Generational Parallel Varying Mutation GAs and their Applications

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at Shinshu University

Hernán Eduardo Aguirre Durán

March 2003
Shinshu University
Japan

Dedication

To my parents.

Contents

List of Figures

List of Tables

Abstract

Evolutionary algorithms (EAs) describe computer-based problem solving systems that use computational models of evolutionary processes as primary elements of its design. Global search abilities, adaptation to the task in hand, and robust performance are favorable characteristics of EAs making them successful solving complex optimization problems.

This work focuses on genetic algorithms (GAs), one of the mainstreams of EAs. Canonical GAs use a binary representation for the individuals, i.e. bit strings with $\{0,1\}$ alphabet. Offspring are generated by randomized process intended to model recombination and mutation. Crossover exchanges segments of information between pairs of individuals with probability p_c and then mutation inverts bits with a probability p_m per bit. In the absence of crossover $(1 - p_c)$ mutation is applied alone. The canonical GA emphasizes crossover as the main genetic operator and considers mutation only as a secondary "background" operator to be used with very small probability.

In addition, a common practice has been to run the GA with its search strategies fixed. A run of a GA, however, is itself an intrinsically dynamic, adaptive process, and there is experimental evidence suggesting that different strategies might be optimal at different stages of the evolutionary process. Methods that modify its search strategies during the run of the algorithm are considered as one of the most important and promising areas of research in evolutionary algorithms, because they could lead to significant improvements in performance. Among these, some approaches that seek to combine crossover with (higher) varying mutation rates during the course of a run have been proposed recently. It has been shown that deterministically varying mutation rates over the generations and/or across the representation can improve the performance of GAs. Self-adaptive mutation rate schedules have also been proposed. Self-adaptive algorithms evolve simultaneously strategy parameters and object variables and are regarded as the most promising way of combining forms of control (strategy parameters co-adaptation).

From the application of operators standpoint, deterministic, adaptive, and self-adaptive varying mutation GAs have been mostly designed similar to a canonical GA. That is, crossover is applied with high probability p_c and then follows mutation. Thus, under these standard varying mutation approaches, higher mutations are mostly applied serial to crossover. This model of standard varying mutation GAs raises several important questions regarding the interference between crossover and high mutation, how this affects performance of the algorithm, whether this affect the mutation rate control itself in the case of (adaptive) self-adaptive varying mutation algorithms, and more generally whether this is an appropriate model for combining forms of control (co-adaptation of strategy parameters).

This work proposes a model of parallel varying mutation GA that addresses the questions raised by the standard model of varying mutation GAs. The proposed model detaches varying mutation from crossover, applies "background" mutation (or none at all) after crossover (CM) and varying mutations (SRM) only parallel to crossover, putting the operators CM and SRM in a cooperative-competitive stand with each other by subjecting their offspring to extinctive selection. The model relies in adaptive (self-adaptive) mutation schedules to increase the effectiveness of SRM and enhance selection by eliminating fitness duplicates, which postpones genetic drift and creates a fair competition between the offspring created by both operators. The motivation of this work comes from the desire to design effective and efficient varying mutation GAs that can be used in real-world applications.

There are several advantages in the proposed model. First, it gives an efficient framework to

achieve better balances for varying mutation and crossover in which the strengths of the operators can be kept without interfering one with the other. Second, since varying mutation is detached from crossover, the instantaneous effectiveness of the varying mutation operator depends only upon itself and its relative success can be directly tied to the mutation rate to create adaptive (self-adaptive) schemes for mutation rate control. The same can be said for crossover, especially if no "background" mutation is applied after it. Third, parallel varying mutation can be studied on its own seeking to increase the performance of GAs. Fourth, the individual roles and the interaction of crossover and varying mutation throughout the run of the algorithm can be better understood, which could be important for co-adaptation studies.

The model is studied using two test problem generators. One of the generators is for 0/1 multiple knapsack problems, which allows to test the model on a broad range of classes of constrained problems by varying the feasible region of the search space, number of constraints, and the size of the search space. Real-world 0/1 multiple knapsack problems with known global optimum are also used. These latter problems allow studying the global search abilities of the algorithms. The second test problem generator is the well known Kauffman's NK-Landscapes. NK-Landscapes are stochastically generated fitness functions on bit strings, parameterized with N bits and K epistatic interactions between bits. The term epistasis describes nonlinearities in fitness functions due to changes in the values of interacting bits. For a giving N, we can tune the ruggedness of the fitness function by varying K. In the limits, $K = 0$ corresponds to a model in which there are no epistatic interactions and the fitness contribution from each bit value is simply additive, which yields a single peaked smooth fitness landscape. On the opposite extreme, $K = N - 1$ corresponds to a model in which each bit value is epistatically affected by all the remaining bit values, yielding a maximally rugged fully random fitness landscape. Varying K from 0 to $N - 1$ gives a family of increasingly rugged multi- peaked landscapes. NK-Landscapes allow testing the model in a broad range of classes of epistatic, non-linear, problems.

First, the internal structure of the proposed model (GA-SRM) is studied in depth using an adaptive schedule for mutation. Important structural issues are the balance for offspring creation between CM and SRM, the ratio between number of parents and number of offspring (extinctive selection pressure), "background" mutation probability in CM, and the threshold to trigger adaptation in SRM. The effect of population size and number of evaluations is observed, too. Two mutation strategies to select the bits that will undergo mutation are investigated. In addition, the importance and effect on performance of extinctive selection and the interaction of varying mutation parallel to crossover is assessed. It is found that GA-SRM greatly improves the performance of GAs for global optimization in constrained problems. Extinctive selection accelerates the search process and parallel varying mutation increase the convergence reliability of the algorithm. Robustness even with small populations is also a remarkable characteristic observed in the improved GA-SRM. Mutation strategy for parallel varying mutation turns out to be an important issue to improve the performance of parallel varying mutation.

Second, the proposed model is compared with the standard model of varying mutations GAs across a broad range of problems using deterministic and self-adaptive schedules for mutation rate control. The statistical significance of the results is verified with ANOVA tests. It is found that the proposed model is more effective and efficient than the standard model. In deterministic varying mutation GAs, a GA with varying mutation parallel to crossover showed faster convergence and higher robustness to initial settings of mutation rate than a GA with varying mutation serial to crossover. In self-adaptive varying mutation GAs, the convergence velocity of a parallel self-adaptive mutation GA was matched by a serial self-adaptive mutation GA only when initial diversity of parameters was allowed. Convergence reliability of the parallel varying mutation model was higher than the standard model of varying mutation in both deterministic and self-adaptive

GAs. It was also found that the standard model of varying mutations in fact affects negatively the (adaptive) self-adaptive mutation rate control. This strongly suggests that the standard model of varying mutation GAs may not be appropriate for combining forms of control.

Then, the behavior of the parallel varying mutation GA-SRM is examined on epistatic problems using NK-Landscapes. Properties of NK-Landscapes are discussed and the effect on performance of selection, drift, mutation, and recombination is verified. Mutation strategy for the varying mutation operator is also studied in detail. Experiments are conducted using NK-Landscapes with nearest neighbor and random patterns of epistasis. Comparisons are made with a canonical GA, a simple GA with extinctive selection, a mutation only EA, and a random bit climber RBC+. It is shown that GAs can be robust algorithms on NK-Landscapes postponing drift by eliminating fitness duplicates and using selection pressure higher than a canonical GA. Different to previous works, even simple GAs with these two features perform better than the single bit climber RBC+ for a broad range of classes of problems. It is also shown that the interaction of parallel varying mutation with crossover (GA-SRM), similar to constrained problems, improves further the reliability of the GA. Contrary to intuition it is found that a mutation only EA can perform as well as GA-SRM that includes crossover for small values of K, where crossover is supposed to be advantageous; but the relative importance of crossover interacting with varying mutation increases with K performing better than mutation alone for medium and high K. Better overall performance by population based mutation only evolutionary algorithms over random bit climbers is also observed. With regards to mutation strategy for parallel varying mutation, it is found that a dynamic segment mutation strategy improves further the performance of GAs on problems with nearest neighbor patterns of epistasis.

After analyzing the proposed model and comparing it with the standard model of varying mutation GAs, it is shown that the fundamental concept of the model can be successfully extended to other important classes of GAs and that it can be effectively applied to real-world problems.

An important area of research is the parallelization of GAs. Evolutionary algorithms are population based methods and it is considered that its full potential would come from implementing the algorithm in parallel architectures. It is shown that the proposed model extended to a parallel distributed GA (DGA-SRM) achieves higher search speed, higher convergence reliability, and less communication cost for migration than a canonical distributed GA. It is also shown that DGA-SRM scales up better as the difficulty of the problem increases and tolerates population reductions better than a canonical distributed GA.

Next, it is shown that GA-SRM can be successfully applied to real world problems in which efficiency in processing time and computer memory is a major issue. The improved GA-SRM is extended to the two dimensional image halftoning problem and an accelerated image halftoning technique using GA-SRM with tiny populations is proposed. Simulation results verify that the proposed scheme impressively reduces computer memory and processing time, making the improved approach appealing for practical implementation.

Furthermore, it is shown that the concept of GA-SRM can also be effective for multi-objective optimization of real world applications, which is important due to the multi-objective nature of most real-world problems. The improved GA-SRM is extended to a multi-objective optimization GA to simultaneously generate halftone images with various combinations of gray level precision and spatial resolution. Simulation results verify that the proposed scheme can effectively generate several high quality images simultaneously in a single run reducing even further the overall processing time.

Finally, this work is summarized presenting conclusions and suggesting future research.

Chapter 1

Introduction

This chapter presents a brief introduction to evolutionary algorithms. Then, it describes the background, motivation and goals of this research. Finally, it outlines the contents of this work.

1.1 Evolutionary Algorithms

An evolutionary algorithm describes a computer-based problem solving system that uses computational models of evolutionary processes as primary elements of its design and implementation.

The origins of evolutionary inspired algorithms for optimization and machine learning can be traced to the 1950s and 1960s[1, 2, 3, 4, 5, 6]. In addition, several evolutionary biologists used computers to simulate evolution for the purpose of controlled experiments[7, 8, 2, 9, 10]. However, historically the three mainstream instances of evolutionary algorithms that have received considerably attention are Genetic Algorithms[11], Evolution Strategies[12, 13], and Evolutionary Programming[14]. These approaches have inspired the development of additional evolutionary algorithms such as genetic programming and classifier systems. Moreover, hybridizations of evolutionary algorithms with other soft computing techniques, such neural networks and fuzzy systems, or with other search heuristics, such local search, tabu search, and simulated annealing, are intensely being developed.

Although a variety of evolutionary algorithms have been proposed, all of them share the following general and basic properties: (i) Evolutionary algorithms utilize the collective learning process of a population of individuals. Each individual represents and encodes a search point in the space of potential solutions to a given problem. (ii) By means of evaluating individuals in their environment, a measure of quality or fitness can be assigned to the individuals. According to the quality measure, a selection process favors fitter individuals to reproduce more often than those that are relatively less qualified. (iii) Descendants of individuals are generated by randomized process intended to model mutation and recombination. Mutation corresponds to an erroneous self-replication of individuals and recombination interchanges information between two or more individuals.

Figure 1.1 outlines a typical evolutionary algorithm (EA). A population P of individuals is initialized and then evolved from generation t to generation $t+1$ by repeated application of fitness evaluation, selection, recombination, and mutation. An evolutionary algorithm typically initializes its population randomly, although domain-specific knowledge can also be used. Evaluation measures the fitness of each individual according to its worth in some environment (problem). Fitness evaluation can be as simple as computing a function or as complex as running an elaborate simulation. Selection is often performed in two steps, parent selection and survival. Parent selection decides who becomes parent and how many children the parents have; higher-fitness individuals are more likely to be parents and have more offspring. Offspring are created via recombination and mutation. Recombination exchanges information between parents and mutation further perturbs the offspring. The offspring are then evaluated. Finally the survival step decides who survives in the population.

Evolutionary algorithms have been used in a large number of scientific and engineering problems and models. Some of the areas where evolutionary algorithms are being used are optimization, automatic programming, machine learning, economics, immune systems, ecology, population genetics, evolution and learning, and social systems[15].

1.1.1 Genetic Algorithms

Genetic Algorithms (GAs) were invented and developed by Holland[11]. Holland's original goal was to formally study the phenomenon of natural adaptation and to develop ways in which its mechanism might be imported into computer systems. GAs were presented as an abstraction of biological evolution and derived its behavior from a genetic/evolutionary metaphor.

Traditionally, GAs use a binary representation for the individuals (chromosomes or structures).

```
procedure EA();
begin
 t = 0; /* Initial Generation */
 initialize_population(P(t));
 evaluate(P(t));
 while (not termination condition)
 begin
  P'(t) = select_parents(P(t));
  P''(t) = recombine(P'(t));
  P'''(t) = mutate(P''(t));
  evaluate(P'''(t));
  P(t + 1) = select_survivors(P'''(t) ∪ Q); /* Q ∈ {∅, P(t)} */
  t = t + 1; /* Next Generation */
 end
end
```

Figure 1.1: The outline of an evolutionary algorithm

Recently, however, many applications have focused on other representations such integers, real-valued vectors, graphs (neural networks), Lisp expressions, and ordered lists.

Selection is a probabilistic function based on relative fitness. With this selection method, known as fitness-proportional selection, the expected number of times an individual will be selected to reproduce is the individual's fitness divided by the average fitness of the population. A simple method of implementing fitness-proportional selection is roulette-wheel sampling[16]. The number of offspring created is the same as the number of parents μ. Later, in the survivors selection step, the μ newly created offspring will replace the μ parents in the population. This form of selection is not elitist[1].

