Summer 2010

Population control in evolutionary algorithms

Jason Edward Cook

Department:

POPULATION CONTROL IN EVOLUTIONARY ALGORITHMS

by

JASON COOK

A THESIS

Presented to the Faculty of the Graduate School of

MISSOURI UNIVERSITY OF SCIENCE AND TECHNOLOGY

In Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE IN COMPUTER SCIENCE

2010

Approved by

Dr. Daniel Tauritz, Advisor
Dr. Ralph Wilkerson
Dr. Ronald Frank

ABSTRACT

Traditional evolutionary algorithms (EAs) are powerful robust problem solvers that have several fixed parameters which require prior specification. Having to determine good values for any of these parameters can be problematic, as the performance of EAs is generally very sensitive to these parameters, requiring expert knowledge to set optimally without extensive use of trial and error. Parameter control is a promising approach to achieving this automation and has the added potential of increasing EA performance based on both theoretical and empirical evidence that the optimal values of EA strategy parameters change during the course of executing an evolutionary run. While many methods of parameter control have been published that focus on removing the population size parameter (μ), most of these methods have undesirable side effects for doing so.

This thesis starts by providing evidence for the benefits of making μ a dynamic parameter and then introduces two novel methods for removing the need to preset μ. These methods are then compared, explaining the strengths and weaknesses of each. The benefit of employing a dynamic value for μ is demonstrated on two test problems through the use of a meta-EA, and the first novel method is shown to be useful on several binary test problems while the second performs well on a real valued test problem.

A condensed version of this thesis has been accepted for publication in the proceedings of the Genetic and Evolutionary Computation Conference 2010 [7].

ACKNOWLEDGMENT

I would like to thank my advisor Dr. Daniel Tauritz for all of the assistance he has given me, from reading and correcting my writing, to motivating me to work as hard as I can. I would not have made it this far without his help.

I would also like to thank my friends and family for supporting me during my time here in Rolla.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

NOMENCLATURE

Symbol	Description
μ	Population Size
l	Length of a representation
$D(x, y)$	Hamiltonian Distance between x and y
$P_{sur}(i)$	Probability of survival for i
i	An individual in the population
$f(i)$	The fitness value of i
$MaxFit$	Maximum fitness value currently in the population
$MinFit$	Minimum fitness value currently in the population
μ_0	Initial value of μ
λ	Offspring Size

1. INTRODUCTION

1.1. MOTIVATION

Modern society increasingly is faced with complex computational problems for which evolutionary algorithms (EAs) are appropriate solvers. However, practitioners in the field often lack the necessary expertise to properly configure EAs, leading to dismal results. This may lead to the practitioner disavowing the use of EAs for future problem solving, even when EAs would have been the best choice of solution method. Automating the configuration of EA strategy parameters (further referred to as parameters) is one approach for making EAs more usable by practitioners. Parameter control is a promising approach to achieving this automation and has the added potential of increasing EA performance based on both theoretical and empirical evidence that the optimal values of EA strategy parameters change during the course of executing an evolutionary run [9, page 131]. While parameter control does not entirely remove the need to configure parameters, it tends to make the behavior of an EA significantly less sensitive to the initial parameter settings than an EA with fixed parameter values [9, page 133]. The overhead caused by the need to converge to the optimal parameter values may be expected to be more than compensated by the huge savings obtained by minimizing the required amount of computationally expensive parameter tuning that is required. A drawback of many published parameter control approaches is the "stealth" introduction of new, control-related parameters; in some cases, these stealth parameters even end up determining the convergence values for the parameters being controlled, which is counterproductive [20].

A considerable body of work exists on parameter control, most of it focused on controlling mutation step size and, to a lesser extent, recombination and population size (μ). The latter has inspired several intriguing approaches, but all hampered by a variety of problems ranging from the introduction of stealth parameters to wasting evaluations on parallel populations. This thesis provides additional evidence that population size control can be beneficial to EA performance on important benchmark problems and introduces two novel approaches for population size control that avoid

some of the drawbacks of previously published methods. Both approaches have in common that they do not control population size directly, but rather make it a derived measure by employing non-traditional survival selection methods. The first method assigns each individual a survival chance proportional to its fitness, while the second method only removes individuals upon exhausting all available memory.

1.2. EVOLUTIONARY ALGORITHMS

An EA is a population-based optimization algorithm that uses artificial evolution to produce solutions to problems for varying difficulty, examples of which are provided in Figure 1.1 and Figure 1.2. It has three inputs: a fitness function, a representation, and a set of strategy parameters. The representation specifies the form of a candidate solution for the problem to be optimized. Commonly used examples of representations are bit strings, real valued vectors and trees. The fitness function maps each representation to a metric that determines how well that representation solves the problem. The final input, the set of parameters, controls how the EA will perform by managing how the various EA operators behave. These parameters include the population size, the offspring size and the mutation rate, among others.

Internally, an EA follows a fairly straight forward procedure, displayed in Figure 1.3. The first step is the creation of an initial population comprised of individuals encoding candidate solutions. Initialization can be performed in a variety of ways, including randomly, with a user defined heuristic, with results seeded from a previous run, or any combination of these or other methods. Each of these individuals is then evaluated and assigned a fitness value, indicating the quality of its particular solution. At this point, the evolutionary cycle begins. The first step in the evolutionary cycle is to select parents that will produce offspring. These parents can be selected in many ways, either randomly or by introducing some form of bias towards picking fitter individuals. After parents are selected, an offspring is created by using recombination. This results in an offspring that has some of the information contained in each parent participating in the offspring's creation. After being generated, the offspring undergoes mutation, modifying its genes slightly, altering the solution that it represents. This modification can vary significantly in severity, and might not even happen at all for a given offspring. Mutation exists to introduce new genetic

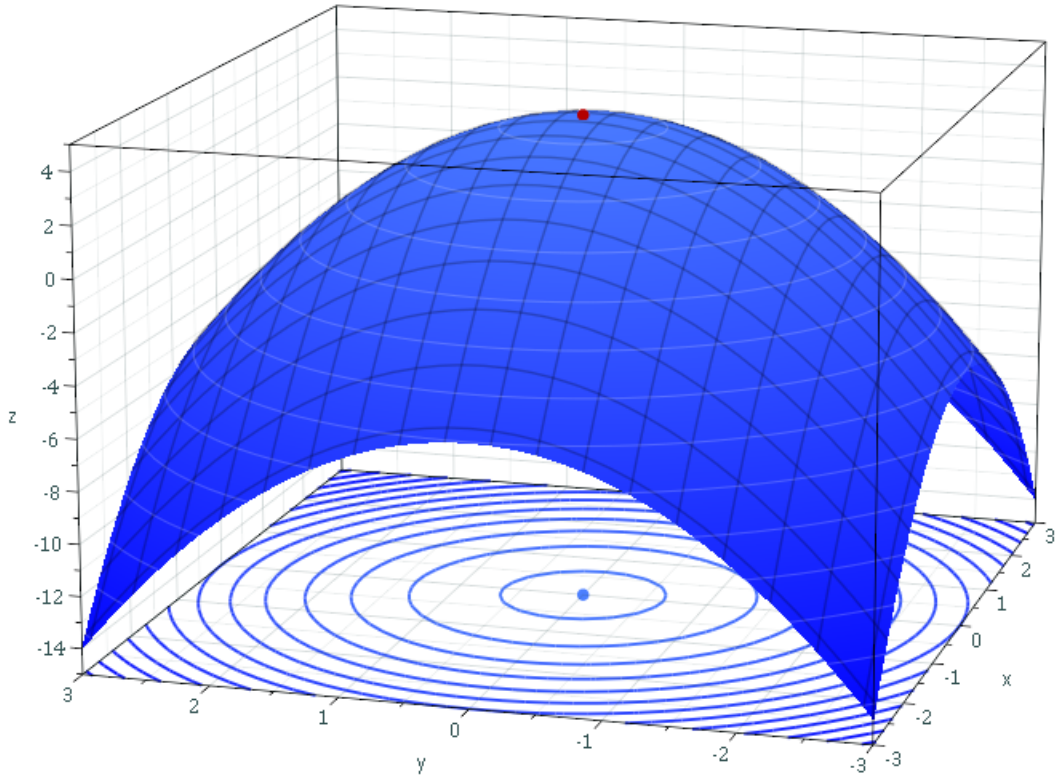

Figure 1.1. An example of a well-behaved search space.

material and maintain some level of diversity in the population, as without it, genes needed to produce a particularly good solution might disappear from the population entirely, assuming they were ever present to begin with. The offspring are evaluated and assigned a fitness value, just as the initial population was. The final step in the evolutionary cycle is to select survivors. These survivors will continue to exist in the algorithm and possibly generate more offspring for at least another generation. There are many different ways to select survivors, most of which are biased towards selecting stronger individuals to survive. The survivors that are selected repeat the evolutionary cycle, creating offspring and selecting survivors until some termination criteria is met. This criteria can be based on a variety of things such as the number of fitness evaluations used, the amount of time that has passed, or the quality of

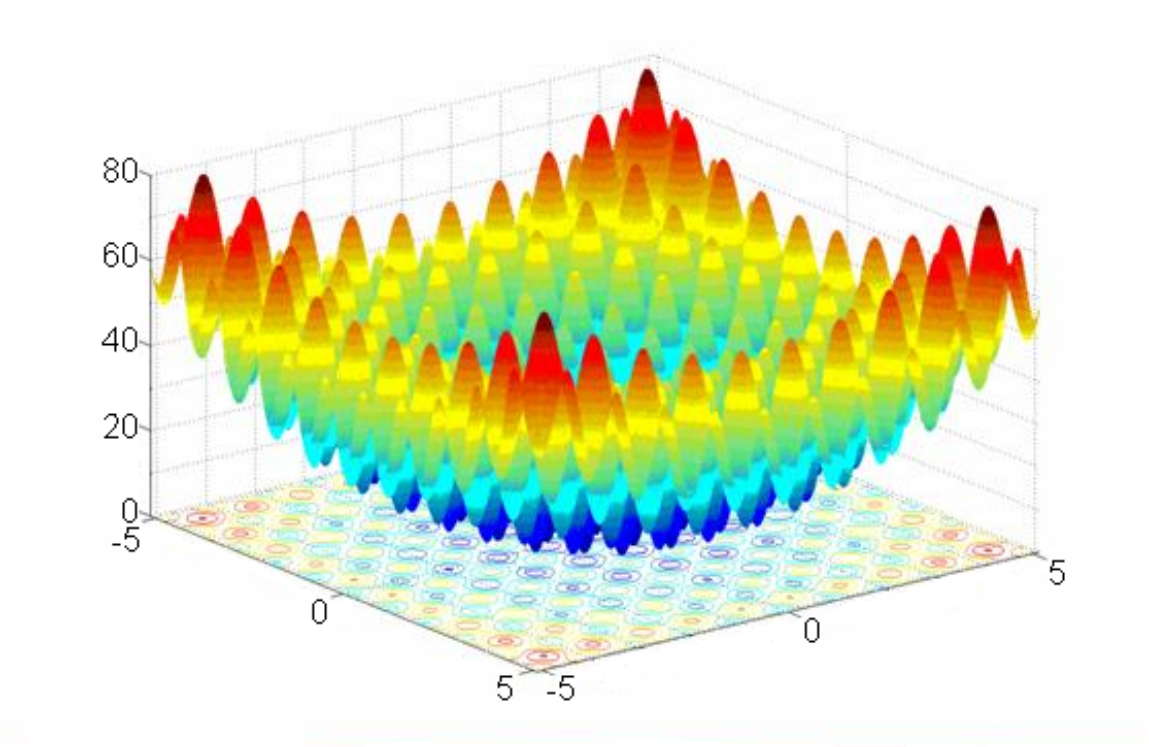

Figure 1.2. An example of an ill-behaved search space.

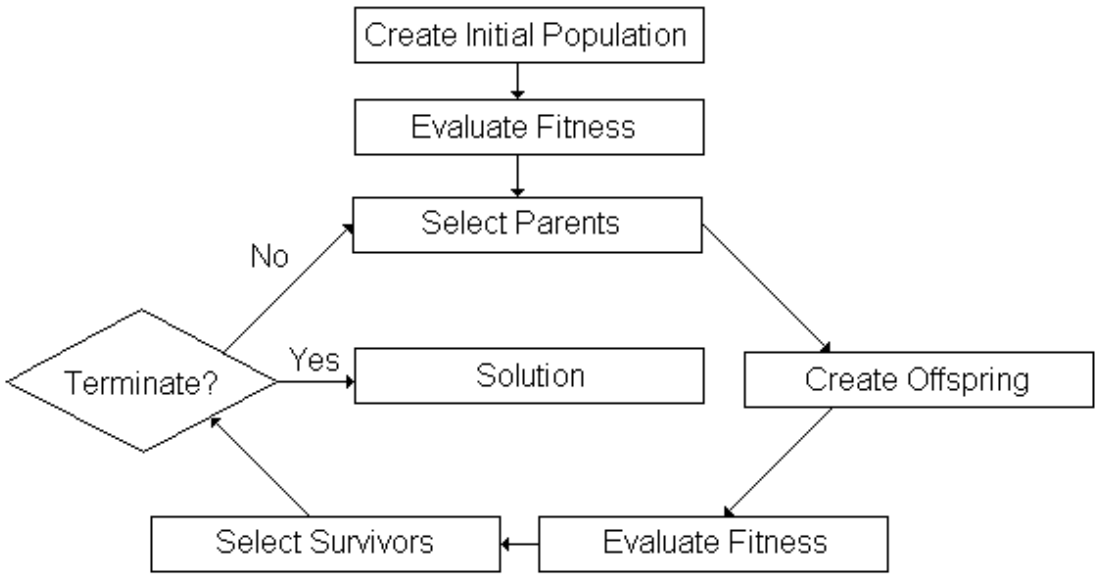

Figure 1.3. The evolutionary cycle.

the best available solution. Once this criteria is met, the individual with the highest fitness value ever found produces the EA's output in the form of its encoded solution, representing the best solution that was discovered.

