each sample point x yields info on all schemata S_x for which $x \in S_x$, i.e. all those schemata whose defined alleles match those found in x. As a result, each x is a member of 2^l schemata, and therefore very few genotypes are needed to sample large numbers of schemata. However, there exists a trade-off between the number of schemata sampled and the amount of information per schema. The amount of information gathered by one genotype about one schema depends on the schema's order. A genotype is an instance of 2^l schemata ranging from $o(1)$ to $o(l)$, but yields less information about lower-order schema. For example, if a schema is $o(l)$, then every position in that schema is fixed, and so only one genotype will be a member of the schema. Therefore the matching genotype will yield complete information on such a schema. Conversely, an $o(0)$ schema will contain l undefined genes, and so 2^l genotypes will be members of the schema. Each member genotype will sample only $1/2^l$th of the schema, and so yield vastly less information about the schema.

Evolutionary algorithms that utilize schemata must rely on averages *observed* in the sample population. It is possible to define the "fitness" of a schema as the average fitness of all individuals belonging to the schema, whether or not these individuals are in the current population. However, except for the highest order schemata, it is impractical to calculate schema fitnesses, due to the large number of individuals that match each schema. As a result, algorithms must rely on limited samples of individuals to collect information on the true average fitness of each schema. The statistics collected in such a way are "observed" statistics, instead of the true statistics of the schemata. "Efficient" processing of schemata through the use of observed statistics remains an open research question, as observed average schemata fitnesses could easily be misleading—low-fitness schemata may have high observed fitnesses, and high-fitness schemata may not even be represented in the population. Because of this "sampling error", a search algorithm must strike a balance between the exploitation of observed high-fitness schemata and the exploration for more information about observed and unobserved schemata. In the remaining parts of this thesis, the "fitness" of a schema will refer its *observed* average fitness.

The greater the amount of knowledge an evolutionary algorithm has about the search space, the greater its chances of success. (Goldberg, 1989) estimated that a random population of size λ will sample on the order of λ^3 schemata. Consequently, the computational burden of explicitly recording statistics about each schema encountered quickly becomes prohibitive as population size is increased. Still, larger populations may be more desirable than smaller populations, as they implicitly provide information on many more schemata than smaller populations, and judicious use of this extra information can increase the probability of discovering highly beneficial genotypic properties. Genetic algorithms deal with this problem of "information overload" through the use of fitness-proportionate schemata representation. Instead of using explicit calculations to keep track of schemata fitness, observed schemata fitnesses are *implicitly* recorded, by including higher fitness schemata in the population more often than lower fitness schemata. A fitness-proportionate selection operator chooses above-average fitness individuals for reproduction more often than below-average fitness individuals. This action engenders a positive feedback phenomena, in which GAs allocate trials to schemata with above-average observed fitness more often than to those of below-average observed fitness, and the population becomes increasing composed of higher (observed) fitness schemata, which in turn increases the probability of the selection of these schemata. This fitness-proportionate representation of schemata has the additional benefit of focusing the search by probabilistically limiting the possible combinations of schemata achievable through the application of the crossover operator; the search collects more information on higher fitness schemata, as opposed to collecting equal amounts of information about all schemata. However, only the selection operator performs schemata analysis, and it examines only $o(l)$ schemata. The mutation and recombination operators of genetic algorithms perform no analysis on schemata; they rely on the selection operator to choose highly fit individuals for manipulation, with the assumptions that these individuals will contain schemata important for the construction of even higher fitness individuals and that enough random mixing of schemata will eventually result in the creation of the "best" combinations of schemata. As a result, the population of the GA allows it to avoid the infeasible task of explicitly recording average schemata performance, while at the same time maintains and adapts the GA's search focus.

The "Golden" Heuristic?

A blanket assumption that fitness proportionate schemata selection is the "best" selection method to use is simply not true. The heuristics an algorithm uses to guide its search may have to change, depending on the specific task to be accomplished. Sometimes the goal of evolutionary search is not approximation, but the maximization of the cumulative fitness of all function evaluations. (Holland, 1975) proved that allocating exponentially increasing trials to schemata of above-average observed fitness is a minimum-loss strategy. It must be recognized that this strategy is only a *heuristic* for guiding evolutionary search. Whether high fitness schemata should be propagated or not depends on the type of problem attacked, the operators employed by the algorithm, and the goal of the algorithm itself. In optimization problems, the goal is to find very high fitness individuals in as few function evaluations as possible. While allocating trials on the basis of observed schema fitness probably is a good heuristic for locating this individual, it is quite possible that the highest fitness schemata do *not* contain the highest fitness individuals. This phenomena is called "deception" (Goldberg, 1989). When deception is present, the blanket assumption that above-average fitness schemata should be utilized more often than below-average schemata is incorrect. In such situations, an algorithm must take action to avoid converging permanently to the deceptive local optima. These actions could include restarting the search with a new individual that is maximally different from the local optimum (in a binary problem, for instance, the maximally different individual would be the bitwise complement of the local optimum). Therefore, the "golden" heuristic of fitness proportionate schemata selection must be supplemented by other heuristics in order to effectively deal with deceptive and other difficult situations.

2.4 Eugenic Combinatorial Approximation

All of the Darwinian evolutionary search algorithms perform minimal analyses of the information contained in their populations; the only information used in the construction of new individuals is a list of one or more previously encountered individuals and their corresponding fitnesses. These algorithms rely on a large number of trials and repeated applications of random operators to overcome their lack of domain knowledge and "blindly" strike about in the search space (albeit in regions biased by the list of historical individuals). The mutations made by these algorithms may be small or large, increasing or decreasing in size, but all the mutations are *random*, in the sense that the choices of what parts to mutate and by how much are not dependent on any analysis of what changes might be best. The genetic recombinations that take place in pattern-based methods are not guided by any criteria of which parts would work most successfully together; parts are just randomly extracted from higher fitness individuals and patched together. Eugenic evolution hopefully reduces some of the need for many trials by performing reasonable amounts of analysis of the information gathered in previous trials and pro-actively exploiting this information to guide the choices made for choosing and combining genetic material.

At least two algorithms, "Binary Simulated Crossover" (BSC) (Syswerda, 1993) and "Population-Based Iterative Learning" (PBIL) (Baluja, 1994), have attempted to pro-actively and explicitly identify and exploit correlations between fitness and genetic parts or complete genotypes. Both are pattern-based methods, but PBIL also can be considered to be a mutation-based method, even though it makes use of a population of samples. BSC proceeds as a SSGA, but does not construct new individuals through random crossover and mutation. Instead, the alleles of a new individual are generated independently, using modified versions of GA selection techniques such fitness-proportionate selection or rank selection. Instead of using these selection techniques to choose *individuals* to participate in reproduction, these techniques select *alleles* for inclusion in a completely new individual. The "fitness" in "fitness-proportionate selection" would refer to an allele's fitness, instead of an individual's fitness. For example, if fitness proportionate selection was being used, and a specific allele of a gene was found in several individuals whose mean fitness was 50, while another allele of the same gene was found in individuals whose mean fitness was 25, then the probability of use of the first allele in new individuals would be twice that of the second—the first allele would be two times more likely to appear in a new individual. The "fitness" of an allele is dependent upon a statistic computed over the individuals in which it is found, and in this case the statistic is the sample mean of the individuals' fitnesses. BSC relies solely on measurements of *marginal* allele-fitness distributions; it makes no effort to determine the dependence of an allele's fitness conditional upon the presence or absence of other alleles. BSC makes no attempt to determine which allele *combinations* go best together; it completely ignore schemata of order greater than one. As a result, BSC only analyzes $o(1)$ schemata and therefore completely ignores epistasis.

PBIL also forsakes random crossover and mutation for an alternate method of offspring creation. A "probability vector" is used to generate each population. This probability vector is simply a vector whose nth component represents the probability that the nth gene will take the '1' form in the next population. An entire population is randomly generated with this single vector. After each population is generated, the single best and worst individuals in the population modify the probability vector so as to increase the likelihood that the alleles found in the best individual but not in the worst individual will be produced in the next generation. In a somewhat similar way to simulated annealing's cooling schedule, the amount that the probability vector is modified by decreases with time. PBIL is pattern-based in the sense that the probability vector represents a pattern of "good" individuals, while it is also mutation-based in that a single pattern that is the "starting point" to generate new solutions. PBIL's use of a single probability vector is basically equivalent to a BSC algorithm that combines the allele-fitness averages of the highest and lowest fitness individuals from many generations. Since PBIL only processes entire genotypes and probability vectors, and never attempts to analyze parts of these genotypes or vector, it processes only $o(l)$ schemata.

In both BSC and PBIL, all alleles participate equally in updating the fitness averages and selection probabilities, and no attempt is made to sort out the possibly conflicting information coming from very diverse individuals. BSC processes only $o(1)$ schemata, and PBIL processes only $o(l)$ schemata—neither algorithm processes intermediate-order schemata. It will be demonstrated in later chapter that the eugenic algorithm, developed independently of both BSC and PBIL, processes schemata of all orders, through the use of measurements of both marginal and conditional allele-fitness distributions. Therefore the EuA additionally has the extremely important ability (lacking in both BSC and PBIL) to track and exploit epistasis by explicitly examining allele behavior that is dependent upon other alleles, even when these dependencies only occur in schemata of less than order l.

Chapter 3

Problem Structure and Difficulty

A clear understanding of the optimization problems faced by evolutionary algorithms is necessary to fully appreciate the mechanism of the EuA. In this chapter, five major qualitative characteristics of discrete combinatorial optimization problems (attractors, hill-climbability, deception, epistasis, and fitness distribution) are defined and discussed. Although none of these concepts are new, they are usually not defined very concretely. This section will attempt to define and relate them into a unified view of problem structure. Attractors are optima (local or global) that have basins of attraction. Hill-climbable problems are ones that can be solved easily using exploitation—when no suboptimal attractors exist. When suboptimal attractors do exist, both deception and epistasis can occur. Deception is encountered when strong suboptimal attractors exist that are very genotypically different from the global optima. Problems with high epistasis have a great number of small attractors. Greater deception and higher epistasis lead to greater problem difficulty. Unlike deception and epistasis, the relationship between fitness distribution and problem difficulty is not so concrete. Extremely small or large fitness variance or ranges can either adversely affect the success of evolutionary optimization algorithms, decrease problem difficulty, or have no effect at all. It can be very difficult to determine by observation the extent to that these problem characteristics manifest themselves in a particular problem, but quantitative methods for measuring a problem's epistasis and fitness distribution will be proposed in this thesis. However, before a full discussion of any of these characteristics is possible, the difference between the concepts of *genotypic distance* and *algorithmic distance* must be explored and clarified.

3.1 “Distance”: Genetic and Algorithmic

In GA literature, the term “distance” has been associated with both genotypic and algorithmic distance. Although both measures of distance are often directly proportional, this is not always the case. A well known measure of genotypic distance is Hamming distance. Hamming distance is a suitable metric for computing genotypic distance for binary genotypes, as it simply counts the number of gene positions that differ between two individuals. As Hamming distance increases, genotypic similarity decreases. Algorithmic distance, on the other hand, is the amount of search an algorithm must perform to generate one point from another. It is inversely related to transitional probability—the probability that a particular offspring will be generated from a given parent. The algorithmic distance between point A and point B increases as the likelihood decreases that an algorithm’s search will progress directly from point A to point B. If the algorithmic distance between point A and point B is infinite, then the transitional probability from point A to point B is zero. Although algorithmic and genotypic distance are often related, they are independent of each other, since algorithmic distance varies depending on the algorithm in use, while genotypic distance remains constant.

There are cases in which algorithmic distance is directly proportional to genotypic distance. This is true for some of the most popular evolutionary algorithms. In these cases, the use of the term “distance” to simultaneously refer to both distance measures has been both convenient and justified. For mutation-based algorithms that use small mutations, Hamming distance can easily be used to estimate the algorithmic distance of two points. These methods rely on small perturbations of the current solutions to generate new solutions, and so points separated by small Hamming distances are highly likely to be generated from each other, while points separated by large Hamming distances very likely will never be generated from each other—they are truly “far away” from each other. Therefore, for these algorithms, the qualitative relationship between genotypic and algorithmic distance is simple and static. The concepts of genotypic proximity and transitional *im*probability are truly interchangeable, and both concepts can be easily unified into a single idea of “distance”.

For pattern-based algorithms, the concept of distance is much more complicated. An intuitive assumed equivalence between genotypic and algorithmic distance is not valid, since extremely large perturbations to the current solution(s) can and do occur from one iteration (generation) to the next. It is often very likely that a new individual will differ significantly from every individual in its parent population, even though the new offspring was created with only a single iteration of the search. All that is likely to remain similar between parents and offspring are the special patterns that the algorithm promotes. When these patterns include most of an individual's alleles, as in mutation-based algorithms, Hamming distances remain small from generation to generation—genotypic distance is small when algorithmic distance is small. When the exploited patterns are more complex, such as the random-length schemata of GAs, the Hamming distances between offspring and their immediate parents can often be very large. Therefore, although the genotypic distance between two individuals may be very large, the *algorithmic* distance could still be very small. The simple relationship between genotypic and algorithmic distance that is encountered in many mutation-based methods breaks down as the complexity of the exploited patterns increase. For pattern-based methods, and for mutation-based methods that utilize extremely high mutation rates, genotypic distance is often largely unrelated to algorithmic distance.

3.2 Attractors

Because of the breakdown of the relationship between genotypic and algorithmic distance, the ideas of "attractors" and "local optima" must be approached very carefully, since the very idea of "locality" depends on the distance measure being used. Local optima are *suboptimal* points or regions that optimization algorithms frequently converge to, for one reason or another. Accompanying these sub-optima are their "basins of attraction", which are regions of the search space such the algorithm is likely to converge to the sub-optima if it enters the region. Whether optimal or suboptimal, points (or regions) having basins of attraction are called "attractors". An attractor for mutation-based algorithms is not necessarily an attractor for pattern-based algorithms. Attractors for mutation-based algorithms (*mutation-based attractors*) are points that are superior to points in the *genotypic* neighborhood. Fitness in this neighborhood monotonically decreases with increasing genotypic distance from the attractor. Points that are more similar to the attractor have higher fitness, while points that are less similar have lower fitness. Therefore, a mutation-based algorithm that encounters a point in the basin of attraction can easily increase the fitness of the solution by mutating the point to become more similar to the attractor. Once inside the basin of attraction, a mutation-based algorithm would converge to the attractor, unless the algorithm performed large mutations ("jumps") that made the current solution less similar to the attractor. The "strength" of a mutation-based attractor depends on how far out its basin of attraction extends (i.e. at what genotypic distance does fitness stop monotonically decreasing), and on how many points inside the basin of attraction violate the rule of monotonically decreasing fitness.

In contrast, attractors for pattern-based algorithms *pattern-based attractors* are rooted in suboptimal *patterns*, not suboptimal points. When a genotypic pattern is observed to have fitness superior to that of its alternatives, and therefore promoted throughout the search, then this *pattern* can be considered to be an attractor for a pattern-based algorithm (see "Deception", below). The fitness of a pattern is generally considered to be the average fitness of all individuals encountered that match the pattern, but the choice of the statistic used (whether mean, mean-squared, etc.) is arbitrary. Patterns arise through observations of many different individuals, but the patterns do not necessarily lead to any particular highly-fit point. The observations are incomplete, because they often do not sample *all* the points matching a particular pattern, and therefore any statistics calculated using the observations are prone to sampling error. Therefore, the patterns are "noisy" attractors that help guide the algorithm towards *seemingly* good groups of points, not necessarily directly toward specific superior individuals. The strength of a pattern-based attractor increases as the mean fitness of the pattern increases and its variation decreases. Strength decreases with increasing pattern length, because the pattern becomes more specific and therefore matches fewer individuals. As a result, it is less likely that the pattern will be encountered and, when encountered, sampling error will be higher since fewer matching individuals will be available. In effect, by increasing pattern length, the basin of attraction of the pattern decreases in size.

3.3 Hill-Climbability

For many evolutionary algorithms, there are some types of discrete optimization problems that are easier to solve than others. In particular, the characteristic of "hill-climbability" can be found in many of the simplest problems. In hill-climbable problems, the exploitation of observed superior properties leads easily to the creation of an optimal individual. In such problems, there is little or no need for exploration and, specifically, there is no need to discard an observed superior pattern or individual in favor of a inferior one. Varying degrees of hill-climbability exist. Problems that have more suboptimal attractors are less hill-climbable, as are problems whose suboptimal attractors have larger basins of attraction, since exploitative algorithms are more likely to converge to the sub-optima in these cases. Therefore hill-climbability decreases with increasing numbers and strength of suboptimal attractors.

There are two types of hill-climbability—mutation-based and pattern-based. In a mutation-based hill-climbable (MBHC) problem, all the mutation-based attractors are global optima. This means that from every non-optimal point in the search space, there are many single-bit mutations that lead to points of higher fitness. Therefore, there is an *exploitative*, single-bit "mutation-trajectory" from every point in the space to a global optimum. A MBHC problem can be solved very easily by a mutation-based "hill-climbing" algorithm, which performs single-bit mutations and replaces the current individual only when the new individual has higher fitness. Later in this thesis, a specific mutation-based hill-climbing algorithm will be developed to determine the MBHC of any problem.

Pattern-based hill-climbability (PBHC), better known as “building block constructivity”, has been discussed extensively in GA literature (Goldberg, 1989). In theory, a PBHC problem could easily be solved by exploiting alleles and allele sets that have superior observed fitness averages. Such alleles or allele sets would surely be present in a global optimum, and therefore it would be a simple matter for an algorithm to combine superior patterns into an optimum individual. However, in practice, even toy problems that have been designed specifically to be building block constructive are *not* that easily optimized by pattern-based algorithms. However,it has been demonstrated that GAs can perform relatively poorly on problems that are specifically designed to be building-block constructive (the “Royal-Road” functions described in (Forrest and Mitchell, 1993)). In particular, far simpler approaches such as random bit-climbing algorithms have been shown to easily outperform GAs on several of these problems. It is now clear that there are many other factors involved in GA performance, and that building-block constructivity cannot be so easily measured. It is usually the case that only “toy” problems can be clearly seen to be optimizable through the recombination of short, low-order schemata. Similar to MBHC, PBHC decreases when the number and strength of pattern-based suboptimal attractors increase.

In order to more fully understand both mutation- and pattern-based hill-climbability, it is useful to study a problem that is completely mutation- and pattern-based hill climbable. The BIT-COUNT problem is such a problem. It is probably the simplest toy problem used to test and illustrate the actions of evolutionary algorithms. In BIT-COUNT, the fitness of each point ($\mathbf{x}$) is its Hamming distance from the zero-vector ($\mathbf{x} = 0...0$). The optimum is therefore reached when all the bits are set to ’1’ ($\mathbf{x} = 1...1$), since this point has the highest Hamming distance from the zero-vector. It is mutation-based hill-climbable because any mutation from a ’0’ allele to a ’1’ allele leads to an individual of higher fitness that is also closer to the optimum, while mutations from ’1’ alleles to ’0’ alleles lead to individuals of lower fitness that are farther away from the optimum. Therefore a completely exploitative mutation-based algorithm will easily and quickly locate the optimum, and the problem is therefore mutation-based hill-climbable. In addition, all schemata that contain more ’1’ alleles than their rivals will have higher average fitnesses than their rivals, if the number of observations over which these averages are computed is high enough. As a result, the complete exploitation of these higher fitness schemata will lead to optimization. Therefore BIT-COUNT is also pattern-based hill-climbable. It should, however, be noted that pattern-based algorithms do not usually have access to complete schemata information—they must rely on only small samplings of schema members to compute schema fitness, and are therefore vulnerable to the errors associated with incomplete sampling. The fitness averages observed in smaller samples may incorrectly reflect relative fitnesses, and therefore an exploitative pattern-based algorithm may erroneously promote patterns not matching the optimum. As a result, such algorithms will not always find the optimum of BIT-COUNT as quickly as a completely exploitative mutation-based algorithm.

3.4 Deception

Even though a problem may display a high degree of *observed* hill-climbability—where exploitation leads to significant and reliable increases in fitness—the problem may still have local optima. When local optima have very large hill-climbable basins of attraction and differ greatly from the global optima, the problem is "deceptive"[1]. In this case, the easy "progress" made by hill-climbing algorithms is deceptive, as it leads only to local optima that are very genotypical different than the global optima. For example, a particular problem may have one schema that is observed to have a greater average fitness than its competitors. However this "better" schema might not be part of the optimum genotype, even though the promotion of this schema above its competitors results in individuals of increasingly higher fitness. In fact, the schema may match individuals that differ significantly from the optimum individuals. In this case, the problem would be pattern-based deceptive. A similar example can be constructed for mutation-based deception, where mutations towards a local attractor would lead farther and farther away from the true global optimum.

Depending on a problem's attractors, both pattern-based deception as well as mutation-based deception are possible, but do not necessarily always coexist. "DECEPT" is standard GA test problem that is both pattern- and mutation-based deceptive. DECEPT is very similar in structure to BIT-COUNT, but DECEPT is a much harder problem for an evolutionary algorithm to solve. DECEPT (equation 3.4) is basically identical to "BIT-COUNT", except that the zero-vector ($\mathbf{x} = 0...0$) itself is the optimum, with slightly higher fitness than the one-vector ($\mathbf{x} = 1...1$). In both problems, when all non-optimal individuals are examined[2], and then the average fitness of both the '0' and '1' alleles are calculated, it will be found that the average fitness of the '1' allele is higher than the average fitness of the '0' allele. For the BIT-COUNT problem, this intuitively confirms the presence of the optimum at $\mathbf{x} = 1...1$. However, for the DECEPT problem, it contradicts the intuition that the observed "average best" allele (allele '1') will be the one present in the optimum, since

[1]Deception is typically said to exist in problems with "misleading" (or locally optimal) *schemata* (Goldberg, 1989), and therefore this definition does not distinguish between mutation- and pattern-based deception.

[2]It is reasonable to exclude the optimum from the calculation of allele or schema fitness, since we are assuming that the optimum is not known *a priori* (that's why we need an optimization algorithm!).

$$l + 1 \text{ when } \mathbf{x} = \mathbf{0} \tag{3.1}$$

$$Hamm(\mathbf{x}, \mathbf{0}) \text{ otherwise} \tag{3.2}$$

Figure 3.1: The DECEPT problem

the optimum contains only '0' alleles. As a result, DECEPT is significantly harder for both mutation-based and pattern-based algorithms. It is hard for mutation-based algorithms to optimize since fitness will increase each time a '0' is mutated to a '1' (except when starting from the optimum). Each seemingly "good" mutation is actually an "incorrect" mutation, in the sense that it will lead to a sub-optimal solution and away from the true global optimum. DECEPT is hard for pattern-based algorithms because, in populations that do not contain the optimum, every "schemata competition"[3] is won by the schema with the greater number of '1' alleles, and so more '1' alleles are propagated through the population. A pattern-based algorithm will allocate increasing amounts of trials to "misleading" schemata that can never be recombined to form the optimum.

3.5 Epistasis

In general, the degree to which a problem exhibits epistasis is the degree to which the "best" setting of an allele is dependent on the settings of other alleles—it is the amount of interdependence among a genotype's alleles (Goldberg, 1989). However, little attention has been given to the interaction between attractors and epistasis. Epistasis occurs when a problem has an extremely high number of very small mutation-based attractors, to such an extent that there are almost no pattern-based attractors, since the basins of attraction of the sub-optima contain so few individuals over which patterns can be discerned. If the problem has only one (global) attractor, then the best setting of every allele is always independent of the settings of the other alleles, and the problem is completely unepistatic—it exhibits 0% epistasis. There is no need to coordinate the selection of alleles; each gene has an elite allele that is consistently found in superior individuals, and so it is a simple matter to construct high fitness individuals using these alleles. BIT-COUNT is an example of a 0% epistatic problem. It has no suboptimal attractors. For any given individual, the best setting of every allele is completely independent of the other alleles.

[3] I define a "schemata competition" as a comparison between two schemata with the same order and fixed positions

Conversely, if the best setting of an allele can never be specified without the examination of all the other allele settings, then the problem is 100% epistatic. The best allele choice for a gene can rely heavily on a suitable and “harmonious” setting of other genes. 100% epistatic problems are hard for both mutation-based and pattern-based optimization algorithms, since the settings of almost all of the genes must be coordinated simultaneously in order to create a superior individual. For example, in a mutation-based algorithm, such as SA, the alleles found in the current sample roughly indicate “good” settings. As the search progresses, fewer alleles are changed and correspondingly more alleles are left untouched, as it is hoped that higher and higher fitness combinations have been found. But in a 100% epistatic problem, the “best” setting of each allele changes any time another allele changes values. Therefore SA’s heuristic of mutating fewer alleles as better solutions are found is much less effective. The heuristic of promoting good sets of alleles (schemata), used by pattern-based algorithms, is also less effective in these circumstances, as the fitness of each schemata becomes too dependent on the presence or absence of other “companion” schemata. As a result, random recombination of schemata must rely on the random chance that two entire schemata will work well with each other. Because of the combinatorial explosion of the number of possible combinations of schemata, this random chance is very small. Consequentially, 100% epistatic problems are the hardest problems for evolutionary algorithms.

Epistasis can be partially studied by examining single-allele competitions between otherwise identical genotypes. Such a competition is performed when a single allele of an individual is mutated into an alternative allele. The fitnesses of both the original and the mutated individual are determined, and the allele associated with the higher fitness individual wins the competition. These competitions can be performed on any individual in the search space. In a 0% epistatic problem, each gene will have a special allele that it *always* wins its competitions with its alternate allele, no matter how the rest of the genotype is configured. In a 100% epistatic problem, however, the winners of these competitions will be equally balanced among all the competing alleles. For example, if the genotypes are binary, then the value of ’0’ for a particular gene will yield superior fitness for half the genotypes, while ’1’ will be better for the other half. In effect, the search space is partitioned into two sets by a particular gene—the first set containing all those genotypes for which ’0’ yields the highest fitness, and the other set that contains those genotypes for which ’1’ is best. In a 100% epistatic problem, these two sets are equal in size—each allele is best only half the time. However, as one set increases in size and the other shrinks, a “pattern” emerges in which one allele more frequently is a better choice, resulting in a corresponding decrease in epistasis. Therefore, by keeping track of the frequency with which alleles yield higher fitness than their complements (i.e. win a competition), epistasis can be estimated. More "balanced" win frequencies imply more epistasis, as they imply that it is very difficult to decide which allele is better, without examining the rest of the genotype. However, this epistasis estimate only deals with o(1)-schemata; it ignores competitions among higher-order schemata and how unbalanced the win frequencies of these schemata are. It is unclear as to whether or not this "o(1)-epistasis" is an under-estimator or over-estimator of overall epistasis; figure 3.5 shows an example problem that exhibits almost 100% o(1)-epistasis, but in which higher order competitions are unbalanced, and therefore epistasis is lower for higher-order schemata. Equation 3.3 defines the relationship between genotypes and fitness; it was designed so that just about every $o(1)$ competition was perfectly balanced.

$$f(\mathbf{x}) = \begin{cases} 2\mathrm{INT}(\mathbf{x}) & \text{when } \mathrm{INT}(\mathbf{x}) < 2^{l-1} \\ 2(2^l - \mathrm{INT}(\mathbf{x})) & \text{otherwise} \end{cases} \tag{3.3}$$

Individual		**Gene 1** (x_1)		**Gene 2** (x_2)		**Gene 3** (x_3)	
x	$f(\mathbf{x})$	x_1='0'	x_1='1'	x_2='0'	x_2='1'	x_3='0'	x_3='1'
000	1	1	-	1	-	1	-
001	3	3	-	3	-	-	3
010	5	5	-	-	5	5	-
011	7	7	-	-	7	-	7
100	8	-	8	8	-	8	-
101	6	-	6	6	-	-	6
110	4	-	4	-	4	4	-
111	2	-	2	-	2	-	2
$f_\mu(x_i = a)$		4.0	5.0	4.5	4.5	4.5	4.5
$p_\mu(x_i = a)$		0.44	0.56	0.5	0.5	0.5	0.5

Figure 3.2: A
3-bit problem with extremely high o(1)-epistasis; the mean fitness of almost every allele is equal to that of its competitor. $f_\mu(x_i = a)$ is the mean fitness of allele a for gene x_i, and $p_\mu(x_i = a)$ is ratio of allele of this mean fitness over the sum of all the mean fitness of all alleles for gene x_i. When fitness-proportionate selection is being used, $p_\mu(x_i = a)$ would be the probability of selecting allele a for gene x_i. Note that for genes 2 and 3, the probability of selecting each allele is identical, and for gene 1, the allele selection probabilities are fairly equal. The allele selection probabilities for gene 1 would become increasingly similar as the number of genes is increased. Since the allele selection probabilities are almost all balanced, it is very hard to decide which alleles are the most responsible for increased fitness.

A measure is proposed here that can be used to directly estimate o(1)-epistasis. It measures how hard it is to immediately solve a problem by randomly sampling a small number of genotypes and then create a very high fitness individual by setting each allele the the average winner of each o(1) competition. In a 100% o(1)-epistatic problem, allele win frequencies would be balanced, and so the decision to use one allele or the other would be difficult. Since there may be hundreds of genes in genotype, it would not be adequate to simply count the win frequencies of the alleles of a single gene; the win frequencies of all the alleles of all genes must be recorded, and how much this distribution of frequencies differs from a completely balanced distribution (expected in a 100% epistatic problem) must somehow be measured. The chi-squared (χ^2) test (equation 3.4) can be used to measure this difference between the observed win frequencies and expected win frequencies. The chi-squared test is used to determine how well theoretical distributions fit empirical distributions. In this case, the theoretical distribution (e_i) of 100% epistasis expects each allele to win 50% of its competitions. Therefore, if there are N competitions per gene, each allele should win $N/2$ competitions (if a binary genotype is being examined). The empirical distribution (o_i) is the actual number of times each allele wins an o(1) competition over the course of the N competitions. For binary genotypes, there are two alleles per gene, and therefore alleles can be numbered from 1 to ($2l$). χ^2 is calculated by summing the frequency "errors" $\frac{(o_i-e_i)^2}{e_i}$ for all ($2l$) alleles. The minimum possible value (0) for χ^2 is achieved when the observed distribution o_i of every allele i exactly matches its expected distribution e_i—when there is 100% epistasis. The maximum possible value ($l \times N$) occurs when the allele competitions are completely unbalanced—either an allele wins all N competitions, or it wins zero competitions and its complementary allele wins all N competitions. When the allele competitions are completed unbalanced in this way, χ^2 is a sum of $2l$ integers, of which l have the value N and l have the value zero. Therefore the maximum possible value of χ^2 is ($l \times N$). Equation 3.5 is used to calculate a value for o(1) χ^2-epistasis E_{χ^2} that ranges from zero to one. A value of zero would indicate 0% o(1) χ^2-epistasis, while a value of one would indicate 100% o(1) χ^2-epistasis. The extension of this method, to measure the dependence of higher order schemata on the settings of alleles outside those schemata, is straightforward but would require exponentially increasing amounts of computation with each increase in schemata order.