Offspring are created by recombination (crossover) of parent individuals with probability p_c. After that, mutation is applied with a very small probability p_m per bit. In its initial conception, GAs emphasize recombination (crossover) as the primary search operator and apply mutation solely as a "background operator". Interest in mutation has increased recently, partly due to the influence of Evolution Strategies and Evolutionary Programming.

1.1.2 Evolution Strategies

Evolution strategies (ESs) were developed by Rechenberg[12], using selection, mutation, and a population of one parent and one offspring. Schwefel[13] introduced recombination and populations with more than one individual, and compared ESs with more traditional optimization techniques.

Evolution strategies typically use real-valued, vector representations. Individuals to be parents are selected randomly from a uniform distribution. The number of offspring λ created is greater than the number of parents μ. The selection of survivors is deterministic and is implemented in one of two methods. The first method selects the best μ out of λ offspring and replaces the parents with this newly created individuals. In other words, only the best μ offspring are allowed to survive. This method is known as a (μ,λ) selection strategy. The second method selects the best μ individuals among μ parents and λ offspring. Thus, in this method both the best parents

[1]Elitist: the best individual always survive, ensuring that once an optimum is found it cannot be lost.

and offspring are allowed to survive. This second method is known as a $(\mu + \lambda)$ selection strategy. Both methods belong to the kind of extinctive (truncation) selection methods. $(\mu + \lambda)$ selection is elitist but (μ,λ) selection is not.

Offspring are created by recombination (when $\mu > 1$) of parent individuals followed by mutation. A variety of different recombination mechanisms are currently used in ESs and the operators are sexual and panmictic. In sexual operators, recombination acts on two randomly chosen parent individuals. Conversely, in panmictic operators, one parent is randomly chosen and held fixed while for each component of its vectors the second parent is randomly chosen anew from the population. Mutation perturbs the individuals using a normal distribution with expectation zero. In ESs considerably effort has been put on (self) adapting the mutations during the run of the algorithm. ESs allow each individual (or each variable within the individual) to have adaptive mutation rates that are normally distributed with a zero expectation.

ESs emphasize recombination and mutation as essential operators for searching simultaneously in the variables space and in the strategy parameters space (self-adaptation).

1.1.3 Evolutionary Programming

Evolutionary programming (EP) was developed by Fogel et al. [14]. EP arose from the desire to generate machine intelligence using selection and mutation to evolve finite-state machines.

EP traditionally has used representations for the individual that are tailored to the problem domain. In the case of finite-state machine applications, the individuals within the population are represented as graphs. For other applications, appropriate representations such real-valued vectors and ordered lists are used.

Selection is a probabilistic function based on tournament. The number of offspring created is the same as the number of parents μ. In the survivors selection step, μ individuals are chosen from the 2μ (parents and offspring) individuals. This form of selection is elitist and can be considered to be a $(\mu + \mu)$ selection strategy.

Offspring are created by mutation of parent individuals. The form of mutation is based on the representation used, and similar to ESs is often (self) adaptive. For real valued optimization problems, for example, it works with normally distributed mutations with expectation zero and extends the evolutionary process to the strategy parameters (self-adaptation). The forms of mutation used are usually quite flexible and can produce perturbations similar to recombination, if desired. However, EP emphasizes mutation and does not incorporate the recombination of individuals. The justification for this is that in EP each individual is usually viewed as the analog of a species, and there is no sexual recombination between species.

1.2 Background

This work concentrates on GAs. Canonical GAs use a binary representation for the individuals, i.e. bit strings with $\{0,1\}$ alphabet. Crossover exchanges segments of information between pairs of chromosomes with probability p_c, and mutation inverts bits with a probability p_m per bit. Crossover and mutation are applied one after the other; in the absence of crossover $(1 - p_c)$, mutation is applied alone. Holland[11] defined crossover as the main genetic operator and considered mutation only as a "background" operator to be used with very small mutation rates. The role of crossover is to construct high order building blocks (hyperplanes) from low order ones; whereas the primary role of mutation is to replace allele values lost from the population, assuring that crossover has a full range of alleles so that the "adaptive plan" is not trapped on local optima.

Common settings recommended for crossover probability p_c are 0.60[17], 0.95[18], and 0.75-0.95[19]. Similarly, common settings recommended for the mutation probability p_m are 0.001[17], 0.01[18], and 0.005-0.01[19]. All these values were obtained by experimental investigations and the combined setting of p_c and p_m, especially in [19], usually depends on population size.

Mutation rates within the ranges mentioned above are still widely used in applications of canonical GAs (i.e. using binary representation), because these settings are consistent with Holland's proposal for mutation as a background operator and Goldberg's recommendation to invert on the order of one thousand bits by mutation[16]. Recently, some analytical results concerning optimal schedules of the mutation rate in the cases of simple objective functions and simplified genetic algorithms suggest that a $1/\ell$ constant mutation rate is almost optimal[20, 21, 22, 23], where ℓ is the bit string length. It is important to mention that not useful analytical results are known for the dependence of the optimal mutation rates on offspring population size[24].

In addition, a common practice has been to run the GA with its parameters set to constant values. A run of a GA, however, is an intrinsically dynamic, adaptive process, and there is experimental evidence suggesting that different values of parameters might be optimal at different stages of the evolutionary process. In order to pursue better balances for crossover and mutation, parameter control methods that modify the values of the strategy parameters during the run of the algorithm by taking into account the actual search process are being proposed. These methods are an alternative form to the common practice of tuning parameters "by hand" and are considered as one of the most important and promising areas of research in evolutionary algorithms[25].

One way to pursue better dynamic balances is to use adaptive or self-adaptive mechanisms to control the rate of operators[26, 27, 28]. Another approach seeks to combine crossover with (higher) varying mutation rates during the course of a run. It has been shown that deterministically varying mutation rates over the generations and/or across the representation can improve the performance of GAs[29, 30, 31]. Self-adaptive mutation rate schedules inspired from Evolution Strategies and Evolutionary Programming have also been proposed to control the mutation rate of generational and steady state GAs[31, 32, 33]. The deterministic approach uses one mutation rate for all individuals in the population. Conversely, the principle of self-adaptation incorporates strategy parameters into the representation of individuals evolving simultaneously strategy parameters and object variables. The self-adaptive approach is regarded as the method having the advantage of reducing the number of exogenous parameters[33] and is thought to be the most promising way of combining forms of control (parameters co-adaptation)[25].

1.3 Issues and Goals

Varying mutation approaches differ from canonical GAs mainly in the mutation rate control. Also, some approaches use a selection mechanism with higher selection pressure; for example, (μ,λ) selection instead of proportional selection. However, from the application of operators standpoint, deterministic, adaptive, and self-adaptive varying mutation GAs have been mostly designed similar to a canonical GA. That is, crossover is applied with probability p_c and then follows mutation; in the absence of crossover $(1 - p_c)$, mutation is applied alone. As mentioned above, the probability p_c is usually set to 0.6 and higher values are often used. Thus, under these standard varying mutation approaches, higher mutations are mostly applied serial to crossover.

This model of standard varying mutation GAs raises several important questions such as: Is there interference between crossover and high mutation? If so, does it affect performance of the algorithm? In the case of (adaptive) self-adaptive varying mutation algorithms, does it affect the mutation rate control itself? And more generally, is it an appropriate model for combining forms

of control (co-adaptation of parameters)?

The objective of this work is to design effective and efficient varying mutation GAs that can be used in real world application. In order to achieve this goal it is important to explore models of varying mutation GAs that address the questions raised by the standard model of varying mutation GAs.

An alternative to standard varying mutation methods is to design approaches that apply background mutation after crossover (or none at all) and higher mutations only parallel to crossover. There are several advantages in these models. First, such approaches could give an efficient framework to achieve better balances for varying mutation and crossover in which the strengths of the operators can be kept without interfering one with the other. Second, since varying mutation is detached from crossover, the instantaneous effectiveness of the varying mutation operator depends only upon itself and its relative success can be directly tied to the mutation rate to create adaptive (self-adaptive) schemes for mutation rate control. Third, parallel mutation can be studied on its own. For example, higher mutation rates raise the question of mutation strategy and its relevance to a given epistatic pattern or class of problems. Fourth, the individual roles and the interaction of crossover and varying mutation throughout the run of the algorithm can be better understood, which could be important for co-adaptation studies.

From this point of view, this work explores a model of GA that applies varying mutations parallel to crossover & background mutation putting the operators in a cooperative-competitive stand with each other by subjecting their offspring to extinctive selection (GA-SRM)[34, 35]. Adaptation and mutation strategy are designed to improve the effectiveness of parallel varying mutations. Selection is also enhanced by eliminating fitness duplicates to postpone drift and to create a fair competition between operators.

Two test problem generators are used to study systematically the proposed model. One of the generators is for 0/1 multiple knapsack problems (constrained problems). The other one is the well known Kauffman's NK-Landscapes (epistatic non-linear problems). These generators are well suited to study the behavior of the model on classes of problems which characteristics resemble those of the difficult, constrained, and non-linear problems found in real world-applications. The model of parallel varying mutation GAs shall be compared against the standard model of varying mutation GAs using deterministic and self-adaptive schedules for mutation rate control. This will help to clarify and answer some of the questions raised above. The model should be able to prove its worth in a real-world application and in order to have wide applicability the fundamental concept of the model should be extended to other important classes of genetic algorithms such distributed GAs and multiobjective GAs.

1.4 Outline

The central theme of this work is the design of efficient and effective generational parallel varying mutation GAs that can be used in practical applications to optimize difficult and highly constrained problems.

Chapter 2 describes and contrasts two models of designing generational varying mutation GAs. One of the models is a simple extension of a canonical GA that applies varying mutations mostly after crossover, which has been the standard approach for designing generational varying mutation GAs. The second, called GA-SRM, is the proposed model that applies varying mutations only parallel to crossover.

Chapter 3 describes two test problem generators used in this work for testing the performance of genetic algorithms (GAs). One is a test problem generator for 0/1 multiple knapsacks problems,

and the other one is the well known Kauffman's NK-landscapes model.

Three chapters are dedicated to study the model of parallel varying mutation GAs. Chapter 4 focuses on studying the structure of the parallel varying mutation model GA-SRM. Important structural issues include the balance between operators, the importance of mutation after crossover, the contribution of parallel varying mutation to the search, extinctive pressure, and the threshold parameter to trigger adaptation of the parallel varying mutation operator. The search velocity and search reliability of the model is observed under various evaluation times and different population sizes. Chapter 5 studies and compares in detail the impact on performance (convergence reliability and convergence velocity) of extinctive selection and higher mutations in the standard and parallel models of varying mutation GAs. Deterministic and self-adaptive mutation rate controls are used for varying mutation in both models of GA. Chapter 6 examines the behavior of GA-SRM on epistatic problems and looks into the effect of elimination of *fitness duplicates* to further improve the model.

Another three chapters are dedicated to show that the fundamental concept of GA-SRM can be extended successfully to other important classes of GAs, such parallel and multiobjective GAs, and that it can be effectively applied to real world problems.

An important area of research is the parallelization of GAs. Evolutionary algorithms are population based methods and it is considered that its full potential would come from implementing the algorithm in parallel architectures. Most models of parallel GAs have considered the parallelization of the evaluation function, the structure of the population, and the selection strategy. However, from a processing time standpoint, the parallel application of operators has been overlooked because it is considered that only minor gains would come from parallelizing simple operators. Chapter 7 extends GA-SRM to a parallel distributed GA (DGA-SRM) arguing that the parallel application of crossover and higher varying mutations within parallel GAs is worth exploring, because the parallelization of operators would exploit their interaction in a more effective way achieving significant gains in performance and robustness. Simulation results show that DGA-SRM achieves higher search speed and higher convergence reliability with less communication cost for migration. It is also shown that DGA-SRM scales up better as the difficulty of the problem increases and tolerates population reductions better than a canonical distributed GA.

Chapter 8 shows that GA-SRM can be successfully applied to real world problems in which efficiency in processing time and computer memory is a major issue. Here, the improved GA-SRM is extended to the two dimensional image halftoning problem and an accelerated image halftoning technique using GA-SRM with tiny populations is proposed. Simulation results show that the proposed scheme impressively reduces computer memory and processing making the improved approach appealing for practical implementation.

The multiobjective nature of most real-world problems makes multiobjective optimization a very important research topic. Chapter 9 shows that the concept of GA-SRM can also be effective for multiobjective optimization of real world applications. The improved GA-SRM is extended to a multiobjective optimization GA to simultaneously generate halftone images with various combinations of gray level precision and spatial resolution. Simulation results verify that the proposed scheme can effectively generate several high quality images simultaneously in a single run reducing even further the overall processing time.

Finally, Chapter 10 summarizes this work, presents conclusions, and suggests future research.

Chapter 2

Generational Varying Mutation Genetic Algorithms

This chapter starts with a brief description of the main components of a canonical genetic algorithm. Then, it describes the standard approach that has been used for designing generational varying mutation GAs. This model is a simple extension of a canonical GA in which varying mutations are mostly applied after crossover. Finally, it presents in detail the proposed model of GA that applies varying mutations only parallel to crossover.

2.1 A Canonical Genetic Algorithm

A canonical GA[11, 16] selects individuals from the parent population $P(t)$ with a selection probability proportional to their fitness and applies crossover with probability p_c followed by mutation with a very small constant mutation probability p_m per bit (background mutation). In the absence of crossover $(1 - p_c)$ mutation is applied alone.