1.3. OVERVIEW

The art of parameter control is still fairly undeveloped. While a considerable amount of work has been published, most of this work focuses on a small subset of parameters. Most commonly examined are the mutation step size, recombination, the parent selection methods, and μ. While a considerable amount of work has been done exploring parameter control of μ, many of these methods introduce new problems in exchange for the benefits that they create.

The purpose of this research is first to explore the benefit of a variable μ, then, provided that varying μ proves to be beneficial, to examine a pair of novel ideas on how to control μ. Both of these methods control μ by altering the way survivor selection is handled. The first method replaces survivor selection with an individual survival chance, allowing the population to fluctuate based on how many individuals survive from one generation to the next. The second method is based on the idea that individuals should not be unnecessarily removed from the population: removing any chance of an individual failing to survive until the system can no longer support more individuals because the memory allotted to the EA is full.

2. RELATED WORK

2.1. PARAMETER CONTROL

Parameter control is any method in which parameters of an EA are varied during a single run of the EA. This can take many forms, as there are countless methods in which one can vary any parameter during a run. In most published work, only a few of the possible parameters have been extensively examined, as mentioned in Section 1.3. Table 2.1 contains a brief summary of pertinent works that examine methods of parameter control. As shown in Table 2.1, most of these examine controlling the mutation rate, but some explore controlling other parameters such as the number of offspring produced (λ), selection methods, and μ.

The two most common types of parameter control are adaptive and self-adaptive parameter control. Adaptive parameter control involves changing the value of a strat-

Table 2.1. Brief summary of prior work examining parameter control and behavior.

Year	Parameter(s)	Type of Results	Source
1991	Reproduction	Theoretical and Empirical	[24]
1991	Mutation Rate	Empirical	[12]
1992	Mutation Rate	Empirical	[2]
1992	Mutation Rate	Theoretical and Empirical	[3]
1993	Mutation Rate	Empirical	[4]
1994	μ	Empirical	[1]
1996	Recombination Strategy	Empirical	[13]
1996	Mutation Rate	Empirical	[14]
1998	Mutation Rate, Crossover Rate	Theoretical and Empirical	[23]
1999	μ	Empirical	[10]
2000	Mutation Rate, Crossover Rate, μ	Empirical	[5]
2001	Mutation Rate	Theoretical and Empirical	[19]
2006	Selection Pressure, μ	Empirical	[8]
2007	μ	Empirical	[20]
2007	Parent Selection	Empirical	[21]
2008	Parent Selection, μ	Empirical	[11]
2009	λ	Empirical	[18]

egy parameter based upon feedback from the search. A classic example of adaptive parameter control would be Rechenberg's 1/5 success rule, which is used for controlling the mutation step size [9, page 72]. This rule states that about one in every five mutations should be successful, i.e., result in an offspring with a higher fitness value than its parents. If fewer mutations are successful, the mutation step size, the amount that mutation can change an individual, is reduced to focus the search near the current solutions. If more mutations are successful, the mutation step size is increased to broaden the area that mutation can reach, allowing for more exploration of the search space. In [4], an example of adaptive behavior is provided, in which the mutation rate is controlled with a predetermined mutation rate schedule based upon its success rate when optimizing a particular fitness function.

Self-adaptive parameter control involves coding additional information into each individual genotype. This information is then used to change how the EA behaves with regards to this individual, often in the form of changing various parameters. These encoded parameters are subject to mutation and recombination, as is the rest of the representation. This idea is based on the theory that high-quality individuals come from high-quality environments. By this reasoning, the individuals that evolve a high-quality set of additional parameters will have better solutions than those that evolve a poor set of parameters. This will generally cause the poorer sets of parameters to disappear from the population, ideally only leaving behind very high-quality sets of encoded parameters. Examples of self-adaptive behavior are very plentiful. In [2], basic self-adaptation of the mutation rate is explored, and [3] expands further on this idea, developing a near-optimal schedule for the mutation rate. The effects of self-adaptation in a steady-state genetic algorithm are throughly examined in [14]. In [23], the effects of self-adaptation on both the mutation rate and the crossover rate are investigated, and [13] introduces self-adaptive recombination by allowing the individuals to create groups of genes that are treated as a single gene for the purposes of recombination.

Some methods exist that combine aspects of both adaptive and self-adaptive parameter control. An example of one of the more common methods for doing so is a voting scheme [8]. In such algorithms, each individual is encoded with a vote. At predetermined intervals, a vote between all of the living individuals in the population

occurs, allowing each individual to influence the behavior of the EA, until the next vote occurs.

2.2. PRIOR WORK ON POPULATION SIZING IN EAS

One of the first attempts to control population size resulted in the Genetic Algorithm with Varying Population Size (GAVaPS) [1]. This was accomplished by removing the population size parameter entirely and introducing an individual survival chance based on age, thus causing population size to become a derived measure. A similar approach is also used to create the Genetic Algorithm with Adaptative Population Size (APGA) [5]; however, APGA does not age the fittest individual in the population, giving it a better chance for survival. This is the inspiration for the two novel approaches introduced in this thesis which also remove the population size parameter entirely and employ an individual survival chance, though not based on age. While GAVaPS and APGA both remove population size as a parameter, they also both introduce two new stealth parameters, MinLT and MaxLT, representing the minimum and maximum number of generations that any individual can survive. Furthermore, it has been shown that this approach does not actually remove the population size, as it simply causes the population size to converge on a fixed value determined by the two stealth parameters [16, 20].

A completely different approach was taken in the Parameter-less GA [10, 16] and GPS-EA [20, 11] which attempt to converge on the optimal population size by generating increasingly larger populations in parallel until no improvement in the fitness is found in the largest population. The main difference is that the Parameter-less GA has no bound on the total number of parallel populations, while the GPS-EA has exactly two parallel populations at all times. While this approach avoids the pitfall of implicitly specifying the population size through the values of the stealth parameters, it does have its own drawbacks. One major drawback is the lack of population size control during the evolution of a particular population, as various researchers have indicated that different population sizes are optimal at different stages of evolution [5, 8]. A second major drawback is that because new populations are initialized randomly, most of the fitness evaluations are used by populations that are discarded during execution and do not contribute to finding the final solution, except

for determining the optimal population size. An attempt to reuse fitness evaluations from smaller populations by seeding new larger populations resulted in poor performance [11, Section 3.4]. The novel approaches introduced in this thesis employ a single population, thus avoiding the overhead of multiple parallel populations, and can adapt the population size during the entire evolutionary run.

Another approach was taken in GASAP [8] which employs a self-adaptive method for population size control via a voting system. Each individual in the population has a gene encoding its vote on population size; the population size is determined by tallying the votes of all the individuals. However, stealth parameters were introduced for specifying the lower- and upper bounds of the vote values, and the sensitivity of EA performance to these bounds was not reported.

Many competitive EAs use either multiple populations evolving in parallel, such as the Parameter-less GA [10] and the GPS-EA [20]. Both of these are effective techniques for finding higher-quality solutions, as they minimize the impact of one population converging to a sub-optimal local optima, since others may avoid it. Some work has also been done on reinitializing a portion of the population at several points throughout a run [22]. While this can be effective for maintaining genetic diversity, most of the new individuals are unlikely to survive for many generations as the fitness values for individuals already in the population are typically higher than for new, randomly generated individuals.

3. EMPIRICAL EVIDENCE OF POPULATION CONTROL BENEFITS

3.1. METHODOLOGY

While some researchers have indicated that the optimal population size varies during evolution [5, 8], there is a lack of published evidence. In this section a systematic approach to gathering such evidence is described and some initial results are presented.

The approach employs a meta-EA to simultaneously evolve the optimal sequence of population size values for each generation of an EA. Note that we are not advocating this as a practical approach for population size control, just as a way to determine whether different population size values are optimal at different evolutionary stages. Every benchmark problem for which we can find an EA configuration that causes the sequence found by the meta-EA to consist of values that differ, adds evidence that the optimal population size can vary during evolution. Note that an EA employing optimal population size control will be at least as good as the exact same EA without population size control as the one with population control can mimic the one without population control, effectively setting the population size to a fixed value, but not the other way around.

To test this idea accurately, first a manually tuned Traditional EA (TEA) was created. The parameters that were found to be ideal for this manually tuned EA were then used as fixed parameters for the EA that the meta-EA will evolve. The solution provided by the meta-EA should be a set of population sizes, μ_0 through μ_n where n is the number of generations, that will produce a high-quality solution. The results from TEA are then compared to the results obtained from the EA produced by the meta-EA.

Ideally, the results obtained from this experiment will show several things. The first is that the population size will differ significantly from generation to generation. Also, hopefully this dynamic μ will result in an increase in performance over the traditionally tuned EA. Provided that the values of μ_0 through μ_n are different and some improvement is made, the usefulness of a dynamic μ is apparent. Additionally,

this test could reveal patterns in the values of μ. If patterns in the population size are found, determining an optimal method for controlling the population size could be made easier.

3.2. EXPERIMENTAL DESIGN

3.2.1. Test Problems. For this particular experiment two test problems were selected. The Spears' Multimodal Problem (SMP) was chosen because it is a multimodal problem in which the difficulty is easily controlled [15]. The SMP generator creates a specified number of bit strings that represent different peaks present in the landscape. Each of these peaks is then assigned a value. The fitness of individual i that represents a bit string of length l is defined as:

$$f(i) = \frac{l - D(i, Peak_n(i))}{l} \cdot height(Peak_n(i)) \tag{1}$$

Because the difficulty of the problem can be altered easily by changing the number of peaks, this particular problem is very useful for gauging how well an algorithm can perform on increasingly difficult problems. As such, this experiment will be repeated using different numbers of peaks, as described in the next section.

The second test was performed on the bounded D-TRAP problem [6]. Essentially, the problem consists of a bit string broken up into 4-bit substrings called traps. The fitness for each trap is defined as:

$$f(u) = \begin{cases} 3 - u & \text{if u} \leq 3 \\ 4 & \text{otherwise} \end{cases} \tag{2}$$

where u is the number of zeros in the trap. The resulting fitness for a given solution is the sum of the fitness values for every 4-bit trap. For these experiments, a solution is comprised of 100, 250 or 500 traps.

3.2.2. Dynamic Population Concept Testing. To test the usefulness of a dynamic population, a set of various experiments was created. A meta-EA is executed on three instances of the Spears' Multimodal Problem, one with 10 peaks, the second with 50 peaks and the third with 100 peaks. All of the parameters for the meta-EA are fixed, except for the population size, which will be represented by

a string of 100 numbers. Each of these numbers will correspond to the value of μ desired at each generation. If at each generation μ changes significantly, then the usefulness of a dynamic population is demonstrated. The fixed parameters used by the meta-EA and the EA are shown in Table 3.1. For this particular experiment, the population values that the meta-EA is capable of finding are restricted to not increasing by more than λ at a time. This is done to allow λ to remain fixed.

The results from this experiment are compared against a traditionally tuned EA with a fixed population size, or TEA. Both will be limited to 100 generations, rather than by the number of fitness values used, to properly examine the benefits of allowing μ to change over those 100 generations. TEA was tuned by first testing all combinations of μ and λ at values of 50 through 500, at increments of 50. After the best values for μ and λ were found, various combinations of parent selection operators and survival selection operators were tested, with the set producing the best results being used. These values are shown in Table 3.2.

Table 3.1. Fixed parameters used by the meta-EA and the EA that is produced by the meta-EA.

Algorithm	Meta-EA	EA
μ	10	—
λ	10	50
Parent Selection Tournament Size	2	5% of μ
Crossover	Uniform Random	
Mutation Rate	1/100	$1/l$
Survivor Selection	Elitist Binary Tournament	
Termination Condition	200 Fitness Evaluations	100 Generations

Table 3.2. Fixed parameters used by TEA.

Problem	SMP	D-TRAP
μ	450	400
λ	500	100
Parent Selection Tournament Size	10% of μ	5% of μ
Crossover	Uniform Random	
Mutation Rate	$1/l$	
Survivor Selection	Truncation	Elitist Binary Tournament
Termination	100 Generations	

3.3. RESULTS

The main purpose behind using this meta-EA was to examine what was considered to be an optimal set of population sizes for an EA. As can be seen in the standard deviation in Figure 3.1 and in Table 3.3, the population size fluctuates wildly at nearly every generation. While this may be caused in part by a lack of sensitivity at various times during the execution of the algorithm, the drastic variation shown in Figure 3.1 as well as the higher, more efficiently obtained results observed for the EA produced by the meta-EA in Table 3.3 indicate that these changes are significant, and that an optimal population should not remain fixed throughout the evolution on at least some problems.