To calculate o(1) χ^2-epistasis, single allele competitions must occur between genotypes that differ by only one allele. An algorithm for calculating o(1) χ^2-epistasis is specified in figure 3.4.

Looking back at the DECEPT example, we see that, even though the problem is fairly difficult, there is actually very little epistasis in the problem since there is only one suboptimal attractor, and since most allele competitions are unbalanced (the '1' allele most

$$
\begin{aligned}
N &= \text{number of competitions performed} \\
o_i &= \text{observed number of wins for allele } i \\
e_i &= \text{expected number of wins for allele } i = N/2 \\
i &\in 1..(2l) \\
E_{\chi^2} &= \text{o}(1)\ \chi^2\text{-epistasis}
\end{aligned}
$$

$$\chi^2 = \sum_{i=1}^{2l} \frac{(o_i - e_i)^2}{e_i} \tag{3.4}$$

$$max_{\chi^2} = l \times N$$

$$E_{\chi^2} = 1 - \frac{\chi^2}{max_{\chi^2}} \tag{3.5}$$

Figure 3.3: Calculating o(1) χ^2-Epistasis

```
DO WHILE ($\mathbf{N} < \mathbf{max - N}$)
1.  Choose random genotype x
2.  Evaluate $f(\mathbf{x})$
3.  Choose random gene $g$ in x and change its allele from $i$ to $i\prime$, yielding x$\prime$
4.  Evaluate $f(\mathbf{x}\prime)$
5.  If $f(\mathbf{x}) < f(\mathbf{x}\prime)$, increment $\mathbf{o_{i\prime}}$ else increment $\mathbf{o_i}$
6.  N++
OD
```

Figure 3.4: Pseudo-Code for Calculating o(1) χ^2-Epistasis

often beats the '0' allele). DECEPT is an example of a difficult problem with practically 0% epistasis, but high deception. It is therefore clear that a problem need not be highly epistatic in order to be difficult to optimize.

In summary, if a problem has no suboptimal attractors (whether mutation- or pattern-based), it is hill-climbable, because exploitation of observed higher-fitness schemata leads easily to global optima. If a problem has an extremely large number of suboptimal attractors, it is very difficult to determine when a schema with higher observed fitness is local to a suboptimal attractor, or if the feature is actually present in a global optimum.

Such a problem is highly epistatic. If a suboptimal attractor leads an algorithm to a solution that is significantly (genotypically) different from the global optima, then the attractor is deceptive. If this attractor is strong enough—if it has a large enough basin of attraction—then the problem is deceptive. Hill-climbability, deception and epistasis are all problem characteristics that arise from the genotypic topology of the search space. If this topology changes, then all three characteristics can radically change. All three of these characteristics are relatively independent of raw fitness values, but they are very sensitive to changes in individuals' fitness *ranking*. The next section will discuss "fitness distribution", which is a problem characteristic that is completely dependent on raw fitness values.

3.6 Fitness Distribution

The distribution of a problem's fitness function can be characterized by many different measures: the range of fitness values, their mean and variation, their ratios, their concentration around specific values, etc.. Since most evolutionary algorithms directly use fitness values to calculate important search probabilities such as selection probabilities and mutation rates, a mismatch between a problem's fitness distribution and the probability-calculating functions used by the particular algorithm can cause dismal performance. For example, if an algorithm relies too much on absolute differences in individual's fitnesses, it can have difficulty solving problems that have a very small range of fitnesses. For instance, a GA using fitness proportionate selection would perform miserably on a problem in which fitness ranged from 10^6 to $(10^6 + 1)$, as the algorithm would allocate trials to the best and worst individuals with virtually the same frequency. The same algorithm can run into difficulty when the range of fitness is too wide. If fitnesses ranged from 0 to 10^6, but most individuals had fitness less than 10, the first individual encountered with fitness 10^2 or 10^4 would "swamp" the population, as fitness proportionate selection would be likely to allocate all trials to this individual. This "bad match" between the algorithm's use of fitness values and the fitness range in the problem being attacked would quickly cause premature convergence.

Bad matches can also exist between a problem's fitness distribution and algorithms that do not use fitness-proportionate selection. For example, simulated annealing has an "acceptance probability" that determines whether lower fitness individuals will replace the current individual. This probability changes in relation to the difference in fitness between a new sample and the old sample. When the difference in fitness between two samples is very low, the probability of acceptance can be very high. If a search space is dominated by a dense concentration of low fitness points, the search may spend much of its time wandering among these points, without ever exploiting small increases in fitnesses necessary to climb out of the low fitness region. Almost every mutation will be accepted, no matter what its effect on fitness (since the changes in fitness will be "small"). However, it might be these tiny changes in fitness are what is necessary to guide the search to the best regions of the space. An example of this phenomena will be demonstrated later in this thesis (see the discussion of F31 in the "Experiments" chapter).

Various methods (such as "fitness scaling" (Goldberg, 1989)) have been proposed that alleviate some of the problems associated with mismatches in fitness distributions and specific algorithms. These methods usually try to convert algorithms that rely on fitness-proportionate selection into hill-climbing algorithms that allocate trials on the basis of the fitness-rank-order of the different individuals or schemata in the population.

With respect to evolutionary optimization, the major characteristic of fitness distributions is their variance—in what ranges can the majority of fitnesses be found, and how concentrated are these fitnesses around particular values? Most importantly, from the standpoint of an optimization algorithm, is the fitness variation caused by a particular genotypic feature? How much effect does a particular allele, gene or schema have on fitness—how "significant" are these genotypic features? In the following, a possible method for the analysis of variance (ANOVA) of allele significance will be proposed.

3.6.1 ANOVA of Allele-Fitness

The more predictable an individual's fitness is, given that it contains a particular allele, the easier it should be to construct individuals with desired levels of fitness. An individual's fitness is "predictable" from the presence of a particular allele if the variation of fitnesses of individuals containing that allele (the allele's fitness variation) is low. When the allele's fitness variation increases, it becomes harder to decide whether or not that that allele has a large effect on fitness, as the fitness of individuals containing that allele become less consistent. When *all* the alleles have high fitness variances—when no single allele is consistently associated with a particular small range of fitness values, it becomes very hard to predict fitness from the entire genotype. One way to measure how predictable fitness is from complete genotypes is to sum all the allele fitness variances. I shall call this sum "within-allele fitness variance" (S_w^2). A low S_w^2 implies that many alleles' fitness variances are low, and therefore predictability of fitness from genotypes is high. High S_w^2 values imply that there are many alleles that appear in individuals of wildly different fitness; allele fitness variance is high overall. High S_w^2 values therefore indicate that the genotype-fitness mapping is hard to predict from individual alleles.

How are "high" and "low" values of S_w^2 defined? An allele's fitness variation should be measured relative to the overall fitness variation of the problem. One measure of the overall variation in a problem is "between-allele fitness variance" S_b^2. It measures how much each gene's allele mean fitnesses differ from each other. It is the sum of the square of the differences between all pairs of allele mean fitnesses. This quantity is mathematically equivalent to the sum of squared differences between a population's mean fitness and all allele fitness means.

$$\begin{aligned} i, j &\in 1..2l \\ f_\mu(i) &= \text{mean fitness of allele } i \\ f_\mu(.) &= \text{mean of all allele mean fitnesses} \end{aligned}$$

$$\begin{aligned} S_b^2 &= \sum_{i,j} (f_\mu(i) - f_\mu(j))^2 \\ &= \sum_{i} (f_\mu(i) - f_\mu(.))^2 \end{aligned}$$

When allele means are very different from the population mean, the alleles most highly correlated with high or low fitness become more evident—the *significance* of the alleles becomes more pronounced. For example, consider an l-bit genotype that simply encodes the binary integers from 0 to $2^l - 1$. The mean phenotypic value (fitness) of this problem would be $f_\mu = \frac{2^l - 1}{2}$. The most significant bit (gene) could have one of two allele values—'0' or '1'. Allele '0' would be found in individuals having mean fitness half that of the population at large, while allele '1' would have mean fitness one and half times that of the mean population fitness. Therefore both alleles for this gene would be highly significant, as they both would have a significant influence on the fitness of the individuals they were found in. S_b^2 is the sum of differences in allele pairs, and so when many genes have significant alleles, S_b^2 increases. Higher S_b^2 implies that it is easier for an algorithm to identify alleles that are commonly associated with high fitness individuals.

By combining both between- and within-allele variance of a problem into a single ratio, $\frac{S_b^2}{S_w^2}$, we can measure the overall ease of predicting an individual's fitness, given that specific alleles are present in that individual, and therefore we have taken a large step in categorizing the difficulty of the optimization problem. The numerator, S_b^2, increases when variations in gene significance increase, resulting more predictability. The denominator, S_w^2 indicates how unpredictable fitness is from single alleles, so as predictability decreases, S_w^2 will increase, and thus the ratio $\frac{S_b^2}{S_w^2}$ will decrease. The implications of these observations will be discussed next.

3.6.2 Gene Significance and Fitness ANOVA

In most mutation-based algorithms, each gene has the same probability of mutation. However, in many problems, some genes have greater significance than others. Changes to these genes result in larger changes in fitness than changes to other genes. Most mutation-based algorithms do not exploit differences in significance. Pattern-based algorithms (and particularly GAs) indirectly exploit differences in significance because genes that are higher in significance lose diversity much faster than lower significance genes, and therefore become less and less likely to change settings as the search progresses. Problems with a wide variety of gene significances will have large S_b^2 values, as many alleles will have mean fitnesses differing significantly from the population mean. Problems with high variation in gene significances might be considered to be easier than problems for which all gene significances are equal, since, for the first type of problems, an algorithm could quickly determine what the best settings were for the most significant genes, and thus make large progress initially in the search. Once this was accomplished, the algorithm could then concentrate on determining the best settings of the lower-significance genes.

Although phenotypic fitness may be highly predictable from single alleles, this does not mean that the problem is easily optimizable. Choices must be made among competing alleles. If complementary alleles have the same mean, at least one of their variances must be high, since the entire range of fitness values must be included in the mean calculations. It could be the case that one of the alleles had very low fitness variation, while the other allele was frequently found in both the best and worst individuals and therefore had very high fitness variation, but still had the same mean as the first individual. What would be the best allele to use when constructing a new individual? If it could somehow be determined *why* the second allele was present sometimes in high fitness individuals, and other times in low fitness individuals, then it might be possible to carefully construct a new individual that met all the conditions found in the high fitness individuals, and therefore the second allele could be used with confidence. An optimization algorithm—the eugenic algorithm—will be presented in the next chapter that accomplishes this goal through the use of an operator (the "restriction" operator) that causes *conditional* allele-fitness distributions to be measured and therefore determines what *combinations* of alleles lead to extremes in fitness.

Chapter 4

The Eugenic Algorithm

The Binary Simulated Crossover (BSC) (Syswerda, 1993) algorithm explicitly analyzes and recombines o(1) schemata to form new individuals. Population-Based Iterative Learning (PBIL) (Baluja, 1994) uses a probability vector as an explicit "fuzzy" representation of a "good" o(l) schema from which similar o(l) schemata are constructed. What is missing from both these algorithms is the explicit analysis and processing of intermediate order schemata—o(2) to o($l-1$). There are a great many of these schemata, and an explicit analysis of every one of them is infeasible. If a search algorithm is to perform explicit analyses of intermediate order schemata, it must limit its computational burden to only a small subset of these possibilities. The chosen subset should include only those schemata that are most relevant to reducing uncertainty and furthering the search.

A new algorithm that accomplishes these objectives is proposed in this chapter. It achieves *smart recombination*, in addition to *smart mutation*. This "Eugenic Algorithm" (EuA) consists of three major elements: (1) selection of promising alleles, based on an explicit analysis of allele fitness distributions, (2) restriction of the population to relevant individuals and (3) steady-state replacement. This chapter will begin with an illustrative description of example execution of the EuA, followed by discussions and justifications of how good alleles are selected, relevant individuals are identified, and why steady-state replacement is used.

4.1 Example of Eugenic Algorithm Execution

Since many of the concepts introduced in the EuA have never previously been integrated into the same algorithm, an illustrative example will be used to help explain the workings of the EuA. I shall list EuA pseudo-code before beginning the example.

```
L01: pop = random-pop() /* Randomly initialize the population: */
     /* N counts the number of individuals created by the EuA: */
     N = 0
     /* Keep on creating new individuals until N exceeds $N_{max}$ */
L02: DO WHILE ($N < N_{max}$)
          u = { $x_1, x_2, x_3, ..., x_l$ }
          rpop = pop
L03:      DO WHILE nonempty(u)
               /* Select and set a gene: */
L04:           $x_g$ = most-significant-gene(rpop, u)
L05:           $x_{g,a}$ = select-allele($x_g$)
L06:           Remove $x_g$ from u
               /* Restrict population when epistasis of unset genes is high: */
L07:           $E$ = epistasis-of-population(rpop, u)
L08:           $p$ = probability-of-restriction($E$)
L09:           if (uniform-random[0,1] ¡ $p$)
L10:                rpop = restrict-population(rpop, $x_{g,a}$)
          OD
          /* Replace lowest-fitness individual with the new individual */
L11:      w = lowest-fitness-individual(pop)
L12:      pop$_w$ = $\mathbf{x}$
          N++
     OD
```

Figure 4.1: Pseudo-Code for the Eugenic Algorithm

4.1.1 Pseudo-Code for the Eugenic Algorithm

In figure 4.1, pseudo-code is shown that outlines the eugenic algorithm. The subroutines it relies upon will be described in the following sections.

The EuA begins by generating a random population of size λ in step L01. In the simplest case, the population size λ will remain constant during the entire execution of the algorithm. Once the random population is created, the algorithm repeatedly creates a new individual (steps L04 through 10) and replaces the lowest-fitness individual in the population with the new individual (steps L11 and L12) until the maximum number of individuals have been created. There are obviously many other possible termination criteria, such as stopping when an individual with particular level of fitness has been encountered.

The EuA constructs new individuals one gene at a time. Initially, all the genes have no set value—they are "unset" with u the set of unset genes (see steps L02, L04, and L10). The EuA orders genes—from most significant to least significant. Once the most significant gene remaining in u is determined (step L04), an allele value is selected for that gene (step L05). Once a gene's value is set, that gene is removed from u (step L06).

After setting the value of a gene, the EuA estimates the amount of epistasis E that the population exhibits among the remaining unset genes (step L07). The EuA uses E to determine p, the probability of restricting the population to only "relevant" individuals (step L08). If epistasis is high, then the necessity of restriction will be high, and therefore p will be high. In step 09, the algorithm probabilistically decides whether or not to restrict the population. If a random sample from a uniform variable in the range [0,1] has a lower value than p, the algorithm then temporarily removes from the population all those individuals that do not carry allele $x_{g,a}$ (step L10). In effect, the population is "restricted" to only those genotypes that are relevant to the allele choices for the remaining unset genes, i.e. those individuals that contain the allele that was just chosen. These individuals are relevant since they contain information about how other alleles interact with allele $x_{g,a}$; individuals that do not contain allele $x_{g,a}$ have no information on allele $x_{g,a}$'s epistatic interactions. At the start of the creation of each individual, all individuals are included in `rpop`, the restricted population. After each restriction, more individuals are removed from `rpop`. If a restriction would cause `rpop` to have fewer than `minpop` individuals, `rpop` is not restricted.

Once a population is restricted, all the statistics used in steps L03-L10 are calculated using only the restricted population. In the next iteration, step L04 determines the gene x_g that is most significant in the restricted population. Step L05 calculates allele selection probabilities for x_g, based on the average fitnesses of alleles found in the restricted population. Step L07 estimates the epistasis among the unset genes of the restricted population, and, by limiting the scope of these calculations of significance, allele selection probability, and epistasis to only those individuals in the restricted population, the EuA increases their relevancy to determining the best alleles to combine with the alleles already chosen.

A description of an example execution of the EuA will be given below.

4.1.2 Example Execution

In this section, I will give an example of the creation of a single individual from the population shown in figure 4.1.2. The EuA creates a single new individual in steps L03-L10.

Initially, the population contains 5 individuals, and the set of unset genes (u) contains all the genes ($u = \{x_1, x_2, x_3, x_4\}$).

The first step in creating a new individual is to determine the most significant unset gene (step L04). The most significant gene is the gene whose allele values seem to have the greatest effect on individuals' fitness. A simple way to estimate gene significance is to use

Individual		**Gene 1** (x_1)		**Gene 2** (x_2)		**Gene 3** (x_3)		**Gene 4** (x_4)	
x	$f(\mathbf{x})$	x_1='0'	x_1='1'	x_2='0'	x_2='1'	x_3='0'	x_3='1'	x_4='0'	x_4='1'
1100	25	-	25	-	25	25	-	25	-
1110	23	-	23	-	23	-	23	23	-
1101	22	-	22	-	22	22	-	-	22
0100	20	20	-	-	20	20	-	20	-
1111	2	-	2	-	2	-	2	-	2
$f_\mu(x_{i,a})$		20.0	18.0	-	15.0	22.3	12.5	22.7	12.0
$p_\mu(x_{i,a})$		0.53	0.47	0.0	1.0	0.64	0.36	0.65	0.35

Figure 4.2: Complete Population

$\mathbf{x}$	: an individual's genotype
$x_i, i \in 1...l$	: the i th gene of the genotype
$a \in \{'0','1'\}$	: the possible allele values
$x_{i,0}, x_{i,1}$	: the two possible alleles ('0' or '1') for gene i
$f_\mu(x_{i,a})$	: mean fitness of all individuals containing allele a for gene i
$p_\mu(x_{i,a})$	: probability of selecting allele a for gene i

the absolute difference between its alleles' mean fitnesses. This quantity is $|20.0-18.0| = 2.0$ for x_1 (gene 1), $|22.3 - 12.5| = 9.8$ for x_3, and $|22.7 - 12.0| = 10.7$ for x_4. Gene 2 (x_2) has no significance since it had no allele diversity and therefore none of its allele could have any observable effect on the fitness of individuals in the populations. Genes with no allele diversity are always assigned the lowest significance possible. As a result, the most significant gene in this population is x_4, followed by x_3, x_1, and x_2.

Since alleles are assigned to the most significant genes first, an allele is chosen for x_4 in step L05. An allele's selection probability is equal to its mean fitness divided by the sum of allele mean fitnesses for all the alleles associated with this particular gene. For gene x_4, the probability of choosing allele '0' ($x_{4,0}$) is 0.65, since $\frac{f_\mu(x_{4,0})}{f_\mu(x_{4,0})+f_\mu(x_{4,1})} = \frac{22.7}{22.7+12.0} = 0.65$. The corresponding probability of choosing allele $x_{4,1}$ is 0.35. Say that allele $x_{4,0}$ is chosen. In the individual being created, gene x_4 is therefore assigned the allele value of '0'. This occurs in step L05.

Once a gene's allele value is set, the gene is removed from the set of unset genes (u) in step L06. Therefore, x_4 is removed from u, resulting in $u = \{x_1, x_2, x_3\}$.

When epistasis is high, the "correct" choice of alleles depends heavily on the alleles already present in an individual. Therefore the EuA should not always use allele selection statistics gathered over the entire population; it should use statistics gathered over individuals that contain the same alleles as those that have been chosen for the new individual. In steps 07 through 10, the EuA decides whether or not to restrict the population to such individuals. Step 07 determines the probability of restriction by measuring the level of epistasis (E) in the *unset* genes remaining in the current (possibly restricted) population. The epistasis in the unset genes is used, since it reflects the uncertainty in the remaining allele decisions to be made, depending upon the allele choices that have already been made. A simple way to calculate E is to use the maximum difference in allele selection probabilities of the remaining unset genes (D_{max}). When D_{max} is very low, the allele selection probabilities are all about equal, indicating high uncertainty about what allele choices would be best—all allele choices seem to be equally good. When D_{max} is high, there are unset genes whose best allele choice is very evident. High uncertainty among allele choices implies heavy interdependence among alleles—high epistasis. The largest difference in allele selection probabilities in unset genes occurs in x_3; this value is $|0.64 - 0.36| = 0.28 = D_{max}$. Since this difference ranges in the interval $[0.0, 1.0]$, an easy way to calculate E is to use $E = 1.0 - D_{max} = 0.72$. The calculation of D_{max} and E occurs in the call to the subroutine `epistasis-of-population()` in step L07.

A simple way of determining the probability of restriction (p) is to directly use $p = E$, since E ranges in the interval $[0.0, 1.0]$, with $E = 1.0$ indicating the highest epistasis possible and therefore the highest justification for restriction. Therefore the probability of restriction is 0.72. Say restriction is not performed in steps L09 and L10. The algorithm then loops back to step L03. Since x_3 is the most significant unset gene, an allele is chosen for x_3—say allele '0'. x_3 is therefore assigned the value of '0' in the new individual in step L05.

Again, the probability of restriction is determined in steps L07 and L08. This time, $D_{max} = |0.53 - 0.47| = 0.06$ and $E = (1.0 - 0.06) = 0.94 = p$. Say restriction is performed this time in steps L09 and L10. Therefore, individuals that do not contain allele '0' for x_3 are temporarily removed from the population, leaving individuals 1100, 1101, and 0100. These individuals have respective fitnesses 25, 23, and 20. Figure 4.1.2 shows the restricted population.

Note that statistics are not calculated for genes 3 and 4 at this point, as genes 3 and 4 have already been assigned values. Alleles for genes 1 and 2 of the new individual must still be chosen. The most significant gene is now x_1. Previously, the mean fitness of the alleles for gene 1 were 20.0 for $x_{1,0}$ and 18.0 for $x_{1,1}$. The mean fitness of $x_{1,0}$ does not change from 20.0 (since, by coincidence, all the individuals containing allele '0' for gene 1 remain in the population), but the mean fitness of $x_{1,1}$ increases to 23.5, and now exceeds that of allele $x_{1,0}$. By restricting the population to only those individuals that contain $x_{3,0}$, the statistics calculated are now "relevant" to the remaining allele choices. Given that

Individual		**Gene 1** (x_1)		**Gene 2** (x_2)		**Gene 3** (x_3)		**Gene 4** (x_4)	
x	$f(\mathbf{x})$	x_1='0'	x_1='1'	x_2='0'	x_2='1'	x_3='0'	x_3='1'	x_4='0'	x_4='1'
1100	25	-	25	-	25	25	-	25	-
1101	22	-	22	-	22	22	-	-	22
0100	20	20	-	-	20	20	-	20	-
$f_\mu(x_{i,a})$		20.0	23.5	0.0	15.0	n/a	n/a	n/a	n/a
$p_\mu(x_{i,a})$		0.46	0.54	0.0	1.0	n/a	n/a	n/a	n/a

Figure 4.3: Restricted Population, x_3 = '0'

x_3 is assigned the value '0', the conditional fitness of $x_{1,1}$ is greater than its complement, $x_{1,0}$, while globally $x_{1,1}$'s fitness is lower than that of $x_{1,0}$. Had the population not been restricted, this change in relative fitnesses would not have been noticed. By restricting the population to only relevant individuals, the EuA can track epistatic dependencies of various alleles.

Through restriction, higher-order schemata statistics are being collected. The global fitness of allele $x_{0,1}$ is identical to the average observed fitness of schema '1###'. This average of this schema is 18.0, which is lower than that of its competitor '0###'. However, when higher-order schemata are examined, we find that the average observed fitness of schema '1#0#' is 23.5, higher than that of '0#0#'. Therefore the EuA's restriction operator allows the algorithm to calculate schema fitness averages "on demand", when these averages are necessary to making correct allele decisions.

The algorithm still must choose alleles for genes 1 and 2. Say $x_{1,1}$ is chosen for x_1. Since there is no allele diversity in the remaining unset gene, x_2, there is no point to restricting the population, since restriction will not allow us to compare the conditional fitnesses of the alleles for x_2. The restricted population only contains information for allele '1'. At this point, the `select-allele` subroutine can either choose allele '1' with the hope that it is better than the '0' allele, since the '0' allele is not present, or the algorithm can experiment a little and choose the '0' allele. The EuA uses a parameter called "creation rate", which is the probability that an allele that is not present in the restricted population will be chosen ("created"). Without a "creation rate", x_2 would always be set to $x_{2,1}$. With a creation rate, there is a chance that $x_{2,0}$ would be chosen. Say that $x_{2,0}$ is chosen, completing the construction of the new individual. The new individual has genotype '1000', and therefore is a completely new individual.

Once a new individual is created, the population is unrestricted—all individuals are added back into the population. Then the EuA replaces the lowest fitness individual in the population with the new individual. In this example, individual '1111' has the lowest fitness, and is therefore replaced by the individual '1000' (steps L11 and L12).

This example illustrated the behavior of a "standard" EuA. Many options of the algorithm can be changed in order to increase performance. For example, the policy of always replacing the lowest-fitness individual in the current population (pop_w) with the new individual may be slightly modified so that the new individual replaces pop_w only when the new individual has equal or greater fitness than pop_w. In the sections to follow, the motivation and mechanism behind each step of this algorithm are explained. Allele selection will be discussed first, then population restriction, and then finally replacement policy.

4.2 Allele Selection

In this section, I will discuss how the EuA performs step L05, `select-allele`. In the EuA, the selection of single alleles to use in the construction of new individuals is based upon the statistical analysis of the correlations between alleles and fitness[1]. New individuals are constructed one gene at a time, by choosing a single allele for each gene from several possible allelic forms. As with BSC, there are many choices for the selection technique used to choose new alleles, and certainly the performance of each technique is heavily, if not completely, problem dependent. Some examples of allele selection techniques are fitness-proportionate selection (equation 4.2) and rank selection. In fitness-proportionate selection, the probability of selecting an allele varies directly with the proportion of that allele's fitness to the sum of all of the allele fitnesses of that gene. In rank selection, an allele's selection probability for a given gene varies with its rank among the other alleles corresponding to that gene—the highest ranking allele has the greatest probability of being selected, followed by the second rank allele, etc.. Since most GA research has dealt with fitness-proportionate selection, this thesis will be limited to EuAs using fitness-proportionate selection.

$$\begin{aligned} f_\mu(x_{i,a}) &= \text{Mean fitness of all individuals having allele value } a \text{ for gene } i \\ &= \text{The ``fitness'' of allele } x_{i,a} \\ p_\mu(x_{i,a}) &= \text{Probability of selecting allele value } a \text{ for gene } i \\ &= \frac{f_\mu(x_{i,a})}{\sum_{b\in\{'0','1'\}} f_\mu(x_{i,b})} \end{aligned}$$

[1]Although developed independently, this strategy is basically identical to that of BSC.

4.2.1 Defining Allele Fitness: Average Fitness versus Maximum Fitness

Fitness-proportionate selection methods can rely on different definitions of allele fitness. Two ways of defining allele fitness will be discussed in this section: mean allele fitness (μ-fitness) and maximum allele fitness (X_n-fitness). In μ-fitness, an allele's fitness is defined as the *average* fitness of individuals that contain that allele. When using fitness-proportionate selection and μ-fitness, the EuA's allele selection mechanism is identical to the technique described previously in relation to BSC. In X_n-fitness, the *maximum* fitness of all the individuals containing a particular allele defines that allele's fitness. Figure 4.4 illustrates how the allele selection probabilities of μ-fitness and X_n-fitness differ for an example 4-bit genotype. Note that, for this example, the X_n-fitness selection probabilities for genes 3 and 4 are much closer to 0.5 than those of μ-fitness, and therefore X_n-fitness would select alleles for these genes more randomly than μ-fitness. A different example could easily be generated in which these effects were reversed.

One of the main reasons μ-fitness has been so widely used in the past is because it has been shown to be the best strategy for maximizing the cumulative "payoff" of repeated trails (Holland, 1975)—μ-fitness is the best choice, on average, to select "good" alleles. Since the goal in this thesis is to design an algorithm to optimize a fitness function, instead of finding "good" solutions on average, X_n-fitness may be a better choice than μ-fitness. X_n-fitness focuses the search only on the best performance of each allele, whereas μ-fitness incorporates information about all aspects of a single allele's performance, including its worst performance, its typical performance, and its best performance. X_n-fitness may lead to more "risky" choices, as these choices are not tempered by all the information available. In addition, X_n-fitness may be more likely to cause the EuA to prematurely converge than μ-fitness, because the alleles belonging to a single super-individual could completely dominate every allele selection choice, and therefore every new individual created would be identical to the super-individual. More theoretical and experiment exploration of "what allele fitness definition is best" is necessary, but for the meantime, the standard EuA will use μ-fitness.

After a gene in the new individual is fixed to a specific allele value, the next most significant gene is examined and each of its allele's probabilities of selection are calculated. Since the domain of the current implementation of the EuA is limited to binary genotypes, it is only required to calculate the selection probability p for one of the alleles from the most significant gene, as the probability of selecting the complementary allele would simply be $(1 - p)$.

$\mathbf{x}$:	an individual's genotype
$f(\mathbf{x})$:	the fitness of individual with genotype x
$x_i, i \in 1...l$:	the ith gene of the genotype
$a \in \{'0','1'\}$:	the possible allele values
$f_\mu(x_{i,a})$:	mean fitness of all individuals containing allele a for gene i
$p_\mu(x_{i,a})$:	using μ-selection, probability of selecting allele a for gene i
$f_{X_n}(x_{i,a})$:	maximum fitness of all individuals containing allele a for gene i
$p_{X_n}(x_{i,a})$:	using X_n-selection, probability of selecting allele a for gene i

$$p_\mu(x_{i,a}) = f_\mu(x_{i,a}) \sum_{b \in \{'0','1'\}} f_\mu(x_{i,b})$$
$$p_{X_n}(x_{i,a}) = f_{X_n}(x_{i,a}) \sum_{b \in \{'0','1'\}} f_{X_n}(x_{i,b})$$

Individual		**Gene 1** (x_1)		**Gene 2** (x_2)		**Gene 3** (x_3)		**Gene 4** (x_4)	
x	$f(\mathbf{x})$	x_1='0'	x_1='1'	x_2='0'	x_2='1'	x_3='0'	x_3='1'	x_4='0'	x_4='1'
1100	25	-	25	-	25	25	-	25	-
1110	23	-	23	-	23	-	23	23	-
1101	22	-	22	-	22	22	-	-	22
0100	20	20	-	-	20	20	-	20	-
1111	2	-	2	-	2	-	2	-	2
$f_\mu(x_{i,a})$		20.0	18.0	0.0	15.0	22.3	12.5	22.7	12.0
$p_\mu(x_{i,a})$		0.53	0.47	0.0	1.0	0.64	0.36	0.65	0.35
$f_{X_n}(x_{i,a})$		20.0	25.0	0.0	25.0	25.0	23.0	25.0	22.0
$p_{X_n}(x_{i,a})$		0.44	0.56	0.0	1.0	0.52	0.48	0.53	0.47

Figure 4.4: How the allele selection probabilities of μ-selection and X_n-selection differ for an example 4-bit genotype.