From the application of operators standpoint, it can be said that within the canonical GA the probability of crossover p_c enables an implicit parallel application of two operators. One of the operators is crossover followed by mutation (CM) and the other one is mutation alone (M). Figure 2.1 presents a block diagram of the canonical GA and illustrates the implicit parallel application of CM and M.

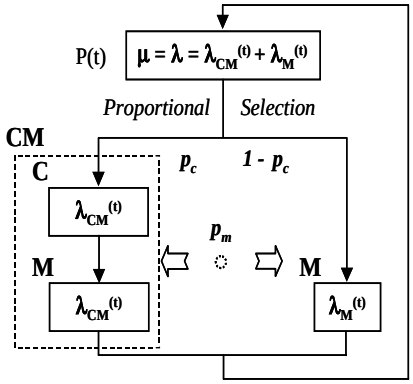


Figure 2.1: Canonical Genetic Algorithm

It should be noted that mutation in both CM and M is governed by the same constant mutation probability p_m and applies the same (bit by bit) mutation strategy. The number of offspring created by CM and M, $\lambda_{CM}^{(t)}$ and $\lambda_{M}^{(t)}$, respectively, depends on the probability of crossover p_c and may vary at each generation t due to the stochastic process. However, the total number of offspring λ remains constant and is equal to the number of parents μ.

2.2 A Generational Standard Varying Mutation Genetic Algorithm

Generational standard varying mutation GAs differ from canonical GAs mainly in the mutation rate control. Also, some of them use a selection mechanism with selection pressure higher than the canonical GA; for example, (μ,λ) selection instead of proportional selection. However, the application of operators has been similar to canonical GAs. That is, crossover is applied with probability p_c and then follows mutation with probability p_m per bit.

Following the same line of thought used in 2.1, a GA that applies crossover with probability p_c followed by mutation with varying probability can also be seen as implicitly applying CM and M in parallel with a sole mutation probability p_m governing mutation rates in both CM and M. Figure 2.2 illustrates the implicit parallel application of CM and M when varying mutations are used.

Since the probability p_c is usually set to 0.6, and higher values are often used[25], it turns out that mutation is mostly applied serial to crossover. In canonical GAs p_m is small, therefore the amount of diversity introduced by mutation either through CM or M is modest. For the same reason, the disruption that mutation causes to crossover in CM is also expected to be small. In

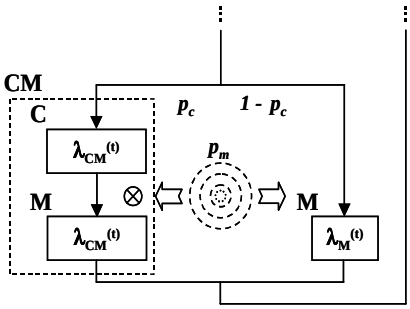


Figure 2.2: Genetic Algorithm with Varying Mutation

varying mutation GAs, however, mutations are higher than canonical GAs and the combined effect of crossover and mutation in CM as well as the effect of mutation alone in M should be carefully reconsidered.

In the case of CM, besides those cases in which crossover and mutation aggregate in a positive manner or are neutral, those cases in which one of the operators is working well but is being hampered by the other should also be taken into account. For example, if mutation probabilities were high then although crossover could be doing a good job it is likely that some of the just created favorable recombinations would be lost, before they become fixed in the offspring, due to the high disruption introduced by mutation. We can think of this case as a mutation interference with crossover in the *creation* of beneficial recombinations. On the other hand, mutation could be working well but crossover may produce poor performing individuals affecting the survivability of beneficial mutations that can contribute to the search. We can think of this case as a crossover interference with mutation in the introduction of beneficial mutations.

In the case of mutation alone M, its instantaneous effectiveness depends only upon itself and does not diminish the effectiveness of other operator. High mutations in M, when are harmful, will have a negative impact on the *propagation* of beneficial recombinations already present in the parent population. However, it will not affect the *creation* of beneficial recombinations by crossover as high mutation can do it in CM.

In the following these approaches that apply varying mutations after crossover are referred as *varying mutation serial to crossover*. Although as explained above, due to the crossover probability they also implicitly apply parallel varying mutation.

2.3 A Generational Model of Parallel Varying Mutation Genetic Algorithm (GA-SRM)

2.3.1 Parallel Operators

The model of standard varying mutation GAs of 2.2 raises several important questions such as

1. Is there interference between crossover and high mutation? If so,
2. Does it affect performance of the algorithm?
3. Does it affect the mutation rate control itself?
4. Is it an appropriate model for combining forms of control (co-adaptation of parameters)?

To answer the questions raised by the standard varying mutation approach, this work explores a model of GA that explicitly differentiate the mutation operator applied parallel to crossover from the mutation operator applied after crossover. There are some advantages to this differentiation.

1. Each mutation operator could be assigned its own mutation probability. Thus, varying mutation can be applied only parallel to crossover and mutation after crossover can be applied with a small mutation rate (or none at all) avoiding interferences between crossover and high mutation.

2. Since in this case the instantaneous effectiveness of the varying mutation operator depends only upon itself its relative success can be directly tied to the mutation rate to create adaptive (self-adaptive) schemes for mutation rate control.

3. Parallel mutation can be studied on its own. For example, higher mutation rates raise the question of mutation strategy and its relevance to a given epistatic pattern or class of problems.

4. The individual roles and the interaction of crossover and varying mutation throughout the run of the algorithm can be better understood, which could be important for co-adaptation studies.

Thus, this work explores a model of GA that in addition to crossover followed by background mutation (CM) it also explicitly applies parallel varying mutation[34, 35]. To clearly distinguish between the mutation operator applied after crossover and the mutation operator applied parallel to crossover, the parallel varying mutation operator is called *Self-Reproduction with Mutation* (SRM). In the following this model of GA is called GA-SRM.

As suggested above, the explicit parallel formulation of CM and SRM can give an efficient framework to achieve better balances for mutation and crossover during the run of the algorithm in which the strengths of higher mutation and crossover can be kept without interfering one with the other. SRM parallel to CM implicitly increases the levels of cooperation to introduce beneficial mutations and create beneficial recombinations. It also sets the stage for competition between operators' offspring.

In the following GAs that explicitly apply varying mutations only parallel to crossover are referred as *varying mutation parallel to crossover*.

2.3.2 Extinctive Selection

Recent works have given more insights to better characterize the roles of recombination and mutation in evolutionary algorithms. An important issue to consider is the deleterious effects caused by the operators and especially how to deal with them.

On the one hand, it has been shown that mutation is more powerful than recombination (crossover) in terms of exploration or disruption[36, 37, 38, 39] and that mutation's disruption capabilities are directly related to the mutation rate[36, 38]. While the explorative effect of mutation is desirable, we should expect that, at all times of the search process, only a number of the individuals created by parallel mutation SRM would offer variability and still keep a reasonable high performance (beneficial mutants). The others would be diverse but poor performing individuals (harmful mutants).

On the other hand, it has also been shown that recombination (crossover) would have a deleterious effect especially on multimodal fitness landscapes[38, 39], performing worse as the number of peaks increase. The recombination of individuals on different peaks will likely produce offspring

in the valleys between peaks, where the fitness is lower. This effect will be more evident during the latest stages of the search when the population moves towards the peaks of the landscape.

The parallel formulation of CM and SRM can avoid interferences between crossover and high mutation; however it cannot prevent SRM from creating deleterious mutations or CM from producing ineffective crossing over operations. To cope with these cases the model also incorporates the concept of extinctive selection that has been widely used in Evolution Strategies[12]. Through extinctive selection the offspring created by CM and SRM coexist competing for survival (the poor performing individuals created by both operators are eliminated) and reproduction.

Among the various extinctive selection mechanisms available in the EA literature (μ, λ) proportional selection[24] is chosen to implement the required extinctive selection mechanism. Selection probabilities for this kind of selection are computed by

$$P_s(\boldsymbol{x}_i^{(t)}) = \begin{cases} \dfrac{f(\boldsymbol{x}_i^{(t)})}{\sum_{j=1}^{\mu} f(\boldsymbol{x}_j^{(t)})} & (1 \leq i \leq \mu) \\ 0 & (\mu < i \leq \lambda) \end{cases} \tag{2.1}$$

where $\boldsymbol{x}_i^{(t)}$ is an individual at generation t which has the i-th highest fitness value $f(\boldsymbol{x}_i^{(t)})$, μ is the number of parents and λ is the number of offspring.

The parallel formulation of genetic operators tied to extinctive selection creates a cooperative-competitive environment for the offspring created by CM and SRM.

2.3.3 Mutation Rate Control

Deterministic, adaptive, and self-adaptive mutation rate control schedules are used to study the serial/parallel application of varying mutations. The adaptive schedule is applied only *parallel to crossover* and the deterministic and self-adaptive schedules are applied both *serial* and *parallel to crossover*.

Deterministic

The deterministic approach implements a time-dependent mutation schedule that reduces mutation rate in a hyperbolic shape. It was originally proposed in [31] and is expressed by

$$p_m^{(t)} = \left(r_o + \frac{n - r_o}{T - 1} t \right)^{-1} \tag{2.2}$$

where T is the maximum number of generations, $t \in \{0, 1, \cdots, T-1\}$ is the current generation, and n is the bit string length. The mutation rate $p_m^{(t)}$ varies in the range $[1/r_o, 1/n]$. In the original formulation $r_o = 2$. Here r_o is included as a parameter in order to study different ranges for mutation. In the deterministic approach the mutation rate calculated at time t is applied to all individuals created by SRM.

Figure 2.3 illustrates the mutation rates over the generations by this schedule for three initial mutation rates, $p_m^{(0)} = \{0.5, 0.10, 0.05\}$.

Self-Adaptive

To include self-adaptation, each individual incorporates its own mutation probability within the representation. SRM to produce offspring first mutates the mutation probability of the selected

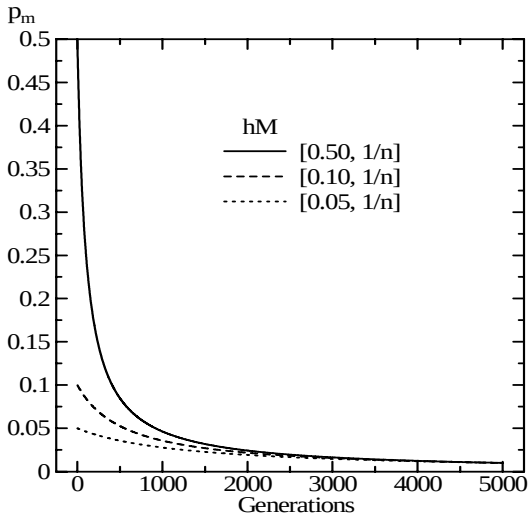


Figure 2.3: Deterministic Hyperbolic Schedule for Mutation Rate Control

individual and then mutates the object variable using the individual's mutated probability. This work applies the self-adaptive approach originally proposed in [31, 33], which uses a continuous representation for the mutation rate and mutates the mutation probability of each individual by

$$p_m^{(t)}(i) = \left(1 + \frac{1 - p_m^{(t-1)}(i)}{p_m^{(t-1)}(i)} exp(-\gamma N(0,1))\right)^{-1} \tag{2.3}$$

where i indicates the i-th individual, γ is a learning rate that control the speed of self-adaptation, and $N(0,1)$ is a normally distributed random number with expectation zero and standard deviation one. Note that individuals selected to reproduce with SRM at generation t could have been created either by SRM or CM at generation $t-1$. Since the mutation rate of each individual is mutated only by SRM, individuals created by CM do not carry an updated mutation rate. Thus, the mutation rate of individuals that were created by CM at generation $t-1$ is first updated by

$$p_m^{(t-1)}(j) = \frac{1}{\mu_{SRM}} \sum_{k=1}^{\mu_{SRM}} p_m^{(t-1)}(k) \tag{2.4}$$

where j indicates an individual created by CM at $(t-1)$, k indicates the individuals created by SRM at $(t-1)$ that survived extinctive selection, and μ_{SRM} is the number of offspring created by SRM that survived extinctive selection. In the case that no offspring created by SRM survived extinctive selection, $p_m^{(t-1)}(j)$ is set to the mutation value of the best SRM's offspring. SRM will mutate this updated mutation in order to mutate the object variable.

It should be mentioned that besides the method described here, other self-adaptive approaches exclusively for parallel varying mutation have been also implemented successfully[40].

Adaptive

Since the instantaneous effectiveness of SRM depends only upon itself its relative success can be directly tied to the mutation rate to create adaptive (self-adaptive) schemes for mutation rate control. In this work the adaptive mutation rate control dynamically adjusts the mutation rate

within SRM every time a normalized mutants survival ratio γ falls under a threshold τ. The ratio γ is specified by

$$\gamma = \frac{\mu_{SRM}}{\lambda_{SRM}} \cdot \frac{\lambda}{\mu} \tag{2.5}$$

where μ_{SRM} is the number of individuals created by SRM present in the parent population $P(t)$ after extinctive selection at time t, λ_{SRM} is the number of offspring created by SRM, λ is the total number of offspring ($\lambda_{CM} + \lambda_{SRM}$), and μ is the number of individuals in $P(t)$. The number of offspring λ_{CM} and λ_{SRM} that will be created by CM and SRM, respectively, is deterministically decided at the beginning of the run.

It should be noted that the deterministic and adaptive schedules use only one mutation rate for all the individuals in the population. The self-adaptive schedule on the other hand, uses one mutation rate per individual.

2.3.4 Mutation Strategy

In the case of background mutation we expect in the average to flip 1 bit (or less) in each individual at each generation. When higher mutations are applied, however, many more bits would be flipped in the same individual. This raises the question of whether a mutation strategy to choose the bits that will undergo mutation would be more effective than other and for which classes of problems. To study this point, two mutation strategies are investigated for SRM: (i) adaptive dynamic-segment (ADS), and (ii) adaptive dynamic-probability (ADP).