3.4. DISCUSSION

The results shown in Section 3.3 are expected, given a few known aspects about how a population behaves. While a small population can converge on a solution more rapidly than a large population, a larger population tends to produce a higher quality solution. In this manner, a larger population can be associated with increased exploration, while a smaller population exhibits more exploitation. Knowing this, it makes sense that at times a larger population would be used to improve the final solution

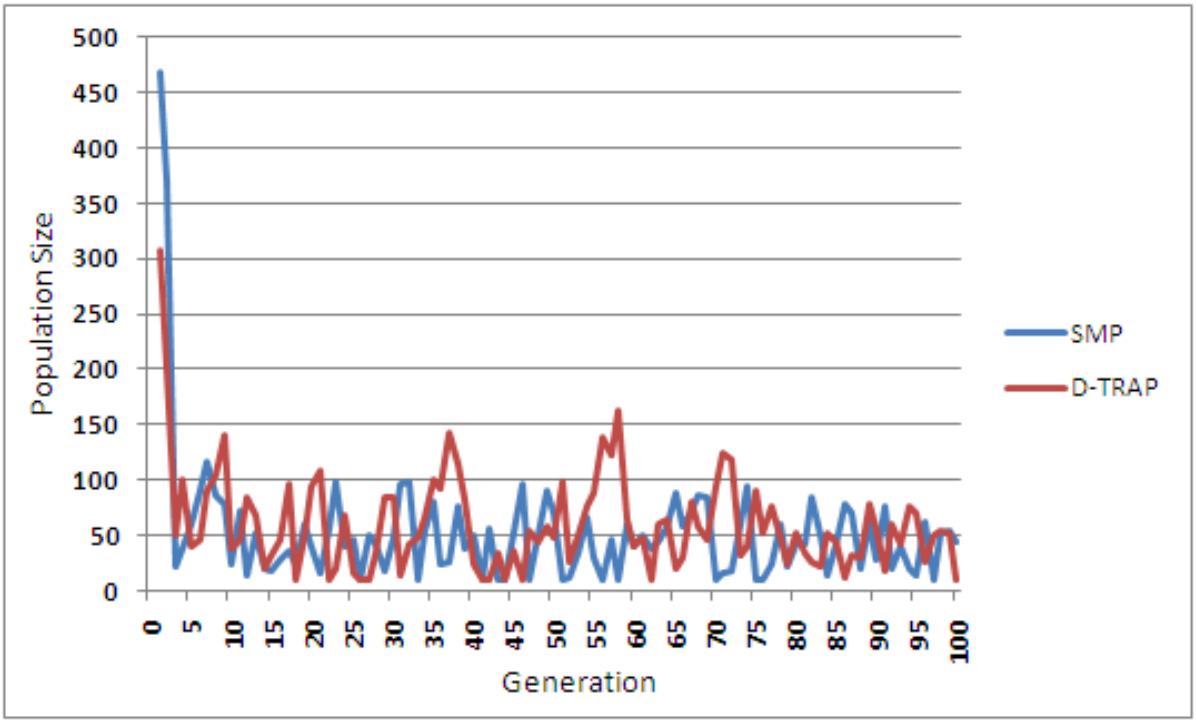

Figure 3.1. Optimal population sizes found by a meta-EA for the SMP and D-TRAP test problems.

Table 3.3. Comparison of meta-EA and TEA results.

Algorithm	EA from meta-EA		TEA	
Problem	SMP	D-TRAP	SMP	D-TRAP
Problem Size	100 Peaks	100 Traps	100 Peaks	100 Traps
Average μ	52.37	59.82	450	400
SD of μ	60.20	45.80	0	0
Fitness Reached	100%	81.5%	99.7%	80.8%
Fitness Evaluations Required	5500	5300	50400	10400

quality by exploring more possibilities, and at other times a smaller population would be used to exploit the information that the larger population has found.

Figure 3.1 sheds a little more insight into what makes a good population, as both tests show one distinctive common trait: the population is very high during

the first generation, then drops off rapidly in the next few generations, followed by oscillating values. This shows that a high initial population size may be very useful, making effective use of an initial random search that is done during the population initialization. This idea is further explored in Section 4.1, as it is a large portion of the inspiration for the Fitness Scaled Individual Survival EA (FiScIS-EA). The oscillating values that follow the initial decrease in population size are likely caused by insensitivity in μ at that point in the evolution, especially if the EA has already converged to a solution.

4. DYNAMIC POPULATION METHODS EXPLORATION

4.1. METHODOLOGY

4.1.1. FiScIS-EA. Fitness-Scaled Individual Survival (FiScIS) EA, replaces traditional survivor selection with an individual-based survival method. Essentially, the idea is to assign each individual a chance to survive at the end of each generation, and allow those chances to determine who survives to the next generation, thus making population size a derived measure.

The exact method for determining the chance for an individual to survive is similar to the idea of linear scaling. At each generation, each individual is assigned a chance to survive to the next generation based on the individuals' fitness when compared to the currently existing best and worst fitness values in the population, namely:

$$P_{sur}(i) = \frac{f(i) - MinFit}{MaxFit - MinFit} \tag{3}$$

where $P_{sur}(i)$ is the chance that individual i survives to the next generation, $f(i)$ is the fitness of i, and $MaxFit$ and $MinFit$ are the highest and lowest fitness values present at the beginning of each generation. This equation is used unless $MinFit = MaxFit$, in which case $\forall i, P_{sur}(i) = 1$ to prevent division by 0. This also prevents the population from going extinct, as this is only possible if $\forall i, P_{sur}(i) = 0$, (i.e., $MinFit = MaxFit$). This method ensures several things. First, the best solution will always survive, as $P_{sur}(\arg\max_i f(i)) = 1$. Also, the worst solution will never survive because $P_{sur}(\arg\min_i f(i)) = 0$. Other solutions randomly survive based upon their quality when compared to the maximum and the minimum.

This formula also encourages population growth to explore an area when the population begins to converge, while keeping the population size smaller while rapid growth of the fitness is still attainable. This occurs because when the population's fitness is rapidly increasing, many weaker individuals die due to the difference in fitness, but when growth slows down the differences created at each generation are no longer as large, allowing many more individuals to survive. Also, this method

allows for a large initial population, but since the large initial population is unlikely to remain large, it still allows for rapid improvement. This effectively combines the advantages of both a large population's exploratory power and a small population's convergence speed. It should be noted that this method still requires the input of an initial value for the size of the population. Because of this, the standard population parameter (μ) has actually just been replaced with an initial population parameter (μ_0). While this does not reduce the number of parameters, it still allows for changes in the population in the middle of a single run, causing the initial choice of μ to have less of an impact.

While functional, as shown in Section 4.3, this method is generally incapable of making use of a large number of fitness evaluations, unlike other powerful population sizing EAs, such as the Parameter-less GA [10]. This can be rectified by allowing the algorithm to reinitialize the population if convergence is detected, essentially restarting the entire EA. Restarting the EA when convergence is detected allows FiScIS-EA to effectively compete with other algorithms that use multiple populations to achieve quality results. FiScIS-EA also has the added benefit over algorithms like the Parameter-less GA of only running one instance of the EA at a time.

4.1.2. GC-EA. The Growth Curve EA (GC-EA) is based on the idea that individuals in the population should never be discarded. Since discarding any individual from the population almost always removes some of the genetic diversity, maintaining every individual could be a very effective method for preventing premature convergence. Using this method, μ_0 is set at the beginning, and at every generation μ is increased by λ.

$$\mu_{n+1} = \mu_n + \lambda \tag{4}$$

This behavior is maintained until the population has expanded to fill all of the available memory that it is allowed to occupy. Once all of the memory has been filled, classic survivor selection methods are used to allow evolution to continue. While potentially effective, this method has one major problem: it places all of the selective pressure upon the parent selection operator. This effectively assumes that the parent selection operator is powerful enough to pick good individuals while not removing any

chance for weaker individuals to be selected, regardless of how large the population grows.

GC-EA differs from FiScIS-EA in several key areas. Unlike FiScIS-EA, it follows a user defined pattern for the values of μ, even though this pattern is determined completely by parameters that are already supplied. This creates additional pressure on picking a successful λ and μ_0, as together, they will determine how the population size will change during every generation.

GC-EA has several obvious downsides, most notably is the dependency on a very strong parent selection operator. After a few generations, assuming μ_0 is not significantly larger than λ, the size of the population will have grown considerably. The ability to handle populations of all sizes well is very difficult to find in most parent selection operators, and since this method has a constantly increasing μ, this is a major concern.

Much like FiScIS-EA, this method is generally not capable of usefully consuming very large numbers of fitness evaluations. As such, it will also be compared with other methods when allowed to restart if convergence is detected, in the same manner as FiScIS-EA.

4.2. EXPERIMENTAL DESIGN

4.2.1. Test Problem Suite. To test these concepts, five different sets of experiments were conducted. All of these sets of experiments consist of comparing TEA with GC-EA and FiScIS-EA, as explained in Section 4.1. These tests are run on several different values of μ and μ_0, as well as several different sizes of each of the problems.

The first problem, ONEMAX, was chosen for its simplicity. This function counts the number of bits in a bit string that are set to one, and returns that as its fitness value. This problem is being used to test the basic functionality of each EA.

The second problem, the Spears' Multimodal Problem (SMP), is defined in Section 3.2.1. This problem was chosen to demonstrate the performance of these two algorithms on a multimodal problem. The third test was performed on the bounded D-TRAP problem defined in Section 3.2.1. The D-TRAP problem is being used here for its deceptive nature, making it a very difficult problem to solve.

The fourth test was performed on the 3-SAT problem. The 3-SAT problem is a boolean satisfiability problem comprised of a series of clauses in conjunctive normal form, each containing three variables. These clauses are randomly generated at the beginning of each run of the experiment. The fitness value returned by the fitness function is the number of clauses made true by a given set of true and false values. For these experiments, the number of variables is held constant at 100, while the number of clauses is changed to alter the difficulty of the problem.

The final test problem is the minimization of the Rastrigin Function. This is a highly multimodal real valued problem, defined by the function:

$$f(i) = 10n + \sum_{i=0}^{n}(x_i^2 - 10cos(2\pi x_i)) \tag{5}$$

This particular problem was chosen for a few main reasons. Since it is a real-valued problem, unlike the other four test problems, it provides a significantly different problem type for comparing these algorithms. Also, because it is a real-valued problem, it is more easily related to a real-world problem. Because of the real-valued nature of this problem, different operators for recombination and mutation need to be used. Recombination will be accomplished by averaging the values of the two selected parents, while mutation will be handled by applying Gaussian mutation.

4.2.2. Dynamic Population Method Testing. Extensive testing was done to determine what strategy parameters to use for these experiments. μ, μ_0 and λ were independently optimized in the order listed for values ranging from 50 to 500. Parent selection was performed using uniform random selection, a binary tournament, a tournament containing five percent of the population, and a tournament containing ten percent of the population. Survivor selection was handled by truncation, binary tournament or an elitist binary tournament for TEA. All combinations of these were tested thoroughly before deciding on the parameters to use for TEA, FiScIS-EA and GC-EA. Values that were held constant throughout all of the experiments are shown in Table 4.1. The final values for the varied parameters are shown in Table 4.2.

The four binary test problems use the same operators for mutation and crossover. The crossover operator randomly picks from which of the two parents each bit comes, and the mutation operator checks every bit and flips the bit if mutation occurs. The

Table 4.1. Fixed parameters used in all experiments.

Parameter	Value
Mutation Rate	$1/l$
Real-Valued Mutation	$N(0,1)$
Binary Recombination	Uniform Random
Real-Valued Recombination	Whole Arithmetic
Alpha	.5
Standard Deviation for Real Valued Mutation	.5
Termination (Maximum Fitness Evaluations)	100000

Table 4.2. Dynamically tuned parameters used in all experiments.

Problem	ONEMAX	SMP	D-TRAP	3-SAT	Rastrigin
Algorithm	TEA				
μ	100	450	400	400	100
λ	100	500	100	300	100
Parent Selection Tournament Size	10% of μ	10% of μ	5% of μ	5% of μ	5% of μ
Survivor Selection	Truncation	Truncation	Elite Binary Tournament	Truncation	Elite Binary Tournament
Algorithm	FiScIS-EA				
μ_0	200	250	100	350	350
λ	100	500	250	400	300
Parent Selection Tournament Size	10% of μ	5% of μ	10% of μ	10% of μ	5% of μ
Algorithm	GC-EA				
μ_0	100	100	100	200	100
λ	100	200	100	100	100
Parent Selection Tournament Size	10% of μ	10% of μ	10% of μ	10% of μ	5% of μ
Survivor Selection	Truncation	Truncation	Truncation	Truncation	Truncation
Maximum μ	5000	5000	5000	5000	5000

Rastrigin function instead uses whole arithmetic recombination and applies Gaussian noise to the values as mutation.

The fixed parameters used in these experiments were obtained by hand tuning each of the parameters one at a time until a better value could not be readily found, starting with the mutation rate, then the recombination method, followed by then mutation step size used in the Rastrigin function experiments. These parameters were chosen not just for the good results they produced, but also because they worked well on most if not all of the conditions that were to be varied.

Using these five test problems, the scalability of both new EAs is compared. ONEMAX uses $l = 100, 500, 1000$, the SMP uses 10, 50, and 100 peaks, the D-TRAP problem uses 100, 250 and 500 traps, ratios of 2:1, 4:1 and 6:1 clauses to variables are used with the 3-SAT problem, and the Rastrigin function uses $l = 10, 50, 100$. These tests are used to see how well these EAs perform on increasingly difficult problems.

Similarly, the sensitivity of the performance of FiScIS-EA to μ_0 is compared to the sensitivity of the performance of TEA to μ by using different values of μ_0 and μ. This is done on the SMP with 50 peaks, the D-TRAP problem with 250 traps, the 3-SAT problem with a 4:1 clauses to variables ratio and on the Rastrigin function with $n = 50$. The values for μ and μ_0 that will be tested are shown in Table 4.3. This is done to show if μ_0 has a more or less significant impact on the final results than the original parameter μ.

Finally, FiScIS-EA and GC-EA will also be compared to the results obtained from the Parameter-less GA [10] and GASAP [8] on the SMP, and with GPS-EA [20] and GPS-EA with ELOOMS [11] on the D-TRAP problem in order to demonstrate how they compare to other population sizing EAs.

To perform these tests, three different performance metrics will be observed. The Mean Best Fitness (MBF) and its standard deviation (SD) are used to determine the quality of a solution. These will both be measured as a percentage of the optimal fitness value. The Average Evaluations until Success (AES), the measure of the number of fitness evaluations needed to find an optimal solution, and the Success Rate (SR), the percentage of runs resulting in an optimal solution, will only be used in the comparisons on the SMP, because the SMP is a test problem that EAs are capable of solving some portion of the time, while D-TRAP is rarely, if ever, solved.