4.2.2 Adaptive Allele Selection

To balance exploration and exploitation, it would be desirable for an evolutionary search algorithm to perform a "smart mutation" when a clear winner among a gene's alleles is apparent, but random mutation/search when the allele fitnesses are roughly equal. When one of a gene's allele has a very high fitness, and the alternate alleles have very low fitness, the high-fitness allele should be exploited. When the fitnesses of the gene's alleles are roughly equal, the choice between them is (for the current population) probably not that important, and so the choice should be more random. The EuA accomplishes these things automatically, when using fitness-proportionate selection. With fitness-proportionate selection, the amount of allele exploitation varies proportionately with the difference between a gene's alleles' fitnesses. Alleles with equal fitnesses have equal probabilities of being selected; for binary alleles, each has a 50% selection probability. In such a case, each allele is equally exploited and therefore the algorithm will equally explore the alternatives. However, an allele with x times the fitness of its alternates has x times the selection probability, and therefore experiences x times more exploitation. The EuA explores when uncertainty among allele choices is high, and exploits when clearly superior alleles exist. In this way, adaptive allele selection is achieved, as the EuA adapts its selection probabilities to the differing situations. This is unlike most mutation-based algorithms, which always randomly choose a gene to mutate, no matter what information has been gathered in the past.

Note that, by relying on an explicit calculation of allele fitness, the EuA differs significantly from the GA. The GA implicitly records allele fitness by relying on an allele's frequency of occurrence in its population. For example, if all the individuals in a GA population had the same fitness, and if allele A occurred in 99 out of 100 individuals, while allele B occurred in only a single individual, allele A would be *99 times more likely* to be used than allele B, even though the average fitness and maximum fitness of both alleles were equal! As a result, by relying on explicit calculations of allele fitness, the EuA more reasonably allocates trails to and pursues promising alleles.

4.2.3 Modifications to Fitness-Proportionate Selection

As a result of fitness-proportionate selection, alleles with relatively low fitness will infrequently be chosen for inclusion in a new individual. Alleles that repeatedly cause low fitnesses in individuals containing them will eventually be eliminated from the population. However, it is not always desirable to have a particular allele completely eliminated from consideration. Consider two alleles, A and B. Allele A consistently has higher fitness than allele B, which consistently has very low fitness. It may be the case that allele B is needed to construct an optimal individual, even though it is detrimental to suboptimal individuals. In this case, allele A would be "deceptive", as it would lead the algorithm away from the allele that was found in the optimum. In order to avoid convergence to deceptive alleles, the EuA must occasionally attempt to use alleles that have been eliminated from the population. The EuA uses a parameter called "creation rate" in order to reintroduce such alleles. This parameter (C) specifies the probability that an allele lost from the population will be reintroduced by including it in a new individual being created. In addition, creation rate allows alleles absent from *restricted* populations to be included in a new individual. If, during the course of creating a new individual, the population is restricted to such an extent that extremely little allele diversity exists—all the individuals in the restricted population are extremely similar—the creation rate parameter allows the EuA to occasionally try out an allele that is not found in these individuals. By doing so, the EuA is afforded a small measure of protection against deceptive and/or premature convergence.

In cases where the chosen selection mechanism is too exploitative, another parameter, "selection noise" (N), can be used by the EuA to encourage exploration. Selection noise is the probability that the allele choice made by the allele selection operator is overridden, and a different allele is chosen. This parameter just adds noise to the allele choices made by the particular allele selection mechanism in place. It can be increased to encourage exploration. We shall see that this is rarely necessary, and otherwise almost always detrimental, when using fitness-proportionate selection.

Even with adaptive allele selection probabilities and a correctly set creation rate, the EuA still faces a possibly large obstacle to performance when dealing with large genotypes. For many problems, even one mutation can cause large fluctuations in fitness. Therefore the *number* of low-fitness alleles present in an individual may be much more important than the *percentage* of low-fitness alleles. As the number of genes of a genotype increases, the number of low-fitness alleles that are included in new individuals increases (assuming that selection probabilities remain constant with respect to genotype size). For example, if, for each gene, there was a 10% chance of choosing a low-fitness allele instead of a high-fitness allele, the number of low-fitness alleles appearing in a new 100-gene individual would be ten times on average the number that would appear in a 10-gene individual, even though the percentage of low-fitness allele would be equal between the two genotypes. Although it may seem reasonable for exploration of low-fitness alleles to remain proportionally constant with respect to problem size, it could easily be the case that this effect is detrimental to the EuA's performance on a large number of problems. Therefore, as genotypes increase in size, the balance of exploration and exploitation may have to be tipped in favor of exploitation. In the standard EuA, no effort is made to counter the problems that may be associated with larger genotypes. However, if desired, allele selection probabilities could easily be modified (using fitness scaling, perhaps) from problem to problem, so as to increase exploration versus exploitation of higher fitness alleles. Selection probabilities could be based upon one-tailed statistical tests of significance, or could be adjusted based upon the observed variance in a particular allele's fitness (H-L. Fang, 1993). Such modifications will not be attempted in this thesis, as no experiments have been conducted to determine their necessity.

4.2.4 Clustering and Allele Significance

The EuA needs to determine gene significance for two reasons. First, the EuA sets genes in order of their significance, so that more important allele decisions are made before less important choices. Second, the EuA uses gene significance as a measure of epistasis. It then uses this measure of epistasis to determine when to restrict the population (subsection 4.3.2. Several different measures of gene significance are available to the EuA, and the choice of this measure will heavily affect the EuA's behavior.

One way to quantify gene significance would be to observe how well a gene's alleles "cluster" individuals into separate groups. An individual containing a particular allele would belong to that allele's group. As the fitness variance of each group decreased and their fitness means became increasingly different, the clustering would improve. As the means diverged, one group would become the "high-fitness group" and the other the "low-fitness group" for that gene. When the difference between the means was high, then allele settings for that gene would obviously have a large effect on fitness, and therefore the gene would be very significant. The standard EuA defines gene significance as the distance between a gene's alleles' means. Future versions could easily incorporate the variance of each allele group into the definition of gene significance.

4.3 Population Restriction

Smart recombination requires not only the identification and comparison of the best genotypic parts, but also the determination of how to best combine these parts to form very high-fitness individuals. Independently setting each allele value is not good enough, as this strategy ignores the epistasis so frequently encountered in nontrivial problems. Such a strategy is equivalent to randomly recombining o(1) schemata, without respect for the high levels of dependence that may exist among the best settings of alleles—such a strategy ignores higher-order interactions. Simple allele fitness averages do not provide enough information to make the right decisions. What is needed is an analysis of *conditional* allele fitness distributions. In one of these distributions, the fitness distribution of a single allele is analyzed, given that other alleles have fixed values. Since the number of possible conditional allele fitness distributions is *more* than even the number of possible schemata, not all conditional distributions can be examined. Therefore only the most important distributions should be examined. The EuA uses the *population restriction* operator to accomplish this goal and to separate global allele fitness distributions into conditional distributions.

4.3.1 The Restriction Operator

The EuA intelligently selects sets of alleles from individuals in the population, but, unlike the GA-crossover operator, these genotypic parts can come from the entire population, instead of from just two parents. Due to epistasis, it is not reasonable to simply mate the entire population, as BSC does, because doing so would ignore allele dependencies found in each individual. Therefore, any population-wide recombination operator must determine which genotypic parts work well together, and selectively choose individuals that contain the most information about these parts.

As the EuA builds a new individual, the choice of what genotypic parts will be most appropriate depends on both gene significance and the already constructed parts of the new individual. The eugenic algorithm builds new individuals from scratch, by proceeding through the genes, from the most significant to the least significant, and probabilistically selecting an allele value for each. At some point, gene significance decreases to such a point that the best choice between two complementary alleles becomes highly uncertain, because the alleles have very similar fitnesses. The EuA can reduce the uncertainty in the choice of the next allele by focusing on those individuals most relevant to this choice. It accomplishes this by restricting the population to those individuals that contain information about the relationship between the unset genes and the alleles already fixed in the new individual. These individuals are those that contain the same alleles as those fixed in the new individual. They contain information on how to best fine tune the remaining alleles, since they belong to the same region of the search space as the new individual. The *restrict-schema*, composed of the allele fixed in the new individual, is used to identify these individuals. Individuals that contain the restrict-schema make up the restricted population.

Once the most relevant individuals are identified, allele fitness statistics are recomputed using only the individuals from this restricted subset. These statistics are no longer global measures, they are *conditional fitness statistics*. They contain information on the performance of the unset alleles in the region of space the restrict-schema defines; they reflect the conditional allele fitness distributions of the "don't care" genes in the restrict-schema.

The use of all of the restrict-schema's fixed positions to limit the population poses a major problem, however; the restricted population can quickly become too small. For diverse populations, in which alleles '0' and '1' occur with about the same frequency for each gene, each application of the restriction operator will reduce the population size by approximately half, since only about half the individuals will have the allele restricted on. And so a restrict schema of order-n will limit the population to only $\lambda/2^n$ individuals. For even a single individual in the population to match the restrict-schema, either λ must be very large, or the restrict-schema must not contain too many fixed positions. Since increasing population size increases the computational demands of the EuA, the latter choice is preferable. One way to limit the number of fixed positions in the restrict-schema is to include only those alleles that were set immediately preceding a restriction, instead of all the alleles fixed in the new individual. Genes are fixed in order of significance, from most significant to least significant. Therefore the last gene to be fixed before a restriction occurs is the least significant gene of the restrict-schema. The lower the significance of a gene, the more likely good values for it depend on other alleles. Consequently, it is more likely that less significant genes take part in heavily epistatic allele interactions. More significant genes are more independent from other genes; they need less help from other allele settings. As a result, if all the genes of the restrict-schema cannot be used to limit the population, the least significant genes should be the ones used, since the values of the remaining unset genes should be specifically tuned for the alleles that need the most help. This is the strategy the current implementation of the EuA employs.

4.3.2 The Probability of Restriction

The restriction operator cannot be applied after every gene that is fixed, since each restriction reduces the population size by roughly half. In addition, while highly significant genes remain, restriction is not necessary. Restriction is necessary when gene significance is low. When gene significance is low, epistasis is high. In order to determine the necessity of restriction, the EuA needs to estimate the level of epistasis among the unset genes in the current restricted population. When epistasis is high, it becomes very important to calculate conditional fitness distributions, since high epistasis implies that global distributions are not that useful. In order to calculate these conditional distributions in situations of high epistasis, the population must be restricted.

Various methods are available to measure epistasis and gene significance. One measure that could be used to determine restriction necessity is χ^2-epistasis (see section 3.5). Alternatively, if allele fitnesses are believed to be drawn from approximately normal distributions, the t-score could be used to test the significance of the differences in mean allele fitnesses. However, the epistasis most relevant to determining restriction necessity is that which prevents a clear choice from being made among competing alternatives. Therefore probably the most appropriate measure of epistasis would be the difference in allele selection probabilities (D), since the selection probabilities directly indicate how much uncertainty is present in the remaining allele choices. For example, when D is very high, it is obvious that there are very clear winners among the remaining unused alleles. This implies that there is a clear dominance of particular genotypic patterns in the remaining unfixed regions of the search space; these patterns have relatively high conditional fitnesses. If, however, the difference in selection probabilities was very low, then the information contained in the population is too contradictory to be useful to the selection method being employed. If it has already been decided that this selection mechanism will be useful for successfully solving the problem, then when this mechanism falters, some action (like restricting the population) must be taken to reduce the interference among competing possibilities.

In the experiments conducted in this thesis, the EuA used this difference in selection probability of a gene's alleles in its measure of epistasis, with epistasis (E) equal to $(1 - D_{max})$, where D_{max} was the largest difference in allele selection probabilities for any gene. Since E varied in the interval [0, 1], with $E = 0$ implying no epistasis and therefore no need to restrict, and $E = 1$ implying extremely high epistasis and a great need to restrict, E could possibly be used directly as the probability of restriction. However, it was observed that this measure of E was often not very close to zero, and so restriction would occur very often. As previously discussed, too much restriction can quickly lead to populations that are too small. Therefore, modifications were made to decrease the probability of restriction. First, a basic probability of restriction was calculated—this value was just E. Then, this value was squared, therefore mapping $E = 0$ to $p_{restrict} = 0$, $E = 0.5$ to $p_{restrict} = 0.25$, and $E = 1$ to $p_{restrict} = 1$. In addition, the probability of restriction was explicitly reduced by half after each restriction to prevent the overuse of restriction on the most significant bits. Therefore, as the population was reduced in size by restriction, p was also reduced.

4.3.3 Consequences of Population Restriction

Incremental Schema Evaluation

What is the order of the schemata explicitly analyzed by the EuA? After each restriction, higher and higher order schemata are analyzed by the EuA. Initially, before any restrictions have occurred, only o(1) schemata are compared. After the first restriction, at least one allele becomes fixed in the entire population. Each allele's fitness distribution is now measured conditionally on the value of the fixed allele. Each schemata analyzed contains the fixed allele. Therefore, after one restriction, schemata of at least order-2 are analyzed. In general, after r restrictions, the number of genes for which only a single allele is present in the population is *at least* r, because each time the population is restricted, there can be a corresponding loss of diversity in alleles *other than* the one restricted on. For example, after 6 restrictions, it could easily be possible that 10 or 15 genes exhibit no allele diversity. Therefore the schema being analyzed would have at least 10 to 15 fixed positions.

There is, however, an upper limit on the number of restrictions that occur when generating a new individual. Since, on average, each restriction reduces the population size (λ) by half, only $\lambda/2^r$ individuals will be in the population after r restrictions. This means that, for a population size of 2^r, it is highly likely that at least order-r conditional allele fitnesses will be explicitly analyzed and compared by the EuA, and very likely that conditional fitnesses of order even more than r will be analyzed as the search progresses and focuses on smaller regions of the search space.

Maximization of *Relevant* Information

Although the *relevancy* of the information contained in the population is increased by restriction, the overall *amount* of the information is decreased, since the number individuals over which information is collected decreases with each restriction. How can the tradeoff between information amount and relevancy be managed?

The EuA maintains the amount of relevant information at the highest possible level by only restricting the population when necessary. If it is clear which alleles are associated with higher fitness, then restriction is not necessary. In these cases, it is most likely that allele *combinations* are not as important as the presence or absence of single alleles. The entire population would provide better, more statistically significant information about these alleles than a small subpopulation. However, in cases of higher epistasis, the conditional allele fitness distributions become increasingly different from the marginal (global) distributions. Larger samples of individuals are not necessarily better samples in these situations. The most relevant information is found in subpopulations, which can be used to more accurately determine conditional distributions than a large diverse population could. Larger populations would contain less relevant information. Since the probability of restriction is based upon the perceived level of epistasis, the EuA adaptively maximizes the amount of relevant information in the population.

Diversity, Schemata Competition, and Subpopulations

Because of the restriction operator, the EuA can and will maintain extremely high levels of diversity in the population, without bogging down the pace of the search with many ineffective schemata recombinations. The restriction operator ensures successful matings among individuals in the population. When epistasis is low, alleles can be drawn from the entire population without ill effect. However, when epistasis is high, if the genotypes of a group of mating individuals are too different, their offspring will most likely have low fitness. The restriction operator attempts to guarantee the compatibility of prospective mates. When high epistasis is detected, the EuA restricts the population to a group of similar individuals, and therefore reduces the probability of incorrect schemata recombination and low-fitness offspring. Because of the randomness of the application of restriction, the EuA causes many different subpopulations to intermate. Unlike many GA-related methods, reproductive compatibility and the subgrouping of individuals is not limited to a single, simple overall measure such as Hamming distance. The randomness of the application of the restriction operator causes many different population subgroups to be matched for reproduction, as compatibility is determined by different genes as each new individual is constructed. As a result, not only does the restriction operator *allow* large amounts of diversity to exist in a single population, it *causes* diversity to be maintained. It is a "multicultural dating service".

Diversity is not always desirable. If it is fairly easy to generate good individuals with very diverse genotypes, then subpopulations of individuals can and will easily coexist in a single large population, and diversity is helpful to generate more representative statistics. In this case, the EuA can simulate the action of a "parallel GA". A parallel GA is a set of GAs that occasionally share individuals between their populations and, by doing so, can collectively maintain search focus in many more regions of the space than a single-population GA. However, there are situations in which homogeneity is preferable to diversity. In search spaces where the number of good individuals is extremely low, it will be difficult to generate even a single good individual and therefore practically impossible to generate a diverse set of good individuals. If the fitness of the single good individual exceeds the population's average fitness by a great enough amount, the EuA will experience a large drop in diversity of bits with very high significance with the occurrence of this first good individual, as the algorithm will have no choice but to exploit the only information it has that can lead to higher fitness. This decrease in diversity is not necessarily detrimental, as the drop in diversity of some genes allows the algorithm to much more thoroughly explore smaller, more promising regions of the search space. This is exactly what is needed when high-fitness individuals are very difficult to locate.

The EuA's smart recombination allows it to maintain much higher levels of diversity in a single population than a GA could maintain. In GAs, schema fitness is implicitly recorded by how often it is found in the population—fitness proportionate representation. Fitness is not explicitly defined using averages, maximum values, or any statistics whatsoever. It is therefore difficult for two or more competing schemata to exist in the population for long, because of the positive feedback between representation and selection. As soon as one schemata achieves higher representation than its competitors, it becomes more likely to be included in reproduction than its competitors, and its representation increases even more. Schemata whose representation fall below that of the others become less likely to be propagated, and their representation falls even more until they vanish from the population. In contrast, competition among schemata in the EuA is not won by higher amounts of representation, but by fitness itself. As a result, a high-fitness schema can exist in just one or two individuals in a large population, but still have a high probability of being used in the construction of new individuals. For example, consider a situation in which 99 individuals of a population contained schema A, and one individual contained a competing schema B. If the mean fitness of the schema A individuals was the same as the fitness of the schema B individual, the probability of the occurrence of each schema in new individuals would be equal, even though only one individual contained schema B. In a fitness-proportionate GA, schema A would be 99 times more likely to occur in offspring than schema B, since the sum of fitnesses of all schema A individuals would be 99 times that of the schema B individual. As a result of the EuA's managed competition of schemata, and the restriction operator's compatibility checking, many diverse groups of individuals can and will be maintained in the population during the search. The experiments to follow will demonstrate this empirically, illustrating that very genotypical different individuals coexist peacefully, yet competitively, in EuA populations.

It should be pointed out that the restriction operator performs exactly the opposite of what has been recommended by previous researchers—"incest prevention" (Eshelman and Schaffer, 1991). Since the problem of premature convergence is very significant in the performance of GAs, some method such as incest prevention must be used to maintain high levels of diversity, without introducing random mutation. When incest prevention is used to prevent diversity loss, individuals mate only when they are significantly *different*—when the Hamming distance between them is large. It has been demonstrated that this is a very good heuristic for the GA, since its highly effective prevention of premature convergence overshadows the increase in incompatible matings. It allows the GA's search to increase exploration without the use of higher mutation rates. Since the amount of exploration the EuA performs is adaptive, and diversity is promoted only when necessary, this artificial method of preventing diversity loss is not needed. Furthermore, it would most likely be detrimental to the performance of the EuA, because it would increase the chance of incompatible schemata being paired together.

4.4 Steady-State Replacement

Instead of generating an entirely new population from the existing population ("Generational Replacement"), the EuA uses "Steady-State Replacement" (Eshelman, 1991; Syswerda, 1991), in which only one individual from the existing population is replaced at a time. Since feedback can be gained from the evaluation of only one new individual, Steady State Replacement is a superior strategy to Generational Replacement because new information can be immediately exploited for use in new individuals. In addition, the action of the EuA is very genotypically "disruptive", as it frequently generates new individuals that have very high Hamming distances from those in the current population. Steady State Replacement guarantees that good schemata found in the current population will be preserved as the search proceeds, while at the same time allows the EuA search to be very explorative. If Generational Replacement were used, then valuable information would be much more likely to be lost from generation to generation, and therefore the search would have to be much more cautious and conservative.

The choice of which individual to replace is yet another heuristic to determine for the EuA. The current implementation of the EuA always replaces the lowest-fitness individual in the population with the newly created individual. "Elitism" can be employed to prevent the possibility of a lower-fitness new individual from replacing a higher-fitness individual already in the population. Yet another variation of the replacement strategy is to probabilistically replace individuals based on their fitness. However, such a strategy would increase the chance that high-fitness schemata would be lost, and, given the high amount of exploration performed by the EuA (as will be demonstrated in the experiments), the current strategy of always replacing the worst is likely a better heuristic for most stationary problems.

4.5 Summary of the Eugenic Algorithm

The EuA is composed of two main parts: individual creation and individual replacement. Individual creation is composed of smart allele selection and population restriction. In the standard EuA, fitness-proportionate selection is used to choose alleles, where an allele's fitness is defined as the mean fitness of all the individuals that contain that allele. Mean fitness selection is a superior strategy to representation-proportionate fitness selection, since only a single instance of a high-fitness allele is necessary to guarantee its exploitation. In addition, it allows much greater levels of schemata diversity to exist. Population restriction is a major innovation of the EuA that is not present in any previous algorithm. Population restriction allows the EuA to purposefully (not randomly) juxtapose compatible schemata and also maintain many diverse subpopulations inside of a single large population. These subpopulations allow the EuA to focus its search in many differing regions. By using population restriction, the EuA can calculate only those conditional allele fitness distributions that are necessary. These calculations lead to the computation of increasingly higher order schema fitnesses, and only the most important of these schemata are analyzed. Population restriction allows epistasis to be intelligently tracked, analyzed and exploited. In effect, population restriction allows the EuA to perform smart recombination. The EuA uses Steady-State Replacement to prevent the loss of valuable genetic information—information on high-fitness alleles and schemata. In Steady-State Replacement, a newly created individual replaces the lowest-fitness individual in the EuA population.

In the chapter to follow, EuA performance will be compared against that of several other optimization algorithms on six different combinatorial optimization problems. We shall see that the EuA performs extremely well in relation to the other algorithms.

Chapter 5

Experiments

In this chapter, six combinatorial optimization problems will be attacked by several evolutionary algorithms, including the EuA. Five of these problems are drawn from the genesys-2.0 test suite[1], and the sixth has been studied in simulated annealing literature comparing the performance of simulated annealing to that of genetic algorithms (Ingber, 1993). These problems are considered to be very hard, as even customized methods (such as direction set, branch and bound, etc.) have great trouble finding the global optima. At least four are NP-complete. Trivial toy problems, such as BIT-COUNT (described in section 3.3), will not be studied, as they would provide little feedback on how these algorithms would perform in real-world applications. A detailed description of each of the problems will be provided in section 5.3.

This thesis focuses on combinatorial optimization, and so all of the independent variables of the problems were encoded using binary genotypes, even though some of these problems might be more easily solved using a straightforward real-number representation (F02 and F38 in particular). It is extremely important to realize that the entire topological structure of the problem search space changes with such a transformation from m-dimensional real space to n-dimensional binary space. For example, there are a great many points that are mutation-based neighbors in the real space, but are separated by great (Hamming) distance in the binary space. This phenomenon is known as "Hamming cliffs". For instance, consider 4-bit binary genotypes that encoded equidistant real numbers in the range $[0, 15]$. The genotypes '0111' and '1000' would encode the real genotypes 7.0 and 8.0, respectively. While their distance in binary space is the greatest possible for a 4-bit

[1]The genesys-2.0 software was provided in pre-release form by Joerg Heitkoetter. In the genesys-2.0 test suite, there are thirty-seven problems, designated F01 through F37. To aid researchers in the future, I will use genesys-2.0's nomenclature. As a result of this nomenclature, the one test problem studied in this thesis that was not included in the genesys-2.0 package shall be designated "F38".

binary genotype, their genotypic distance in real space is fairly small. Therefore genotypes that are neighbors in real space do not necessarily maintain this relationship when they are encoded into another representation. As a result, the difficulty of an optimization problem can be greatly increased or decreased when the representation of the inputs change. In particular, local optima present in real space do not necessarily retain their same neighborhoods of attraction in binary space, and therefore they may no longer be locally optimal, since they are no longer superior to their immediate neighbors. Even a transformation from one binary representation to another that maintains local neighborhoods can cause local optima to appear or disappear, since the bit patterns that matched specific individuals in the original space may no longer match the individuals in the new space. In problem F02, we shall see that the change from real to binary space greatly increases the number of local optima (and therefore problem difficulty). In contrast, problem F38, a problem with a great many strong local optima in real space, seems to lose this characteristic upon transformation of its inputs into a binary encoding. Almost all the algorithms tested were never trapped by F38's real-space local optima. Furthermore, to make a fair comparison, all the algorithms searched using the same genotype-phenotype mapping, since changing this mapping could easily alter a problem's epistatic and local optima properties. By comparing each algorithm's performance on a number of differing problems, it is hoped that the special bias of each algorithm can be identified and eliminated from the comparison. In order to make this comparison, the algorithms must be tested on the same problems using the same representation.

Since the ideas of epistasis and problem difficulty are so closely linked, one algorithm (IMHC) was specifically designed to be the "silver bullet" for zero epistasis problems (specifically 0.0 o(1) χ^2-epistasis). The relative speed at which IMHC solves a problem, compared to the other algorithms used, will be considered a measure of problem triviality (and therefore problem insignificance). As we shall see, IMHC easily optimized one of the problems (F38), and very quickly solved a second problem (F32), relative to the more complicated algorithms used. Hence, we may assume that F32 and F38 have little epistasis, and are therefore "easier" problems than the other four. We shall see that the main hurdle to IMHC's performance is *not* high epistasis (i.e. a great many local optima), but deception.

Finding the best solutions for these six problems is not the goal of this study. The goal is to discover the problem-independent characteristics of the best performing algorithms, and subsequently determine whether the theorized "important" characteristics of the EuA—smart mutation and recombination—are useful when attacking hard discrete optimization problems, and, in addition, whether or not these characteristics are efficiently performed by the EuA. It will be shown that this is indeed the case. The next section will describe what is meant by the phrase "best performing algorithm", and will be followed by descriptions of the algorithms tested and their performance on the test problems.

5.1 Quantifying Performance

Optimization algorithm performance is usually measured by two quantities: the quality of solutions found and the speed at which high quality solutions are found. A standard method of representing an optimization algorithm's performance on a particular problem has been to construct "learning curves", which plot the change in quality of the best solutions found versus algorithm resource expenditure. Solution quality is usually measured with an average of the fitnesses of the best individuals discovered over a number of different runs, and resource expenditure is usually measured using raw computation time or the number of function evaluations conducted. While computation time and function evaluations are both very practical and valid measures of resource usage, the use of mean fitness values as a measure of quality must be used with caution. For instance, it may be the case that the mean fitness of the best individuals found by one algorithm was twice the mean fitness of the best individuals found by another algorithm. In spite of this, it must not be assumed that, the first algorithm performed twice as well, or even much better than the second algorithm. For example, consider a problem in which the optimal individual had fitness of 100.0, the second best individual had a fitness of 95.0, and all the other individuals had fitness ranging from 0.0 to 5.0. Assume that algorithm A_1 can locate the optimum in half of the runs, but in the other runs A_1 can only find points of fitness 5.0 or less. Algorithm A_2 can locate the *second* best point 100% of the time, but can never locate the optimum. If we evaluate algorithms on the basis of the mean fitness of the best individuals they found, we would rate algorithm A_2 to perform almost twice as well as algorithm A_1, even though A_2 could never find the optimum. Thus the use of mean fitness to judge algorithm performance can be misleading, since it does not clearly demonstrate an algorithm's probability of finding optimal or almost-optimal points. This problem could be made even worse if the range of all fitness values was small, relative to their absolute values. For example, if we added 10^6 to the fitness of all the individuals in the search space, then the two algorithms would seem to perform identically, since the difference in their mean best fitness values would be almost nonexistent and easily attributed to sampling error.

A better way to quantify an algorithm's performance on a particular problem might be to compute statistics using the overall *ranks* of the best points found, instead of the absolute fitnesses of these points. By using rank instead of raw fitness, relative search capabilities of the algorithms could be compared without being affected by order-preserving distortions of the fitness space. However, determining the rank of each individual encountered is practically infeasible, as it would require enumeration of almost the entire search space. In addition, reducing performance to a simple statistic (such as an average) will almost inevitably lead to a loss of important performance-related information. The cumulative distribution function of the ranks of points found by a given algorithm after a specific number of function evaluations is probably one of the best practical representations of an algorithm's performance, as it simultaneously demonstrates both the quality and frequency of discovery of solutions.

In spite of the problems with mean fitness values, the standard practice of plotting mean fitness versus time (function evaluations) will be used in this thesis to represent the behavior of the algorithms, for the sake of simplicity and ease of understanding. In future studies, it might be practical to determine the ranks of created individuals. At this time we will be limited to mean fitness metrics and, in a few cases, a limited study of points close to the optima. Algorithms will be judged by how many function evaluations they require to solve the given problems, and not by their raw computational usage, as it is assumed that, in "real-world" optimization problems, the cost of a function evaluation is much greater than the computational cost of a particular search algorithm.

The next section will include a detailed description for each algorithm, as well as the algorithm parameters used in the experiments. Then, each experiment and its results will be discussed in depth.