ADS (Adaptive Dynamic-Segment)

ADS directs mutation only to a segment of the chromosome using constant mutation probabilities per bit

$$p_m^{(SRM)} = \begin{cases} \alpha \; (if \; the \; bit \; is \; in \; the \; segment) \\ 0 \; (otherwise) \end{cases}$$

while the mutation segment size ℓ is dynamically adjusted every time γ falls under τ. The segment reduction is summarized below:

$\ell = n \; (t = 0)$
if $(\gamma < \tau)$ **and** $(\ell > 1/\alpha)$
  $\ell = \ell/2$

where the segment size ℓ varies from n (bit string length) to $1/\alpha$, $[n, 1/\alpha]$ following a step decreasing approach as shown in Figure 2.4.

The segment initial position, for each chromosome, is chosen at random, $s_i = N[0, n)$, and its final position is calculated by

$$s_f = (s_i + \ell) \bmod n. \tag{2.6}$$

With this scheme, the average number of flipped bits goes down from $n\alpha$ to 1, $[n\alpha, 1]$.

ADP (Adaptive Dynamic-Probability)

With ADP, every bit in the chromosome is always subject to mutation with probability $p_m^{(SRM)}$ varying each time γ falls under τ:

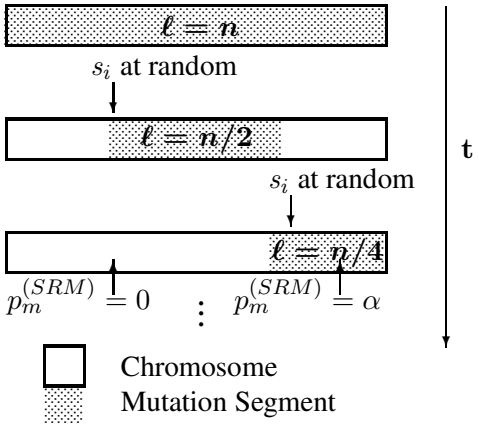


Figure 2.4: ADS (Adaptive Dynamic-Segment) mutation.

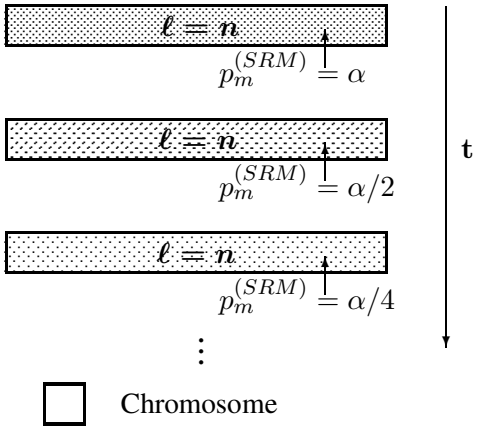


Figure 2.5: ADP (Adaptive Dynamic-Probability) mutation.

$p_m^{(SRM)} = \alpha \ (t = 0)$
if $(\gamma < \tau)$ **and** $(p_m^{(SRM)} > 1/n)$
$\quad p_m^{(SRM)} = p_m^{(SRM)}/2$

In other words, the segment size is kept constant, $\ell = n$, but $p_m^{(SRM)}$ follows a step decreasing approach from α to $1/n$ per bit, $p_m^{(SRM)} = [\alpha, 1/n]$ as shown in Figure 2.5.

Both schemes, ADS and ADP, impose an adaptive mutation rate control with the same expected average number of flipped bits; the difference lies whether mutation is applied locally inside the segment (ADS) or globally inside the whole chromosome (ADP).

2.3.5 Duplicates Elimination

Genetic drift is postponed enhancing selection by eliminating fitness duplicates. If several individuals have exactly the same fitness then one of them is chosen at random and kept. The other

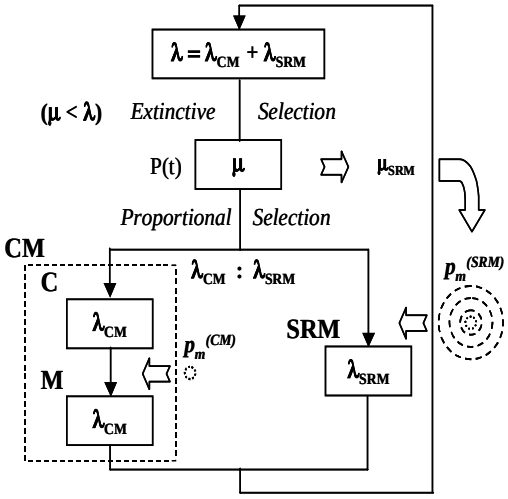


Figure 2.6: GA with Parallel Varying Mutation

equal fitness individuals are eliminated from the population. The fitness duplicates elimination is carried out before extinctive selection. Note that under this approach it is possible that two individuals with the same fitness may actually be different at the genotype level. Preventing duplicates removes an unwanted source of selective bias[41] and allows a fair competition between offspring created by CM and SRM.

2.4 GA-SRM's Algorithm

The algorithm of the parallel varying mutation GA based on the proposed model is presented below and its block diagram is shown in Figure 2.6.

procedure GA-SRM
begin
$t := 0$;
initialize $(P(0))$;
evaluate $(P(0))$;
while (**not** termination condition)
begin
$P'(t)$ = crossover and mutation $(P(t))$; /* creates λ_{CM}*/
$P''(t)$ = self-reproduction with mutation $(P(t))$; /* creates λ_{SRM} */
evaluate $(P'(t) \cup P''(t))$; /* $\lambda = \lambda_{CM} + \lambda_{SRM}$ */
$P'''(t)$= eliminate fitness duplicates $(P'(t) \cup P''(t))$;
$P(t+1)$ = (μ, λ) proportional selection $(P'''(t))$; /* $\mu < \lambda$ */
$t := t + 1$;
end
end

Chapter 3

Test Problems Generators

Test function generators for broad classes of problems are seen as the correct approach for testing the performance of genetic algorithms. This chapter describes two test problems generators and the test data used in this work. One of generators is for 0/1 multiple knapsack problems, which allows to test the algorithms on a broad range of classes of constrained problems by varying the feasible region of the search space, number of constraints, and the size of the search space. Real-world 0/1 multiple knapsack problems with known global optimum are also used. These latter problems allow studying the global search abilities of the algorithms. The second generator is the well known Kauffman's NK-landscapes model of epistatic interactions that has been the center of several theoretical and empirical studies both for the statistical properties of the generated landscapes and for their *GA-hardness*. This generator allows testing the algorithms on a broad range of classes of epistatic non-linear problems. Both, 0/1 multiple knapsack problems and NK-Landscapes, are known to be NP-hard combinatorial problems.

3.1 0/1 Multiple Knapsacks Problems

In the 0/1 multiple knapsack problem there are m knapsacks and n objects. The capacities of the knapsacks are $c_1, c_2, ..., c_m$. For each object there is a profit p_i $(1 \leq i \leq n)$ and a set of weights w_{ij} $(1 \leq j \leq m)$, one weight per knapsack. If an object is selected its profit is accrued and the knapsacks are filled with the object' weights. The problem consists on finding the subset of objects that maximizes profit without overfilling any of the knapsacks with objects' weights. The 0/1 multiple knapsack problem can be formulated to maximize the function

$$g(\boldsymbol{x}) = \sum_{i=1}^{n} p_i x_i \tag{3.1}$$

subject to

$$\sum_{i=1}^{n} w_{ij} x_i \leq c_j \qquad (j = 1, ..., m) \tag{3.2}$$

where $x_i \in \{0,1\}$ $(i = 1, ..., n)$ are elements of a solution vector $\boldsymbol{x} = (x_1, x_2, ..., x_n)$, which is the combination of objects we are interested in finding. Solutions to this problem have a natural binary representation in the GA constructed by mapping each object to a locus within the binary chromosome. A 1 in locus i indicates that the object i is being selected and a 0 otherwise. A solution vector $\boldsymbol{x}$ should guarantee that no knapsack is overfilled and the best solution should yield the maximum profit. An $\boldsymbol{x}$ that overfills at least one of the knapsacks is considered as an infeasible solution. Figure 3.1 illustrates the problem.

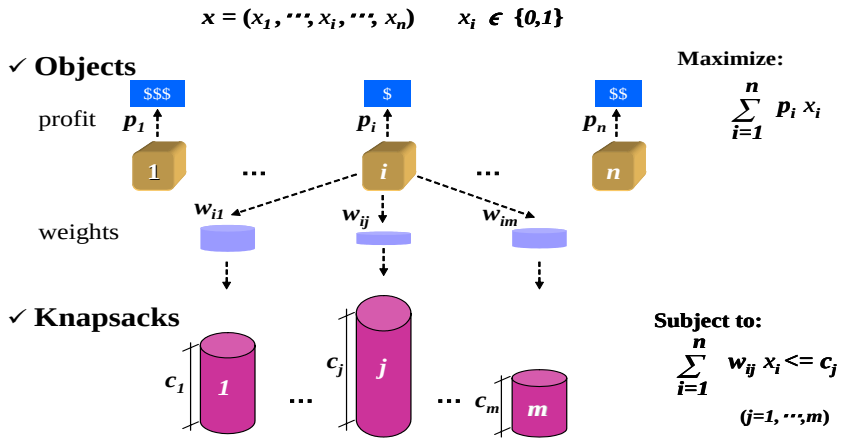


Figure 3.1: Knapsack Problem

Besides defining the number of knapsacks m (number of constraints) and the number of objects n (size of the search space, 2^n), it is also possible to define the tightness ratio ϕ between knapsack capacities and object weights (which implies a ratio between the feasible region and the whole search space). Thus, by varying m, n, and ϕ, 0/1 multiple knapsacks allows us to carefully observe the behavior and scalability of the algorithms in three important aspects that are correlated to the difficulty of a problem.

The 0/1 multiple knapsack problem is also known as the 0/1 multidimensional or m-dimensional knapsack problem and can be regarded as a general statement of any zero-one integer programming problem with non-negative coefficients. Indeed much of the early work on the problem viewed the problem in this way.

Table 3.1: Real world 0/1 multiple knapsack test problems.

Name	n	m	*Max*
Petersen 3	15	10	4015
Petersen 4	20	10	6120
Petersen 5	28	10	12400
Petersen 6	39	5	10618
Petersen 7	50	5	16537
Sento 1	60	30	7772
Sento 2	60	30	8722
Weing 7	105	2	1095445

The 0/1 multiple knapsack problem is a NP-hard combinatorial optimization problem and its importance is well known both from a theoretic and practical point of view. It is a generalization of the 0/1 simple ($m = 1$) knapsack problem. Simple knapsack problems can be used as subproblems to solve more complicated ones[42] and other combinatorial problems, such the partition problem, can be polinomially transformed into it [43]. Also, well known NP-complete problems, such satisfiability problems (SAT), can be formulated as special instances of a 0/1 multiple knapsack problem [44].

In addition, many practical problems can be formulated as a 0/1 multiple knapsack problem. For example, the capital budgeting problem where project i (object) has profit p_i and consumes w_{ij} units of resource j (knapsack). The goal is to find a subset of the n projects such that the total profit is maximized and all resources constraints are satisfied. Other applications of the problem include allocating processors and databases in a distributed computer system[45], project selection and cargo loading[46], cutting stock problems[47], and maximizing the number of served clients or the use of bandwidth for ad hoc networks[48].

Two sets of 0/1 multiple knapsacks problems are selected to test the algorithms. One of the sets consists of real-world problems with known optimum solution. The other set consists of larger randomly generated problems with unknown optimum solution. Both sets of problems were obtained from the OR-Library[1].

3.1.1 Real World Problems

The problems in this set range from 2 to 30 knapsacks and from 15 to 105 objects (see Table 3.1). Problems *Petersen 3-7* are due to Petersen[49], *Sento 1- 2* were introduced by Senyu and Toyoda[50], and *Weing-7* was introduced by Weingartner and Ness [51]. Since the optimum solution is known for these problems we can observe the behavior of the genetic algorithms for global optimization. These problems have been used by other authors to test canonical GAs and standard varying mutation GAs, allowing initial relative comparisons with other GA approaches.

The fitness function for the real-world knapsack problems introduces the same penalty term used in [52] to deal with infeasible solutions (no repair strategy is used). Thus, the fitness function is specified by

$$f_1(\boldsymbol{x}) = g(\boldsymbol{x}) - s \cdot max\{p_i\} \tag{3.3}$$

where s ($0 \leq s \leq m$) is the number of overfilled knapsacks.

[1] http://mscmga.ms.ic.ac.uk/jeb/orlib/info.html

Table 3.2: 7 Subclasses of problems (10 random problems in each subclass)

	Paremeters			
Subclass	*m*	*n*	ϕ	*Comment*
1	30	100	0.75	reducing
2			0.50	feasible
3			0.25	region
4	5	100	0.25	increasing
5	10			number
(3)	30			constraints
(3)	30	100	0.25	increasing
6		250		search
7		500		space

3.1.2 Large Random Problems

The problems in this set consists of classes of larger and highly constrained instances of 0/1 multiple knapsack. These classes of problems were created using a knapsack test problem generator. The problems and the generator were initially proposed in [53]. The generator itself is based in the procedure suggested in [54] and its main characteristics are as follows:

1. The weights w_{ij} are drawn at random from a uniform distribution $U(0, 1000)$.

2. For each combination of m-n, the capacities of the knapsacks are set by

$$c_j = \phi \sum_{j=1}^{n} w_{ij}$$

   where ϕ is the tightness ratio.