Table 4.3. Values for μ and μ_0 used to test the sensitivity of μ_0.

Test Problem	TEA	FiScIS-EA
SMP	350, 450, 550	150, 250, 350
D-TRAP	300, 400, 500	50, 100, 200
3-SAT	300, 400, 500	250, 350, 450
Rastrigin	50, 100, 150	50, 100, 150

Another set of tests will be performed using the same setup just described, except after ten generations passing with no better fitness value being found, the population will be reinitialized. This will allow for a better comparison with leading algorithms, as it will be more capable of utilizing large numbers of fitness evaluations.

4.3. RESULTS

The goal behind making μ a dynamic value is to remove a user parameter without causing a significant loss in performance and ideally even improve performance. This section compares the effectiveness of FiScIS-EA, GC-EA, and TEA. In order to statistically validate the comparison of these algorithms, 30 runs were conducted for each EA for each test problem and ANOVA with $\alpha = .05$ was used to determine the signifcance of the differences observed.

As can be seen in Table 4.4 and Figure 4.1, TEA, FiScIS-EA and GC-EA performed similarly with regards to MBF on ONEMAX. All of these EAs were capable of finding the correct solution every time and converge at about the same number of fitness evaluations. Figure 4.2 demonstrates how μ changes over the course of a representative run when using FiScIS, shrinking early to speed up convergence while expanding later to explore the search space. Figure 4.3 examines how scalable each EA is on the ONEMAX problem, demonstrating that the increase in time required to find a solution for a more difficult problem is similar for all of the tested EAs.

The second experiment was performed on the Spears' Multimodal Problem and the results from it are shown in Table 4.4 and Figure 4.4. In this experiment the differences between FiScIS-EA and TEA were not significant for any set of parameters. GC-EA converged more quickly and still found a high MBF, but unlike the

Table 4.4. Results obtained from all tested problems.

Algorithm	TEA			FiScIS-EA			GC-EA		
Problem	ONEMAX								
Length	100	500	1000	100	500	1000	100	500	1000
MBF	100	100	100	100	100	100	100	100	100
(SD)	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)
AES	1560	14380	86300	1733	14933	92600	1900	17533	96400
SR	100%	100%	100%	100%	100%	100%	100%	100%	100%
Problem	SMP								
Peaks	10	50	100	10	50	100	10	50	100
MBF	100	99.22	99.14	100	99.15	99.26	100	98.8	98.3
(SD)	(0)	(0.94)	(0.94)	(0)	(1.27)	(0.73)	(0)	(1.75)	(1.53)
AES	8450	8450	8450	5750	5750	5750	2100	2080	2200
SR	100%	70%	37%	100%	60%	47%	100%	50%	20%
Problem	D-TRAP								
Traps	100	250	500	100	250	500	100	250	500
MBF	81.91	80.70	72.60	82.41	80.14	70.80	80.5	77.8	68.75
(SD)	(1.14)	(0.71)	(0.51)	(1.15)	(0.54)	(0.54)	(1.14)	(0.66)	(0.63)
AES	—	—	—	—	—	—	—	—	—
SR	0%	0%	0%	0%	0%	0%	0%	0%	0%
Problem	3-SAT								
Clauses to Variables	2:1	4:1	6:1	2:1	4:1	6:1	2:1	4:1	6:1
MBF	100	99.43	98.11	100	99.63	98.21	100	99.25	97.83
(SD)	(0)	(0.30)	(0.27)	(0)	(0.29)	(0.27)	(0)	(0.30)	(0.32)
AES	7075	29200	—	4750	10750	—	2260	—	—
SR	100%	10%	0%	100%	7%	0%	100%	0%	0%
Problem	Rastrigin Function								
Length	10	50	100	10	50	100	10	50	100
MBF	100(0)	97.27	91.70	100	97.68	92.04	99.79	99.42	98.96
(SD)	(0)	(0.70)	(0.85)	(0)	(0.76)	(0.69)	(0.28)	(0.69)	(0.98)
AES	1553	—	—	1443	—	—	3700	—	—
SR	100%	0%	0%	100%	0%	0%	60%	0%	0%

other two algorithms it has considerably more difficulty in finding the optimal solution. Figure 4.5 shows how μ changes throughout a representative run when using FiScIS-EA. In this particular experiment, the high value of λ causes μ to rise early and remain high. Figure 4.6 examines how scalable each algorithm is on the SMP problem, showing that GC-EA's performance drops off much faster than the other two algorithms tested.

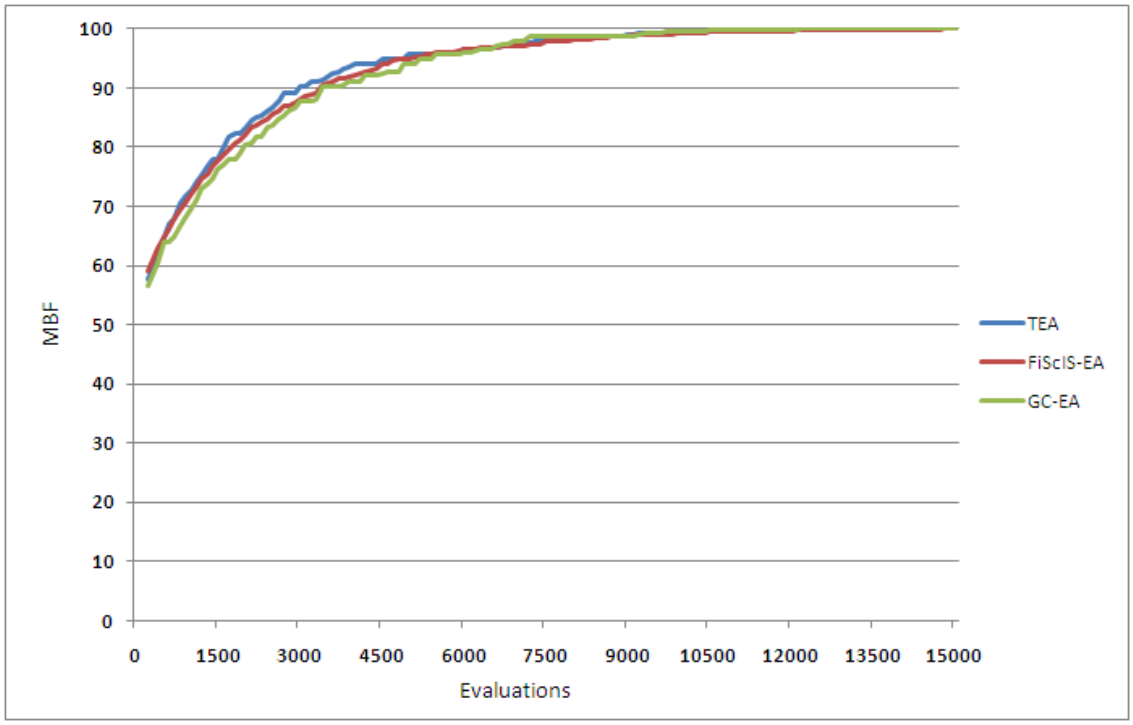

Figure 4.1. Comparison of fitness values between three algorithms on the ONEMAX problem using a bit string of 500 bits.

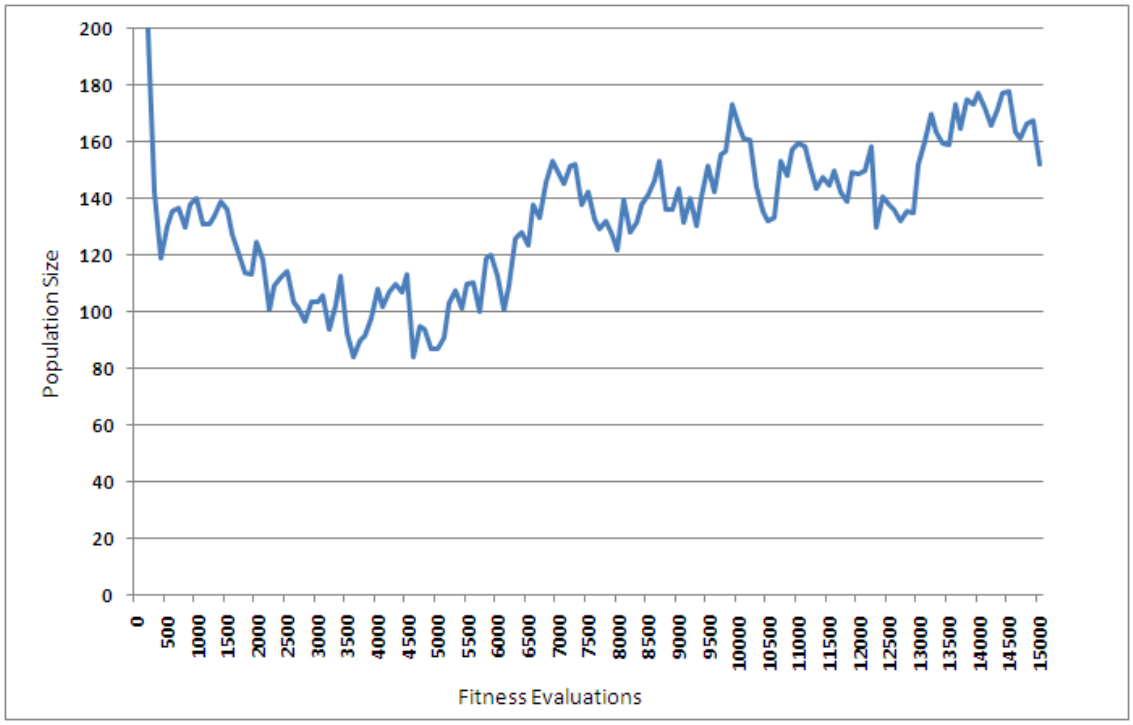

Figure 4.2. Examination of population values of FiScIS-EA on the ONEMAX problem using a bit string of 500 bits.

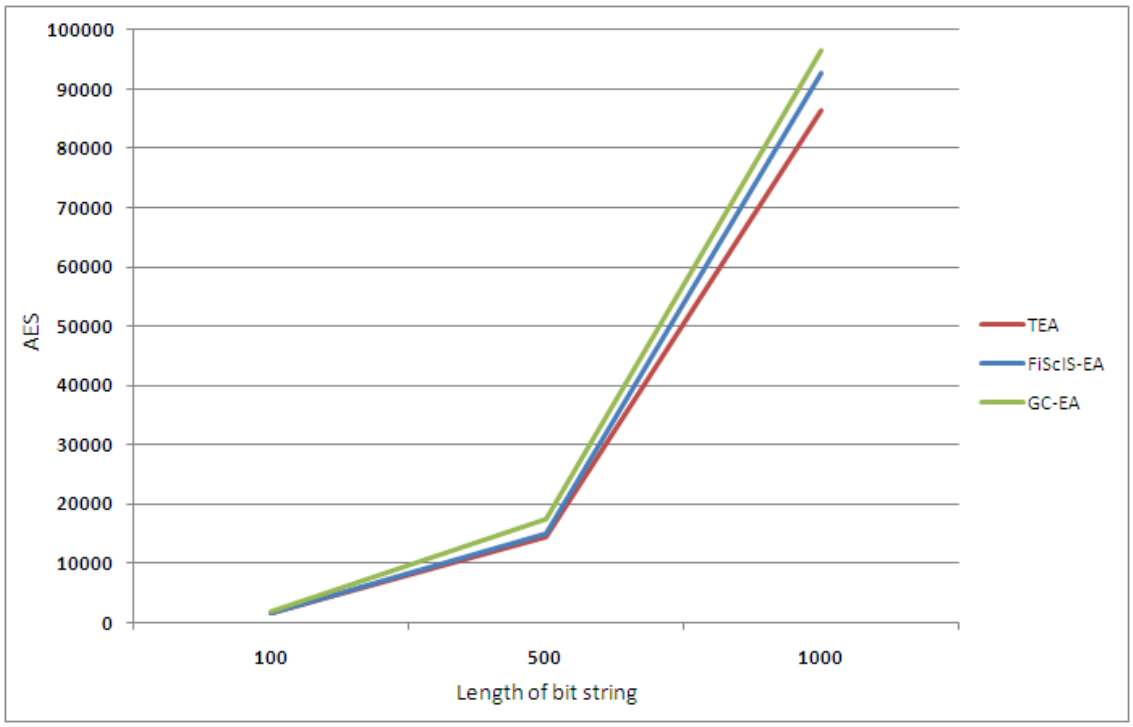

Figure 4.3. AES of the TEA, FiScIS-EA and the GC-EA for the ONEMAX problem.