5.2 The Algorithms

Three mutation-based and two pattern-based algorithms are evaluated in this section. The mutation-based algorithms are "Increasing Mutation Hill Climbing" (IMHC), "Decreasing Mutation Hill Climbing" (DMHC), and simulated annealing (SA) (Kirkpatrick and Sherrington, 1988). Both IMHC and DMHC were derived from "Random Mutation Hill Climbing" (RMHC) (Forrest and Mitchell, 1993) especially for this thesis. The mutation based-algorithms can be broken down into two classes—heating algorithms (IMHC) and cooling algorithms (DMHC and SA). Heating algorithms start with a random individual and slowly increase the "heat" (the size of mutations) using an arbitrary schedule. Heating algorithms' main emphasis is exploitation. Therefore they can quickly find excellent solutions in some instances, but are prone to become trapped in local optima in others (at least until heat increases to a point where the search can escape the local optimum's basin of attraction). Cooling algorithms start with high mutation rates and decrease these rates over time. Cooling algorithms are initially very explorative, with the hope of initially avoiding the basins of attraction of local optima. It is assumed that the basins of attraction of true optima will be stronger than those of local optima. However, high initial exploration may lead to very little progress, as promising areas are ignored in favor of further research. Cooling algorithms also are liable to be trapped in local minima, if exploration decreases too much at the wrong time. If this occurs, the algorithm has little chance of escaping. The pattern-based methods studied can also be divided into two classes—genetic and eugenic. The genetic algorithm studied relies on random mutation and recombination, while the eugenic algorithm uses smart, statistically justified mutation and recombination. The EuA will be the only eugenic algorithm studied, although tests will be conducted with EuAs that do not use the restriction operator, and are therefore effectively equivalent to BSC (Syswerda, 1993).

Different parameter settings for these algorithms were tested in the experiments. For SA and DMHC, two settings of the "cooling rate" parameter are studied. For the GA, two recombination methods are tested. IMHC has no parameters, and therefore it wasn't possible to study more than one incarnation of it. The parameters varied for the EuA were population size, selection noise, creation rate, and the use of the restriction operator. The goal of the study was not to determine the "absolute best" algorithm for use on a particular test problem, but to identify the prevalent biases, strengths and weakness of the algorithms, and to develop a unified theory of why a particular algorithm *type* would perform well on some problems, but not on others. Therefore no effort was made to tune any of the algorithms parameters used, even when hindsight suggested that some parameters were obviously ill-chosen for a particular problem. Please note that there are *many* variations of SA and GAs; the ones used for the experiments, as well as their parameters, were chosen to represent the "standard" incarnations, for which a high level of performance should be expected.

5.2.1 Increasing Mutation Hill Climbing (IMHC)

In IMHC, a random individual is initially chosen. Random locus mutation of varying size (from one-bit to l-bit) is then applied to this individual. Mutations are accepted when they increase fitness or cause no change to fitness, otherwise the individual remains unchanged. Initially, mutation size (i) is one. Mutation size is slightly increased when no increases in fitness are encountered after a number of attempts. In particular, when no increase in fitness is encountered after $l \cdot 2^i$ i-bit mutations are attempted, mutation size is increased to $i + 1$ bits. For example, if $l \cdot 2^1$ one-bit mutations cause no increases in fitness, then $l \cdot 2^2$ two-bit mutations are attempted, and if still no increases in fitness are encountered, $l \cdot 2^3$ three-bit mutations are attempted, and so on.. The number i can increase until $i = l$. If this happens, i is reset to one. In order for IMHC to escape a local optimum, a point outside that local optimum's basin of attraction must be discovered that has a fitness equal to or greater than that of the local optimum's.

IMHC is the "silver bullet" against trivial toy problems such as BIT-COUNT. Out of all the algorithms tested, IMHC will find the optimum the fastest if a problem is 0% epistatic, since its one-bit mutations will quickly hill climb to points of higher and higher fitness. In such a case, the expected number of fitness evaluations it will have to perform is $O(l)$. However, if it encounters a very large group of genotypically similar individuals with *equal* fitnesses, it can waste large amounts of function evaluations by mutating back and forth among members of this group. A slight optimization to IMHC (and all the mutation-based algorithms) would be to keep track of previous mutations, and prevent the algorithm from retrying the same mutation on the same individual more than once.

5.2.2 Decreasing Mutation Hill Climbing (DMHC)

DMHC is a very simple cooling algorithm. After an initial individual is randomly chosen, random locus mutations of decreasing size are attempted. When mutation causes no change or an increase in fitness to current individual, the mutated individual replaces the current individual. The mutation probability of each bit, p_M, is a function of the "cooling rate" r_c, the number of function evaluations t and the maximum number of function evaluations t_{max}:

$$p_M = 0.5(1.0 - \frac{t}{t_{max}})^{r_c} \tag{5.1}$$

At $t = 0$, p_M is 0.5, and so half the bits in a genotype will be mutated on average. As r_c is increased, p_M decreases more quickly, with a resulting decrease in exploration, in favor of exploitation. DMHC is very similar to simulated annealing (see below), except that newly generated points with lower fitness than the current individual have absolutely no chance of replacing the current individual. In the discussions to follow, the notation DMHC-x will be used to denote DMHC with a cooling rate of $r_c = x$.

5.2.3 Simulated Annealing (SA)

The simulated annealing algorithm used in the experiments is only a rough approximation of "true" simulated annealing. It is identical to DMHC, except that points of fitness lower than the current individuals' fitness have a non-zero probability of being accepted and replacing the current individual. This probability decreases as more function evaluations are performed. This is the "acceptance probability" (p_A) is defined as follows:

$$p_A = \frac{1.0}{1.0 + exp(\frac{\Delta_f}{0.5(1.0 - \frac{t}{t_{max}})^{rc}})} \tag{5.2}$$

$$\tag{5.3}$$

$$\Delta_f = \frac{\mid f(\mathbf{x}_1) - f(\mathbf{x}_2) \mid}{\mid f(\mathbf{x}_1) \mid} \tag{5.4}$$

$$\tag{5.5}$$

where $f(\mathbf{x}_1)$ is the fitness of the current point, $f(\mathbf{x}_2)$ is the fitness of the mutated point, and r_c controls the rate at which the acceptance probability and mutation size decreases. The desired benefit of accepting mutations that cause decreases in fitness is to decrease the probability of being trapped in a local minima and increase exploration. For four of the six problems (F31, F32, F33, and F38), this tactic appeared to be slightly detrimental, as SA under-performed DMHC, and, for the three more difficult problems (F31, F32, and F33), the tactic was severely harmful to the SA's performance relative to DMHC's performance. Against a problem (F2) that was specifically designed to have an extremely significant local minimum, this tactic was clearly beneficial. However, it must be noted that F2 is an artificial test problem created to *simulate* hard problems, while F31-F33 are classic combinatorial optimization problems. In the discussions to follow, the notation SA-x will be used to denote SA with a cooling rate of $r_c = x$.

5.2.4 The Genetic Algorithm (GA)

The GA used in these tests was "genesys-2.0"(Bäck, 1992). Fitness-proportionate selection and elitism were used. The population size was fixed 50 individuals, while the crossover probability and per-bit mutation probability were set to 0.6 and 0.001, respectively. The fitness values of the population were rescaled every 5 generations. Both simple 2-point crossover (2X) and uniform crossover (UX) were tested independently, with UX seeming to have a slight advantage. In all the experiments, the GA's consistently converged to unimpressive populations, in spite of the fitness rescaling used.

5.2.5 The Eugenic Algorithm (EuA)

A complete description of the eugenic algorithm was given in chapter 4. Many different combinations of parameters were studied:

Population Size: 20,50, and 100

Selection Noise: 0.01 to 0.25 (usually 0.05)

Creation Rate: 0.01 to 0.25 (usually 0.05)

Restriction Operator: off or on

The primary parameters studied were population size and selection noise. Also, the EuA was tested with the restriction operator shut off. Removal of the restriction operator degraded performance for all problems except for F02.

5.3 Combinatorial Optimization Experiments

In this section, six experiments are described. Each experiment features a particular combinatorial optimization problem and the performance and behavior of all five optimization algorithms on this problem. All of the problems are *minimization* tasks, and so "higher fitness" individuals produce smaller phenotypic values for these functions. For a given individual, I shall refer to the actual function value as the individual's phenotype, while I will use fitness to refer to the "goodness" of the individual.

All of the results were averaged over 100 runs of a particular algorithm/parameter combination. The mean phenotypic value of the best individuals found by the 100 runs was computed every 100 function evaluations, and learning curves were constructed that graphed the number of function evaluations versus the average phenotypic value of the best individual found by each run. Both axes of the learning curves are plotted using base ten logarithms of the actual numbers encountered. This is done in order to more clearly display differences in behavior. When the range of phenotypic values is huge, differences in the average phenotypic values can be more easily seen on log-log plots. Also, most of the learning curves of the experiments were highly exponential in nature, with initial extremely large changes in phenotypic value followed by very little change for many function evaluations. Few performance differences could be detected when examining linear plots. If the logarithms of both the number of function evaluations and average phenotypic value were not used, most of the plots would resemble many "L"-shaped curves squashed together on the left side of the graphs. The appearance of the learning curves is more linear using log-log plots, and behavioral differences are clearer.

Each experiment in this chapter will be presented with four accompanying graphs. All the graphs show the performance of three "standard" EuAs, which use a 5.0% selection noise and creation rate. The three standard EuAs use different population sizes—20, 50 and 100 individuals. Each graph compares the performance of these three standard EuAs with either "nonstandard" EuAs or different algorithms altogether. Nonstandard EuAs are those that have extremely high selection noise, creation rates, or do not use restriction. The notation EuA-λ is used to denote eugenic algorithms with population size λ. The variables N and C will denote EuA selection noise and creation rates, respectively. The four graphs presented for each experiment are:

1. Eugenic vs Genetic Search: Graphs the behavior of the standard EuAs relative to two or more GAs that use different recombination operators (uniform or simple 2-point crossover).

2. Eugenic vs Mutation-Based Search: Graphs the behavior of the standard EuAs relative to SA, DMHC, and IMHC. Two cooling rates, 1.0 and 4.0, were studied with both SA and DMHC.

3. Selection Noise and Creation Rate Tests: Graphs the behavior of the standard EuAs relative to EuAs that use various differing combinations of population size, selection noise and creation rate.

4. Restriction Operator Tests: Graphs the behavior of the standard EuAs relative to EuAs that do not use the restriction operator.

The o(1)-epistasis (E) and fitness variation (F) of each problem was estimated for each problem. A random sampling of individuals was generated and then epistasis ($o(1)\chi^2$-Epistasis) and fitness variation ($\frac{S_b}{S_w}$) were calculated using the random sampling. For each paired o(1)-competition, two genotypes that differed by a single bit were compared. Five hundred o(1)-competitions occurred per gene (bit). Therefore the total number of random samples taken for an l-bit problem was $l \cdot 500$.

5.3.1 F02: The Two-Dimensional Rosenbrock Function

Problem Description

The two-dimensional Rosenbrock function has one global optimum and one major local optimum, when encoded in 2-dimensional real space (De Jong, 1993). It is an artificial problem that was intentionally created to have a local optimum with a basin of attraction much larger than the global optimum's basin of attraction. It is therefore very deceptive in real space. The two dependent variables can vary in the region $[-5.12, 5.12]$. This problem's phenotypic values can range from 0 to approximately 10^5.

$$F02(\mathbf{y}) = (100 \cdot (y_2 - y_1^2)^2 + (y_1 - 1)^2) \tag{5.6}$$

$$y_i \in [-5.12, 5.12] \tag{5.7}$$

The genotype $[1.0, 1.0]$ yields the global optimal phenotype (0.0), while the local optimum (0.017) occurs at $[1.13111, 1.28]$. When $y_1 = 1.13111$ and $y_2 = 1.28$, $y_1^2 \approx 1.28 = y_2$, and therefore both the first term in F02 $(100 \cdot (y_2 - y_1^2)^2)$ and the second term $((y_1 - 1)^2)$ are very close to zero.

$32,000$ samples were generated for the random sampling conducted for estimating epistasis and fitness variance. The measured $o(1)\chi^2$-Epistasis) was 0.70 and the fitness variation $(\frac{S_b}{S_w})$ was 1.84. The average phenotypic value was 14418.5; the minimum value encountered in the random sampling was 0.0159001 and the maximum was 97766.4.

When the two real variables are encoded in n-dimensional binary space, however, the problem becomes rife with many additional mutation-based and pattern-based local optima (due to hamming cliffs). Obviously, the global optimum still occurs when the both groups of 32 bits are decoded to yield the real number 1.0. However, due to limited precision, this number can be coded in two different ways:

$$G_1 = 00000000000000000000000010011001 \quad (= 1.0)$$
$$G_2 = 11111111111111111111111100011001 \quad (= 1.0)$$

Note that the rightmost bits are the most significant in this binary encoding.

In addition, the strong local optimum 0.017 still occurs when the first group of 32 bits decodes to 1.13111 and the second group decodes to 1.28. The coding for this local optimum is:

$$y_1 = 01010010010001001110001000111001 \quad (= 1.13111)$$
$$y_2 = 00000000000000000000000000000101 \quad (= 1.28)$$

The reason why this is a local optimum is because a large mutation would be required to decrease the phenotypic value, while intermediate-sized mutations would cause *large* increases in the phenotypic value. As a result, even though the second group of bits differs by only 4 bits from the G_1 setting (causing 1.0), changing these 4 bits would require a simultaneous change of either 11 bits (to the G_1 setting) or 16 bits (to the G_2 setting) in y_1's bits, so that y_1's bits would also decode to the value 1.0. Therefore, to decrease the phenotypic value from 0.017 down to 0.0 would require a fairly precise jump of $(4+11) = 15$ or $(4 + 16) = 20$ bits—roughly 25%-30% of the bits. The jump would also have to be well coordinated, because the large first term in F02 causes large increases in the phenotypic value when y_1^2 differs too much from y_2. Because of the transformation from a real space genotype to a binary space genotype, many additional local optima are introduced into the problem(including another fairly strong one at phenotypic value 1.0, requiring a perfect 12-bit mutation). Many of the optimization algorithms tested frequently converged to one of these other local optima (see below).

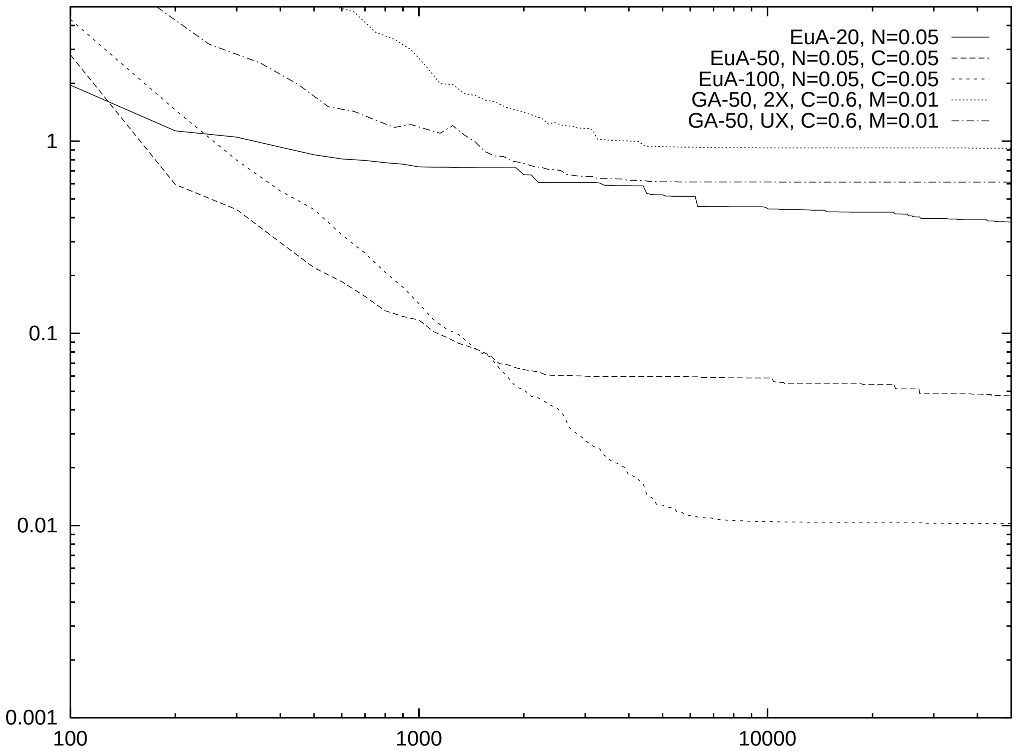

Figure 5.1: F02: Eugenic vs Pattern-Based Search. Larger populations helped the EuA to avoid the strong local optima. UX-crossover, since it is more disruptive than 2X-crossover, resulted in better GA performance.

Results

The local optima mentioned in the problem description dominated the landscape of the binary search space, and therefore dominated the performance of all the search algorithms. Over the course of their 100 runs, both EuA-20 and IMHC converged to about 40 different optima (including the global optimum at (1.0, 1.0). These optima ranged in phenotypic value from 0.0 to 1.0000000024. Out of the 39 *genotypical* different locally optimal individuals, only 33 different *phenotypic* values were encountered. For example, there were four different individuals whose phenotypic value was 0.0, and three whose phenotypic value was 0.0784000008. This multiplicity of individuals per phenotype was due to the limited precision of the binary encoding and the Hamming cliffs inherent in any binary encoding. As with the major local optimum at (1.13111, 1.28) discussed earlier, very large and accurate mutations are needed to escape the basins of attraction of these local optima.

Due to the great many local optima, exploration was far more valuable than ex-

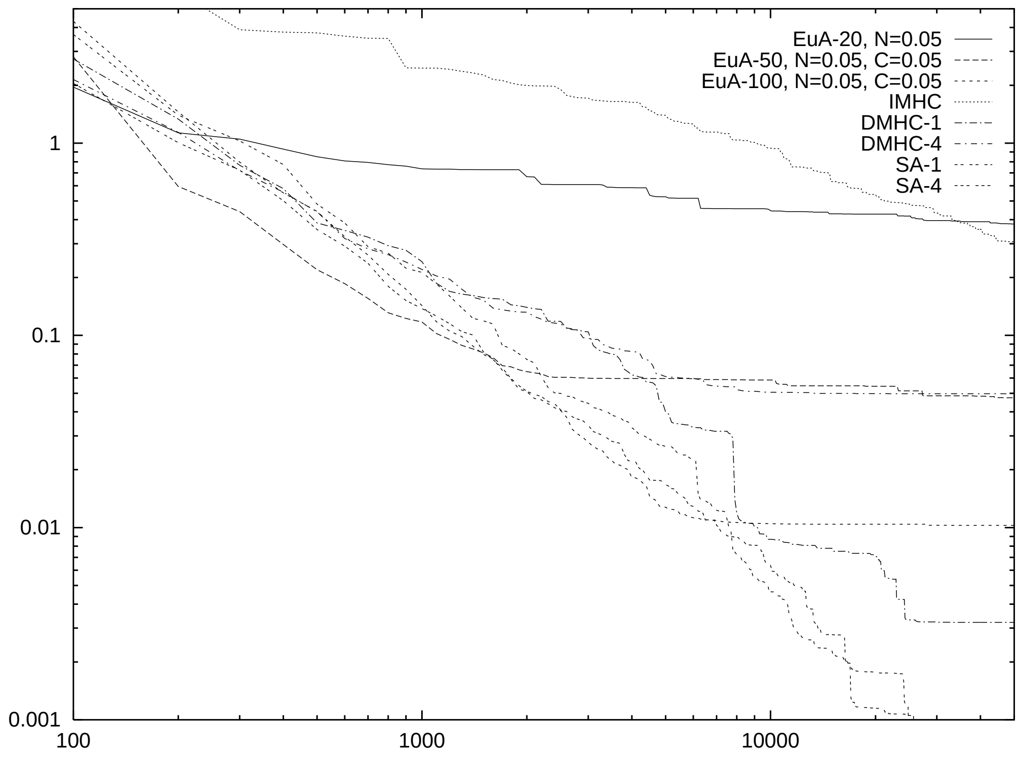

Figure 5.2: F02: Eugenic vs Mutation-Based Search. IMHC was frequently trapped by local optima. SA, due to the good match between its major heuristic and the problem structure, performed well. DMHC did not perform as well as SA, due to its determinism. Greater stochasticity clearly aided DMHC.

ploitation. For example, very high settings of the EuA's selection noise (N) and creation rate (C) greatly improved the chances of the EuA finding the global optimum. In addition, EuAs with larger population sizes found the optimum more frequently than EuAs with smaller populations. The higher diversity levels in the larger EuA populations allowed them to more thoroughly explore a wider variety of samples, and so the larger populations were more able to avoid local minima. Also, larger population EuAs are less likely to be dominated by a single individual or small group of individuals, and therefore they are less likely to prematurely converge. An initially surprising result was that the EuA performed better *without* the restriction operator. This was due to the vastly increased exploration that was performed, at the cost of greatly decreased exploitation of epistatic interactions. When using the restriction operator, the EuA would attempt to fine tune its settings for each local optima, instead of making the large random jumps needed to escape local optima. The EuAs that used the restriction operator initially performed better than those without,

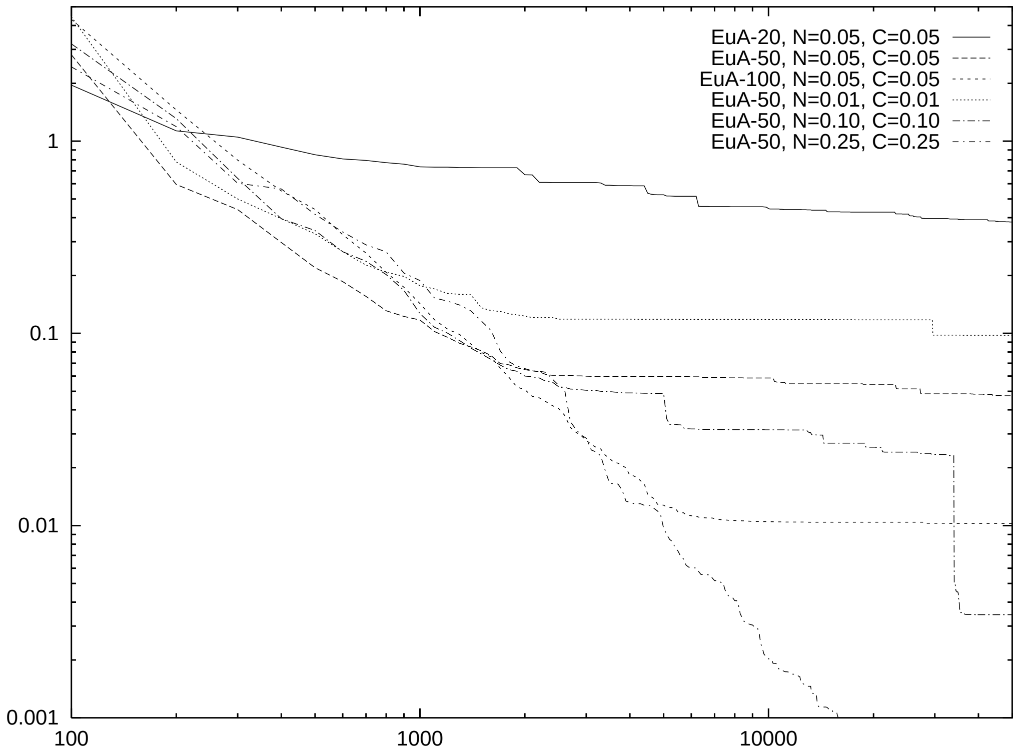

Figure 5.3: F02: Selection Noise and Creation Rate Tests. The performance of the EuA-50 was clearly helped by greater stochasticity. With a 25% selection noise and creation rate, the EuA-50 was the best performing algorithm overall.

but their propensity to converge to local optima eventually damned their initial successes. In summary, final EuA performance was determined by how well local optima were avoided, not by how well the EuA could exploit promising schemata.

The experiments performed on this problem using non-eugenic algorithms also stressed the importance of exploration at the expense of exploitation. The most greedy mutation-based algorithms, IMHC and DMHC-4, and the most greedy EuA (the EuA-20) faired the worst. By 50,000 evaluations, IMHC had often found the same local optima as the EuA-20, but IMHC was doing a better job of fine tuning these solutions, and so it was still finding better points while EuA-20 had already converged. IMHC's heuristic of taking very small one-bit mutations was bound to fail on this problem, since it would most likely end up exploiting local optima. As would be expected, DMHC faired better than the more exploitative IMHC but worse than the more exploratory SA. DMHC-4 and the EuA-50 performed almost identically, while DMHC-1 (with its slower cooling schedule and therefore greater exploration) fared much better than DMHC-4 and even the EuA-100.

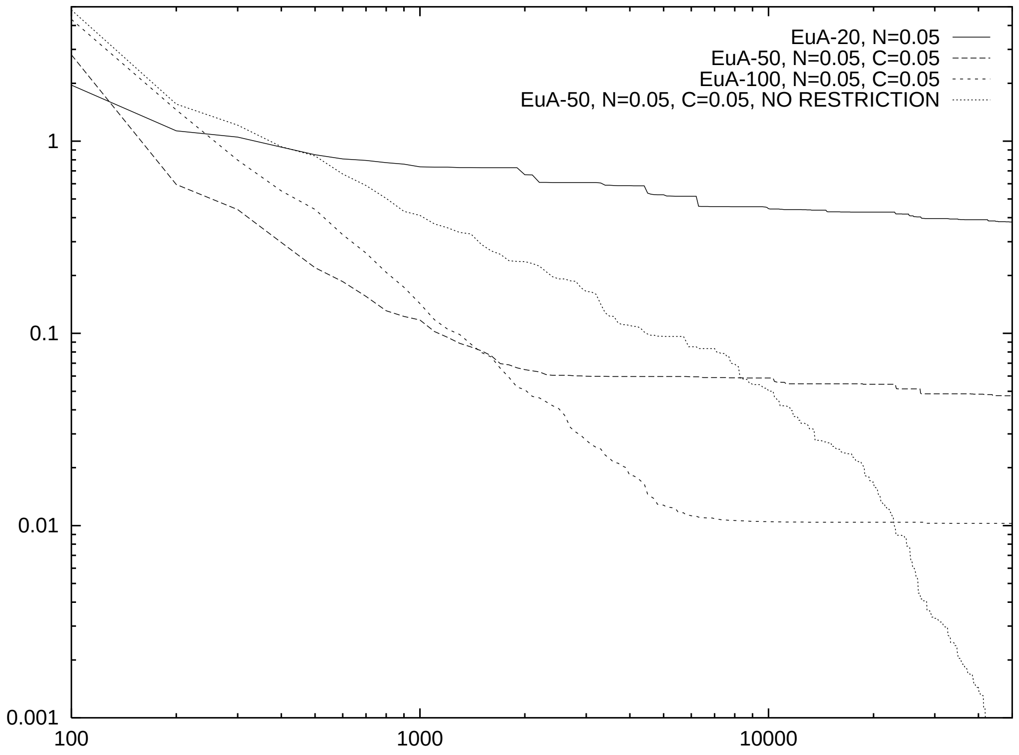

Figure 5.4: F02: Restriction Operator Tests. Restriction actually harmed EuA performance, since its epistatic tracking lead to increased exploitation of local optima.

The most exploratory mutation-based algorithms, SA-1 and SA-4, achieved excellent results, even beating the EuA-100 (but not the EuA-50 with selection noise and creation rates of 0.25). SA's "generous" (less greedy) strategy of allowing lower-fitness individuals to replace the current individuals was very successful. DMHC and SA performed very well on F02, compared to the other algorithms, but DMHC did not perform as well as SA. Both DMHC and SA rely on decreasing size mutations which start out large enough to avoid local optima, but the two algorithms differ in that SA will randomly accept lower-fitness replacements for the current individual, while DMHC never does. Because of this, SA was more able to avoid F02's local optima.

F02, as all the other problems, demonstrated the weaknesses of the GA. Both GAs made very slow progress initially, and then prematurely converged. The GA was clearly the worst performing algorithm on F02. The GAs had neither the ability to maintain high levels of "educated" exploration, nor did they have a trace of the exploitative ability of the EuA or the mutation-based algorithms. For this problem, UX faired better than 2X most likely because of the higher amount of disruption it caused on the 32-bit groups.

The experiments confirmed the hypothesis that this space was filled with very strong local minima, and that the best strategy for such a space is to sacrifice exploitation for increased exploration. The strong performance of the EuA (and in particular the winning performance of the EuA-50 with high selection noise and creation rates) demonstrated the versatility and flexibility of the eugenic algorithm.

5.3.2 F30: Subset Sum Problem

This problem involves a set I of positive integers in the range [0,1000], and a large positive integer M. The goal is to choose a subset of I in such a way that the sum of the integers in the subset is as close to M as possible, but does not exceed M. If this sum does not exceed M, then the solution is "feasible". If the sum exceeds M, then the solution is "infeasible". The representation chosen for this problem is described in (Khuri et al., 1993). This problem is included in the genesys-2.0 package. Each bit in the genotype represents the presence or absence of a particular integer from the set I. The problem tested had 50 integers in the set I, and so the genotype was 50 bits long. If a solution is feasible, its phenotype is the difference between M and the sum of the chosen integers. Otherwise, the phenotype is simply the sum of the chosen integers. Therefore, lower phenotypic values lead to higher fitness.

$$F30(\mathbf{x}) = \begin{cases} M - \sum_{i=1}^{l} I_i \cdot x_i & \text{when feasible} \\ \sum_{i=1}^{l} I_i \cdot x_i & \text{otherwise} \end{cases} \tag{5.8}$$

$25,000$ samples were generated for the random sampling conducted for estimating epistasis and fitness variance. The measured epistasis was 0.04 and the fitness variation was 0.21. The average phenotypic value was 5561; the minimum value encountered in the random sampling was 0 and the maximum was 24945. The overall phenotypic range of the problem is 0 to approximately 25,000.

Results

Except for the genetic algorithms, all of the optimization algorithms tested had little difficulty solving this problem. Even the random sampling discovered several optimal points. Very few differences can be discerned among the various algorithms. Epistasis is fairly low for this problem—the χ^2-epistasis estimated for F30 was far lower than that of F02, and, in addition, IMHC (the algorithm specifically designed for low-epistasis problems) was the second best performer.