3. The profits of the objects are correlated to the weights of the objects[2] by

$$p_i = \sum_{i=1}^{m} w_{ij}/m + 500q_j$$

   where q_j is a real number drawn from a continuous uniform distribution $U(0, 1)$.

To obtain a broader perspective on the performance of the GAs and to have a clear idea on scalability the GAs are applied to several knapsacks problems systematically varying ϕ, m, and n. Each combination of ϕ, m, and n defines a subclass of problem. Table 3.2 shows the combination of values of the parameters ϕ, m, and n used to define the subclasses of problems. This allows to observe the robustness of the algorithms reducing the feasible region (tightness ratio between knapsack capacities and object weights $\phi = \{0.75, 0.50, 0.25\}$), increasing the number of constraints ($m = \{5, 10, 30\}$ knapsacks), and increasing the search space ($n = \{100, 250, 500\}$ objects). We use 7 subclasses and 10 random problems in each subclass.

The quality of the solutions found by the algorithms are measured by the average percentage error gap in a subclass of problems, which is calculated as the normalized difference between the

[2]The difficulty of the problems is greatly increased by the correlation between profits and weights[42].

best solutions found and the optimal value given by the linear programming relaxation (LP) (the optimal integer solutions are unknown)[53]. The average error gap is taken for the 10 random problems in each subclass performing 50 runs for each problem by

$$\%\ Error\ Gap = \frac{1}{10}\sum_{p=1}^{10} 100 \times \frac{LP - Problem\ Average}{LP}$$

$$Problem\ Average = \frac{1}{50}\sum_{r=1}^{50} Best\ Solution$$

Similar to the real-world problems, to deal with infeasible solutions a penalty term is introduced into the fitness function. The two following fitness functions are tried

$$f_1(\boldsymbol{x}) = g(\boldsymbol{x}) - s \cdot max\{p_i\} \tag{3.4}$$

$$f_2(\boldsymbol{x}) = \begin{cases} g(\boldsymbol{x})/s \cdot max\{o_j\} & (s > 0) \\ g(\boldsymbol{x}) & (s = 0) \end{cases} \tag{3.5}$$

where s ($0 \leq s \leq m$) is the number of overfilled knapsacks and o_j (> 1) is the overfilling ratio of knapsack j calculated by

$$o_j = \sum_{i=1}^{n} w_{ij}x_i/c_j. \tag{3.6}$$

Note that the penalty term of f_1 is merely a function of the number of violated constraints (s), but has no direct correlation with any metric that indicates the distance from feasibility. On the other hand, the penalty term of f_2 is a function of both number of violated constraints (s) and distance from feasibility (o_j).

The fitness function f_1 is the one used for the real-world problems, which has been successfully used before with smaller 0/1 multiple knapsacks problems [31, 34, 35, 52, 55]. In the larger randomly generated problems, however, f_1 did not produce feasible solutions or the results were very poor especially on problems with restrictive knapsack capacity and small number of knapsacks. This is because the penalty term used in f_1 is too weak for these problems, causing a major portion of the infeasible region to end up with fitness higher than the feasible region. In such case, the fitness function misleads the algorithm to search within the infeasible region of the search space. Similar behavior with other penalty functions was observed in [30] for single ($m = 1$) 0/1 knapsack problems of restrictive capacity. Results by f_1 are in agreement with [56]. That is, penalties that are solely functions of the number of violated constraints are not likely to find feasible solutions for problems having few constraints and reduced feasible region.

As mentioned above, the fitness functions f_2 of Eq. (3.5) includes a penalty that is also a function of the distance from feasibility. In this case, feasible solutions were found on all test problems. In general, a good penalty function should balance the preservation of information of the infeasible solutions with the pressure for feasibility[56]. The results reported here for the randomly generated problems are obtained using f_2.

3.2 NK-Landscapes

NK-Landscapes are stochastically generated fitness functions on bit strings, parameterized with N bits and K epistatic interactions between bits. In biology, the term epistasis is used to describe a range of non-additive phenomena due to the non-linear inter-dependence of gene values, i.e. the expression of one gene masks the genotypic effect of another. In the context of GAs this

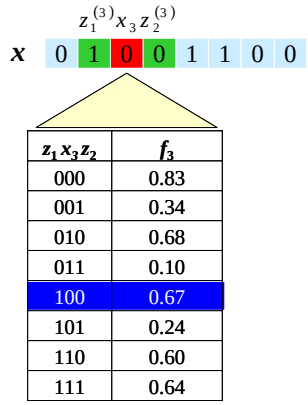


Figure 3.2: An example of the fitness function $f_3(x_3, z_1^{(3)}, z_2^{(3)})$ associated to bit x_3 in which x_3 epistatically interacts with its left and right neighboring bits, $z_1^{(3)} = x_2$ and $z_2^{(3)} = x_4$ ($N = 8$, $K = 2$)

terminology is used to describe nonlinearities in fitness functions due to changes in the values of interacting bits[3].

More formally, an NK-Landscape is a function $f : \mathcal{B}^N \rightarrow \Re$ where $\mathcal{B} = \{0, 1\}$, N is the bit string length, and K is the number of bits in the string that epistatically interact with each bit. Kauffman's original NK-Landscape[57, 58] can be expressed as an average of N functions as follows

$$f(\boldsymbol{x}) = \frac{1}{N} \sum_{i=1}^{N} f_i(x_i, z_1^{(i)}, z_2^{(i)}, \cdots, z_K^{(i)}) \tag{3.7}$$

where $f_i : \mathcal{B}^{K+1} \rightarrow \Re$ gives the fitness contribution of bit x_i , and $z_1^{(i)}, z_2^{(i)}, \cdots, z_K^{(i)}$ are the K bits interacting with bit x_i in the string $\boldsymbol{x}$. That is, there is one fitness function associated to each bit in the string. NK-Landscapes are stochastically generated and usually the fitness contribution f_i of bit x_i is a number between [0.0, 1.0] drawn from a uniform distribution. Figure 3.2 shows an example of the fitness function $f_3(x_3, z_1^{(3)}, z_2^{(3)})$ associated to bit x_3 for $N = 8, K = 2$, in which x_3 epistatically interacts with its left and right neighboring bits, $z_1^{(3)} = x_2$ and $z_2^{(3)} = x_4$, respectively.

For a giving N, we can tune the ruggedness of the fitness function by varying K. In the limits, $K = 0$ corresponds to a model in which there are no epistatic interactions and the fitness contribution from each bit value is simply additive, which yields a single peaked smooth fitness landscape. On the opposite extreme, $K = N - 1$ corresponds to a model in which each bit value is epistatically affected by all the remaining bit values yielding a maximally rugged fully random fitness landscape. Varying K from 0 to $N - 1$ gives a family of increasingly rugged multi- peaked landscapes.

Besides defining N and K, it is also possible to arrange the epistatic pattern between bit x_i and K other interacting bits. That is, the distribution of the K bits among the N. Kauffman investigated NK-Landscapes with two kinds of epistatic patterns: (i) *nearest neighbor*, in which a bit interacts with its $K/2$ left and right adjacent bits, and (ii) *random*, in which a bit interacts with K other randomly chosen bits in the chromosome. By varying N, K, and the distribution of K among the N, we can study the effects of the size of the search space, intensity of epistatic

[3]In this work a gene can be interpreted as a bit and an allele (gene value) as a bit value, i.e. 0 or 1

interactions, and epistatic pattern on the performance of genetic algorithms.

The works of Altenberg[59, 60] Heckendorn et al.[61], and Smith and Smith[62, 63] are important contributions extending NK-landscapes to a more general framework of tunable random landscapes and on their analysis. The simplest of the generalized NK-Landscapes[59, 61, 62] can be expressed as the average of P functions as follows

$$f(\boldsymbol{x}) = \frac{1}{P}\sum_{j=1}^{P} f_j(x_i^{(j)}, z_1^{(i,j)}, z_2^{(i,j)}, \cdots, z_K^{(i,j)}) \tag{3.8}$$

where $f_j : \mathcal{B}^{K+1} \rightarrow \Re$ gives the fitness contribution of the $K+1$ interacting bits. That is, in this model there could be zero, one, or more than one fitness function associated to each bit in the string.

An implication of epistatic interactions among bits is that the fitness function develops conflicting constraints[59]. For example, a mutation in one bit may improve its own contribution to fitness but may decrease the contributions of other bits with which it interacts. Conversely, if a bit value at other interacting position in the bit string changes, a bit value that had been optimal may not longer be optimal. Thus, epistatic interactions increase the difficulty in trying to optimize all bits simultaneously.

The study of the effects of epistasis is of great interest to the evolutionary algorithm's community because epistatic interactions are directly correlated to the underlying representation that the evolutionary algorithm uses and to the multimodality and non-linearity of the fitness landscape that the algorithm searches. Lately, the influence of epistasis on the performance of Genetic Algorithms (GAs) is being increasingly investigated. NK-Landscapes, particularly, have been the center of several theoretical and empirical studies both for the statistical properties of the generated landscapes and for their *GA-hardness*[60, 64, 65, 66, 67]. Previous works that investigate properties of GAs with NK-Landscapes have mostly limited their study to small landscapes, typically $10 \leq N \leq 48$, and observed the behavior of GAs only for few generations. Recently, studies are being conducted on larger landscapes expending more evaluations[61, 68, 69, 70] and the performance of GAs is being benchmarked against hill climbers[61, 68, 70].

In this work experiments are conducted on landscapes with *random* and *near neighbor* patterns of epistasis for $N = 48$ and $N = 96$ varying K from 0 to $N-1$ in intervals of 4. Each combination of pattern of epistasis, N, and K defines a class of problem. For each one of these classes 50 random problems are created. This allows testing the performance of GAs on a broad range of classes of epistatic non-linear problems.

There are several empirical and theoretical studies of the statistical properties of the generated NK-Landscapes. The following is a synopsis of the results for one-mutant adaptive walks on NK landscapes taken from [59].

For $K = 0$, the fitness function becomes the classical adaptive multilocus model.

1. There is a single globally attractive genotype.

2. The adaptive walk from any genotype will proceed by reducing the Hamming distance to the optimum by one at each step, and the number of fitter one-mutant neighbors is equal to this Hamming distance. Therefore, the expected number of steps to the global optimum is $N/2$.

3. The fitness of one-mutant neighbor genotypes are highly correlated, as $N-1$ of the N fitness components are unrelated between the neighbors.

For $K = N-1$, the fitness function is equivalent to the random assignment of fitness over the genotype space.

1. The probability that a genotype is a local optimum is $1/(N+1)$.
2. The expected total number of local optima is $2^N/(N+1)$.
3. The expected fraction of one-mutant neighbors that are fitter decreases by $1/2$ each step of the adaptive walk.
4. The expected length of adaptive walks is approximately $ln(N-1)$.
5. The expected number of mutants tested before reaching local optimum is $\sum_{i=0}^{log_2(N-1)-1} 2^i$.
6. As N increases, the expected fitness of the local optimum reached from a random initial genotype decreases towards the mean fitness of the entire space, 0.5. Kauffman[57, 58] calls this the *complexity catastrophe*.

For intermediate K, it is found that

1. For small K, the highest local optima share many of their alleles in common. As K increases, this allelic correlation among local optima falls away, and more rapidly for random neighbors than adjacent neighbors.
2. For large K, the fitness of local optima are distributed with an asymptotically normal distribution with mean approximately.

$$\mu + \sigma \left(\frac{2\ln(K+1)}{K+1} \right)^{1/2}$$

and variance approximately

$$\frac{(K+1)\sigma^2}{N[K+1+2(K+2)\ln(K+1)]}$$

where μ is the expected value of F_i, and σ^2 its variance. In the case of uniform distribution, $\mu = 1/2$ and $\sigma = (1/12)^{1/2}$.

3. The average Hamming distance between local optima, which is roughly twice the length of a typical adaptive walk, is approximately

$$\frac{N\log_2(K+1)}{2(K+1)}$$

4. The fitness correlation between genotypes that differ at d loci is

$$R(d) = \left(1 - \frac{d}{N}\right)\left(\left(1 - \frac{K}{N-1}\right)^d$$

for the random neighborhood model, and

$$R(d) = 1 - \frac{K+1}{N}d + \frac{1}{\left(\begin{array}{c} N \\ d \end{array}\right)} \sum_{j=1}^{min(K,N+1-d)} (K-j+1) \left(\begin{array}{c} N-j-1 \\ d-2 \end{array}\right)$$

for the adjacent neighborhood model.

Chapter 4

Studying the Structure of the Parallel Varying Mutation GA-SRM

This chapter studies in detail the internal structure of the parallel varying mutation GA-SRM using adaptive mutations. It analyzes the impact of important issues that affect the performance of GAs and study the contribution of extinctive selection, adaptation, mutation strategy, and the interaction between parallel varying mutation and crossover. Real-world and randomly generated 0/1 knapsack problems are used to study and test the model. Comparisons are made with simple GAs and other improved GAs.

4.1 Introduction

The GA-SRM model applies varying mutations parallel (SRM) to crossover & background mutation (CM) putting the operators in a cooperative-competitive stand with each other through extinctive selection. Important structural issues to be studied are the balance between CM and SRM for offspring creation, the mutation probability in CM, the ratio between number of parents, number of offspring (extinctive selection pressure), and the threshold to trigger adaptation in SRM. Recombination in canonical GAs has been emphasized and deeply studied. It is known that different crossover operators create dissimilar degrees of disruption and construction of building blocks. Therefore it is also important to verify whether the kind of diversity that we expect from SRM can be produced by other kinds of recombination operators. The impact of other important issues related to the performance of GAs, such as population size and evaluation time, is observed, too. Besides the internal structure, the effect on performance of extinctive selection, the interaction of varying mutation parallel to crossover, and the effect of adaptation and mutation strategy are also investigated. In this chapter, all these issues are studied using real-world and randomly generated 0/1 knapsack problems.