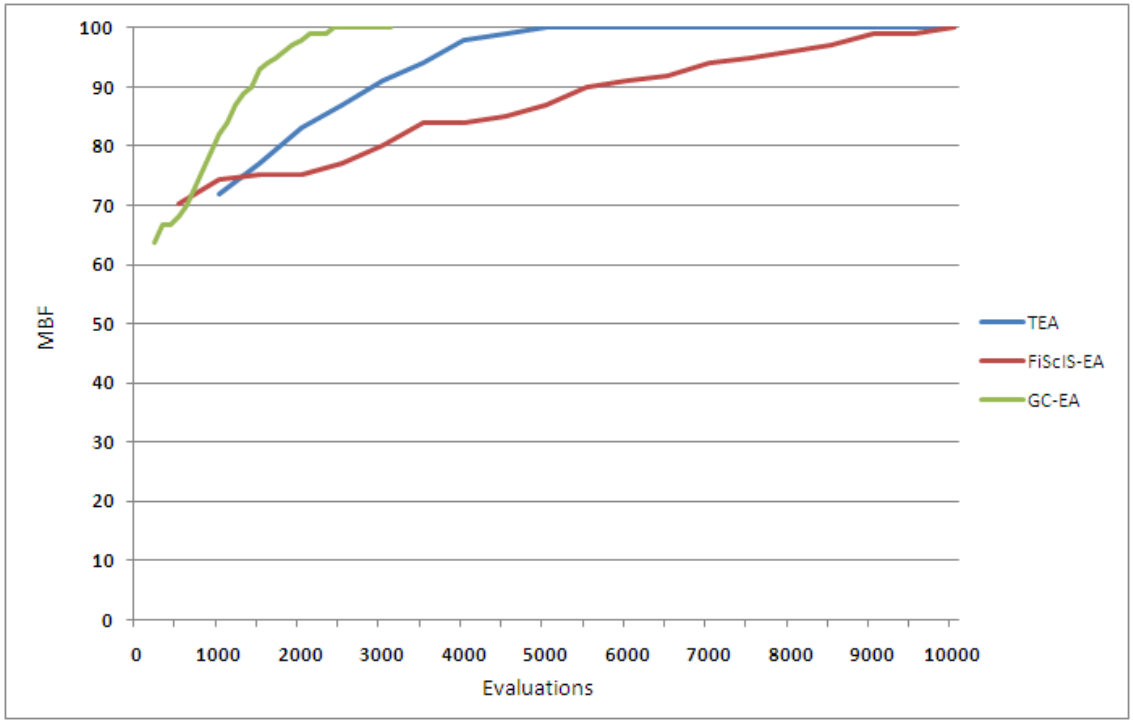

Figure 4.4. Comparison of fitness values between three algorithms on the Spears' Multimodal Problem using 50 peaks.

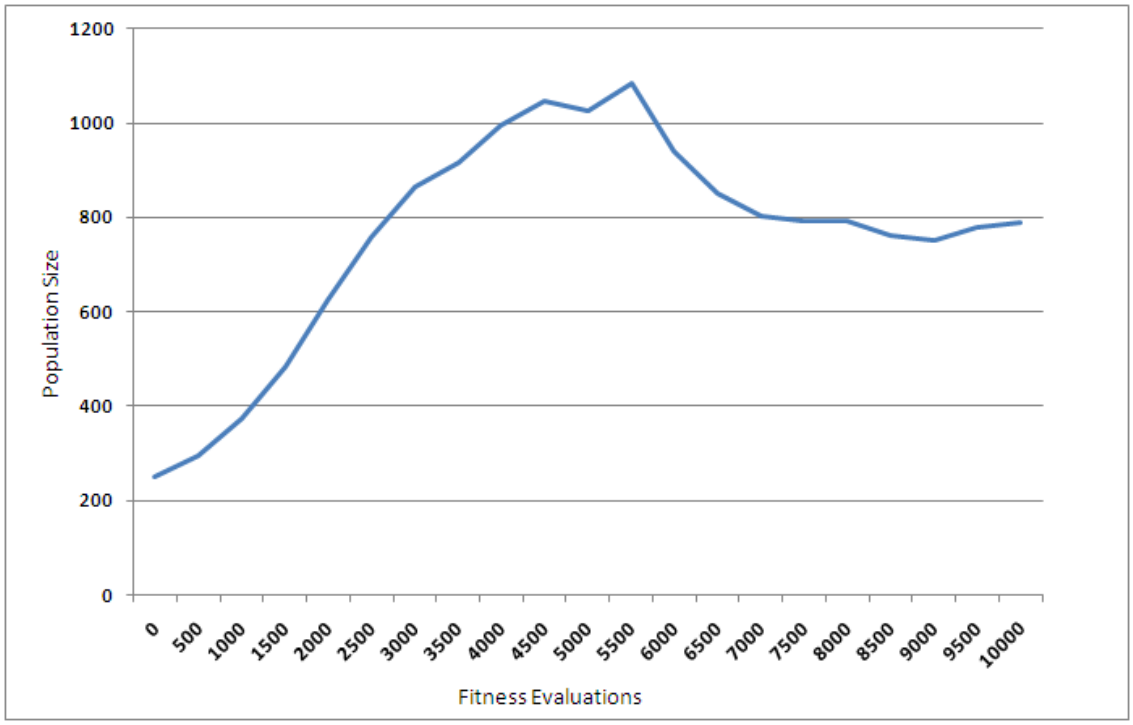

Figure 4.5. Examination of population values of FiScIS-EA on the Spears' Multimodal Problem using 50 peaks.

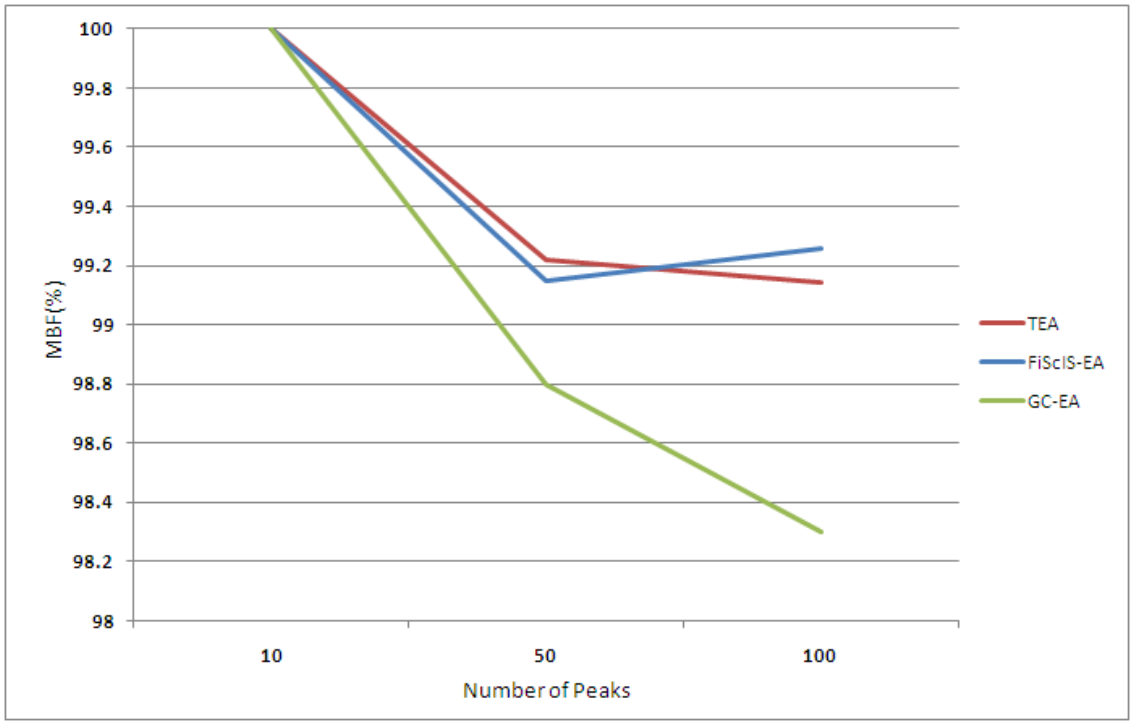

Figure 4.6. MBF of the TEA, FiScIS-EA and the GC-EA for the Spears' Multimodal problem.

The thrid experiment was performed on the D-TRAP problem, with results shown in Table 4.4 and Figure 4.7. Here, altering the population size using either FiScIS-EA or GC-EA had a significant negative impact on the performance. Figure 4.8 shows the changes in μ observed during a representative run using FiScIS-EA, showing that the population grew quickly, then tended to fluctuate between λ and 2λ. Figure 4.9 illustrates how scalable each EA is on the D-TRAP problem. For this problem, the fitness values degrade at a similar rate for each of the EAs.

The results for the fourth experiment are shown in Table 4.4 and Figure 4.10, which shows the performance, and Figure 4.11, which shows the changes in μ observed during a representative run using FiScIS-EA. In this test, FiScIS-EA converged faster, produced a better MBF, and was more capable of finding the optimal solution than the other two algorithms, though the improvement in MBF was insignificant except when using a 4:1 clauses to variables ratio. The population values recorded with FiScIS-EA show a slow climb for most of the evolution, followed by a steep climb

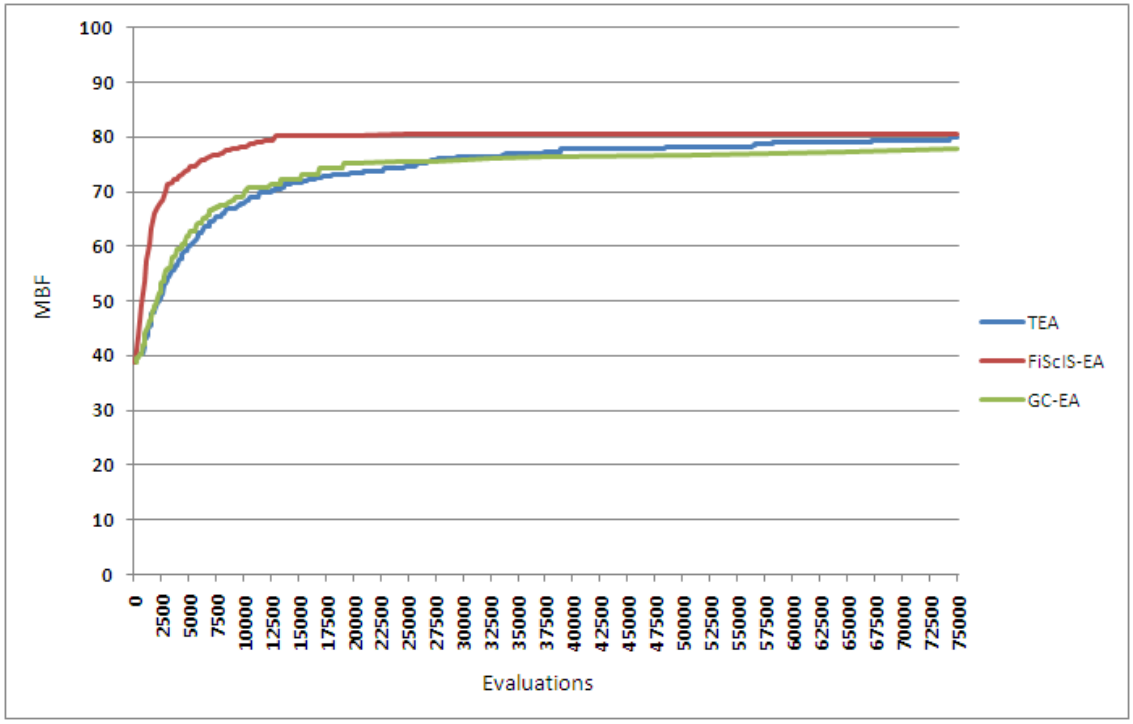

Figure 4.7. Comparison of fitness values between three algorithms on the D-TRAP problem using 250 traps.

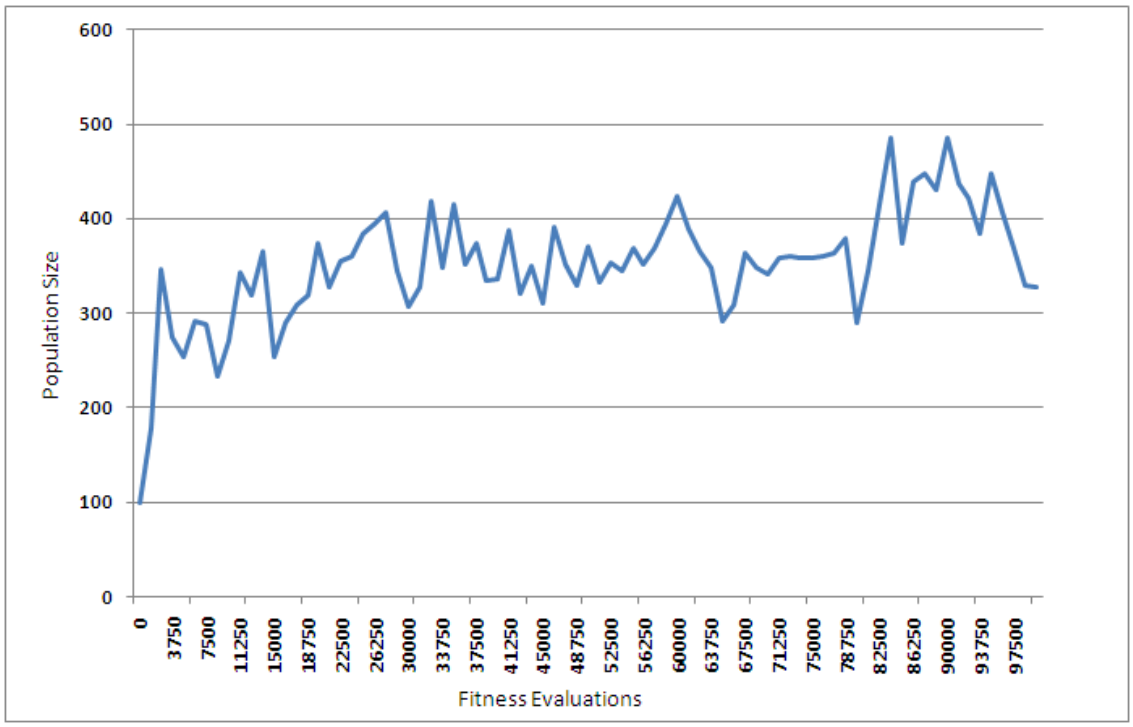

Figure 4.8. Examination of population values of FiScIS-EA on the D-TRAP problem using 250 traps.