Among the mutation-based algorithms, algorithms using faster cooling rates (and therefore more exploitation) fared better than their more exploratory counterparts. The

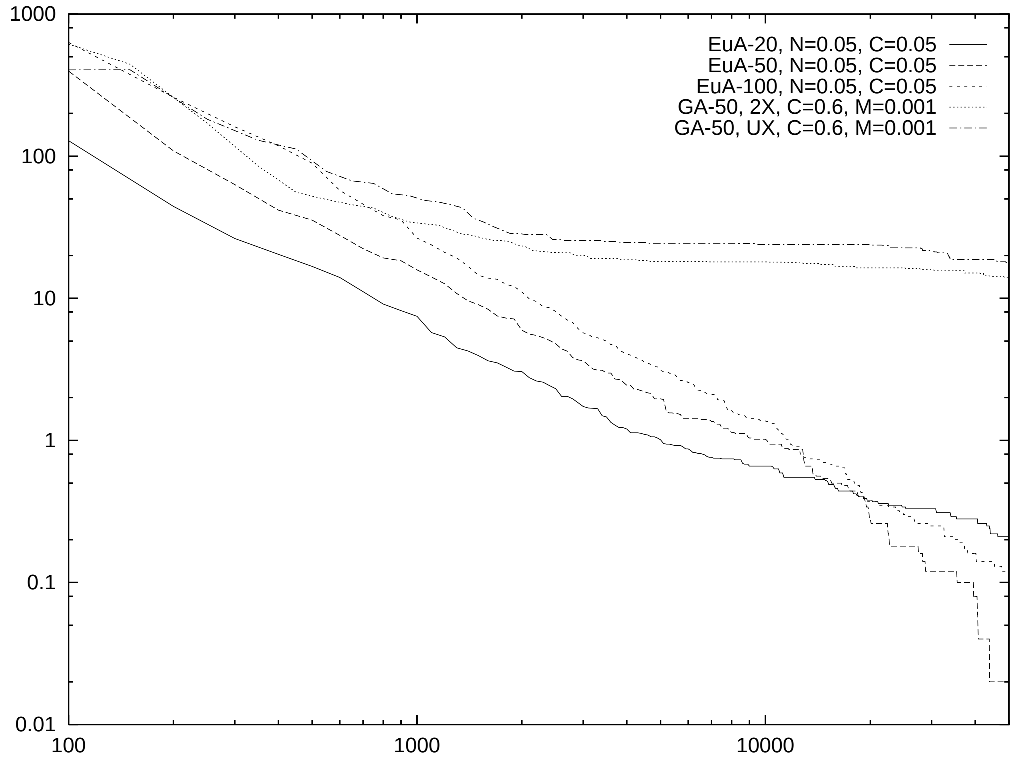

Figure 5.5: F30: Eugenic vs Pattern-Based Search. Greater exploitation and less search inertia enabled smaller-population EuAs to initially make more speedy progress than larger-population EuAs. However, this increase in exploitation lead to an increased tendency to prematurely converge.

eugenic algorithm lagged somewhat behind the mutation-based algorithms, but this behavior would be expected for such a simple, non-epistatic problem, as the EuA was designed for much more epistatic problems.

Larger populations slowed the search pace of the EuA, due to the smaller effect that new high-fitness individuals have on the statistics of larger populations[2]. When the EuA-20 actually found global optima, it did so faster than the EuAs with larger populations. However, the EuA-20 was less likely to find optimal points than the EuA-50 or the EuA-100. Compared to the EuA-20 and the EuA-100, the EuA-50 achieved a very good balance between the speed of finding optima and the probability of finding optima.

Very low levels of selection noise ($N = 0$) made little difference in EuA performance, except for the EuA-20. Initially, the EuA-20 with no selection noise made the same progress as EuA-20 with 5.0% selection noise, but after about 10,000 evaluations, the lower level of

[2]A more in-depth discussion of the effect of population size on EuA behavior will be discussed in the next chapter.

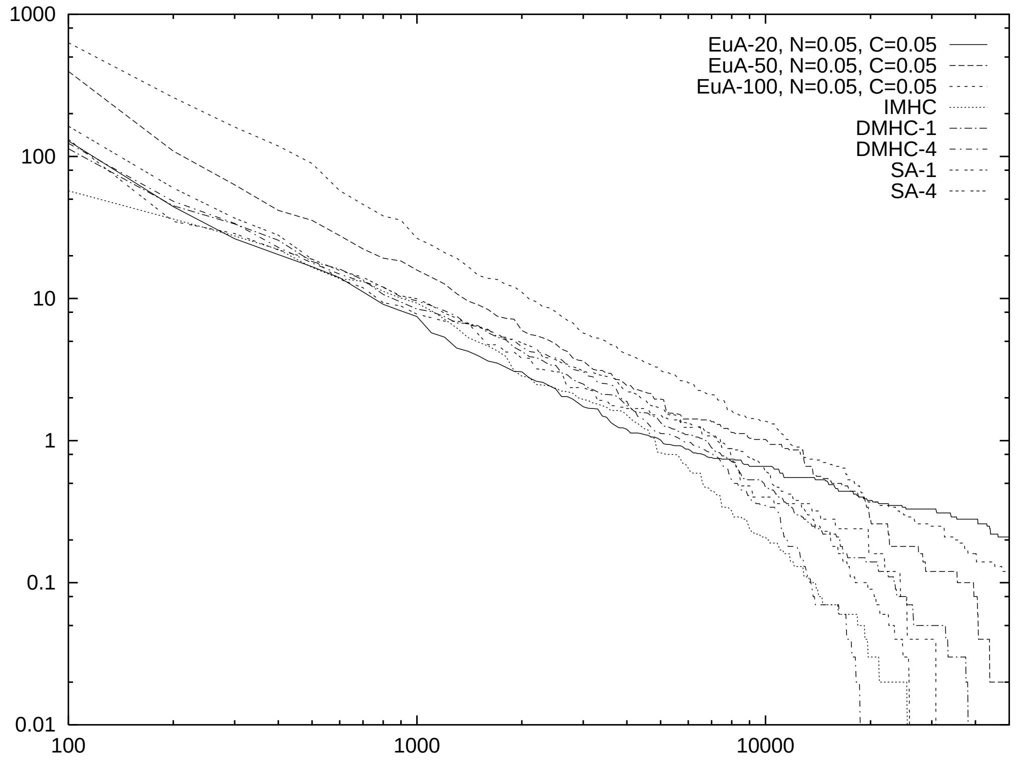

Figure 5.6: F30: Eugenic vs Mutation-Based Search. The highly similar, very good performance of all the algorithms demonstrated that F30 is a fairly easy problem. Greater stochasticity enabled DMHC and SA to progress slightly faster. IMHC was the second best performing algorithm, demonstrating that a very small amount of epistasis was present.

selection noise allowed the EuA-20 to find optima much more frequently than all the other EuAs, most likely due to the increased accuracy of the allele choices and recombinations that it performed.

Without the restriction operator, EuAs were able to make fair progress (and still outperformed the GA's tested), but were not able to compete with the EuAs using the restriction operator. This demonstrated that, although this was a fairly easy problem, it still had some epistasis that needed to be tracked. Since exploitation worked well on this problem, it is not surprising that performance was decreased when the restriction operator was not used. This is because EuAs without the restriction operator use statistics from complete populations, instead of concentrating on groups of a few similar individuals, and therefore being more exploitative. Population size—whether 50 or 100 individuals—seemed to make little difference to the EuA's that did not use the restriction operator.

In summary, algorithms performing heavy exploitation were the clear winners. The

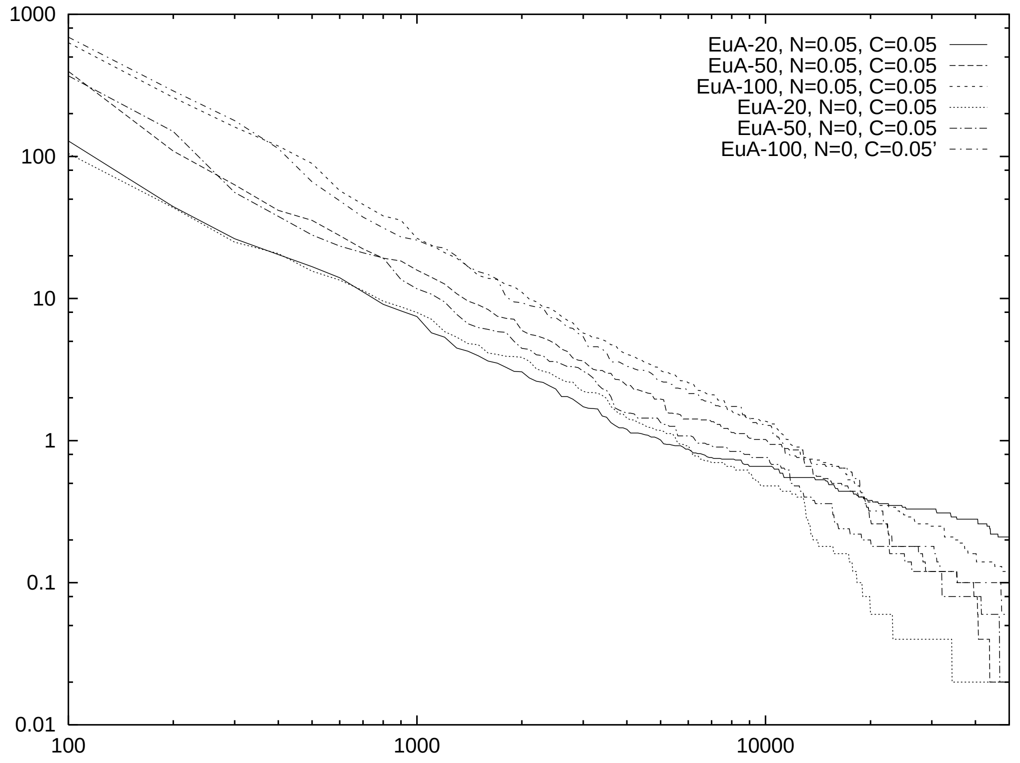

Figure 5.7: F30: Selection Noise and Creation Rate Tests. Lesser stochasticity aided performance.

most exploitative two mutation-based algorithms—DMHC-4 and IMHC—did much better than the other algorithms. The most exploitative EuA—the EuA-20 with no selection noise—performed better than all the other EuA's. While at first random recombination allowed both GAs to make fair progress, both GAs prematurely converged even though this problem was very easily solved using greedy small mutations.

5.3.3 F31: Multiple-Knapsack Problem

Problem Description

This was the largest problem in the test suite (2^{105} possible solutions). Even after 50,000 evaluations, many of the algorithms were still clearly making progress. This problem is similar to the Subset-Sum Problem. There is a set of m knapsacks, each with its own capacity $c_1 ... c_m$, and n objects, each yielding a profit $p_1 ... p_n$. The weight of each object is different for each knapsack. Therefore there is a matrix of weight values, such that $w_{i,j}$ is the weight of the ith object in the jth knapsack. The profit of each object remains constant,

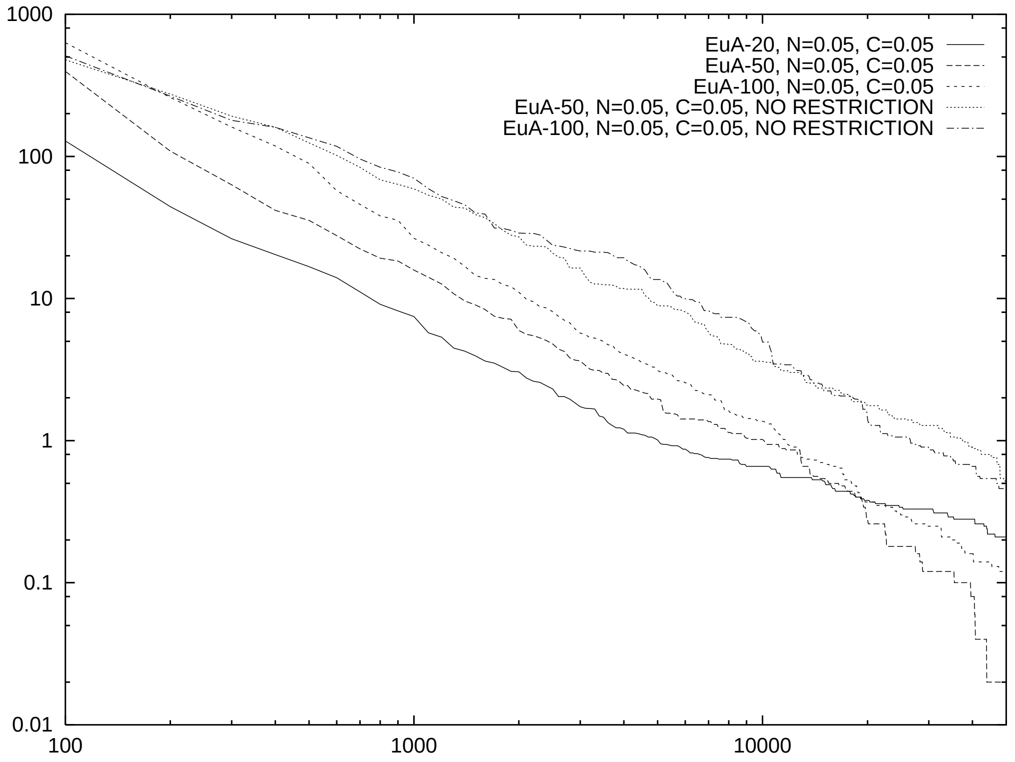

Figure 5.8: F30: Restriction Operator Tests. The EuAs without restriction lagged behind the standard EuAs, but not by much. This demonstrates that some epistasis was present, but not very much.

and either an object is placed in *all* the knapsacks, or not used at all. As with the Subset-Sum Problem, each bit in the genotype denotes the presence or absence of a particular object. A point is feasible when no knapsacks capacities are exceeded. This particular problem analyzed was drawn from the genesys-2.0 test suite, and had 2 knapsacks and 105 objects, but the fitness function was modified in order to provide a smoother transition from infeasibility into feasibility, by indicating "how infeasible" particular infeasible points were. There appears to be only a single optimum, as all the algorithms that found an optimal individual found the same individual.

$$F31(\mathbf{x}) = \begin{cases} \sum_{i=1}^{n} p_i - \sum_{i=1}^{n} p_i x_i & \text{when feasible} \\ \sum_{i=1}^{n} p_i + \sum_{j=1}^{m} (\sum_{i=1}^{n} w_{i,j} x_i - c_j) & \text{otherwise} \end{cases} \tag{5.9}$$

52,500 samples were generated for the random sampling conducted for estimating epistasis and fitness variance. The measured epistasis was 0.00 and the fitness variation was 0.00. The average phenotypic value was 626992; the minimum value encountered in the

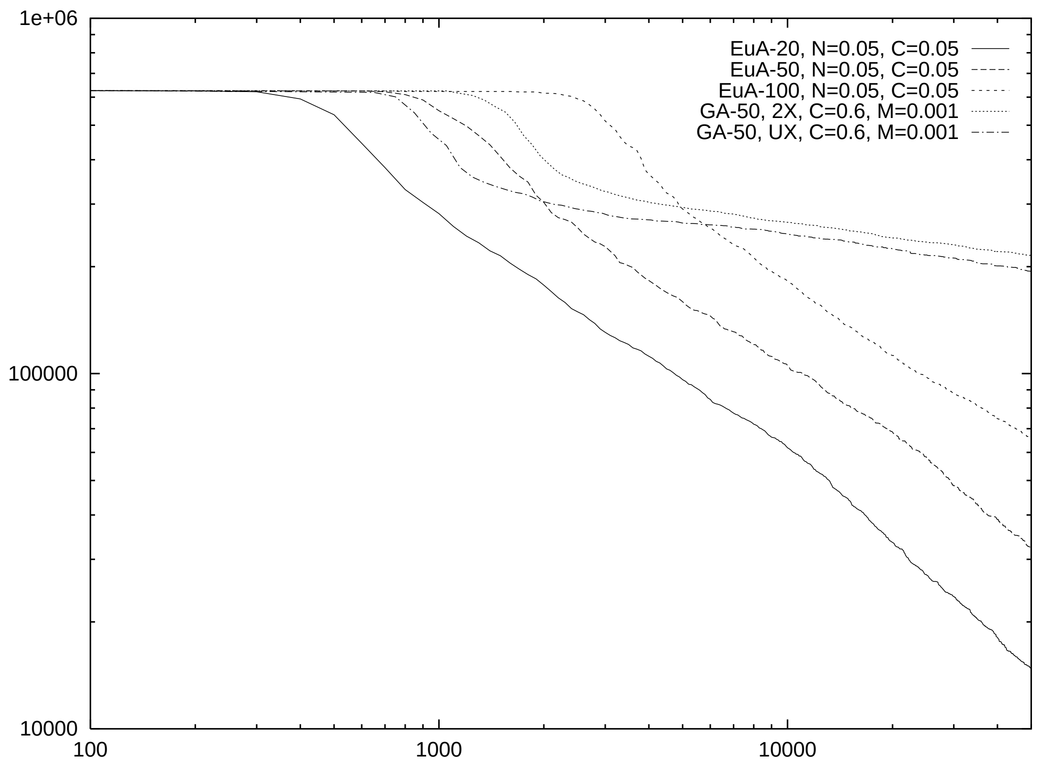

Figure 5.9: F31: Eugenic vs Pattern-Based Search. The greater exploitation of the smaller-population EuAs enabled them to stay ahead of the larger-population EuAs, but the log-log rate of convergence was not affected by population size. The more disruptive UX-crossover operator caused a slight increase in GA performance over the 2X-crossover operator. However, both GAs' convergence rate was very small.

random sampling was 624811 and the maximum was 629237. The overall phenotypic range of the problem is 0 to approximately 10^6.

Results

The random sampling produced no feasible points. Very low epistasis is present in the random sampling—there are clear winners in most of the allele competitions. Most likely the winners were the '0' alleles, as they would always produce smaller total weights, and therefore "less infeasible" solutions. There are *many* infeasible points, and there is a clear heuristic for producing feasible points—replace as many '1' alleles with '0' alleles. Therefore the epistasis is low in this region of the space, as each gene has a clear allele winner ('0'). The infeasible region was so large and exploitation was so necessary, that if an algorithm attempted too much exploration while still in the infeasible region, it would not fair very

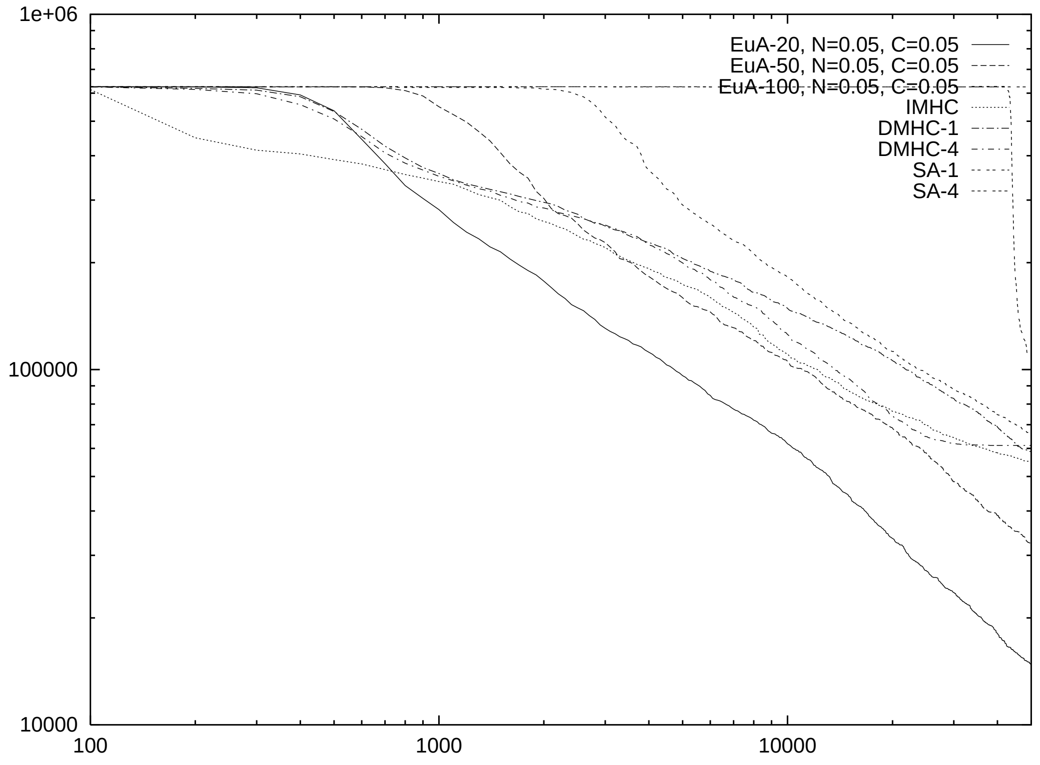

Figure 5.10: F31: Eugenic vs Mutation-Based Search. IMHC was able to quickly reduce infeasibility, but was often trapped by the deceptive local optimum. DMHC initiallly made slower progress than IMHC, but was able to catch up later in the search. Both SA algorithms were unable to any significant progress.

well. SA and the non-restricting EuA far too exploratory for this region of space, and therefore were unable to make reasonable progress.

As would be expected, the most exploitative algorithms quickly made progress. IMHC was by far the fastest algorithm to break into the feasible region, however it was soon surpassed by DMHC and the better EuAs (no matter what their mutation rates), which all discovered the feasible region at about the same time. Although IMHC and DMHC made quick progress out of the infeasible region and into the feasible region, they were not as able as the EuA to make fast progress in the feasible region. DMHC-4 prematurely converged. This indicates the existence of mutation-based local minima, from which large mutations must be made in order to escape.

In contrast to the fairly good performance of IMHC and DMHC, SA performed dismally. SA's nondeterministic acceptance of lower-fitness points was the clear reason for SA's inferior performance, since DMHC and SA are otherwise functionally equivalent.

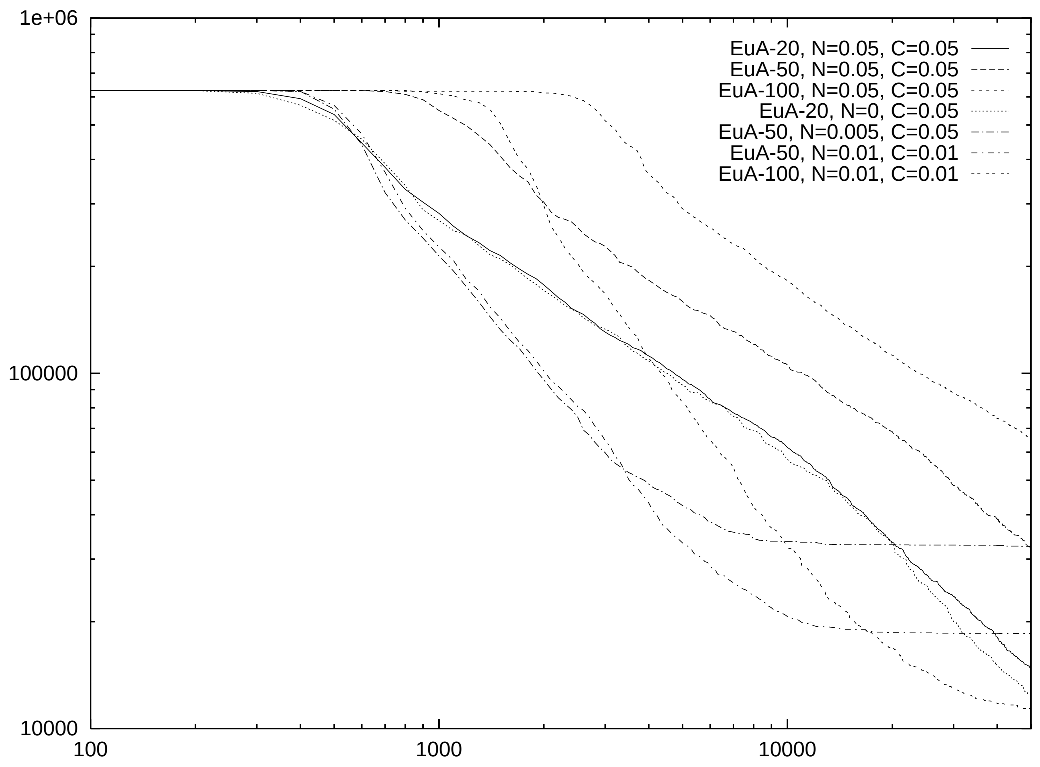

Figure 5.11: F31: Selection Noise and Creation Rate Tests. Lower stochasticity initially aided the convergence rate, but increased the chances of converging to a local optimum.

This is because of the bad match between the fitness variation in the infeasible region and the formula SA used to calculate its acceptance probability of lower-fitness points. In the infeasible region, the range of values encountered initially (and throughout the search, for SA) was very limited. The random sampling of 52,500 individuals produced phenotypic values only ranging from 624811 to 626992. This implies that an extremely large portion of the space had very large values, in a very small range. Points in this range varied only 0.34% in phenotypic value. As a result, the fitness variation in this region of the space is obviously very small, and this is confirmed by the fact that the $(\frac{S_b}{S_w})$ fitness variation measured was zero. SA would therefore be unable to move out of this initial infeasible region, due to its probabilistic acceptance of lower-fitness points. The probability of accepting lower-fitness individuals is based upon the percentage difference of the phenotypic values of the current and mutated points (new individuals of higher fitness than the current individual always replace the current individual). If the new individual has a fitness lower than the current individual, and the percentage difference between the new and current individual is small, there is a high probability that SA will replace the current individual with the new

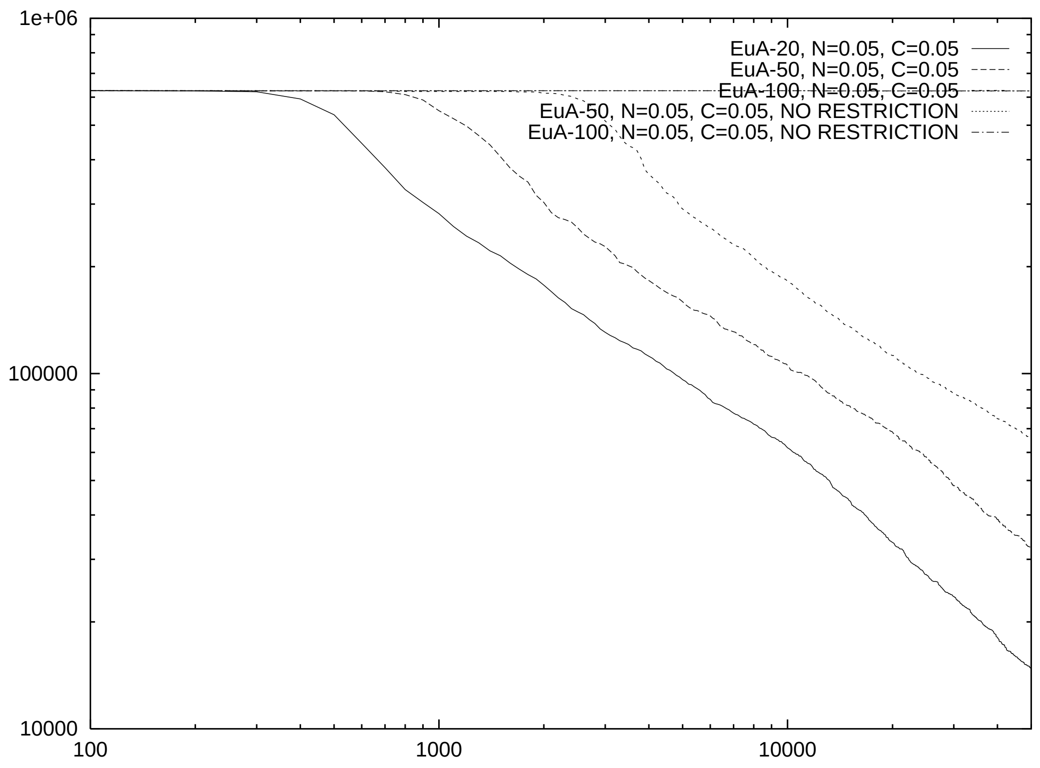

Figure 5.12: F31: Restriction Operator Tests. The non-restricting EuAs were unable to make any progress; therefore epistasis was clearly present.

individual. Therefore a mutation that causes a small (percentage) decrease in phenotypic value from the current point will have a high probability of replacing the current point. Initially, SA samples the space randomly, therefore the values initially encountered by SA are very similar to those found by the test random sample. Out of these points, the largest percentage difference that would have ever been encountered by SA would have been 0.34%, and so, with high probability, *every* point in this region would be accepted. Even if the best point of this region was encountered by SA, it would quickly be replaced by one of the myriad of other points in this initial region. No progress could be made from this region, except due to random chance, and the random sampling demonstrated that this random chance is indeed very small. It is clear that a better choice for calculating the acceptance probability would have greatly helped SA, and that the bias of SA was extremely unsuited to this problem.

Lower selection noise seemed to increase the EuA's chances of being trapped by local minima, while creation rate did not seem to have that much effect. The EuA-50 with 0.5% selection noise and a 5.0% creation rate converged to the same local optimum that trapped

DMHC-4, but not as frequently. The EuA-50 with 1.0% selection noise and a 1.0% creation rate also was trapped by the same local optimum, but far less frequently. However, the EuA-50 with 5.0% selection noise and 5.0% creation rate was not caught by this local optimum. Lower selection noise seemed to slightly increase EuA-20 performance, but only late in the search. This may be due to the fact that, initially these small populations are very diverse, and therefore the statistics calculated are initially very rough and therefore higher selection noise initially has little effect. However, later in the search, the populations become more homogeneous, and therefore the calculated statistics become more accurate, and selection noise's effect becomes more pronounced. Decreased selection noise and creation rates greatly aided the EuA-100. Perhaps this was because the statistics computed by the EuA-100 were very accurate, and increasing selection noise only added unnecessary noise. In general, EuAs with lower search stochasticity were able to initially make much faster progress, although they seemed to more often prematurely converge.

The fact that EuAs with low levels of selection noise initially made better progress implies that fitness proportionate selection is a very good heuristic for the first part of the search space, as little deceptiveness is encountered. Because of this, even the rough estimates of schemata-fitness provided by the EuA-20's population statistics are enough to quickly guide the creation of better and better individuals. However, fitness-proportionate selection can be too exploitative at times, as the premature convergence of lower selection noise EuAs demonstrates. Both GAs achieved mediocre performance, with UX seeming to have a slight edge. Surprisingly, neither GA had converged by the time the experiment ended.

It is clear that many of the algorithms would have benefited from a longer test period. It is also clear that the EuA-20 had the greatest potential for optimizing this problem, far greater than the mutation-based algorithms. Therefore even rough indicators of high-fitness patterns allowed smart recombination to outperform random mutation hill climbing.

5.3.4 F32: Maximum Cut Problem

Problem Description

This was one of the hardest problems in the test suite. In this problem, a connected graph of 100 nodes and 10000 edges must be divided into two separate sets of nodes. The sum of the weights of the edges that connect nodes from both sets must be maximized.