4.2 Experimental Setup

Experiments are conducted using the following algorithms: (i) a canonical GA denoted as cGA (CM and proportional selection), (ii) a simple GA with (μ, λ) proportional selection denoted as GA(μ, λ) (CM and extinctive proportional selection), and (iii) the proposed algorithm denoted as GA-SRM(μ, λ) (CM, SRM and extinctive proportional selection). In GA-SRM, SRM is used with ADS and ADP mutation strategies. Unless stated otherwise, the genetic algorithms used here are set with the parameters specified in Table 4.1. It should be noted that throughout this chapter elimination of *fitness duplicates* is not considered and that SRM's mutation rate control is the adaptive mechanism explained in 2.3.3.

Experiment with the real-world knapsack problems consisted of 100 independent runs for each problem. For the randomly generated problems 50 runs were performed for each one of the seventy problem in the seven subclasses of problems, as explained in 3.1.2. Each run was set with a different seed for the random initial population and ended after T evaluations were performed. The values of T used for each real-world problem are indicated in Table 4.4. For all randomly generated problems the number of evaluations was set to $T = 5 \times 10^5$. The number of generations for each experiment is calculated as T/λ. In GA-SRM, the values of τ (threshold to trigger adaptation in SRM) are sampled and the one that produces the overall higher performance on the seven different classes of random problems is chosen. The threshold is set to $\tau = 0.64$ for ADS and $\tau = 0.54$ for ADP. The values of τ used for the real-world problems are indicated in Table 4.4.

4.3 Operators' Balance

First, the importance of SRM and the operators' balance for offspring creation is investigated. Three general cases are considered.

1. The size of the parent population $P(t)$ is equal to the size of CM's and SRM's offspring populations, $\lambda_{SRM} = \mu = \lambda_{CM}$.

2. $P(t)$ is smaller than CM's but bigger than SRM's population, $\lambda_{SRM} < \mu < \lambda_{CM}$.

Table 4.1: Genetic algorithms parameters.

$Parameter$	cGA	$GA(\mu,\lambda)$	$GA-SRM(\mu,\lambda)$
$Representation$	Binary	Binary	Binary
$Selection$	Proport.	(μ,λ) Proport.	(μ,λ) Proport.
$Scaling$	Linear	Linear	Linear
$Mating$	$(\boldsymbol{x}_i,\boldsymbol{x}_j),\ i\neq j$	$(\boldsymbol{x}_i,\boldsymbol{x}_j),\ i\neq j$	$(\boldsymbol{x}_i,\boldsymbol{x}_j),\ i\neq j$
$Crossover$	one point	one point	one point
p_c	0.6	0.6	1.0
$p_m^{(CM)}$	$1/n$	$1/n$	$1/n$
$p_m^{(SRM)}$	-	-	$\begin{cases}\alpha=0.5,\ \ell=[n,1/\alpha]\ (ADS)\\ \alpha=[0.5,1/n],\ \ell=n\ (ADP)\end{cases}$
$\mu:\lambda$	-	$1:2$	$1:2$
$\lambda_{CM}:\lambda_{SRM}$	-	-	$1:1$

3. $P(t)$ is bigger than CM's but smaller than SRM's population, $\lambda_{SRM}>\mu>\lambda_{CM}$.

In the case of equal size populations, both operators could allocate all their offspring to $P(t)$. Therefore, there is competition between the two operators' offspring for every spot in $P(t)$. The normalized mutant survival ratio γ, specified by eq.(2.5), reflects the number of mutants *winners* that survive after competing with CM's offspring. Also, in this case the number of mutants that survive selection equals the number of CM's offspring being eliminated.

However, if one of the offspring populations is smaller than $P(t)$, then it could at most cover a fraction of the parent population. Hence, competition for survival between operators is not for the μ spots but rather for μ^c specified by the size of the smaller population. This is because the best $\mu-\mu^c$ of the exceeding population need not to compete in order to survive. For example, if the bigger population corresponds to SRM, μ_{SRM} in eq.(2.5) includes not only the mutants *winners* but also those that survive without competition. Also, note that in this case the number of mutants that survive selection does not equal to the number of CM's offspring being eliminated.

To reflect the competition between operators when different offspring population sizes are used the mutants survival ratio of eq.(2.5) is extended to

$$\gamma=\frac{\mu_{SRM}^w}{\lambda_{SRM}^c}\cdot\frac{\lambda^c}{\mu^c} \tag{4.1}$$

where μ_{SRM}^w is the number of individuals created by SRM that compete and survive selection (mutants *winners*), λ_{SRM}^c is the offspring number created by SRM that undergoes competition, λ^c is the total offspring number that compete for survival ($\lambda_{CM}^c+\lambda_{SRM}^c$), and μ^c is the number of spots that SRM's and CM's offspring compete for in the parent population $P(t)$. Note that eqs.(2.5) and (4.1) are the same if equal size populations are used (case 1).

The balance between operators for offspring creation is studied using eq.(4.1). Setting the parent and offspring population to $(\mu,\lambda)=(50,100)$ several experiments are conducted especially for *Weing 7*[1] varying the number of offspring created by CM and SRM from an all CM regime to a 90% SRM regime. A 100% CM regime represents a genetic algorithm that applies "background" mutation after crossover and uses (μ,λ) Proportional Selection, i.e. GA(50,100). Also, note that

[1]For this problem the number of evaluations is set to $T=2\times10^5$. The same number of evaluations were used in [31] and [52] using offspring populations of 100 and 50 individuals, respectively.

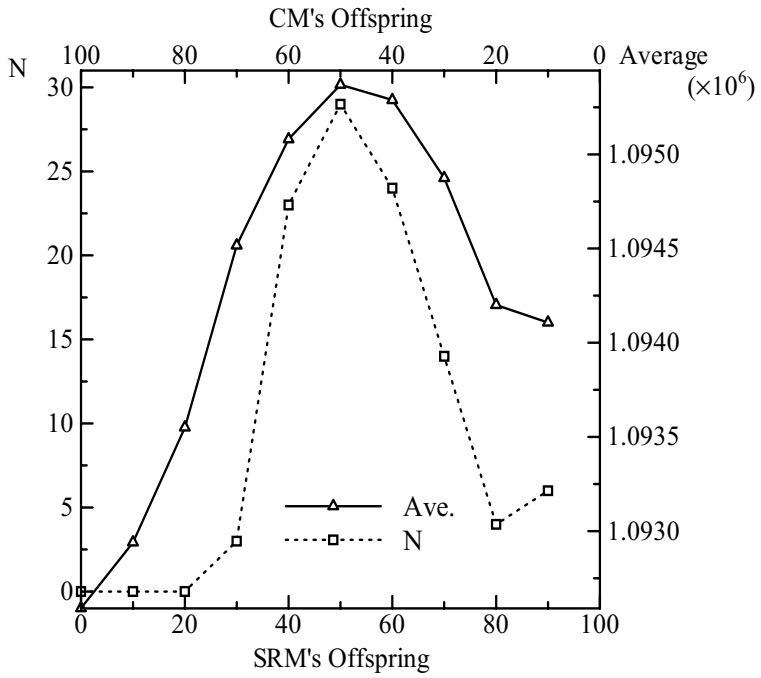


Figure 4.1: Operators' balance and search ability.

because SRM's adaptive mutation schedule is based on a mutant survival ratio, which reflects the competition among SRM's and CM's offspring, it is not possible to test the algorithm with an all SRM regime and simultaneously keep its adaptive feature.

The relationship between operators' offspring balance and search ability (best individuals' average and number of times the global optimum was found in 100 runs, *Average* and *N* respectively) is shown in Figure 4.1 when SRM is implemented with ADS. From this figure the following observations can be drawn. Ratios that favor SRM's offspring, i.e. $\lambda_{SRM} > 50\%$, produce better results than its opposites. A $\lambda_{CM} : \lambda_{SRM} = 1 : 1$ ratio is the best choice for stable and robust performance that simultaneously maximizes *N* and *Average*. In the following sections we use the best $1 : 1$ operators' balance.

4.4 Mutation Probability in CM

Second, the relevance of CM's mutation probability is studied fixing $\lambda_{CM} : \lambda_{SRM} = 50 : 50$. The model's searching ability for $p_m^{(CM)}$ values in the range $[0.5/n, 1.5/n]$ are shown in Figure 4.2, where n is the bit string length. From this figure the following observations are relevant. A $p_m^{(CM)} = 1/n$ turns out to be the probability that gives the highest values for $Average$ and N, that is a coincidence with the results in [24]. Values of $p_m^{(CM)} > 1/n$ are less deteriorative than values of $p_m^{(CM)} < 1/n$ are.

Segment size reduction, ℓ, as well as the number of individuals produced by SRM that survive selection, μ_{SRM}, are shown for one of the runs for $p_m^{(CM)} = 1/n$ in Figure 4.3 (a). Here it can be observed that SRM contributes to the survivor parent population in every generation of the search process. The key factor for SRM to be an effective operator lies in its own regulation mechanism, i.e. the mutation rate is adjusted every time the number of mutants that survive selection falls

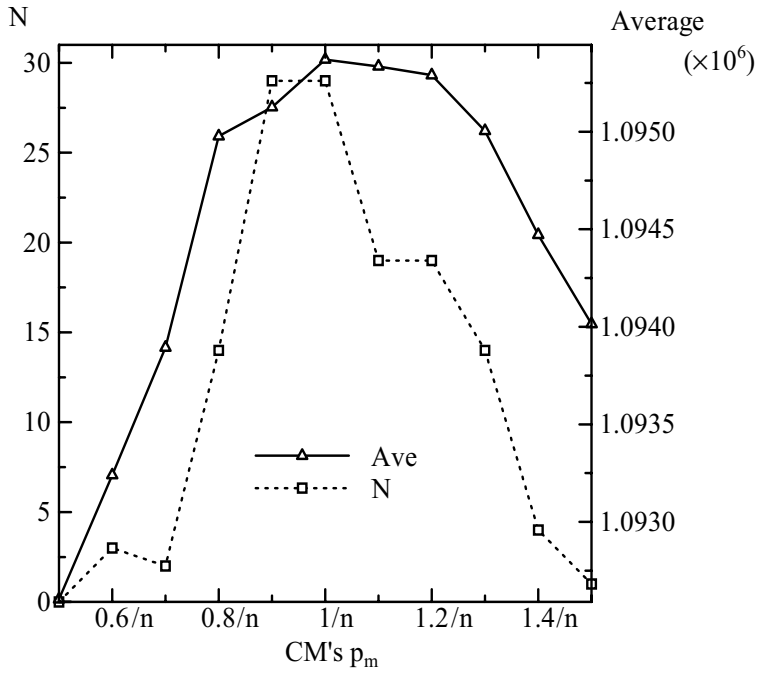


Figure 4.2: CM's mutation probability and search ability (elimination of *fitness duplicates* is off).

under a minimum level τ. Also, the average number of SRM's offspring that survive selection increases as the mutation segment is reduced.

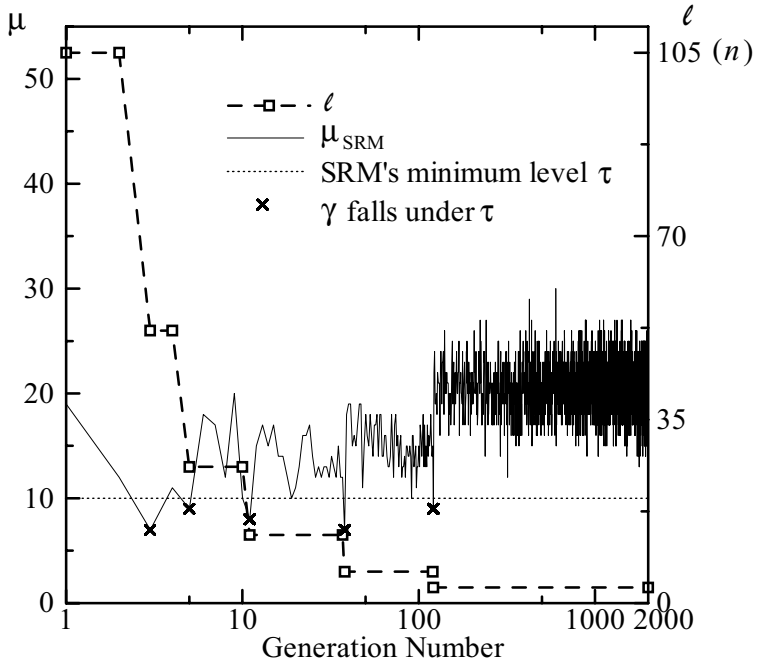
For $p_m^{(CM)} \leq 0.5/n$ it is observed that SRM's offspring fitness cannot compete with CM's offspring, which is specially critical during the early stages of the search, causing a premature reduction of SRM's mutation rate, lost of diversity in the population and convergence to a local optimum. Another typical figure on SRM's contribution μ_{SRM} and segment size reduction ℓ for $p_m^{(CM)} = 0$ (without mutation after crossover) is shown in Figure 4.3 (b) to compare with Figure 4.3(a). Small mutation after crossover is required in CM to achieve a robust search performance by keeping an appropriate balance between CM and SRM. Note that in this case, elimination of *fitness duplicates* is not being considered.