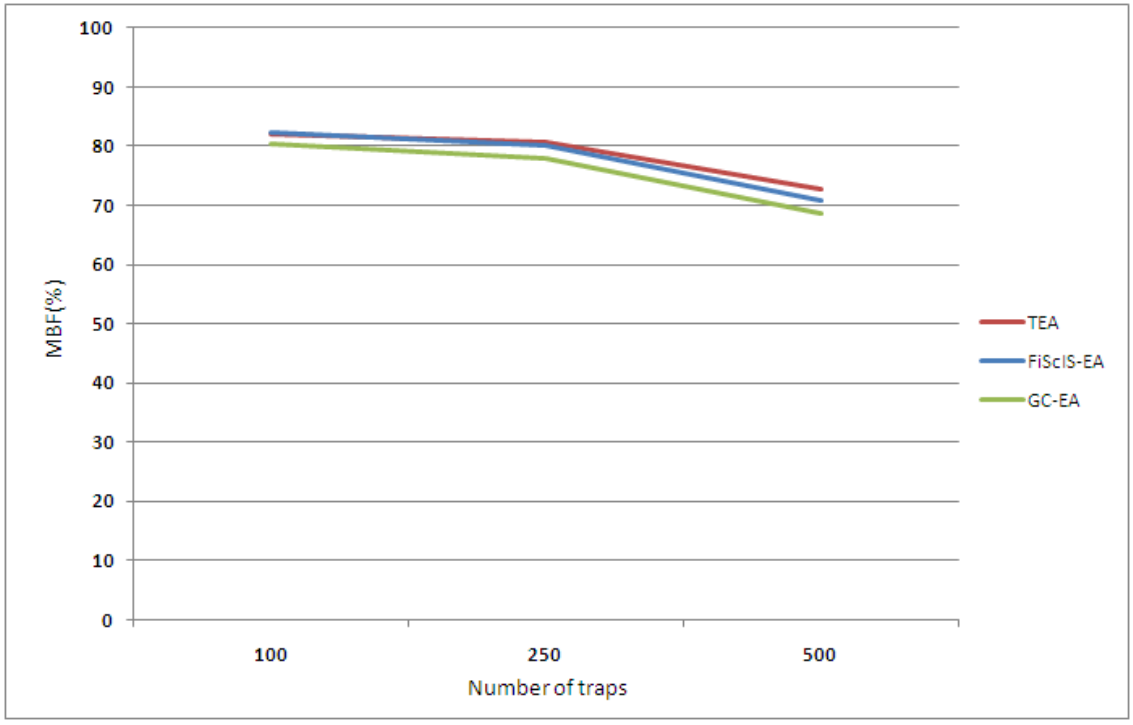

Figure 4.9. MBF of the TEA, FiScIS-EA and the GC-EA for the D-TRAP problem.

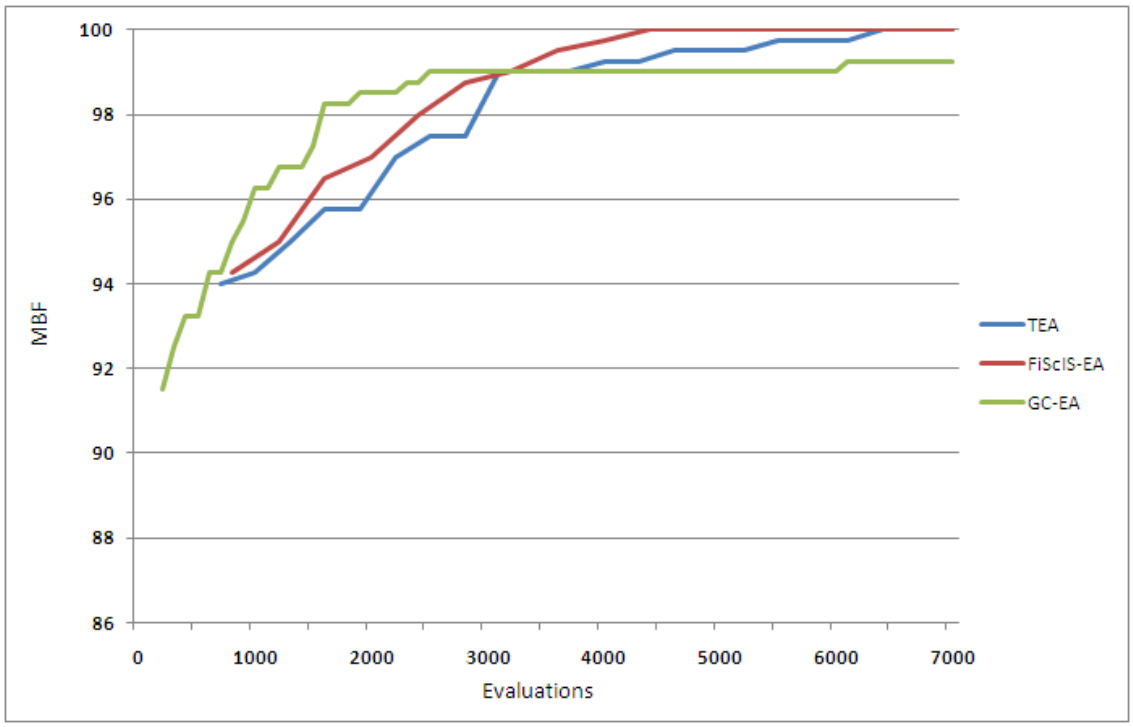

Figure 4.10. Comparison of fitness values between three algorithms on the 3-SAT problem using a 4:1 clauses to variables ratio.

throughout the remainder of the run. Overall, the difference in performance between FiScIS-EA and TEA was insignificant, while GC-EA performed significantly worse. Figure 4.12 shows the MBF of all three algorithms with different clause to variable ratios, showing that all three algorithms are equally scalable on the 3-SAT problem.

The fifth experiment was performed on the Rastrigin Function, with the results shown in Table 4.4 and Figure 4.13. In this experiment, GC-EA achieves a significantly higher MBF than the other EAs for $l = 50, 100$. Figure 4.14, which shows the changes in μ observed during a representative run of FiScIS-EA. Once again, the behavior in the population is similar to the previous experiments. Figure 4.15 examines how scalable each algorithm is on the Rastrigin function minimization problem. In this case, the GC-EA possesses a higher degree of scalability, as the MBF for the GC-EA degrades by less than half of what the TEA or FiScIS-EA lose.

Table 4.5 shows the sensitivity of the performance of FiScIS-EA to μ_0 compared to the sensitivity to TEA to μ. Comparing the change in MBF shows that, although

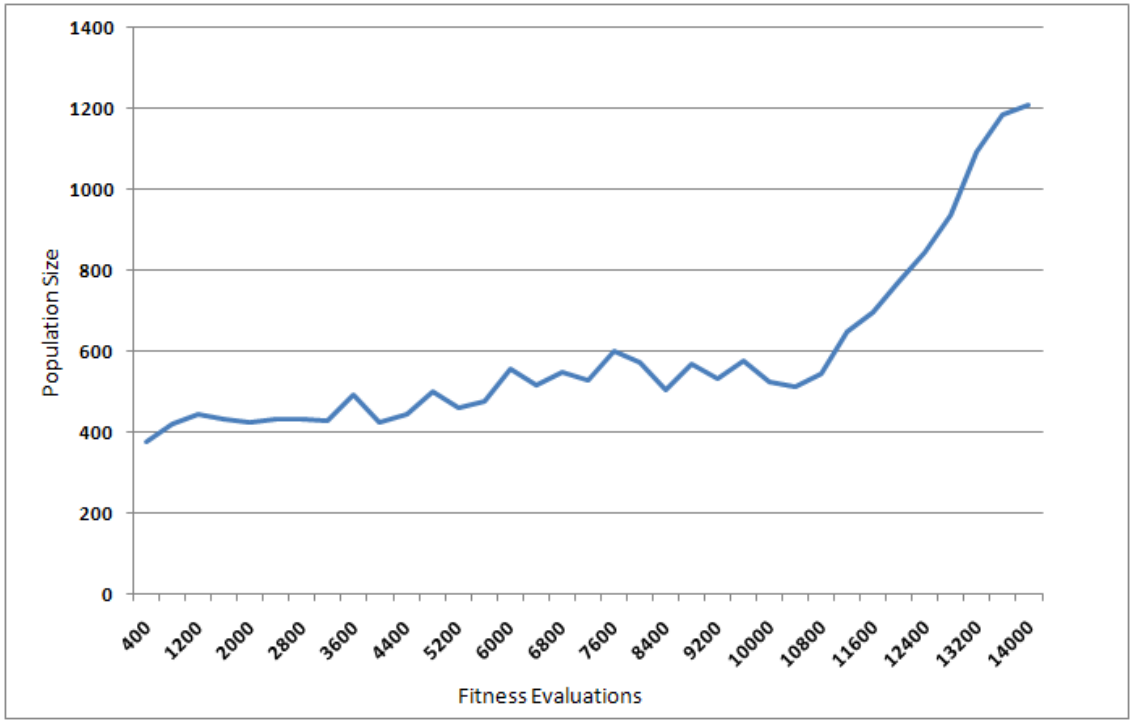

Figure 4.11. Examination of population values of FiScIS-EA on the 3-SAT problem using a 4:1 clauses to variables ratio.

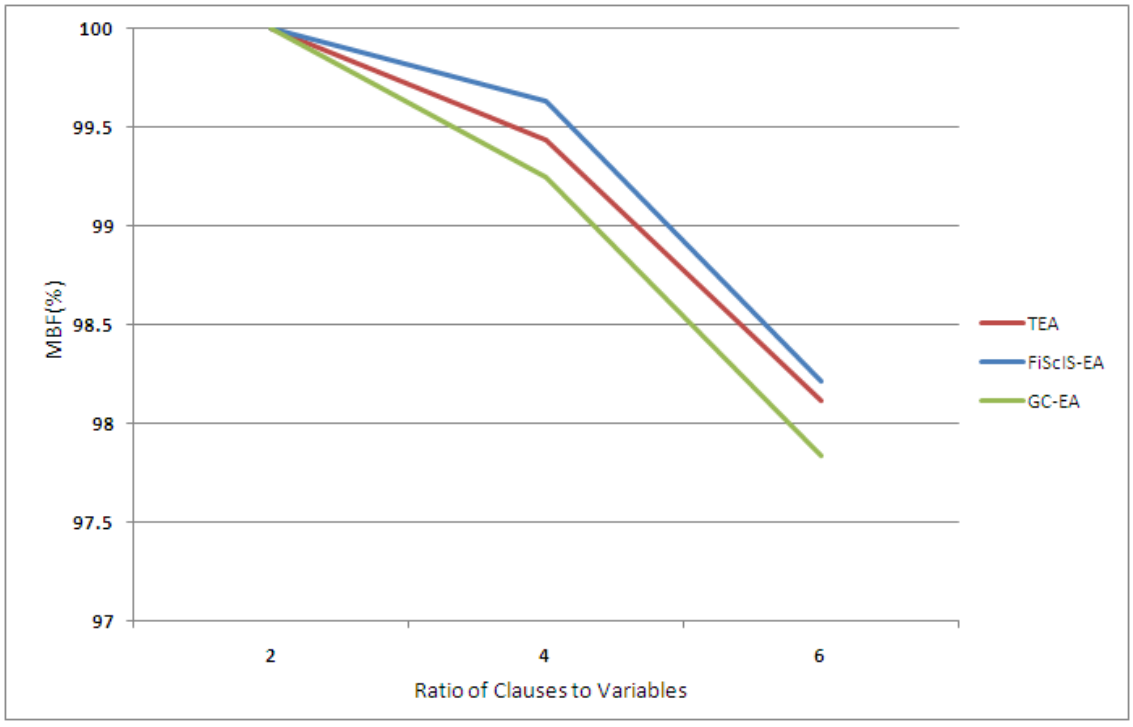

Figure 4.12. MBF of the TEA, FiScIS-EA and the GC-EA for the 3-SAT problem.

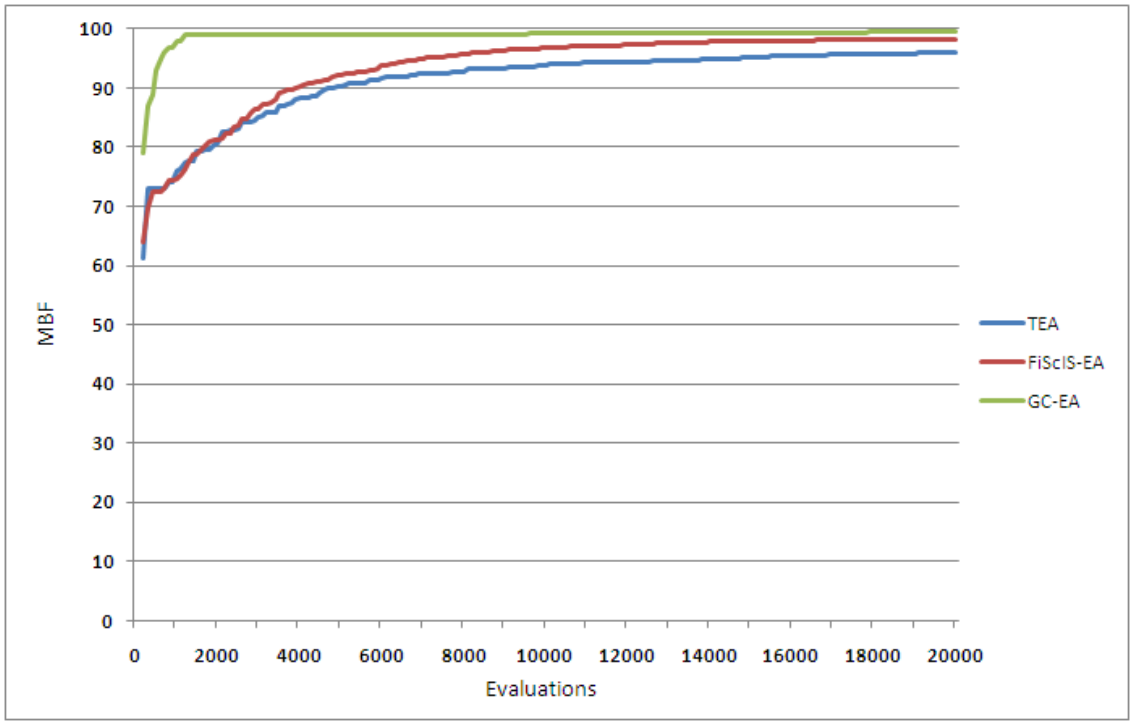

Figure 4.13. Comparison of fitness values between three algorithms on the Rastrigin function using 50 dimensions.