This was the only function that had negative phenotypic values. The best phenotypic value generated was about -203.5. The representation chosen for this problem matched that described by Khuri et al. (1993). This problem was randomly generated by the genesys-2.0 package. In the graphs to follow, 205 was added to the phenotypic values in order to present

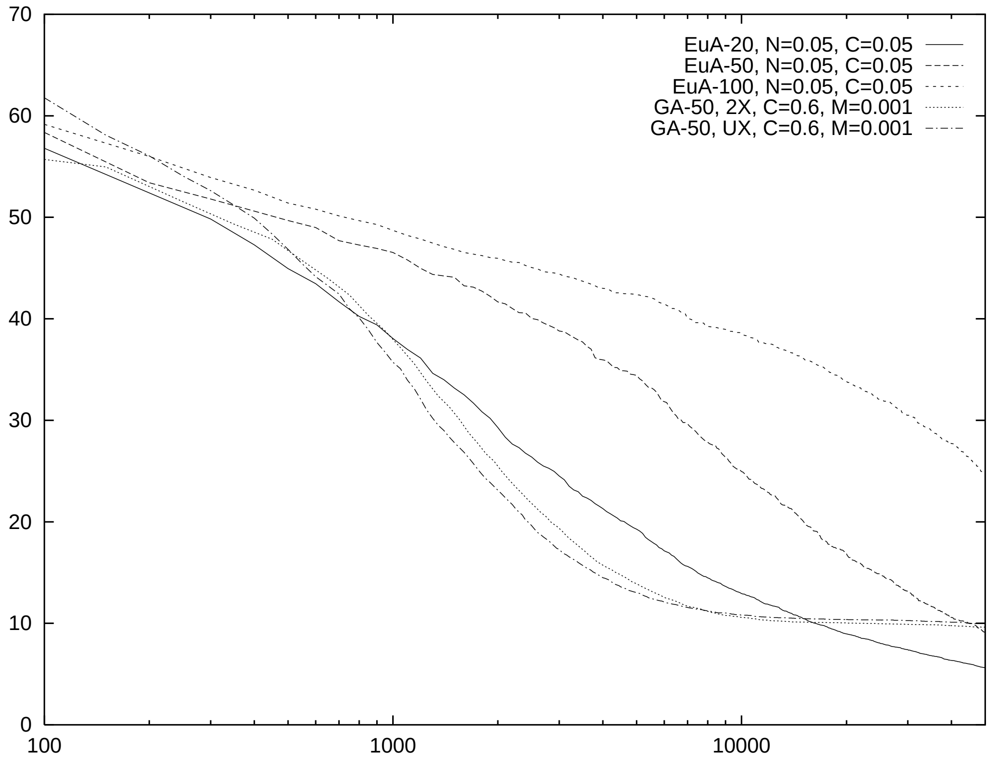

Figure 5.13: F32: Eugenic vs Pattern-Based Search. The GAs were able to progress as fast as the EuA-20, but prematurely converged. Smaller EuA population sizes clearly increased the EuA convergence rate.

positive phenotypes. Please note that these graphs, unlike those of the other experiments, are log-linear, since the range of phenotypic values was not that great.

$$F32(\mathbf{x}) = -\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} w_{i,j} \cdot [x_i(1-x_j) + x_j(1-x_i)] \tag{5.10}$$

50,000 samples were generated for the random sampling conducted for estimating epistasis and fitness variance. The measured epistasis was 1.00 and the fitness variation was 0.01. The average phenotypic value was -115.9; the minimum value encountered in the random sampling was -164.4 and the maximum was 0. The overall phenotypic range of the problem is -205 (the best point) to 0 (the worst point).

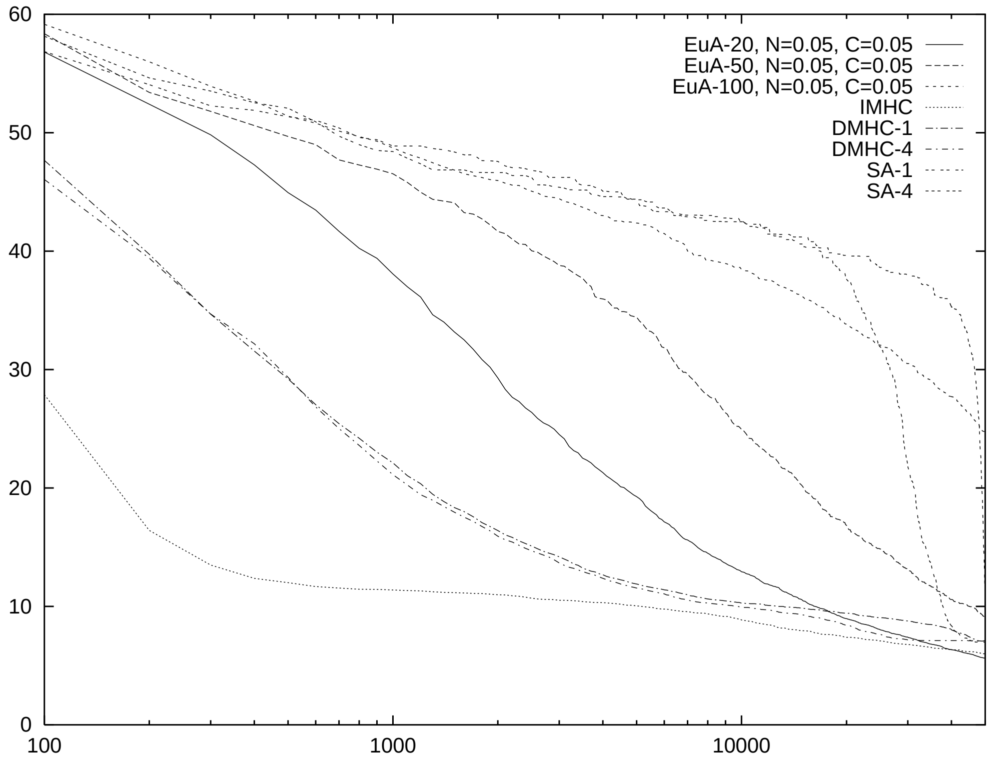

Figure 5.14: F32: Eugenic vs Mutation-Based Search. IMHC was clearly the best performing algorithm, although at the very end of the search the EuA-20 seemed to be finding better points and improving faster than IMHC. DMHC's performance was very good, but unaffected by changes in its cooling rate. SA performed poorly initially, but then was able to catch up to the other algorithms when its acceptance probability decreased sufficiently.

Results

The best point ever found had a phenotypic value of -204.76. This point was found by the EuA, with a population size of 20. The best point from each run of every algorithm was recorded, and later compared with each other. Out of 1000's of runs, the same best point was *never* discovered twice. This demonstrates that an extremely large number of points had phenotypic values very close to the optimum. The best performer was the EuA-100 with a very low selection noise (0.01) and creation rate (0.01). This algorithm was most likely to *eventually* find excellent points, although other algorithms could initially find good points faster. Some very slow progress was possible without restriction, indicating that epistasis was too high for statistics computed over many dissimilar individuals to have any meaning, but that some genes were definitely more significant than others. The restriction operator was necessary in order to compare "apples to apples". Lower stochascity

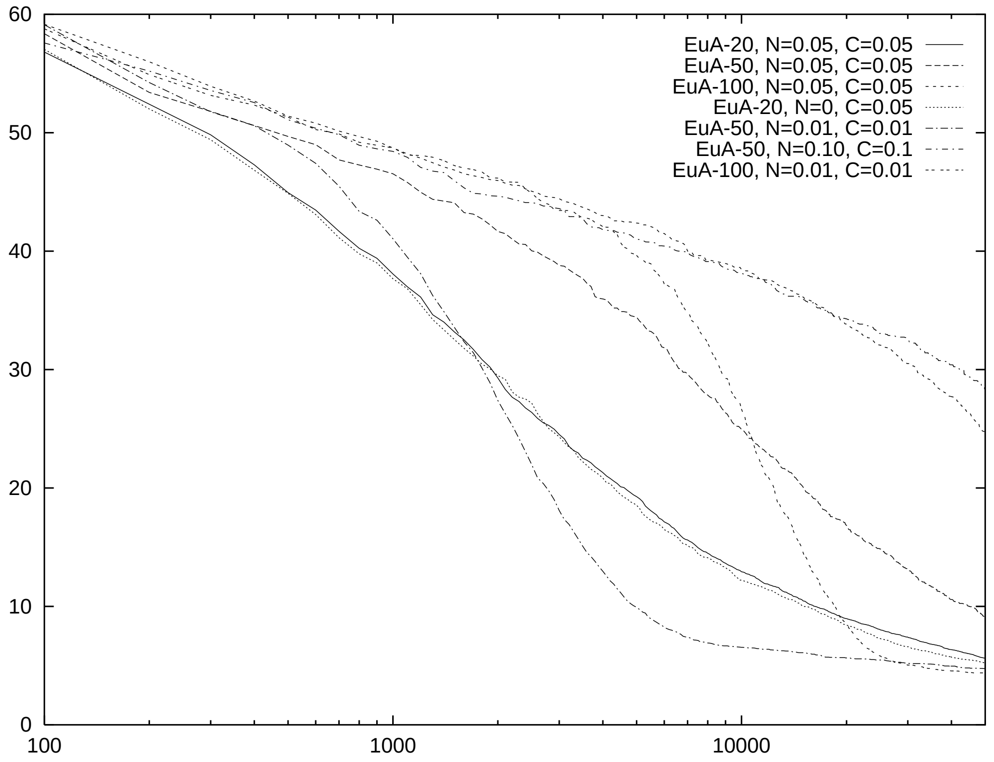

Figure 5.15: F32: Selection Noise and Creation Rate Tests. Lower selection noise had no significant effect on the EuA-20. Lower stochasticity greatly aided both the EuA-50 and the EuA-100, making them the best performing algorithms overall.

universally aided EuA performance, indicating that statistical patterns were definite and well justified in restricted regions of the search space. EuAs with smaller population sizes generally progressed more quickly, but it could not be discerned from the experiments what the asymptotic effect of population size would be.

Initially, IMHC's and DMHC's performance was extremely impressive. However, their asymptotic performance was not as great as that of some of the EuA's. IMHC found points close to the optimum far more quickly than any other algorithm. After only 500 function evaluations, most runs of IMHC had found almost-optimal solutions. However, IMHC's search foundered at this point, and never was able to find points as good as the ones found (albeit much more slowly) by the EuA-20, and by low noise EuA-50s and EuA-100s. DMHC-4 and DMHC-1 performed almost identically to each other. They did not improve as quickly as IMHC, and they usually converged to points of lower fitness than those found by IMHC. Once again, SA's probabilistic acceptance of lower-fitness points was detrimental to its performance, even though the fitness variation among the points it

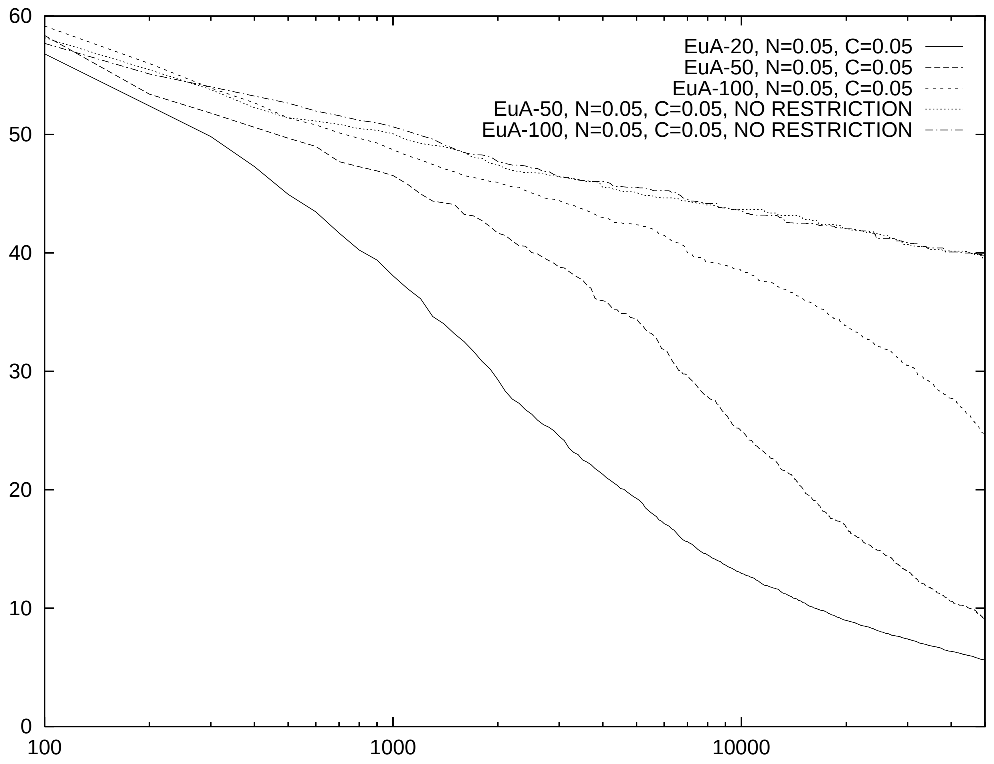

Figure 5.16: F32: Restriction Operator Tests. The loss of the restriction operator detrimented EuA performance, demonstrated a medium level of epistasis in the problem.

sampled was much greater than that found in F31 (and therefore the probability of replacing the current individual with a lower-fitness individual was much less).

A surprising result was that of the GAs, which initially performed almost as well as the EuA-20. Both GAs converged to good points, but not to the excellent points found by most of the other algorithms. One conclusion that may be drawn from the performance of IMHC and the GAs is that, in situations where random recombination is a good heuristic, highly exploitative mutation-based hill-climbing may be a better heuristic.

5.3.5 F33: Minimum Tardy Task Problem

Problem Description

This problem is a scheduling problem in which there are n tasks to be performed. Each task has an execution duration l_i, a completion deadline d_i, and a penalty p_i for late or non-completion. All these parameters are positive integers. For the solution to be feasible, no more than one task at a time may be scheduled, and every task must either complete before its deadline or remain unscheduled. A task that is scheduled but cannot be completed before its deadline is an "infeasible job". Note the feasibility of each job depends on the other jobs that are currently scheduled.

For these particular experiments, 100 tasks will be used. Each bit i of the genotype indicates whether or not task i is to be scheduled. A solution is feasible if no scheduled task's deadline is violated; this is, no infeasible jobs exist. The fitness function used to rank phenotypes was drawn from the genesys-2.0 package.

$$F33(\mathbf{x}) = \begin{cases} \sum_{i=1}^{n} p_i(1-x_i) & \text{when feasible} \\ \sum_{i=1}^{n} p_i(1-x_i) + \sum_{i=1}^{n} p_i x_i G_i + \sum_{i=1}^{n} p_i & \text{otherwise} \end{cases} \tag{5.11}$$

When genotypes are feasible, their phenotype is simply the sum of the penalties of the uncompleted tasks. Therefore smaller phenotypic values are associated with higher fitnesses. When genotypes are infeasible, G_i indicates which tasks violated deadlines. Therefore the second term of the infeasible fitness function sums the penalties all the late-completing tasks, and the third term is a constant "infeasibility penalty" that ensures that any feasible solution will have a lower phenotypic value than any infeasible solution. Complete details on this problem can be found in (Khuri et al., 1993).

$50,000$ samples were generated for the random sampling conducted for estimating epistasis and fitness variance. The measured epistasis was 0.50 and the fitness variation was 0.10. The average phenotypic value was 41582; the minimum value encountered in the random sampling was 8175 and the maximum was 57029. The overall phenotypic range of the problem is 200 to 10^5.

Results

The results of this experiment were very interesting, due to the deception present in the fitness function. IMHC, which had performed well on all the other experiments, failed relatively often in F33. Eight out of 100 runs of IMHC converged almost immediately to the "maximally infeasible" solution—the solution that schedules *all* jobs. IMHC troubles were caused by deception. In F33, solutions are infeasible when not all jobs scheduled can be completed before their deadlines. In order to make an infeasible solution "more feasible", the number of scheduled jobs must be decreased, or different sets of jobs must be substituted for the existing jobs. A penalty is assessed for jobs that are either unscheduled or late. However, the infeasible fitness function assesses equal penalties for both conditions, so unscheduling a job that cannot be completed on time (given the other existing jobs) results in *no change in fitness* if the solution still remains infeasible. So there is no positive feedback in the infeasible region for reducing the number of scheduled jobs. In addition,

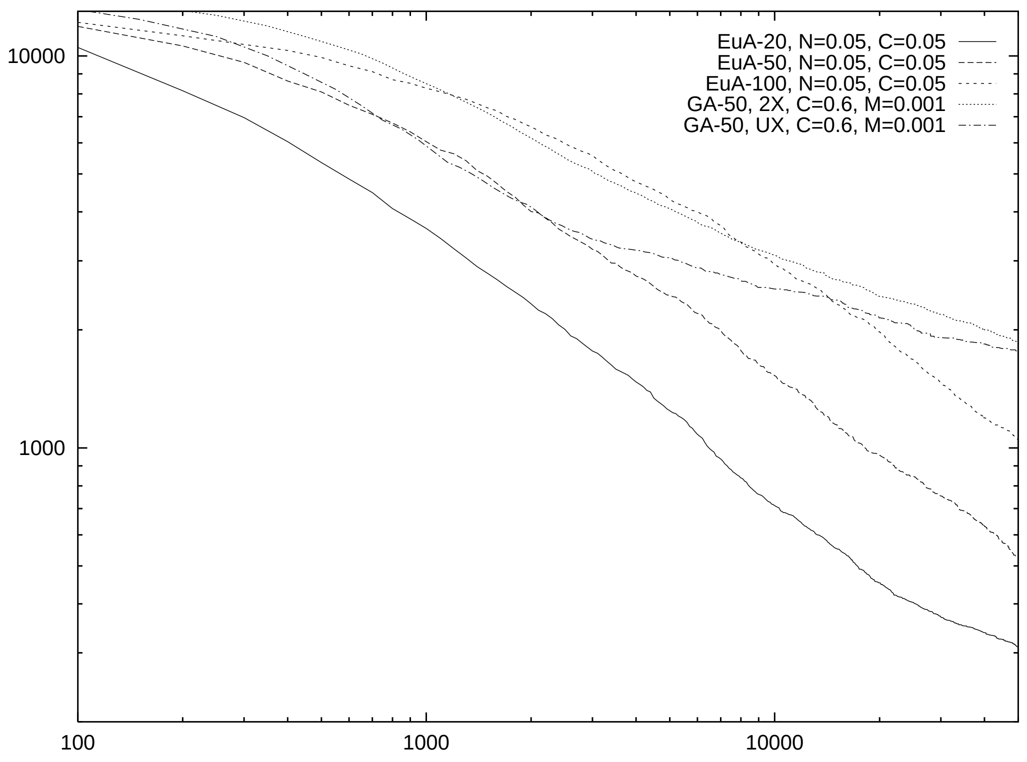

Figure 5.17: F33: Eugenic vs Pattern-Based Search. The GAs were initially able to converge at reasonable rates, but began to prematurely converge. Greater exploitation due to smaller population sizes allowed the EuA to progress faster.

negative feedback can arise, when an "easy" job is added to an infeasible solution. It may be possible to complete this easy job before its deadline, in spite of the fact that some of the already scheduled jobs cannot be completed on time. If this is the case, the penalty associated with this easy job is no longer assessed, and fitness *increases*. Therefore, the fitness of an infeasible solution is never decreased with the addition of new jobs, and it is even possible that adding jobs to an infeasible solution can increase its fitness, if the additional jobs are easy enough to be completed on time. So an optimization algorithm can increase the fitness of a solution by increasing the number of jobs scheduled, even when a solution is infeasible. This will make the infeasible solution even more infeasible. Fitness will eventually reach a local optimum—the maximally infeasible solution—when all of the jobs have been scheduled. This local optimum is very deceptive, as it has a very large basin of attraction, and its genotypic distance from the globally optimal solution is very large.

Highly exploitative ("greedy") hill-climbing algorithms that rely on small mutations (such as IMHC) will often converge to this local optimum. They may avoid this local

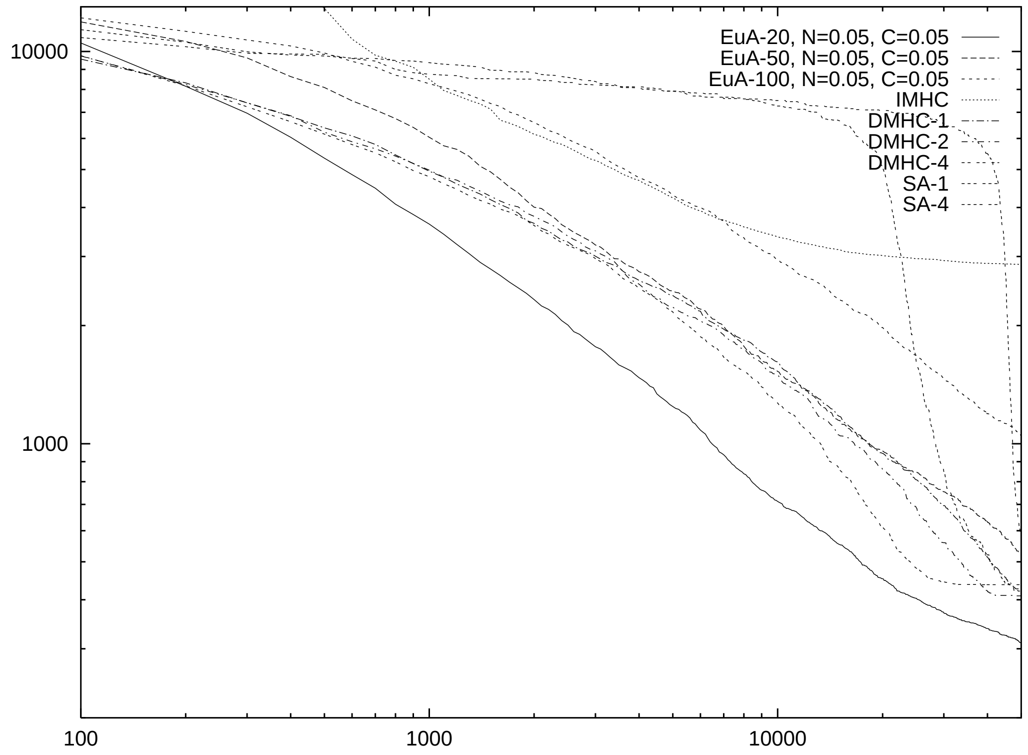

Figure 5.18: F33: Eugenic vs Mutation-Based Search. IMHC converged almost immediately to local optima on eight of its runs. Otherwise, it converged quickly to the global optimum. The shape of its learning curve demonstrates that simple averages of fitness versus time can be clearly misleading. SA initially converged very slowly, but then caught up with the other algorithms. DMHC had middling performance.

optimum only if search stochasticity increases—whether due to larger mutations or the probabilistic replacement of the current individual with lower-fitness individuals. All of the mutation-based algorithms have one or both of these qualities. DMHC and SA both start out with very large mutation sizes, and SA probabilistically replaces its current individual with lower-fitness individuals. IMHC increases its mutation size when it detects no increase in fitness after many mutation attempts. The relative performance of the mutation-based algorithms demonstrated the relative efficacy of their strategies in avoiding the local optimum and homing in on the global optimum. SA initially had a great deal of trouble, but was able to find points of average fitness by the end of its search. Most likely SA's initial trouble was due to the acceptance of infeasible points as valid replacements for a current feasible point. SA's nondeterministic acceptances of worse-fitness individuals once again prevented it from making much progress until this probability had decreased sufficiently.

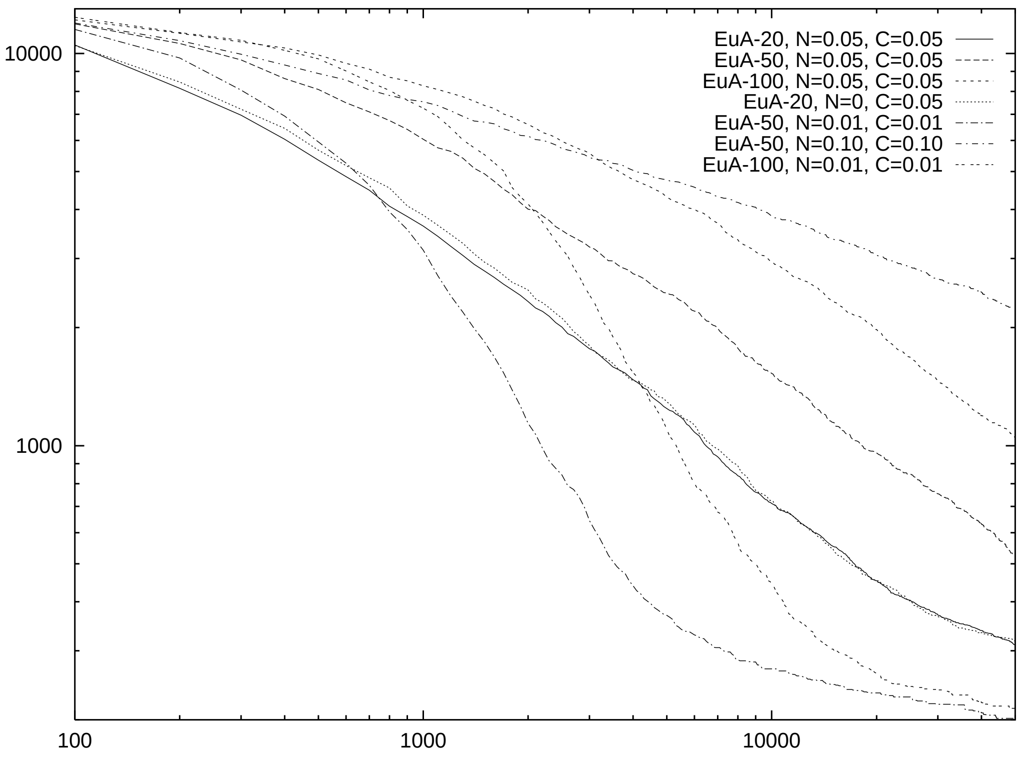

Figure 5.19: F33: Selection Noise and Creation Rate Tests. For the EuA-20, there was little difference in behavior when selection noise was changed. Lower selection and creation rates clearly aided EuA performance when population sizes were larger.

Once it had decreased to a critical point, SA was able to quickly catch up to DMHC, which had good overall performance. Both DMHC and SA consistently avoided the local optimum. However, they did not converge on the global optimum as skillfully as many of the EuAs. The behavior of IMHC was radically different from that of DMHC and SA. On eight of its 100 runs, IMHC converged almost immediately to the local optimum. On the other 92 runs, IMHC almost always found the global optimum. Therefore IMHC experienced more "extremes" in behavior than DMHC and SA. It is likely that the best runs of IMHC were those that either started in the feasible region or found the maximally infeasible point quickly. If IMHC started in the feasible region, there is no chance that it would stray into the infeasible region, as all feasible points had higher fitness than all infeasible points. If IMHC started in the infeasible region and found the maximally infeasible point, then it would also increase its mutation size at the greatest possible rate (since mutation size continues to increase until a point of higher fitness is found). If the local optimum was found early in its search, IMHC had a very good chance of escaping from the infeasible region.

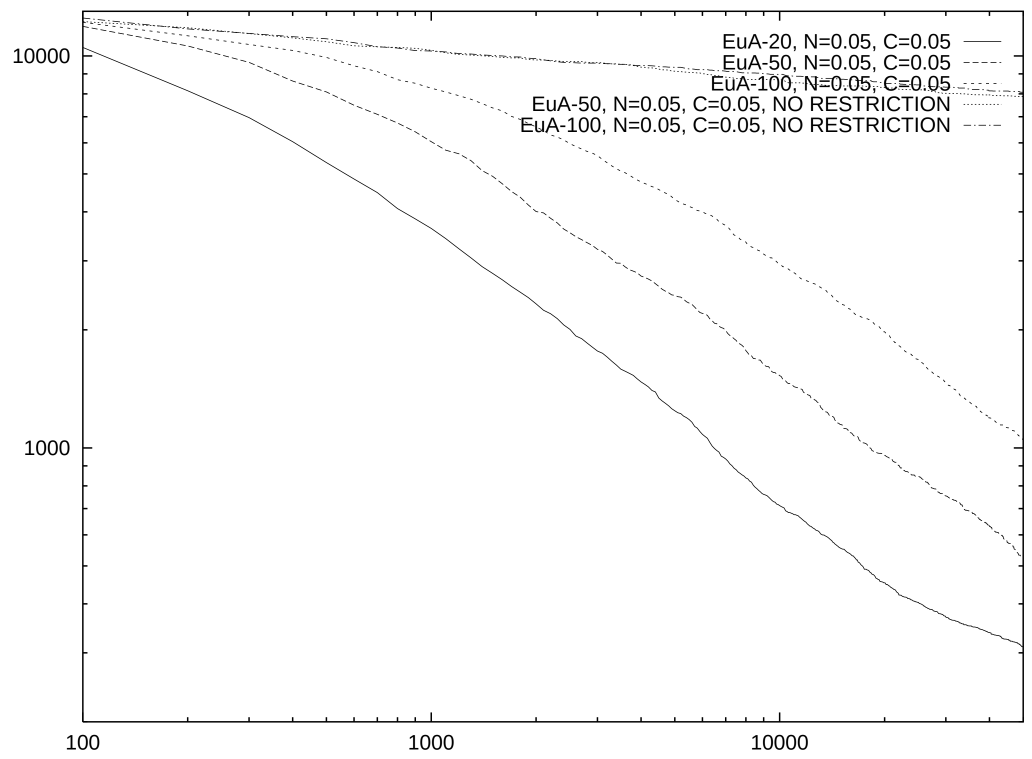

Figure 5.20: F33: Restriction Operator Tests. The EuA without restriction was not able to make good progress, once again demonstrating the presence of epistasis.

However, if IMHC took too long to find the local optimum, it did not have enough time to escape it. The behavior of IMHC, DMHC, and SA on F33 demonstrate their overall strengths and weaknesses. DMHC and SA were less likely to produce catastrophic results, but they were also less likely to produce excellent results. IMHC's behavior was much more "hit-or-miss", as it could consistently find optima, but the quality of these optima varied greatly.