4.5 Extinctive Selection Pressure

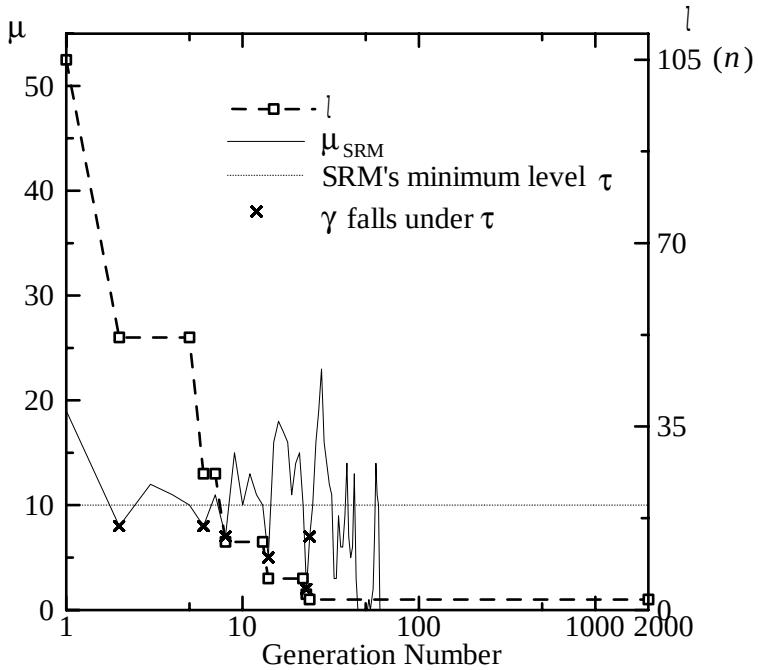
Next, the effect that extinctive selection has in the model is studied varying the size of the parent population μ and setting $\lambda_{CM} : \lambda_{SRM} = 50 : 50$, $p_m^{(CM)} = 1/n$. Figure 4.4 shows results for $(\mu, \lambda) = (\{10, 20, ..., 90\}, 100)$.

High values of *Average* are attained for ratios of extinctive selection pressure in the range μ/λ=[40/100, 70/100]. For $\mu < 50$ both CM and SRM produce offspring in excess of the parent population's requirement ($\lambda_{CM} > \mu$ and $\lambda_{SRM} > \mu$). In this case, there exists competition for survival even among CM's offspring, and SRM's offspring have to outperform CM's best offspring to survive. As the parent population size is reduced competition conditions become severer.

Conversely, for $\mu > 50$ neither CM alone nor SRM can cover the parent population's demand ($\lambda_{CM} < \mu$ and $\lambda_{SRM} < \mu$). In this situation, even if CM totally outperforms SRM, the latter has guaranteed at least $\mu - \lambda_{CM}$ of its best progeny for the next generation. However, with this scheme



(a) $p_m^{(CM)} = 1/n$.



(b) $p_m^{(CM)} = 0$.

Figure 4.3: SRM's offspring number after extinctive selection.

the removal of CM's offspring that are performing poorly is not facilitated. Note that for the worst CM's offspring to be eliminated it has to be worse than the best $\mu - \lambda_{SRM}$ SRM's offspring.

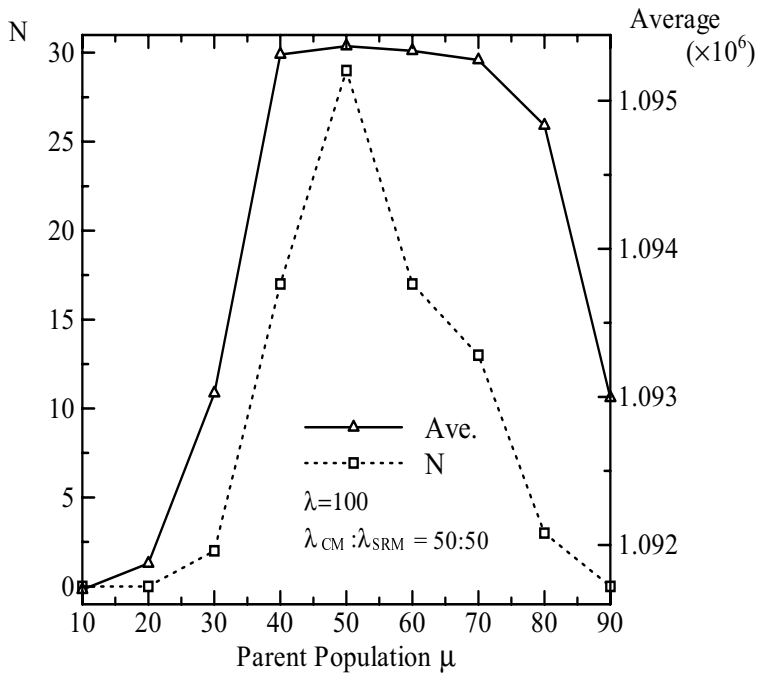


Figure 4.4: Extinctive selection pressure and search ability.

4.6 SRM's Adaptive Minimum Level τ

Figure 4.5 plots *Average* and *N* for values of τ in the range [0.20,0.52] for *Weing 7*. SRM's offspring that survive selection increases as mutation rates are reduced. Therefore, if τ is too small the mutation rate could remain too high at the end of the search, i.e. after certain point there are no further reductions in SRM's mutation rate because the mutants survival ratio γ is always higher than τ. In the experiments the minimum mutation rates were $3/n$ in more than 90% of the runs for $\tau \leq 0.28$ and $1.5/n$ in 92% of the runs for $\tau = 0.32$. In both cases *Average* is high but there is a big difference in *N*.

As τ is increased the minimum value of the mutation rate will tend to be $1/n$ and its reduction will be faster. In this example, the average time (on the hundred runs) at which $1/n$ mutation rate is reached is about $0.5T$ for $\tau = 0.48$, and $0.25T$ for $\tau = 0.52$. A proper reduction speed of SRM's mutation rates guarantees a high *Average* and a SRM's mutation rate close to $1/n$ during the final stage of the search helps to locate the global optimum.

From Figure 4.5, it can be observed that there is a broad range for the threshold τ in which the *Average* is very high. Also, that there is a safety-range in which both *Average* and *N* are high. Similar behavior is observed on other problems used to test the model.

4.7 Two-point and Uniform Crossover

Experiments using two point and uniform crossover are also conducted. Results using two point crossover by cGA and GA(μ,λ) after T evaluations are (*N*, *Ave*, *Stdev*)=$\{$(0, 1086886.8, 1590.8), (0, 1092647.5, 3032.0)$\}$, respectively. Similarly, results using uniform crossover are (*N*, *Ave*, *Stdev*)=$\{$(0, 1090392.1, 1020.7), (0, 1093748.4, 1776.9)$\}$.

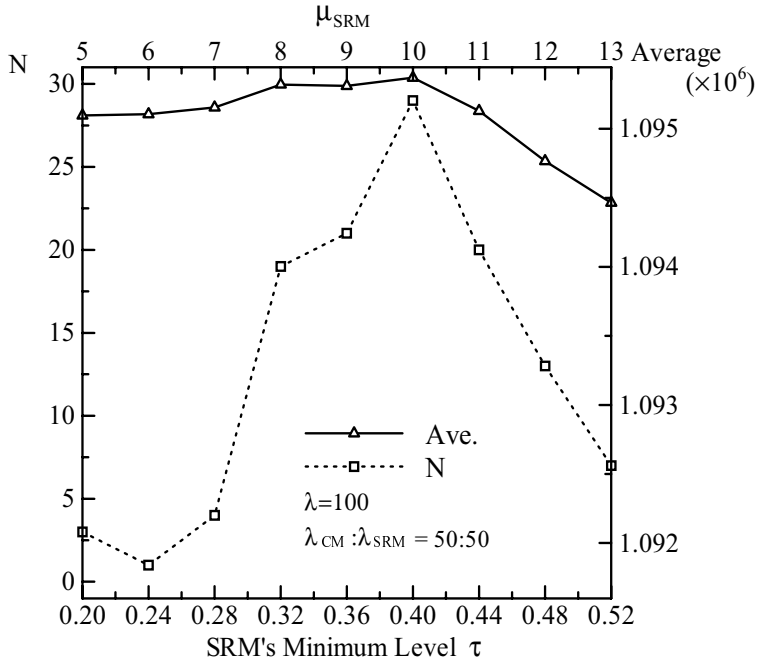


Figure 4.5: SRM's minimum level and search ability.

These results are better than those obtained with one point crossover, yet the global optimum solution could not be found. In the case of GA-SRM(μ,λ), however, the obtained results are quite similar to one point crossover.

4.8 Impact of the Population Size

The impact of the population size in the method's robustness is verified in Table 4.2 (a) and (b) by GA-SRM using ADS and ADP, respectively. All population configurations use the same T evaluations. The right number indicates the value of the global (local) optimum and the left one the number of times it was found. At the bottom of each column, *Average* and *Stdev* are also presented.

From Table 4.2 it can be seen that the model using only 40% of the population size still produces high values for *Average* and *N*. These results are encouraging and suggest that another important benefit of the cooperative-competitive model could be the reduction of the population size. Results by a larger population configuration, i.e. GA-SRM(100,200), are also included. Under the same evaluation time, no considerable difference in results by GA-SRM could be seen varying its configuration from (100,200) to (30,60) for this particular problem.

4.9 Search Ability and Number of Evaluations

The global search ability at various evaluation times is observed by letting the algorithms run for $4T$ evaluations. Table 4.3 presents results by GA-SRM for some intermediate times, where *Stdev* denotes the value of standard deviation around *Average*. Results by cGA and GA(μ,λ) after $4T$ evaluations are (*N*, *Ave*, *Stdev*)=$\{$(0, 1087312.0, 1555.2), (0, 1093797.3, 1624.1)$\}$, respectively.

Table 4.2: Results for Weing 7 using GA-SRM with different population sizes.
(a) ADS.

GA-SRM(100,200) $p_m^{(CM)} = 0.01$ $\tau = 0.46$		GA-SRM(50,100) $p_m^{(CM)} = 0.01$ $\tau = 0.40$		GA-SRM(30,60) $p_m^{(CM)} = 0.01$ $\tau = 0.33$		GA-SRM(20,40) $p_m^{(CM)} = 0.01$ $\tau = 0.30$	
22	1095445	26	1095445	22	1095445	15	1095445
58	1095382	47	1095382	43	1095382	34	1095382
2	1095357	15	1095357	12	1095357	13	1095357
4	1095352	2	1095352	3	1095352	5	1095352
1	1095295	3	1095266	2	1095266	1	1095295
4	1095264			5	1095264	4	1095266
						7	1095264
$9 < 1095264$		$7 < 1095264$		$13 < 1095264$		$21 < 1095264$	
Ave = 1095344.1 *Stdev* = 267.9		*Ave* = 1095345.47 *Stdev* = 337.17		*Ave* = 1095350.46 *Stdev* = 174.50		*Ave* = 1095265.45 *Stdev* = 498.52	

(b) ADP.

GA-SRM(100,200) $p_m^{(CM)} = 0.01$ $\tau = 0.46$		GA-SRM(50,100) $p_m^{(CM)} = 0.01$ $\tau = 0.40$		GA-SRM(30,60) $p_m^{(CM)} = 0.01$ $\tau = 0.33$		GA-SRM(20,40) $p_m^{(CM)} = 0.01$ $\tau = 0.30$	
11	1095445	11	1095445	13	1095445	5	1095445
21	1095382	25	1095382	24	1095382	29	1095382
11	1095357	8	1095357	14	1095357	4	1095357
1	1095352	2	1095352	3	1095266	3	1095295
3	1095295	2	1095295	3	1095264	2	1095266
15	1095264	4	1095266			13	1095264
		6	1095264				
$38 < 1095264$		$42 < 1095264$		$43 < 1095264$		$44 < 1095264$	
Ave = 1095050.63 *Stdev* = 752.3		*Ave* = 1094908.34 *Stdev* = 1106.3		*Ave* = 1094877.47 *Stdev* = 986.56		*Ave* = 1094823.49 *Stdev* = 1032.36	

Table 4.3 empirically show the effect of the proposed cooperative-competitive model in terms of higher search velocity and higher search reliability (reach better solutions with small *Stdev* values). Note the *Average*, *N* and *Stdev* for $0.25T$ and $0.5T$. They also indicate that SRM is a continuous and effective source of diversity, which at the expense of time could be used to improve the search results. For example, when the algorithm was allowed to run for $2T$ evaluations for this particular problem, a remarkable improvement was achieved finding the global maximum 57% of the times with *Average* greater than the second known optimum and very small *Stdev* values.

4.10 Contribution of Parallel Genetic Operators and Extinctive Selection

In order to isolate the contributions of parallel genetic operators and higher selection pressure induced by extinctive selection several additional experiments are conducted using cGA, GA(μ, λ), and GA-SRM(μ, λ). Figure 4.6 plots the average objective fitness in 100 runs of the best-so-far

Table 4.3: Results for Weing 7 using GA-SRM(50,100) with ADS under various evaluation times.

Maximum	0.25 T	0.5 T	$T = 2 \times 10^5$	2 T	4 T
1095445	2	11	26	57	77
1095382	8	29	47	41	23
1095357	2	8	15	1	
1095352	1	2	2	1	
$<$ 1095352	87	50	3		
Average	1094854.1	1095177.6	1095345.5	1095417.4	1095430.5
Stdev	602.8	545.3	337.2	41.6	27.63

individual over the generations by a cGA(100), GA(50,100), and GA-SRM(50,100). From this figure it can be seen that the higher selection pressure of extinctive selection causes an increase on search velocity. GA(μ,λ) in this problem also exhibits higher convergence reliability than cGA without extinctive selection; however, GA(μ,λ) is still not able to find the global optimum and the $Average$ is lower compared to GA-SRM(μ,λ) (See Table 4.5 for *Weing 7*).