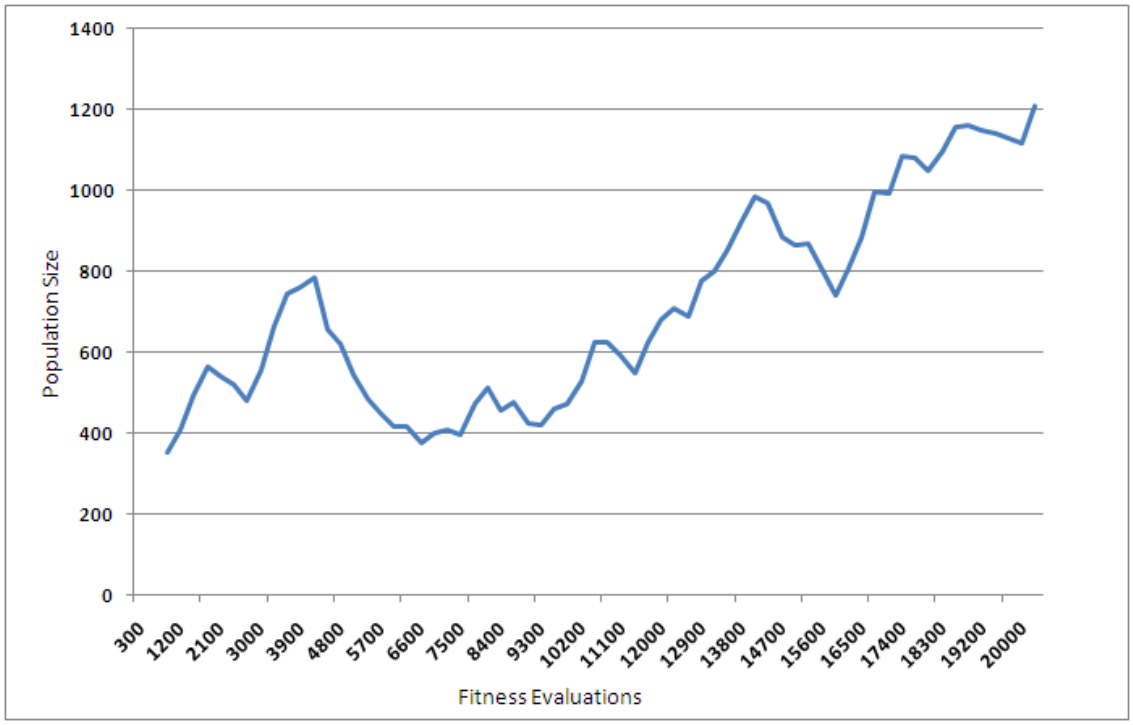

Figure 4.14. Examination of population values of FiScIS-EA on the Rastrigin function using 50 dimensions.

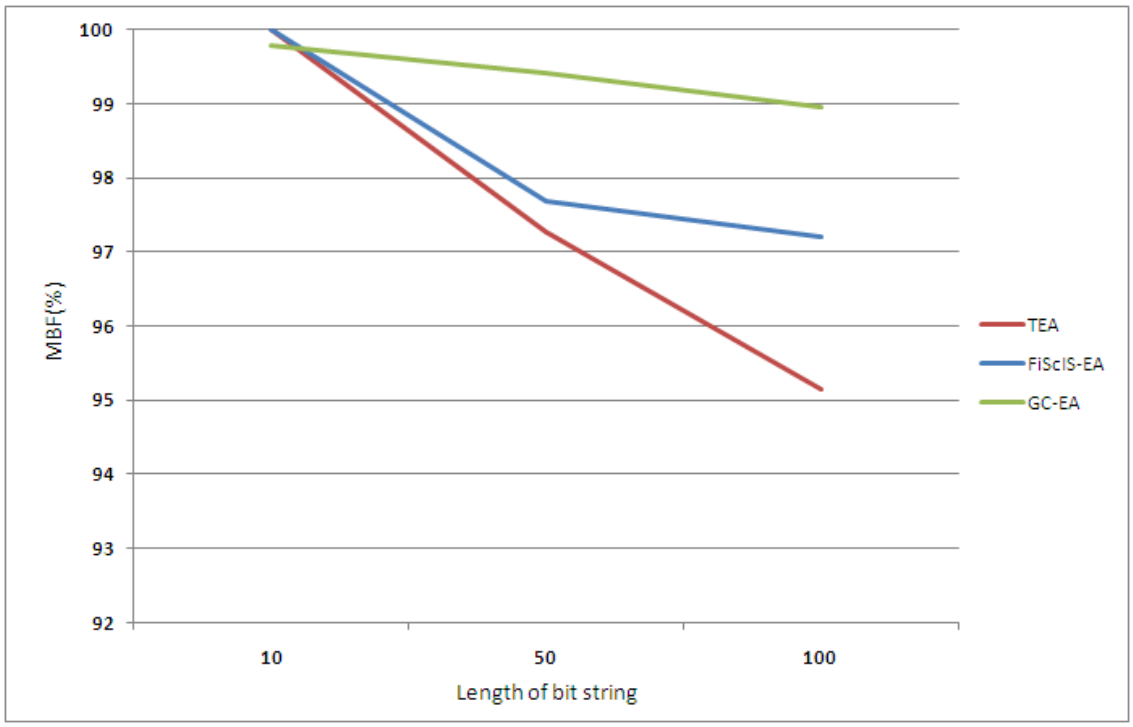

Figure 4.15. MBF of the TEA, FiScIS-EA and the GC-EA for the Rastrigin problem.

the differences are small, the performance of FiScIS-EA is significantly less sensitive to μ_0 than TEA to μ in every case based on ANOVA with $\alpha = 0.05$. Figure 4.16 shows the differences in the F values produced by ANOVA. Larger F values indicate a higher probability of the performance values being significantly different, and, therefore, the performance being more sensitive to the parameter being varied. On average, the MBF for FiScIS-EA changes by about half of the amount that TEA changes. This lack of sensitivity in FiScIS-EA is unsurprising. FiScIS-EA is capable of quickly changing μ if μ_0 is initially set at a value that is too high or too low, allowing FiScIS-EA to compensate for a sub-optimal value. This lack of sensitivity is a significant improvement for the ease of tuning μ.

Table 4.6 shows a comparison of FiScIS-EA, GC-EA, Parameter-less GA [10] and GASAP [8] on the Spears' Multimodal problem, using the results published in their respective papers. As shown in Table 4.6, while both FiScIS-EA and GC-EA are capable of finding a good result very quickly, they have difficulty finding the optimum

Table 4.5. Sensitivity results of the performance of FiScIS-EA and TEA to μ_0 and μ, respectively.

Test Problem	μ	TEA	μ_0	FiScIS-EA
SMP	350	98.57(1.71)	150	98.98(0.71)
	450	99.22(0.94)	250	99.15(1.27)
	550	99.18(0.85)	350	98.99(1.23)
D-TRAP	300	80.38(0.52)	50	80.04(0.57)
	400	80.70(0.72)	100	80.14(0.54)
	500	80.56(0.49)	200	80.06(0.44)
3-SAT	300	99.35(0.14)	250	99.60(0.11)
	400	99.43(0.30)	350	99.63(0.29)
	500	99.40(0.22)	450	99.60(0.42)
Rastrigin	50	97.05(0.75)	50	97.37(0.19)
	100	97.27(0.70)	100	97.68(0.76)
	150	96.71(0.93)	150	97.54(0.92)

solution as often as the Parameter-less GA. This comparison is extended by including the use of restarts for both FiScIS-EA and GC-EA. As shown here, the SR of both FiScIS-EA and GC-EA improves dramatically with the addition of restarts.

Table 4.7 shows a comparison of FiScIS-EA, GC-EA, GPS-EA [20, 11] and the GPS-EA with ELOOMS [11] on the D-TRAP problem, showing the results published in [11, page 78, table 6.1]. Unlike the previous comparison, the addition of restarts has a significant effect on the performance of both FiScIS-EA and GC-EA when done with 25 traps, a small effect when used with 125 traps, and a negative impact when used with 250 traps. This negative impact is likely due to the high number of fitness evaluations required to converge, and spans of time in which no improvement were made. Only a few restarts were possible because of the large number of fitness evaluations used, and those restarts were initiated before the EA had converged.

4.4. DISCUSSION

Most of the results show similar performance for the three algorithms on the five different test problems; however, in some cases one of the algorithms did perform significantly different than the other two. Looking at the ONEMAX results reveals that GC-EA tends to converge slowly. This is likely caused by the occasional usage of

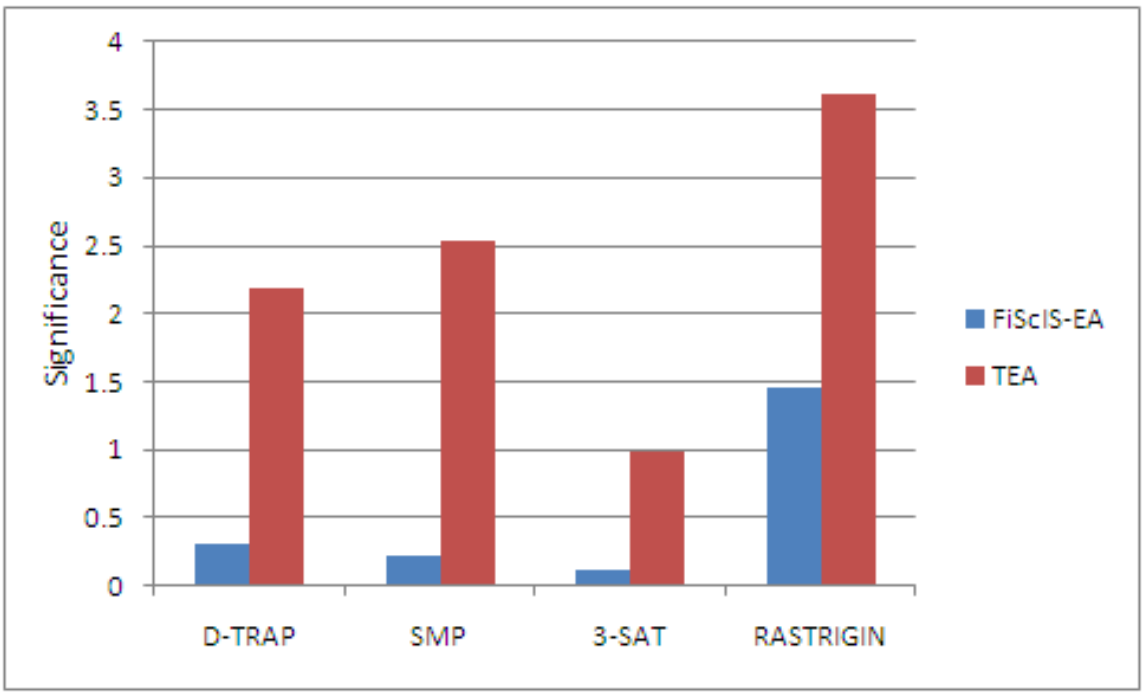

Figure 4.16. Sensitivity results of the performance of FiScIS-EA and TEA to μ_0 and μ, respectively.

inferior parents, caused by maintaining all individuals in the population. Even with a scalable parent selection operator, this still allows many individuals to be selected to mate that are of lower quality. While in some problems this is a desirable behavior, as it assists in holding off premature convergence, for the ONEMAX problem, it merely slows down the algorithm.

Examining the results obtained on the Spears' Multimodal problem, an unexpected behavior is observed. GC-EA either arrives at a successful solution very quickly compared to both FiScIS-EA and TEA, or it fails to arrive there at all. Unfortunately, this behavior lowers the MBF and the SR for GC-EA on this test problem, though the differences in MBF are not significant except when using 100 peaks. While this trade-off is not uncommon, GC-EA exhibited the opposite behavior on the ONEMAX problem. The only apparent reason for this difference is the difference in optimized values for λ. Both TEA and FiScIS-EA used a high value for λ, while GC-EA used a lower value. GC-EA performs better with lower values of λ both because it keeps the population size smaller and more manageable for a longer period of time, as well

Table 4.6. Comparison between FiScIS-EA, FiScIS-EA with restarts, GC-EA, GC-EA with restarts, Parameter-less GA and GASAP on the Spears' Multimodal Problem.

Problem	Measure	FiScIS -EA	FiScIS-EA + restarts	GC -EA	GC-EA + restarts	Parameter -less GA	GASAP
10	SR	100%	100%	100%	100%	—	93%
Peaks	AES	5750	5750	2100	2100	—	2060
	MBF	100	100	100	100	—	99.6
50	SR	60%	100%	50%	100%	100%	41%
Peaks	AES	5750	10483	2100	12663	40142	2098
	MBF	99.2	100	98.8	100	100.0	98.7
100	SR	47%	100%	20%	100%	96%	26%
Peaks	AES	5750	13667	2200	17268	74654	2341
	MBF	99.0	100	98.3	100	99.9	98.7

Table 4.7. Comparison between FiScIS-EA, FiScIS-EA with restarts, GC-EA, GC-EA with restarts, GPS-EA and GPS-EA with ELOOMS on the D-TRAP problem.