The performance of the EuA was excellent. The worst EuAs faired about as well as the mutation-based algorithms, but the average and above-average EuAs performed much better. Once again, small population EuAs made faster progress than equivalent large population EuAs. Fitness proportionate exploitation seemed to be a good heuristic, as EuAs with lower selection noise and creation rates performed much better than those with higher stochasticity. Just as with F32, selection seemed to have no effect on the performance of the EuA-20. However, the combination of lower selection noise coupled with lower creation rates *greatly* increased the performance of the EuA-50 and the EuA-100. With 1.0% selection noise and creation rate, both the EuA-50 and the EuA-100 performed excellently. This would imply that, for this problem, creation rate has a greater effect on EuA performance than selection noise. The deception of the infeasible region was not a problem for any of the EuAs. Very little progress was made when not using restriction, indicating that the epistasis among the scheduled jobs was high—jobs had to be considered in groups, not just one at a time. The relative performance of the EuAs tested in this experiment demonstrated that the EuA can be an extremely reliable and more successful alternative to mutation-based algorithms.

5.3.6 F38: Simulated Annealing Test Function

Problem Description

F38 was a test function in (Ingber, 1993). The real-space encoding of this problem has an extremely large amount of local minima, for which small changes in the real parameters only decrease fitness values. Therefore algorithms that attempt to optimize this problem by using a simple gradient descent method in real space are almost guaranteed to converge to a local optima. F38 is generalized so that it can have anywhere from 2 to n real parameters. For the problem we shall study, four real parameters were used. Each of these four parameters was encoded into a 32-bit binary number, for a total of 128 bits. Therefore the genotype for this problem will be 128 bits, and the optimization algorithms tested will be attempting to find solutions in 128 dimension binary space. We shall see that most of the algorithms tested had very little difficulty solving this problem. Therefore, it would seem that the myriad local optima present in the real space encoding are "smoothed out" in binary space.

$$F38(\mathbf{y}) = \sum_{i=1}^{4} \begin{cases} (t_i \cdot sgn(z_i) + z_i)^2 \cdot cd_i & \text{if } |y_i - z_i| < |t_i| \\ d_i y_i^2 & \text{otherwise,} \end{cases} \tag{5.12}$$

$$z_i = \lfloor |\frac{y_i}{G_i}| + 0.49999 \rfloor sgn(y_i) G_i, \tag{5.13}$$

$$G_i = 0.2, t_i = 0.05, c = 0.15 \tag{5.14}$$

$$d_i = 1, 1000, 10, 100 \tag{5.15}$$

$$y_i \in [-1000.0, 1000.0] \tag{5.16}$$

64,000 samples were generated for the random sampling conducted for estimating epistasis and fitness variance. The measured epistasis was 1.00 and the fitness variation was 1.48. The average phenotypic value was 2.10946e+08; the minimum value encountered in the random sampling was 55115.7 and the maximum was 1.09611e+09. The overall phenotypic range of the problem is 0 to approximately $1.1 \cdot 10^9$.

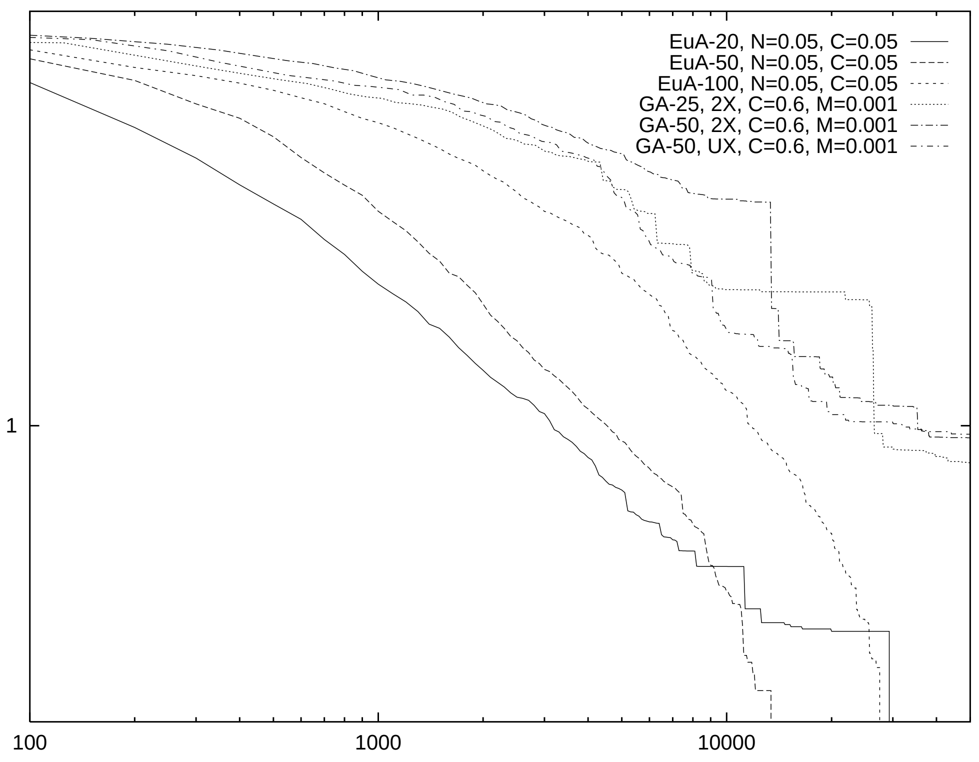

Figure 5.21: F38: Eugenic vs Pattern-Based Search. Smaller-population EuAs were able to make faster progress than larger-population ones, but lost this edge in the latter stage of the search. All of the EuAs were able to find the global optimum. All the GAs tested made slow and very discountinuous progress, and prematurely converged.

Results

Due to limited precision encoding, this problem actually had many global minima with phenotypic values of zero. The smallest non-zero phenotype representable in the experiments was 10^{-10}, and whenever the absolute values of all of the parameters was less than 0.05, the phenotypic value would be zero. As a result, the 32-bit binary encoding of the real parameters allowed a great many different mutation-paths to be taken to the minima, since there are many ways for each 32-bit group (y_i) to take on absolute values less than 0.05. The local minima that supposedly existed in the real-number encoding of the problem were simply not encountered. This is a clear case were the "genetic" representation of each part of the phenotype caused a great reduction in the problem's difficulty. Here are some examples of how $|y_i| < 0.05$ can be coded:

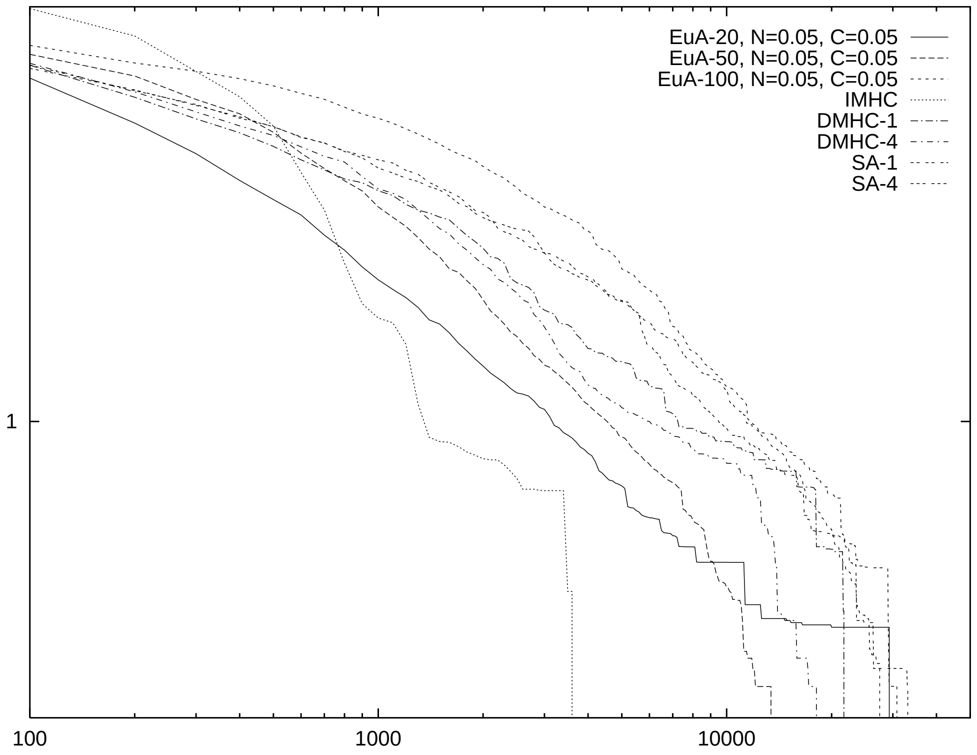

Figure 5.22: F38: Eugenic vs Mutation-Based Search. IMHC clearly found the global optimum very quickly. SA made reasonable progress, while DMHC faired a little better.

$$11110011001001001111111111111110 \quad (= -0.02612300) \tag{5.17}$$
$$11001000101011101111111111111110 \quad (= -0.01654050) \tag{5.18}$$
$$10010001111010011111111111111110 \quad (= -0.01245120) \tag{5.19}$$
$$10111101011011111111111111111110 \quad (= -0.00115967) \tag{5.20}$$
$$11011110001100110000000000000001 \quad (= 0.02429200) \tag{5.21}$$
$$01101011111100001000000000000001 \quad (= 0.03240970) \tag{5.22}$$
$$11010011011101101000000000000001 \quad (= 0.04364010) \tag{5.23}$$

The values of the 17 least significant bits in each 32-bit group did not seem to matter, so long as the most significant 15 bits had values of "000000000000001" or "111111111111110". Therefore the problem size was greatly reduced, as only $4{\cdot}15 = 60$ bits (instead of 128) were significant. As a result, only 2^{60} genotypes had to be searched, instead of 2^{128}. Just about every run of every algorithm was eventually able to find a phenotypic value of zero.

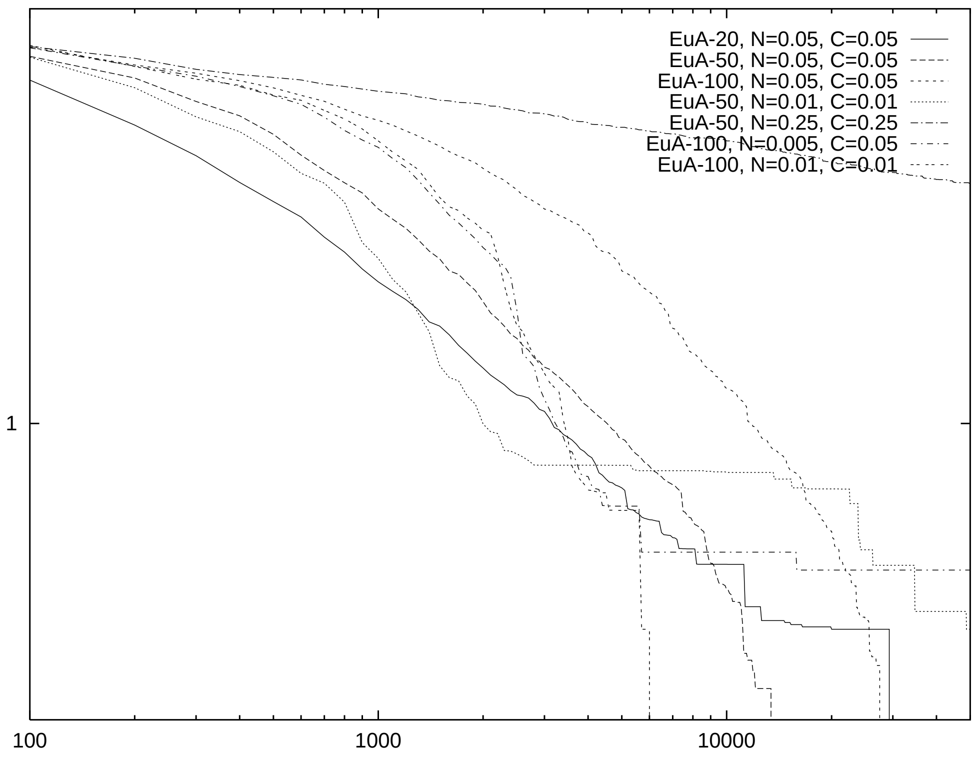

Figure 5.23: F38: Selection Noise and Creation Rate Tests. For the EuA-50, high stochasticity was obviously harmful, while lower stochasticity initially increased the convergence rate, but then caused premature convergence. An increased creation rate caused the EuA-100 to prematurely converge, but a low selection noise and creation rate caused the EuA-100 to converge very quickly to the global optimum.

Mutation-based search obviously was a good heuristic for optimizing F38. IMHC, the algorithm most prone to being trapped by mutation-based local optima, solved the problem the fastest. IMHC found a global optimum in just 3500 function evaluations, on average. DMHC and SA found optima in roughly 20,000 and 30,000 function evaluations, respectively. Therefore the extremely exploitative and small-mutating heuristic of IMHC was obviously a great match with F38. All the mutation-based algorithms exhibited no tendency to prematurely converge. IMHC's avoidance of local optima reinforced the supposition that the change of representation of the parameters of this problem (from real to binary inputs) removed any strong local optima.

The EuA performed well, even though some of the parameter combinations exhibited a tendency to prematurely converge on some of the runs. When selection noise and creation rates were low (1.0%), the EuA-100 beat all the other algorithms except IMHC, finding

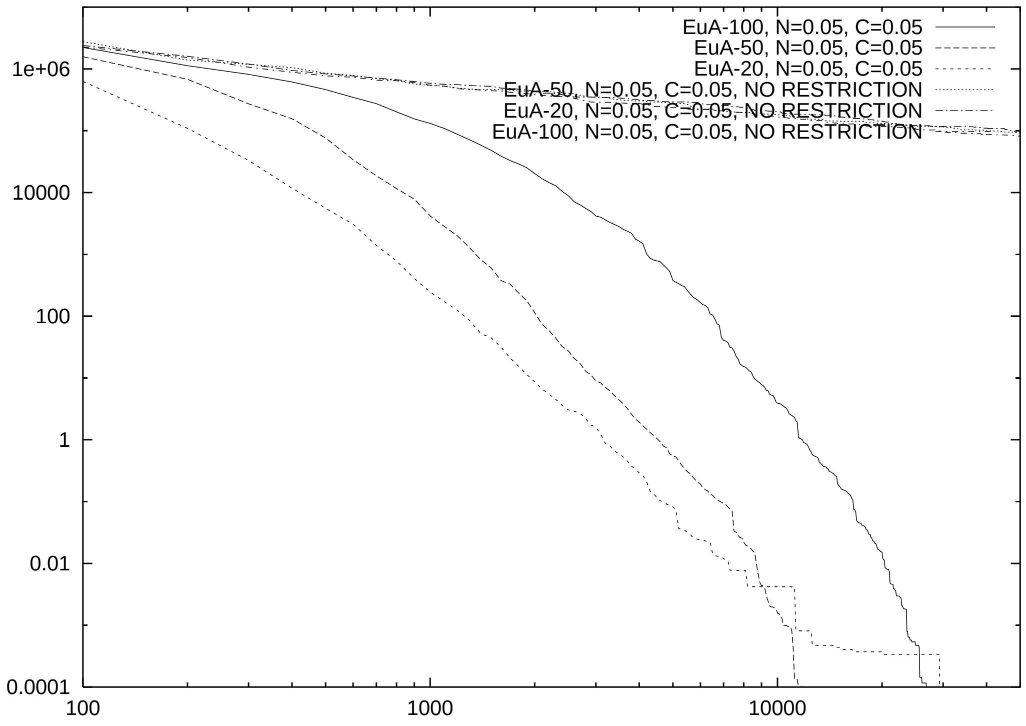

Figure 5.24: F38: Restriction Operator Tests. Restriction obviously aided the EuA in tracking and exploiting epistatic interactions.

an optimum with an average of 6000 function evaluations. Lower selection noise initially helped speed up EuA search, by helping the algorithms home in on very good regions of the space quickly. This is most likely because random disturbances to the selection of the most significant bits could easily result in wild changes in fitness. However, without greater creation rates or population sizes, premature convergence was a slight problem for EuAs with lower selection noise. Even so, for the EuAs that prematurely converged at least once, more than 95% of their runs did not prematurely converge. Creation rates could be set at lower levels for larger-population EuAs, since larger populations reduce the chance of getting trapped by local minima. Lower creation rates were necessary for larger-population EuAs, since high creation rates, coupled with the higher diversity of larger populations (which causes greater search stochasticity), caused the EuA-100 to have trouble homing in on the optimum.

For the EuA, the large variation in fitness values lead to large drops in diversity when "super" phenotypes were encountered. For example, when an individual of fitness 10^6 was found, while all the existing individuals had fitnesses hovering around 10^4, the super individual's fitness would completely dominate the statistics the EuA uses to generate new individuals. The population would then experience a large drop in diversity, as all the lower-fitness individuals would be replaced by new "copy-cat" individuals similar to the super individual. In spite of this, all the EuAs (even those that prematurely converged) were able to maintain high diversity levels throughout the search, as even the copy-cat individuals consistently differed from super individuals.

In the first one thousand function evaluations, the EuA-20 was a clear winner, far out-pacing IMHC and all other algorithms. However, its eventual performance was average. This may be because 20 individuals provided too little information on the 128-bit genotype, even when the population was highly homogeneous. The large search inertia of 100 individuals initially worked against the EuA-100, as after it found a super individual, which far surpassed all those in the current population, it had to "wait" for many individuals in its population to be replaced with the better individuals it was then able to create, before it could start producing many individuals like the super individual. Therefore, the rate at which EuA-100 could exploit information from new super individuals was much slower than the rate for the EuA-20 (in fact, five times slower). However, later on in the search, when many individuals in the EuA-100's (or the EuA-50's) population were fairly close to optimal, and the population was fairly homogeneous, the more accurate statistics garnered using the larger sample sizes could be used to great avail.

Without the restriction operator, the EuA was unable to make much progress at all, no matter what the population size. The restriction operator was obviously necessary, due to the high amounts of initial epistasis and the great number of phenotypically equivalent, but genotypic disparate, individuals. When many genotypically diverse individuals can represent the same phenotype, the restriction operator is necessary to prevent the incorrect mixing of very different schemata. Restriction greatly helped because of the large, competing schemata that were present. The previously-discussed 15-most significant bits "000000000000001" and "111111111111110" of each 32-bit group were a prime example of two very large competing schemata. Without restriction, the EuA would "average" these schemata together (just as the uniform-crossover operator of the genetic algorithm), thereby creating a very low-fitness individual. Larger population EuAs would just average more of these diverse individuals. Since even 100 individuals would still only be a tiny fraction of the search space, no particular advantage was granted to non-restricting EuAs with larger populations. The GA performed better than the non-restricting EuA, most likely because the random two-parent recombination that the GA used would eventually pair two compatible individuals. A smaller population size of 25 helped the GA, and UX seemed to work slightly better than 2X. But still, the GA performed badly relative to the other algorithms, and most often did not find an optimum.

In chapter 3, high epistasis was associated with highly difficult problems. The epistasis measured for F38 was $E = 1.0$. If epistasis is truly a measure of problem difficulty, then why did all the algorithms do so well? Most likely this is because a great number of the o(1)-competitions measured were between very insignificant bits, as less than half of the bits had to be set correctly around the optimum. Epistasis is high when the best choice of each gene's allele depends heavily on a simultaneous complementary settings of other gene's alleles. In high epistasis problems, there is no clear choice for the best setting of each gene, unless the rest of the genotype is known. Epistasis is low when each gene has a definite "best" allele choice, independent of the allele choices for the other genes. Therefore, when relatively few genes have clear "best" settings, while the a majority have no systematic effect on fitness (and therefore no observed "best" settings), it would seem that epistasis is high. There would be many o(1)-competitions that had no clear winners, and so it would appear that the allele settings were highly interdependent and therefore highly epistatic. This observation leads to the realization that functions with many insignificant parameters can appear to be very epistatic. Perhaps a measure of o(1)-epistasis that gave increased credit to highly significant bits and very little credit to genes with little effect on fitness would provide more accurate measurements of epistasis as a measure of problem difficulty.

5.4 Summary

The experiments illuminate many differences in algorithm bias as well as problem structure. IMHC was shown to perform well on non-deceptive problems but would often converge to deceptive optima. DMHC and SA were less likely to converge to deceptive local optima due to their relatively higher stochasticity (and therefore more exploration). However, this ability to avoid deceptive optima came with a high cost—DMHC and SA were also less likely than IMHC to converge to *any* attractor—local or global. Through recombination, the GA often performed well in the initial stages of the search, but, once good building blocks had been created, the GA mutation operator did not create enough exploration to find better building blocks.

F02 was a very nontypical problem, due to the extreme amounts of exploration needed to optimize it correctly. SA's better performance on F02, relative to DMHC, was probably only due to its different acceptance criteria, which allowed it to more readily avoid F02's local optima. In all the problems *except* for F02, DMHC performed better than SA, implying that the non-deterministic acceptance of lower-fitness individuals conducted by SA had only limited applicability. F02's excessive need for exploration was further evidenced in the EuA's behavior. In all the other problems besides F02, the EuA benefited from low stochasticity. However, for F02, the EuA-50 with the *most* stochasticity—a 25% selection noise and creation rate—was definitely the best performing algorithm, always finding the global optimum within 20,000 iterations. The greater success of this EuA versus DMHC and SA demonstrated that eugenic evolution is still very useful even when other algorithms' bias has been explicitly matched to a particular problem's structure. However, if too much faith is given to explicit measurements of allele distributions (i.e. when EuA selection noise or creation rate is very low), the attraction of large sub-optima's becomes too great and the EuA will converge to the wrong optimum.

F30 was a problem that was easily solved using exploitation. It demonstrated that the EuA can optimize simple problems as well as the other algorithms. The other test problems showed that the EuA can also optimize very difficult problems.

F31 demonstrated what happens when there is a "mismatch" between a problem's fitness variation and a particular algorithm. SA performed very poorly on F31 since its acceptance probability of points with fitness lower than its current point was almost always 1.0. Therefore SA never (until the *very* end of its search) exploited the mutations necessary to move out of F31's infeasible region. In spite of this, it is quite possible that SA's acceptance probability could be changed to combat this mismatch. However, this would still not alleviate SA's difficulty with F31, because SA's *bias* is mismatched with F31's structure! Algorithms optimizing F31 must *first* be very exploitative, and *then* become more explorative. SA's major design feature is for it to start out very explorative and then become very exploitative. Therefore no amount of parameter tuning would truly make SA's bias match F31's structure. DMHC and IMHC also suffered on F31, since they were not as able as the EuA to adapt their balance between exploration and exploitation.

On F32, even when IMHC found the globally optimal region far more quickly than the other algorithms, the EuA could still *eventually* find better points. And on F33, the EuA was clearly the best performing algorithm due to its ability to avoid the deceptive local optimum while simultaneously being able to home-in on the global optimum.

The F38 experiments demonstrated that a problem riddled with local optima in real-dimensional space doesn't necessarily have any local optima in Hamming space. They also showed that, even when IMHC quickly found global optima, a EuA-100 could find optima almost as quickly.

A surprising result of the experiments was that, on difficult combinatorial optimization problems, even a simple algorithm such as IMHC can perform far better than much more sophisticated algorithms. Each of the optimization tasks had "silver bullet"—an algorithm with performance much greater than the others. For two of the six problems (F32 and F38 - very low epistasis problems), IMHC was this silver bullet, even though it was developed for this thesis solely for benchmark purposes. In F32, IMHC found highly-fit points in 300 fitness evaluations that the other algorithms didn't find until after at least

3000 fitness evaluations. In F38, IMHC *always* found the optimum within 3500 fitness evaluations, while it took 6000 evaluations for the next best performing algorithm (the EuA-50) to achieve this feat. For three other problems (F02, F31, and F33), the silver bullet was the EuA. In F02, the EuA with a 25% selection noise and creation rate always found the global optima in 15,000 evaluations, while the second best algorithm took 25,000 evaluations. In F31 and F33, no algorithm *always* found the global optima, but the EuA found the global optima much more frequently than the other algorithms. The remaining problem (F30) was easily optimized by all the algorithms (except the GA), but the more highly exploitative mutation-based algorithms generally faired better.

The EuA proved to be an extremely robust, reliable and flexible combinatorial optimization algorithm, relative to the other algorithms tested. All the mutation-based algorithms (including SA) encountered "killer" problems, on which they performed very poorly. The genetic algorithm, the only pattern-based algorithm tested, consistently performed poorly, due to premature convergence caused by a lack of exploitation and mutation rate tuning. The EuA, the only eugenic algorithm tested, performed the best on several problems, but still performed very well in the others. In no case did a standard EuA perform poorly, although in several cases nonstandard EuA's (those with extremely high mutation rates, or those that did not use population restriction) performed poorly. The resistance of the EuA to premature convergence allowed it to continue to improve where other algorithms had failed. In most cases, the eugenic algorithm was still finding better individuals long after the other algorithms had converged to less-fit points.

More in-depth analysis of what these experiments illuminated about particular aspects of problem structure and algorithm bias will be included in the next section.

Chapter 6

Discussion and Future Work

The relative performance of the various algorithms on the test problems in chapter 5 demonstrated many of their strengths and weaknesses, in addition to illuminating much of the test problems' structure. This chapter will discuss the results of these experiments and outline possible future enhancements to the EuA and new directions of research.

6.1 The Eugenic Algorithm versus the Genetic Algorithm

In order to evaluate the EuA's abilities, many evolutionary algorithms were discussed and studied in this thesis. Of these, the genetic algorithm was the major inspiration for the design of the EuA. While the GA has repeatedly been demonstrated to be an excellent general optimization algorithm, it does not rely upon any explicit analysis of the correlations between schemata and fitness, and the experiments in this thesis showed that the GA can fall far short of an algorithm that performs such an analysis—the EuA. The EuA differs from the GA in every important aspect of genetic selection and construction. First, GAs do not select alleles only on the basis of fitness of the individuals containing them, but also on the *frequency* of the alleles in the population. It is therefore possible for the GA to choose alleles from lower fitness individuals *more frequently* than alleles from higher fitness individuals, if the ratio of the numbers of lower fitness individuals to higher fitness individuals is high enough. In contrast, the EuA allocates trials to alleles based *directly* upon the average fitness of the individuals containing them; it makes no difference how often the individuals are found in the population. Therefore even a single individual can greatly aid allele selection, even if this individual does not have an extremely low or high fitness. Second, the GA combines randomly selected genotypic parts to form new individuals, while the EuA deliberately promotes genetic combinations that have proven themselves in the past, using explicit analysis of conditional fitness distributions. As a result, the EuA more effectively and intelligently exploits the information at it disposal. And third, while the GA makes no use of differing gene importance, the EuA specifically attempts to optimize genes in order of their significance and therefore tunes its solutions to regions of the search space that are determined by the most significant genes. This ability of the EuA is a direct result of its restriction operator and its specific ordering of gene optimizations (from genes of greatest importance down to genes of least importance). With this ability, the EuA can accurately explore and exploit optima, without having to wait for its population to converge into (and possibly become trapped in) the optima's basin of attraction. For the GA to thoroughly explore a particular region of the search space, its population must be significantly composed of individuals from this region. But such GA populations lack genetic diversity and therefore become highly susceptible to premature convergence to this region.

It is never necessary to preserve schemata when optimizing non-epistatic problems. However, it is obvious that any powerful combinatorial approximation algorithm must have some method of preserving the epistatic interactions in schemata even on low-epistasis problems. It is typically thought that one of the major benefits of the recombination operator is that it preserves and promotes these epistatic interactions, by allowing differing solutions to share schemata. However, although the GA recombination operator may appear to be *helpful* for epistatic problems, this does not mean that recombination is *necessary* for these problems. For example, the GA outperformed the non-restricting EuA on problems F02 and F30, demonstrating that random recombination can be beneficial for such epistatic problems. However, the mutation-based algorithms, which make no effort to combine differing schemata, consistently outperformed the GA. This cast doubt on whether it is necessary to *share* epistatic allele groups among solutions, and therefore whether recombination is necessary. For instance, DMHC and SA optimized both F02 and F30 much more often than the GAs tested. This result would suggest that recombination is unnecessary for *both* epistatic (F02) and non-epistatic (F30) problems. Mutation-based algorithms preserve and exploit epistasis by other means. When a mutation-based algorithm changes only some of its current individual's genes, all the epistatic interactions that are present in the unmutated genes are preserved. Therefore mutation-based algorithms need not resort to recombination, since they already have a method for exploiting significant epistatic combinations.

6.2 Problem Structure

This section will discuss what was learned about the structure of combinatorial optimization problems through the use of both the new measurements developed in chapter 3 and empirically through the experiments.

6.2.1 Fitness Variation

The one method used to calculate fitness variation, $F = \frac{S_b}{S_w}$, was successful at finding the two problems that had large variations in gene significance. F was relatively high (above 1.47) for F02 and F38, and relatively low (less than 0.21) for the other four problems. F02 and F38 were the only two problems whose genes had a definite hierarchy of significance, since they were the only two problems in which real number parameters were encoded as binary parameters. All the other problems had genotypes that represented set membership of one sort or another, where all genes had equivalent coding significance (although not equivalent functional significance!). This result suggests that $F = \frac{S_b}{S_w}$ is a very good measure of the variation in gene significance. However, the use of F as a measure of global fitness variation turned out to be unreliable. The reason is that the distribution of the sample points used in its calculation can be very unrepresentative of the entire distribution. In F31, for example, the points encountered in the random sampling represented less than 0.34% of the range of the full search space. Therefore the fitness variation of the random sampling was entirely unrepresentative of the *global* fitness variation of the function, even though the statistics gathered were a good approximation of *typical* distribution. In addition, F seemed to have little predictive value when used to determine which of the optimization algorithms would perform better. This was because F makes no attempt to estimate local optima and deception, which both seemed to have a much larger influence on algorithm performance than fitness variation.

6.2.2 Epistasis

The epistasis measure E (paired χ^2-epistasis) turned out to be an ineffective measure of epistasis. A very effective measure of epistasis was the relative performance of EuAs with the restriction operator versus those without. Paired χ^2-epistasis had little predictive use for determining problem structure or algorithm performance. There was no apparent connection between the E measured for a particular problem and the relative performances of the different algorithms. In spite of this, we could still conclude that epistasis definitely exists and is measurable through other means. For instance, there was a vast difference in performance of the EuA with and without the restriction operator. EuAs that did not use restriction ignored epistasis. Removing the restriction operator from the EuA ruined its performance on F31, F32, F33, and F38, indicating that these problems had high levels of epistasis, at least in regions around the optima. However, on F02, non-restricting EuAs actually performed better than most EuAs that used restriction. The less-than-impressive performance of restricting EuAs was due to their proficiency in recombination, which allowed them to quickly find and exploit strong *sub*optima and lead them to prematurely converge. Non-restricting EuAs found F02's global optimum more frequently because they did not attempt to associate interdependent alleles; they only exploited global gene significance and were not attracted to the epistatic allele combinations of local optima. However, the restriction operator was still very useful if search randomness was high enough, as demonstrated by the fact that the EuA-50 *with* restriction and a 25% selection noise and creation rate easily outperformed *all* the other algorithms tested on this problem.