Problem	Measure	FiScIS -EA	FiScIS-EA + restarts	GC -EA	GC-EA + restarts	GPS -EA	GPS-EA w/ ELOOMS
25	SR	0%	0%	0%	0%	0%	0%
Traps	MBF	84.97	88.17	82.80	84.07	90.33	95.07
125	SR	0%	0%	0%	0%	0%	0%
Traps	MBF	82.27	82.97	78.08	78.47	83.55	86.25
250	SR	0%	0%	0%	0%	0%	0%
Traps	MBF	80.14	77.09	77.80	77.37	82.56	84.00

as improving its ability to converge upon a solution. In this case, while a smaller λ performed better than a larger one, the smaller value caused problems in finding the optimal peak.

The results obtained from the D-TRAP problem show that FiScIS-EA is significantly faster at converging than TEA, and generally finds solutions with similar MBF; however, on more difficult problems the differences between TEA and FiScIS-EA are significant. This is likely because of FiScIS-EA's ability to quickly remove undesirable individuals from the population, thus increasing convergence speed. However, even when coupled with a high λ to help maintain genetic diversity, TEA is more

capable of solving difficult instances of the D-TRAP problem. On the problems with 500 traps, the EAs all terminated due to the maximum number of fitness evaluations being reached, rather than from convergence; as such, they failed to achieve the expected quality of results. For D-TRAP, it is expected that at convergence, all traps should be comprised of homogeneous bits. Despite the fact that the evolution is tilted away from finding the optimal solution, about 1/16 of the randomly initialized traps should start with the optimal pattern, and a small portion should also be evolved during the run. Because of this, the expected outcome is at least $\frac{15}{16} \cdot 0.75 + \frac{1}{16} = 77\%$ of the maximum MBF.

The results from the 3-SAT tests showed that GC-EA produced results of significantly lower quality than either TEA or FiScIS-EA. Despite its ability to maintain genetic diversity throughout the experiment, GC-EA's solution quality tends to remain poor, even once the termination condition is met. This is likely due to a lack of need to preserve tremendous amounts of genetic information in binary problems. In this case, GC-EA would likely benefit from keeping fewer individuals while exploring more with a higher λ, similar to the parameters used for this problem with TEA. Comparing TEA with FiScIS-EA only shows a significant difference when using a 4:1 ratio, where FiScIS performs significantly better. There is no obvious explanation for this, and it might be a statistical anomaly.

Despite an unimpressive performance on the other four test problems, the GC-EA excelled on the Rastrigin function. On this test problem, it produced significantly better quality solutions using less time than either of the other algorithms. For this problem, the ability to maintain genetic information proves extremely useful as the algorithm is very capable of finding high-quality solutions very quickly. This behavior is somewhat expected from the GC-EA on real-valued problems, as the number of different possible solutions is much larger than on a binary problem. As such, maintaining genetic information is much more useful on real-valued problems like the Rastrigin function.

Figures 4.2, 4.5, 4.8, 4.11 and 4.14 demonstrate some unexpected behaviors of FiScIS-EA. The expected results were a high initial value for the population followed by a dramatic drop after the first generation to allow for more rapid evolution. The results show something considerably different, as only on the ONEMAX problem does

the population behave as expected. The other problems start with a moderate value for μ, but then it increases rapidly to a point where it then fluctuates up and down. For these problems μ tends to fluctuate between λ and 2λ, suggesting that FiScIS-EA's value of μ is influenced more by λ than anything else. While, in a sense, this does reduce the pressure on selecting good values for μ_0, it also places more pressure on selecting a good value for λ, which causes a similar need for manual tuning. To further demonstrate FiScIS-EA's lack of sensitivity to changes in μ_0, Table 4.5 shows that the MBF of FiScIS-EA is significantly less sensitive to changes in μ_0 than the MBF of TEA is sensitive to changes in μ.

Table 4.6 and Table 4.7 show that the addition of restarts to EAs is not always beneficial. While the SR of the SMP tests was improved dramatically, the MBF observed on the D-TRAP problem was not. This makes some sense when examining the data obtained from those test problems without the usage of restarts. The optimal solution for the SMP was always within two standard deviations of the MBF. This, combined with the high convergence speed allowed the algorithms to utilize restarts to find the optimal solution every time. The D-TRAP problem, however, lacks both of those qualities. The optimal solution was never close to the MBF, and convergence speed for the D-TRAP problem is usually slow. As such, the MBF improved by a small amount, only reaching one to two standard deviations higher than before, at best.

Also worth noting here is that, even though they required fewer fitness evaluations to tune than TEA, both FiScIS-EA and GC-EA still required a significant amount of extra evaluations when tuned as stated in Section 4.2. Because of the large number of fitness evaluations used, possible improvements in this area could be obtained by combining either FiScIS-EA or GC-EA with a powerful parameter tuning method, such as REVAC [17].

5. CONCLUSIONS AND FUTURE WORK

EAs are powerful algorithms capable of solving difficult problems. However, this power comes with a considerable investment in time, not just in the tuning of the algorithm, but in learning how to properly tune an algorithm as well. While many different methods of parameter control have been developed, only a few methods have been created for controlling the population size, and all these methods have significant drawbacks. This thesis examined optimized value sequences for μ, explored controlling μ, introduced two novel methods for controlling μ, and compared their performance on a diverse set of test problems.

The examination of optimal values for μ was handled by a meta-EA and showed not only the value of a dynamic population, but also a considerable amount of insight as to what properties an optimal control scheme for μ requires. In particular, the higher initial value of μ warrants further exploration, as this was present in all of the optimal population size sequences.

The first novel method introduced was FiScIS-EA which works by replacing standard survivor selection with a random chance applied to each individual to survive to the next generation. The chance of survival for a given individual is simply that individual's fitness when all of the fitness values in the current population are scaled between zero and one. FiScIS-EA proved to be very powerful, but also showed some draw backs. While generally it obtained good results, it failed to remove μ as a parameter, instead replacing it with μ_0, to which performance is less sensitive, and increased the pressure placed on picking a good value for λ. Despite the original idea that went into this algorithm's design, it was generally found to have higher values for μ than μ_0 throughout most of the evolution, as opposed to the high initial value followed my a sharp drop that was observed in the optimal values found by the meta-EA.

The second novel method introduced was GC-EA. This method operates under the theory that old individuals should not be discarded unless it is absolutely necessary. To this end, it uses a small initial population and a high maximum population. At each generation, individuals are only removed from the population if the maxi-

mum population has been exceeded. In this manner, genetic information is preserved so that those genes can be used in future generations. GC-EA generally performed rather poorly; however, that might be more caused by a selection of test problems that are mostly outside its area of specialty. Since it only performed well on the Rastrigin function, it is reasonable to assume that it functions better on problems that share similarly large search spaces, including other real-valued problems. This also provides some amount of insight on how much genetic diversity must be maintained on various test problems. It would make sense that more diversity maintenance would be needed for a real-valued problem, as they generally have a much larger search space than a binary problem. As such, GC-EA performs significantly better on the Rastrigin function than the other algorithms. This is likely due to its better ability to maintain diversity.

While the usefulness of a dynamic population has been demonstrated through the use of a meta-EA, the methods for implementing an algorithm with a constantly optimal population size is still elusive. While both FiScIS-EA and GC-EA provide methods for altering the population size during the execution of the EA, neither are capable of maintaining an optimal population size. As such, more examination of what makes a population scheme perform well or poorly is required before significant advances can be made.

While the usefulness of both FiScIS-EA and GC-EA has already been shown to a limited extent, additional test problems are needed to clarify the types of problems that each algorithm excels on. In particular, GC-EA needs to be tested on additional real-valued test problems to substantiate the hypothesis that it will outperform FiScIS-EA on such problems because of its ability to deal with larger search spaces. Additionally, employing a meta-EA to further explore the optimal values for μ on more test problems is expected to provide more insight into what values for μ are most beneficial at various points in the evolution. These insights could then in turn be used to improve the performance of existing algorithms, or to create entirely new algorithms.

In addition to removing μ as a parameter, removing other parameters simultaneously may dramatically improve the performance of an EA. Since the quality of a parameter is based on what values for other parameters are being used as well as

its own value, altering only a single parameter has limits to what it can achieve. By allowing alterations to more parameters at the same time, more useful combinations become possible, potentially reaching past what limits are currently present. For instance, combining dynamic population sizing with dynamic offspring sizing [18] would allow for more extensive control of the ratio between exploration and exploitation. Also, altering selection operators can allow for a wider range of acceptable values for population size. For example, selective pressure needs to be at different values for different population sizes.

BIBLIOGRAPHY

[1] J. Arabas, Z. Michalewicz, and J. Mulawka. GAVaPS-a genetic algorithm with varying population size. In *Proceedings of the First IEEE Conference on Evolutionary Computation*, pages 73–78, 1994.

[2] T. Back. Self adaptation in genetic algorithms. In *Towards a Practice of Autonomous Systems: Proceedings of the 1st European Conference on Artificial Life*, pages 263–271, 1992.

[3] T. Back. The interaction of mutation rate, selection and self-adaptation within a genetic algorithm. In *Proceedings of the 2nd Conference on Parallel Problem Solving from Nature*, pages 85–94, 1992.

[4] T. Back. Optimal mutation rates in genetic search. In *Proceedings of the 5th International Conference on Genetic Algorithms*, pages 2–8, 1993.

[5] T. Back, A. Eiben, and N. A. L. van der Vaart. An empirical study on GAs without parameters. In *Proceedings of PPSN VI: the Sixth International Conference of Parallel Problem Solving form Nature*, pages 315–324, 2000.

[6] J. Clune, S. Goings, B. Punch, and E. Goodman. Investigations in meta-GAs: panaceas or pipe dreams? In *Proceedings of GECCO 2005 Workshops on Genetic and Evolutionary Computation*, pages 235–241, 2005.

[7] J. E. Cook and D. R. Tauritz. An Exploration into Dynamic Population Sizing. In *Proceedings of GECCO 2010: the Genetic and Evolutionary Computation Conference*, 2010.

[8] A. Eiben, M. Schut, and A. deWilde. Is self-adaptation of selection pressure and population size possible? - a case study. In *Proceedings of PPSN IX: the Ninth International Conference on Parallel Problem Solving from Nature*, pages 900–909, 2006.

[9] A. E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing*. Springer, 2007.

[10] G. Harik and F. Lobo. A parameter-less genetic algorithm. In *Proceedings of GECCO 1999: the Genetic and Evolutionary Computation Conference*, volume 1, pages 258–265, 1999.

[11] E. Holdener. *The Art of Parameterless Evolutionary Algorithms*. PhD thesis, Missouri University of Science and Technology, 2008.

[12] R. M. J. Hesser. Towards an optimal mutation probability for genetic algorithms. In *Proceedings of the First Conference on Parallel Problem Solving from Nature*, pages 23–32, 1991.

[13] T. F. J.E. Smith. Recombination strategy adaptation via evolution of gene linkage. In *Proceedings of the 1996 IEEE Conference on Evolutionary Computation*, pages 826–831, 1996.

[14] T. F. J.E. Smith. Self adaptation of mutation rates in a steady state genetic algorithm. In *Proceedings of the 1996 IEEE Conference on Evolutionary Computation*, pages 318–323, 1996.

[15] K. A. D. Jong, M. A. Potter, and W. M. Spears. Using problem generators to explore the effects of epistasis. In *Proceedings of the International Conference on Genetic Algorithms*, pages 338–345, 1997.

[16] F. G. Lobo and C. F. Lima. Revisiting evolutionary algorithms with On-the-fly Population Size Adjustment. In *Proceedings of GECCO 2006: the 8th annual conference on Genetic and Evolutionary Computation*, pages 1241–1248, 2006.

[17] V. Nannen, S. K. Smit, and Á. E. Eiben. Costs and Benefits of Tuning Parameters of Evolutionary Algorithms. In *Proceedings of PPSN X: the Tenth International Conference on Parallel Problem Solving from Nature*, pages 528–538, 2008.

[18] A. Nwamba. Automated Offspring Sizing in Evolutionary Algorithms. Master's thesis, Missouri University of Science and Technology, 2009.

[19] J. Smith. Modeling GAs with self-adaptive mutation rates. In *Proceedings of GECCO 2001: the Genetic and Evolutionary Computation Conference*, pages 599–606, 2001.

[20] E. A. Smorodkina and D. R. Tauritz. Greedy Population Sizing for Evolutionary Algorithms. In *Proceedings of CEC 2007 - IEEE Congress on Evolutionary Computation*, pages 2181–2187, 2007.

[21] E. A. Smorodkina and D. R. Tauritz. Toward Automating EA Configuration: the Parent Selection Stage. In *Proceedings of CEC 2007 - IEEE Congress on Evolutionary Computation*, pages 63–70, 2007.

[22] M. Solano and I. Jonyer. Performance Analysis of Evolutionary Search with a Dynamic Restart Policy. In *Proceedings of the Twentieth International Florida Artifical Intelligence Research Society Conference*, pages 186–187, 2007.

[23] C. R. Stephens, I. G. Olmedo, J. M. Vargas, and H. Waelbroeck. Self Adaptation in Evolving Systems. In *Artifical Life*, volume 4, pages 183–201. 1998.

[24] G. Syswerda. A study of reproduction in generational and steady state genetic algorithms. In B. M. Spatz, editor, *Foundations of Genetic Algorithms*, pages 94–101. Morgan Kaufmann Publishers, San Mateo, CA, 1991.

VITA

Jason Edward Cook was born in Hillsboro Missouri. He graduated from Hillsboro High School in May of 2003 and enrolled as a undergraduate at University of Missouri Rolla (now Missouri University of Science and Technology) in the fall of that year. He received a BS in Computer Science in December of 2006, and enrolled in the Computer Science graduate program for the following year. He received his Master's degree in August 2010.