Therefore the exploitation of epistatic combinations always seems to be useful for epistatic problems, even when there are many strong local optima. Non-restricting EuAs do not maintain any epistatic interactions, and therefore are of limited use for optimizing epistatic problems, but it is this failing that allows them to be used to estimate the amount of a problem's epistasis.

6.2.3 Local Optima and Deception

Deception is increased by larger genotypic distances between local optima and global optima and larger/stronger suboptimal basins of attraction. Local optima played a large role in two of the problems, F02 and F33, but F33 was a much more deceptive problem than F02, since the local optima in F02 did not lead algorithms *away* from its global optimum; they merely caused algorithms to end their search fairly close to the optimum. For example, F02's local optima differed from its global optimum by about 18 bits, on average. In contrast, F33's local optima were radically different from the global optima, and therefore highly deceptive. F33's major local optimum (the maximally infeasible solution) differed from its global optimum by more than 40 bits. Therefore F33's local optimum was much more misleading (deceptive) than F02's. In addition, F33's local optimum was much stronger than the local optima of F02 - the basin of attraction of F33's local optimum was the *entire* infeasible region[1]—much larger than the basins of attraction of the local optima in F02. Therefore the attraction of F33's local optimum was much greater than that of F02's local optima.

Overall, IMHC's performance was a good indication of deception. IMHC performed well on all the problems except F02 and F33 (the two most deceptive problems), as it frequently converged quickly to the sub-optima of both F02 and F33. The hypothesis that F33 was more deceptive than F02 was reinforced by the fact that IMHC prematurely converged much more often in F33 than in F02, and therefore F33's local optima's attraction was stronger than that of F02.

[1]The basin of attraction did not include the infeasible points on the boundary of the feasible and infeasible regions—infeasible points that were one bit away from being feasible.

6.3 Eugenic Algorithm Parameters

The experiments provided a wealth of information on how to the EuA's parameters affected its behavior. This section will discuss the effects of population size, selection noise, and the restriction operator on the performance of the EuA.

6.3.1 Population Size

The experiments clearly demonstrated the effects of population size on the EuA. Smaller EuA populations were clearly beneficial to quickly finding very good points. For every problem except F02, the EuA-20 was initially the fastest eugenic algorithm. However, larger population EuAs were more easily able to avoid local optima, and often final performance depended on this. F02 was obviously a problem with extremely many local optima, and there was a direct increase in asymptotic performance when population size was increased. Larger populations performed better on F02 due to their decrease exploitation. F30 and F38 had somewhat confusing results about the effect of population size on eventual EuA performance, with the EuA-50 beating both the EuA-20 *and* the EuA-100 on these two problems. This could be because the EuA-50 struck the right balance between exploration and exploitation for these two problems. F30 and F38 were the easiest problems tested, as most runs of all the algorithms found global optima within 50,000 function evaluations, and since IMHC did so well on both of them. Therefore, the middling size population of the EuA-50 might never be that useful, because the problems for which it helps can be more easily solved by other algorithms. For the other three problems (F31, F32, and F33), smaller population size was definitely a great benefit (at least up through 50,000 function evaluations). However, these three problems were so large and difficult that most EuAs had not begun to converge, and so the asymptotic relationship between performance and population size was not fully established. It could have been the case that the greater initial performance of the small population EuAs was once again eventually defeated by the final performance of the larger population EuAs. If this were the case, then a strong argument for EuAs that increase their population sizes as the search slows could easily be made.

It would seem that by increasing the size of an EuA's population, allele-fitness distributions could be more accurately approximated. Larger populations can support more diversity, and so larger populations would be able to contain more representative samplings of the search space than smaller populations. More accurate statistics could be collected using larger populations, and an EuA armed with these statistics should easily be able to identify the best alleles and quickly generate highly-fit individuals. Therefore EuAs with larger populations could conserve precious function evaluations by avoiding the needless generation of low-fitness individuals. However, three key issues prevent this line of reasoning from being entirely true—immense problem search spaces, search focus “inertia”, and (most importantly) epistasis. Because of these issues, increasing the population size does not necessarily improve the performance of the Eugenic Algorithm. These issues will now be discussed.

Problem Size and Sampling Error

The search space of the typical combinatorial optimization problem is usually far too large for statistically representative samples to be taken. Even for a fairly small, 50-bit problem, a standard GA population size of 100 samples represents only one ten-trillionth of the search space (($2^{50} \approx 10^{15}$; $10^{15}/100 = 10^{13}$). Even a huge population of 10^6 individuals is still only one-billionth of the search space. Since the sample size of any reasonably sized population would be infinitesimal compared to the problem size, statistics collected using such populations will be highly unrepresentative of the search space as a whole, as such statistics would most likely be highly abnormal with respect to the global distributions of the search space. This effect is called “sampling error”. Normative inferences (such as which alleles are the “best”) made using such statistics are prone to error. As a result, statistics computed over typical population sizes (from one to several hundred individuals) can only serve to point to *better* regions of the search space, instead of pointing to the *best* solutions. Only through the repeated application of its search operators will an algorithm be able to “hill-climb” or “boot-strap” to better regions of the search space. It is therefore it is unreasonable to expect that simply increasing population size (even by huge proportions) will increase the accuracy of the collected statistics to such a point that the search becomes trivial.

Search Focus Inertia

Most of the problems studied in the experiments demonstrated that larger populations resulted in slower search than smaller populations. This is because EuAs with smaller populations have lower “search inertia” than those with larger populations - in general, the statistics of smaller populations can change more rapidly than larger populations. An individual in a small population has a larger effect on the statistics of its population than an individual in a larger population. Fewer replacements of individuals are necessary to radically change the search focus of a smaller population. In contrast, the effect of each new individual is less pronounced on the statistics of a larger population. As a result, each time a point of relatively high fitness is encountered, EuA’s with smaller populations can more quickly exploit the new information, while in larger populations, the overwhelming numbers of existing individuals drown out the newly gained information.

Epistatic Interference

For non-trivial problems, the statistics collected over the population will contain little useful information about the entire problem space; their purpose is to provide a *direction* for the search to proceed in—"where to go next". When the region searched contains intermediate or high levels of epistasis, increasing the population size can actually detriment the search, since a larger population will likely contain more *conflicting* instances of interdependence among allele fitnesses than a smaller population. Sampling error can therefore be *beneficial* to the search! For instance, a larger population will be more likely to contain individuals from differing optima's basins of attraction, and therefore the simple single-allele statistics collected will be more "confused" than those of a smaller population, which would be more likely to contain individuals from only one optimum's basin of attraction. As a result, when epistasis is high, statistics of large populations will no longer point to better regions; they will just be oversimplified descriptors of the current population. Only through the use of the restriction operator can the complex schemata dependencies be untangled, as the restriction operator allows the identification and computation of the most important conditional fitness distributions. After couple of restrictions, population size is reduced enough that interference is no longer a problem. Therefore, epistatic interference due to overly large populations may slow down searches, but most likely will not cause premature convergence since restriction allows a large population to act like a small population. This is evidenced by the very good performance of the EuAs with the restriction operator, and the relatively poor performance of those without.

Exploration versus Exploitation: The "Best" Population Size

Although the lower search inertia of smaller populations may often be helpful, EuA's with smaller populations were more likely to be trapped in local optima than larger population EuA's. As stated previously, a few individuals will not influence larger population EuA statistics as much as they would influence smaller population EuA statistics. Therefore the larger EuAs continue to perform high amounts of exploration even after finding very good individuals. Conversely, smaller populations will be more exploitative of the information contained in the high-fitness individuals. Since larger populations perform more exploration and less exploitation than smaller population EuAs, they are less likely to converge to sub-optimal points. Larger populations increase the EuA's resistance to local optima by decreasing exploitative behavior, but this comes at the cost of increasing the time it takes for the EuA to take advantage of information leading to global optima. As a result, the "best" population size for a EuA would be the smallest one possible that can avoid local optima. Such a population would rapidly respond to new information, without the risk inherent in exploiting harmful information that leads to sub-optima.

6.3.2 Selection Noise and Creation Rate

Just as smaller populations often sped up initial performance of the EuA, so did lower search stochasticity in the form of lower selection noise and creation rates. In most cases, lower search stochasticity aided eventual performance more often than smaller population size, but in other cases (particularly for F02), lower search stochasticity caused premature convergence just as often as small populations did. Once again, there is a clear relationship between exploration and exploitation: greater exploitation aided in all problems except for the one (F02) that was specifically contrived to defeat exploitative algorithms. More study is needed to fully establish the relationship between premature convergence, selection noise, and creation rate.

6.3.3 The Restriction Operator

As was seen in the F02 experiments, the use of the restriction operator could increase the possibility of getting trapped in local optima, since its major effect is to cause the EuA to fine tune the genotypes generated for a particular region of the space. This allows the EuA to quickly home-in on optima, but it also increases the EuA's vulnerability to local optima. Without the restriction operator, EuAs become very explorative, completely ignoring the epistasis information contained in the conditional fitness distributions. This resulting increased search stochasticity causes the non-restricting EuA to avoid all but very large largest basins of attraction. That is why the non-restricting EuA was able to outperform all but the most stochastic restricting EuAs. In general, the relationship between the size of an attractor's basin of attraction and the attractor's optimality will determine a non-restricting EuA's success. If very bad attractors have very large basins of attraction, then a non-restricting EuA will have an advantage against EuA that use restriction, unless the restricting EuA also have very high settings of selection noise and creation rate. Since EuAs using the restriction operator, in one form or another, beat EuAs without the restriction operator, a profitable approach to approximating a problem with unknown properties would be to just vary a restricting EuA's selection noise in a wide range, from 0% to 25% for example. The only usefulness in employing a non-restricting EuA would be to gauge the amount of epistasis or the prevalence of large sub-optima in the problem.

6.4 Extensions and Future Work

This section will describe three major areas of EuA enhancement. These include automatic parameter tuning, more effective statistical descriptions of epistasis and schemata fitness, and finally the extension of the EuA to continuous optimization.

6.4.1 Adaptive Parameters and the "Multi-EuA"

The hit-or-miss performance of SA, compared to DMHC, demonstrated that even very small changes to an algorithm can cause huge differences in performance. This phenomena was also exhibited by the EuA, for which there usually was a "best" combination of parameter settings that yielded greatly superior performance. This section will discuss how the EuA could determine its own most effective parameter settings.

The parameters of the EuA determine its bias. Lower population size and selection noise cause the algorithm to be more exploitative, while large populations or high selection noise cause it to become more explorative. An exploitative algorithm is more likely to find optima faster than an explorative algorithm, but it is also more likely to become trapped by local optima, if they exist. In order take best advantage of the strengths of both exploitation and exploration, the EuA could automatically adapt its search bias to respond to its pace of improvement. In effect, such an algorithm could "change" its bias in mid-search when not enough progress is made in a certain amount of function evaluations. Determining whether more exploration or exploitation would be fairly straightforward. If little progress was being achieved, and newly created individuals were not genotypically "different enough" from existing individuals, then exploration could be increased by enlarging the population and increasing selection noise. Conversely, if new individuals were too genotypically "different" from existing individuals, exploitation could be increased by shrinking the population by removing the worst individuals deterministically or by using (un)fitness proportionate selection. In addition, selection noise could be decreased to increase the exploitation of higher fitness alleles and schemata.

Although straightforward, this scheme for adapting EuA parameters faces several major problems. First, the relationships among exploration, exploitation, population size, and selection noise do not always follow concrete patterns. For example, increasing population size may not always increase exploration. Second, defining genotypic similarity is highly problematic. Would the similarity measure depend on gene significance? Would new individuals be compared against the best existing individual, or some sort of genotypic "average" of the existing individuals? Fortunately, both of these problems can be completely avoided by the use of multiple EuAs running in parallel—a "Multi-EuA". Each EuA would have a distinct combination of parameter settings. EuAs that were not performing as well as their counterparts would modify their parameter combinations in order to become *more like* those combinations found in the better performing EuAs, while the better performing EuAs would modify their parameters in order to become *more unlike* the under-performing combinations. For example, if a EuA with a population size of 20 was outperforming its competitor whose population size was 100, then it could decrease its population size to 15 and the competitor could decease its population size to 50. If further search showed that the larger population size EuA was outperforming the smaller one, then the smaller one's population could increase its population to 30 individuals and the larger population could increase to 60 individuals.

For each parameter that is to be adapted, there would be a pair of EuAs—one with a low parameter setting, and one with a high parameter setting. Therefore if both population size and selection noise were being adapted, four different parameter combinations would be tested, and therefore four different EuAs would compose the Multi-EuA. The four parameter combinations would be: low selection noise and small population, high selection noise and small population, low selection noise and large population, and high selection noise and large population. Then, for example, if the average performance for large populations was greater than that of small populations, then all population sizes would be increased. The same type of analysis and modification would be performed for selection noise. In this way, the Multi-EuA could adapt EuA parameters to exactly the right combinations at the right time.

A Multi-EuA could be implemented using only a single population. For instance, if selection noise was being tuned, *two* individuals could be created each iteration. The first individual would be created using a low selection noise and the next would be created using a high selection noise. A record would be kept of which selection noise setting resulted in the most improvements, and the two selection noise settings could then be changed accordingly. The same EuA could simultaneously optimize for population size, by creating individuals using smaller and larger subsets of the complete population. In this case, four individuals would be created each iteration.

Since the number of combinations of parameter settings doubles with each additional parameter, it is imagined that only two or three parameters would be adapted at a time (resulting in four or eight separate EuAs, respectively). In addition, each additional parameter would roughly double the number of function evaluations necessary. However, in light of the fact that many parameter combinations *never* achieve the same levels of performance as others, the cost of the additional function evaluations would come with the benefit that the parameters would always be tuned to roughly the right combinations. Therefore a Multi-EuA would be much more likely to find the very best individuals possible than a sequence of runs of individual EuAs with fixed parameters. Whether the value of a more robust, better optimizing search outweighed the cost of a more extensive search is a question left to the practitioner.

The concept of the Multi-EuA could be easily extended to other algorithms for that parameter optimization is desired, resulting in Multi-IMHC, Multi-SA, etc..

6.4.2 The Statistics used by the EuA

The statistics calculated by the EuA determine its success, and the composition of the population (whether restricted or unrestricted) determine the statistics. This section will discuss how these statistics might better calculated, or how the composition of EuA populations can be altered to generate different and/or better statistics. These modifications would most likely yield greater improvements than simply replacing fitness-proportionate selection with max-fitness proportionate selection or some other similar simple method of allele selection.

Information-Theoretic Measures

Instead of using "traditional" statistics to quantify the relationship between schemata and fitness, information-theoretic analysis could be used to determine the important quantities used by the EuA, such as the probabilities of selection and restriction. For example, the amount of mutual information between particular alleles and fitness values could be used to determine gene significance or to measure the level of epistasis. This could be accomplished by calculating the allelic entropy (the average amount of information conveyed per allele) present in the current population. Information-theoretic analysis may lead to a more disciplined approach of generating and analyzing algorithm heuristics and bias, not only for the EuA but for other algorithms as well. For a more complete discussion of information-theoretic measures and machine learning, see (Haykin, 1994).

The mention of information-theoretic measures brings to light a different view of the EuA's restriction operator. The restriction operator causes the EuA to probabilistically traverse paths of decision-tree separating high fitness individuals from low fitness individuals. The EuA basically reconstructs one path (from root to leaf) of a decision tree each iteration. The nodes of this decision-tree correspond to genes, and the branches from each node correspond to alleles. Nodes higher in the tree have greater gene significance, and branches weighted by allele fitness. Since gene-significance varies across the search space, the use of a dynamic tree structure is necessary. Due to the similarities with between the EuA and decision trees, improvements to the EuA might be gained by a greater integration of the two methods.

The Unrestriction Operator

When restriction causes the restricted population to become "too small", an operator complementary to the restriction operator, the "unrestriction operator", could be used to add individuals from the unrestricted population back into the restricted population. How small is "too small" could be determined by the information content of the restricted subset. For example, the unrestriction operator could be used when a large drop in gene significance or allelic diversity was encountered. The use of the unrestriction operator would help prevent the EuA making decision based on too little information.

Replacement Policy and Learning for Negative Examples

The standard replacement strategy of the EuA is to always replace the worst existing individual with the newly created individual, regardless of the fact that the existing individual may have a higher fitness than the new individual. Furthermore, in all of the optimization problems studied, the average fitness of new individuals generated by the EuA was extremely low relative to the average fitness of individuals already in the population, As a result of these two facts, the EuA population would *almost always* contain one "low outlier"—an individual whose fitness was much worse than the mean population fitness. This individual would heavily bias the allele-fitness statistics by significantly detrimenting the average fitness of any allele found in this individual. Since new individuals are more likely to share alleles with the best individuals than with the average or worst individuals, and since these new individuals were frequently *extremely* low fitness (low outliers), it was often the case that a new individual would severely detriment the statistics of many of the alleles found the in *best* individuals! As a result, alleles in the best individuals would randomly have extremely low average fitnesses, while alleles found in average individuals would remain unaffected by the low outlier. The EuA would then select against the set of alleles shared between the best individuals and the low outlier, and thus the next child produced would have a lower probability of having these alleles. But for the EuA to find good solutions, alleles found in high fitness individuals should not be selected against! Two solutions to this problem are immediately obvious: either use statistics that ignore the low outliers altogether (as X_n-selection does) or use the low outliers to the EuA's advantage. Next, we will discuss a method for implementing the second solution.

The GA implicitly learns from negative samples by decreasing the rate at which low fitness schemata are sampled. In contrast, PBIL learns explicitly from negative examples. In PBIL, only the alleles that *differ* between the very worst and best individuals are selected against, while alleles that are *shared* remain unaffected by the information gained from the low fit individual (Baluja and Caruana, 1995). In order to identify low fitness alleles, it is not necessary to explicitly search for alleles shared among the best and worst individuals. In fact, it is not necessary to depart from the standard statistical basis of the EuA at all. Only the replacement procedure of the EuA needs to be slightly modified to make this search automatic. The normal replacement procedure is to replace the worst individual in the population with the newly created individual. As explained above, most new individuals are "low outliers", and so most often the new individual immediately becomes the lowest fitness individual in the population—a new low outlier replaces the existing low outlier. However, if any new individual is protected from replacement for one cycle, then the EuA must replace the second-most low fitness individual in the population when the next individual is created. This next created individual, like the previously created individual, will probably have much lower fitness than most of the other individuals in the population. Therefore, the population will most often contain *two* very low fitness individuals, by simply protecting individuals from replacement for one cycle after their creation. By increasing the minimum "lifespan" of an individual - i.e. the number of cycles a new individual is allowed to exist without being replaced—a "replacement queue" consisting of very low fitness individuals is created.

The replacement queue could have a very beneficial effect on the EuA. By maintaining several low fitness individuals in the population, it would become much easier for the EuA to distinguish bad alleles from good alleles through the use of simple averages. When only one very low fitness individual was contained in the population, *any* allele in this low outlier would be selected against. But as more low outliers are added to the population, the number of alleles shared among *all* of them decreases quickly. It becomes more apparent what alleles are "really" responsible for the extremely low fitnesses of the low outliers. Alleles that differ among bad individuals are all penalized roughly equally, and so their selection probabilities are relatively unaffected. Only those alleles that demonstrate repeated, consistent very low fitness would be heavily selected against. It would still be possible that alleles found in highly fit individuals would be selected against, but this would happen only when *all* of the low outliers share this allele with the highly fit individual.

The size of this replacement queue may have to be adjusted when the problem size changes. As genotypic length increases, the less likely it is for two individuals from a diverse population to share a fixed percentage of their alleles. It becomes increasingly difficult to differentiate "bad" alleles from "good" ones as the number of schemata matching each individual increases. Because of this, it may be necessary to increase the size of the protection queue (in addition to the size of the population) when the number of genes in each individual increases.

It is important to note that the use of a replacement queue would probably be a superior strategy to simply enforcing a policy of elitism—i.e. disallowing an individual's entrance into the population if that individual's fitness is less than the worst existing individual. Elitism would prevent low outliers from entering the population, and therefore the population would remain static for many iterations, since most new individuals are low outliers. Such elitism does not have the benefit of allowing the EuA to learn from negative examples, and could greatly increase the chances of premature convergence.

6.4.3 Continuous Optimization

The EuA in this thesis was specifically designed to optimize combinatorial problems, for which each gene is a discrete variable. The EuA could be enhanced to attack continuous or mixed optimization problems, where at least some of the genes are real variables. To make the EuA suitable for continuous optimization tasks, several changes would have to be made. The allele-fitness computations would have to be modified, since there would be an infinite choice of allele values for each real-valued gene, as opposed to finite number of choices for combinatorial optimization. A clustering technique could be used to find regions of high fitness (and therefore high selection probability) in each allele's observed range of values. The determination of gene significance would have to be modified to more efficiently gauge a gene's effect on fitness. Gene significance could be based upon cluster distinctness, simple linear correlations, or more complex measures such as factor analysis (Haykin, 1994). Modifications would also have to be made to the restriction operator. Currently, the restriction operator simply searches for individuals containing a particular allele to include in the restricted population. For continuous genes, the restriction operator would have to be given the ability to identify allele values "close to" the target allele value; this could be accomplished simply by setting a distance threshold between the allele value of the individual under consideration and the target value, or by using a "restriction probability" that varied with the difference between the two values.

Chapter 7

Conclusion

No single algorithm can be superior in performance to all other algorithms over all possible problems (Radcliffe and Surry, 1995), and therefore algorithms must be designed with particular specializations in mind. It was shown in the experiments that the EuA's major design goals allowed it to perform reliably on a variety of combinatorial optimization problems, ranging from medium to extreme difficulty. Its robustness, coupled with the success of its explicit design considerations, has advanced the state-of-the-art in combinatorial optimization. No attempt was made to evaluate the EuA's performance on trivial problems, such as BIT-COUNT, since this type of problem can easily be solved by such simple algorithms as IMHC[1].

Using "standard" parameter settings, the EuA consistently performed very well across all the problems, and, on some of the problems (F02 in particular), the EuA performed the best out of all the algorithms even when it used "extreme" parameter settings. The EuA's flexibility was most obviously demonstrated in the experiments for F02, where a change in the selection noise (from roughly 1% to 25%) allowed the EuA to move from the ranks of middling performance to become the best performer. This flexibility will allow the EuA to be the algorithm of choice for approximating difficult problems of unknown structure. As discussed in the previous chapter, many improvements and enhancements to the EuA are possible that could further expand its robustness and applicability, in addition to advancing its performance.

A new form of evolution was introduced in this thesis—eugenic evolution. Eugenic evolution replaces the random, undirected variation of Darwinian evolution with purpose-

[1]The EuA was specifically designed to optimize *hard* problems. Evaluating the EuA's performance on trivial problems would be akin to rating a jet fighter's ability as a crop duster, an aircraft carrier's effectiveness as a fishing boat, or a CEO's ability to balance his checkbook. Such evaluations would not suitable in light of the specialized design considerations used in the creation of the EuA.

ful, deliberate actions designed to increase fitness. Where Darwinian evolution grants a differential propagation advantage to higher fitness individuals, and therefore *indirectly* promotes schemata of higher fitness, eugenic evolution *directly* attempts to identify, select and promote higher fitness schemata. The EuA accomplished these tasks by calculating allele fitness distributions, and through the use of the restriction operator, conditional allele fitness distributions. The experiments with non-restricting EuAs demonstrated that it is essential to calculate *conditional* schemata fitness distributions in order to account for the epistasis found in more difficult problems. Without information on conditional fitness distributions, performance of the EuA was often severely impaired, except in exceptional cases where exploration was an extreme necessity. In all cases, EuA progress towards optima (whether global or local) was steady and efficient. By implementing a evolutionary process of reasoned and purposeful creation, "eugenic evolution", the EuA has taken a large step toward bringing sight to the "blind watchmaker" of Darwinian evolution.

Bibliography

Bäck, T. (1992). A user's guide to genesys 1.0. Technical report, Department of Computer Science, University of Dortmund.

Bäck, T., Rudolph, G., and Schwefel, H.-P. (1993). Evolutionary programming and evolution strategies: Similarities and differences. In Fogel, D. B., and Atmar, J. W., editors, *Proceedings of the 2nd Annual Conference on Evolutionary Programming*, 11–22. Evolutionary Programming Society, La Jolla, CA.

Baluja, S. (1994). Population-based incremental learning. Technical Report CMU-CS-94-163, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213.

Baluja, S. (1995). An empirical comparison of seven iterative and evolutionary function optimization heuristics. Technical Report CMU-CS-95-193, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213.

Baluja, S., and Caruana, R. (1995). Removing the genetics from the standard genetic algorithm. Technical Report CMU-CS-95-141, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213.

Darwin, C. (1859). *On the origin of species by means of natural selection or the preservation of favored races in the struggle for life*. London: Murray.

De Jong, K. A. (1993). *Foundations of Genetic Algorithms 2*, chapter Genetic Algorithms are NOT Functions Optimizers, 5–17. San Mateo, CA: Morgan Kaufmann.

Eiben, A. E., Raue, P.-E., and Ruttkay, Z. (1994). *Parallel Problem Solving from Nature - PPSN III*, chapter Genetic algorithms with multi-parent recombination, 78–87. Berlin: Springer-Verlag.

Eshelman, L. J. (1991). *Foundations of Genetic Algorithms*, chapter The CHC Adaptive Search Algorithm: How to Have Safe Search When Engaging in Nontraditional Genetic Recombination, 265–283. San Mateo, CA: Morgan Kaufmann.

Eshelman, L. J., and Schaffer, J. D. (1991). Preventing premature convergence in genetic algorithms by preventing incest. In Belew, R. K., and Booker, L. B., editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, 115–122. San Mateo, CA: Morgan Kaufmann.

Forrest, S., and Mitchell, M. (1993). *Foundations of Genetic Algorithms 2*, chapter Relative Building-Block Fitness and the Building-Block Hypothesis, 109–126. San Mateo, CA: Morgan Kaufmann.

Garey, M. R., and Johnson, D. S. (1979). *Computers and Intractability.* New York, NY: W. H. Freeman and Company.

Garfinkel, R. S., and Nemhauser, G. L. (1972). *Integer Programming.* John Wiley & Sons.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning.* Reading, MA: Addison-Wesley.

H-L. Fang, P. Ross, D. C. (1993). A promising genetic algorithm approach to job-shop scheduling, rescheduling, and open-shop scheduling problems. In Forrest, S., editor, *Proceedings of the Fifth Annual Conference on Genetic Algorithms*, 375–382. San Mateo, CA: Morgan Kaufmann.

Haykin, S. (1994). *Neural Networks, A comprehensive foundation.* New York: Macmillan College Publishing Company.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence.* Ann Arbor, MI: University of Michigan Press.

Ingber, L. (1993). Simulated annealing: Practice versus theory. *Journal of Mathematical Computer Modelling*, 18(11):29–57.

Khuri, S., Bäck, T., and Heitkötter, J. (1993). An evolutionary approach to combinatorial optimization problems. In *Proceedings of the CSC'94*.

Kirkpatrick, S., and Sherrington, D. (1988). Infinite range models of spin-glasses. *Physical Review B*, 17:4384–4403.

Lamarck, J. B. (1914). *Zoological Philosophy.* London.

Luria, S. E., Gould, S. J., and Singer, S. (1981). *A View Of Life.* Menlo Park, CA: The Benjamin/Cummings Publishing Company, Inc.

Radcliffe, N. J., and Surry, P. D. (1995). Fundamental limitations on search algorithms: Evolutionary computing in perspective. *Lecture Notes in Computer Science*, 1000:275–??

Syswerda, G. (1991). *Foundations of Genetic Algorithms*, chapter A Study of Reproduction in Generational and Steady State Genetic Algorithms, 94–101. San Mateo, CA: Morgan Kaufmann.

Syswerda, G. (1993). *Foundations of Genetic Algorithms 2*, chapter Simulated Crossover in Genetic Algorithms, 239–255. San Mateo, CA: Morgan Kaufmann.

Whitley, D., Gordon, V. S., and Mathias, K. (1994). Lamarckian evolution, the baldwin effect and function optimization. In Davidor, Y., Schwefel, H.-P., and Maenner, R., editors, *Proceedings of the International Conference on Evolutionary Computation*, vol. 866. Jerusalem, Israel.

Vita

John Prior was born in (1) Newport News, Virginia. Then he moved to: (2) Idaho, (3) Virginia, (4) New Orleans, Louisiana, (5) Tacoma, Washington, (6) Huntington Beach, California, (7) Jakarta, Indonesia, (8) Dallas, Texas and then (9) Manhattan (New York), where he graduated with a B.S. in Applied Mathematics from Columbia University. Then he moved to (10) New Mexico, where he worked at Los Alamos National Laboratory, doing research in speech recognition, speech synthesis, intelligent control, and advanced traffic management. He also snow-boarded, skied, mountain-biked and camped a lot. Then he moved to (11) Austin, Texas, and enrolled as a Master's student in the Department of Computer Science and the the University of Texas at Austin. There he stared at the ceiling, wall, or floor and thought about the structure of combinatorial optimization problems, and then made up an algorithm to solve these problems. He also ate a lot of Tex-Mex food. And windsurfed. During the summer of '95, he went back to (12) New York and worked for Morgan Stanley, for whom he wrote a neural-network simulator that predicted the price of corn futures. But then he returned to (13) Austin, and tried to finish his Master's degree. But he didn't. Instead he moved to (14) Denver, Colorado, where he now works for a small company that uses AI technology to determine which people are most likely to pay off their bad debt. He finished up this thesis in the meantime, and finally got his M.A.. Once again, he's snowboarding, skiing, mountain-biking and camping. But now he's also racing as an amateur in the Sports Car Club of America. He may get a Ph.D. sometime in the future.

Permanent Address: 316 South Centre St.
Lancaster, TX 75146
USA
`jprior@cs.utexas.edu`

This thesis was typeset with LaTeX 2_ε[2] by the author.

[2]LaTeX 2_ε is an extension of LaTeX. LaTeX is a collection of macros for TeX. TeX is a trademark of the American Mathematical Society. The macros used in formatting this thesis were written by Dinesh Das, Department of Computer Sciences, The University of Texas at Austin.