The only difference between GA(μ,λ) and GA-SRM(μ,λ) is the inclusion of adaptive mutation SRM in the latter. Therefore any difference in performance between these algorithms can be attributed to SRM. To better observe SRM's contribution experiments are conducted in which starting with a GA(μ,λ) configuration (all CM and extinctive selection) after a predetermined number of evaluations the algorithm switches to a GA-SRM(μ,λ) configuration (CM, SRM and extinctive selection). Figure 4.7 plots results by an algorithm that makes the configuration transition from GA(50,100) to GA-SRM(50,100) at $\{0.02T, 0.05T, 0.10T, 0.20T, 0.5T\}$ evaluations, respectively. As a reference it also includes the results presented in Figure 4.6 by GA(μ,λ) and GA-SRM(μ,λ). From Figure 4.7 it can be seen that as soon as SRM is included fitness starts to pick up increasing the convergence reliability of the algorithm. Also, early transitions produce higher performance. For example, final results for the algorithms that perform transitions at $0.10T$ and $0.50T$ are (*N*, *Ave*)=$\{$(22, 1095242.1), (10, 1094912.8)$\}$, respectively.

Summarizing Figure 4.6 and Figure 4.7, GA-SRM(μ,λ) gains its increase on search velocity from extinctive selection and its higher convergence reliability from the inclusion of parallel adaptive mutation.

To further clarify the contribution of the interaction between CM and SRM during the latest stages of the search experiments are also conducted in which starting with a GA-SRM(μ,λ) configuration, after the mutation rate on SRM has reached a predetermined value, the algorithm switches either to a all CM regime with extinctive selection or to a all SRM regime with extinctive selection (in the latter case no further reductions on SRM's mutation rate are done). Figure 4.8 plots results by an algorithm that makes the configuration transition from GA-SRM(50,100) when the mutation segment length in SRM has reached $\ell = \{6, 3\}$. As a reference it also includes the results presented in Figure 4.6 by GA-SRM(μ,λ). From Figure 4.8 it can be seen that neither CM nor SRM alone but the interaction of both CM and SRM leads to a higher convergence reliability.

To explain why the interaction of both CM and SRM works better than CM alone diversity values and performance should be looked at simultaneously. Figure 4.9 presents the fitness value of the best individual in the population and the average hamming distance to the best individual $\bar{h}$ over the generations for one of the runs by cGA and GA-SRM. The SRM's mutation segment reduction ℓ is also presented for GA-SRM.

It can be seen that cGA ends up with values of $\bar{h}$ higher than GA-SRM after T evaluations.

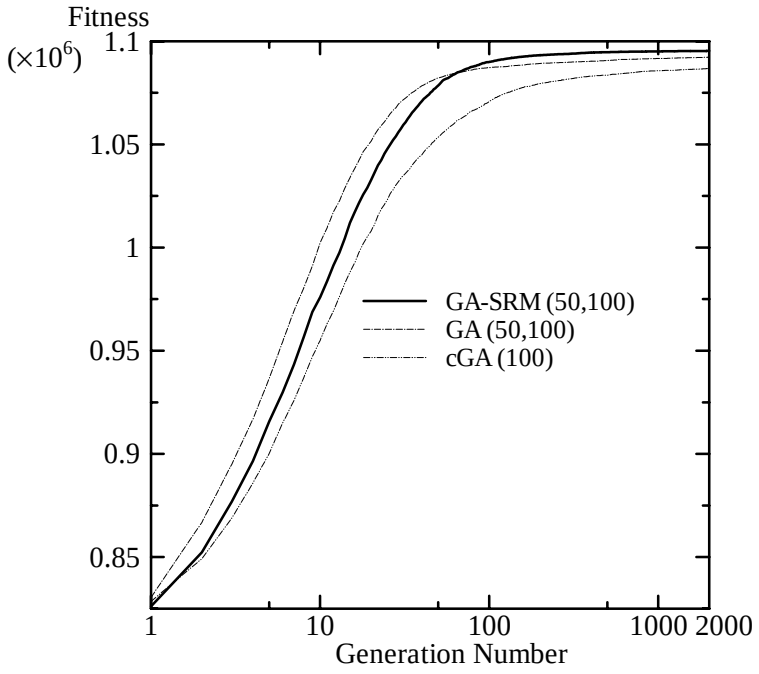


Figure 4.6: Average fitness in 100 runs of the best-so-far individual.

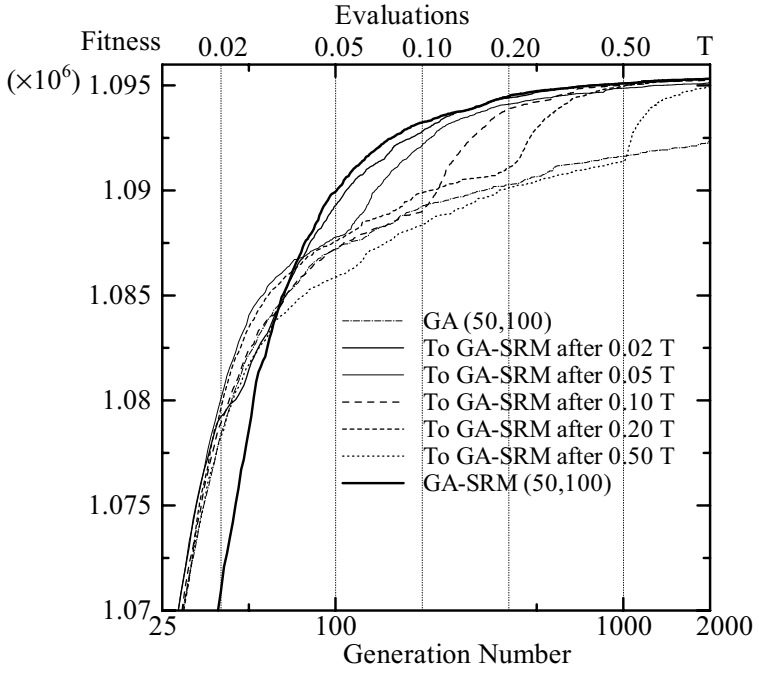


Figure 4.7: Configuration transition: GA(50,100) to GA-SRM(50,100).

Also, at the end of the run the number of diverse individuals in the parent population is about 95% in cGA and 83% in GA-SRM. Letting the cGA run for $4T$ it is found that $\bar{h}$ and the number of

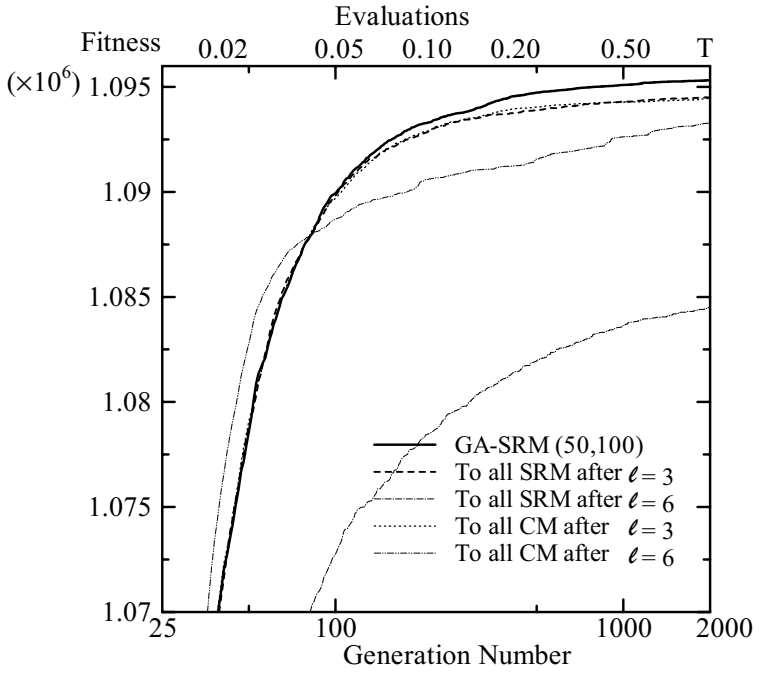


Figure 4.8: Configuration transition: GA-SRM(50,100) to all SRM or all CM.

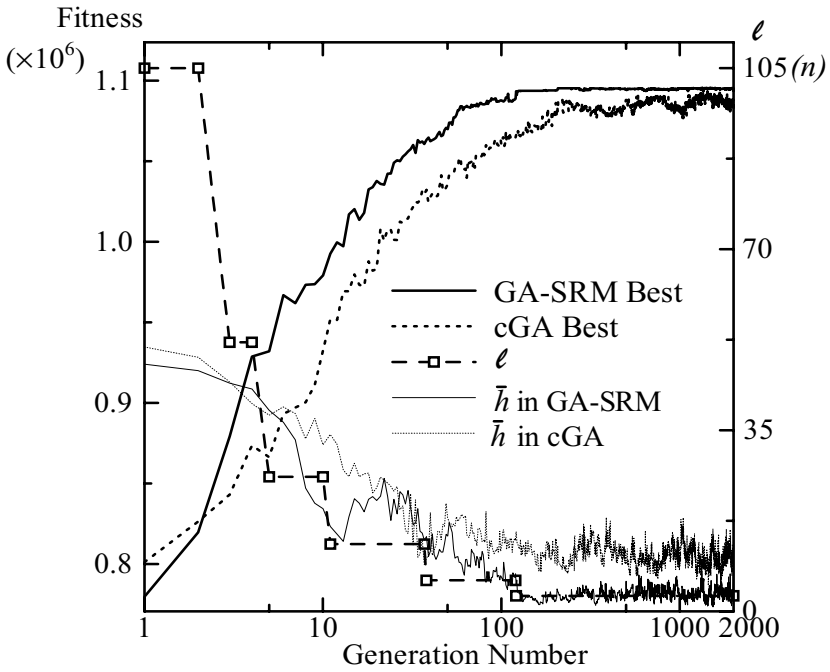


Figure 4.9: Diversity and search ability.

diverse individuals remains at the same levels and that the quality of the solution does not increase significantly (See 4.9). The higher levels of diversity observed in cGA after T evaluations and the

lack of improvement in the quality of solutions after $4T$ evaluations seem rather contradictory. The explanation for this comes from the highly multimodal nature of the landscape that the problem used in our simulations has and it is in accordance with the findings in [38]. In [38] it is shown that the recombination of high fitness individuals located on multiple peaks can often produce poor performing individuals in the valley between peaks, with the side effect of increasing the values of diversity metrics such as $\bar{h}$ and the number of diverse individuals. On the other hand, the presence of multiple peaks will have less influence on the mutation of a high fitness individual on a particular peak (assuming small mutation probabilities). The values of $\bar{h}$ in Figure 4.9 rather than being an indication of the cGA's ability to keep diversity show that CM alone has difficulties pulling the population to higher peaks. In the case of GA-SRM, the lost of effectiveness by CM during the final stages of the search is supplemented by the augment of SRM's contribution conform mutation rates are reduced as shown in Figure 4.3(a).

4.11 Results for Real-World 0/1 Multiple Knapsack Problems

In this section additional results for various real-world 0/1 multiple knapsack problems are presented. In Table 4.4 column *Problem* indicates the knapsack instance *Name*, the number of objects n (it corresponds to the chromosome bit string length), the number of knapsacks m and the known global optimum value *Max*. Column *Parameters* shows the specific values set for τ (used only in GA-SRM), CM's mutation probability $p_m^{(CM)} \approx 1/n$, and number of evaluations T. Table 4.5 show results by cGA (population of 100 individuals), GA(50,100), and GA-SRM(50,100) using either adaptive dynamic segment *SRM-ADS* or adaptive dynamic probability *SRM-ADP*. As a reference for comparison, Table 4.6 presents results for the same problems reported by Khuri et al.[52] running a genetic algorithm for the same T evaluations with a population of 50 individuals. Table 4.7 shows the latest results by Bäck et al.[31] particularly for *Weing 7* running a genetic algorithm with constant mutation rates and other enhanced genetic algorithms that apply varying mutations for the same $T = 2 \times 10^5$ evaluations using offspring populations of 100 individuals.

From Table 4.5 it can be seen that the proposed method outperforms cGA and GA(μ,λ) in every knapsack test problem where simulations were conducted. Also, although direct comparisons are not possible between GA-SRM(μ,λ) and the algorithms used in [31] and [52], looking at Table 4.5, Table 4.6 and Table 4.7 it can be seen that the proposed algorithm gives better results. It should be specially noticed the results obtained for *Weing 7* where the proposed GA-SRM found the global optimum 26% of the runs. In the same problem, genetic algorithms with either constant mutation rate or self-adaptive mutation rates could not find the global optimum. Note that the algorithm that uses a time dependent hyperbolic deterministic schedule for mutation rate control with a (15,100) selection mechanism found it only 3% of the runs[31, 52]. Based on observations of the final feasible solutions reached by the algorithms used in the simulations, it can be seen that for *Weing 7* in the ranges [1095000,1095445], [1094000,1095445], and [1093000,1095445], there are at least 30, 134, and 189 peaks of different heights, respectively. This data might help to visualize the kind of landscape and the global search ability of the algorithms.

It should be mentioned that ADS and ADP exhibit similar behavior. Although better results were obtained with ADS for most of the knapsack test problems used here, at this time it cannot be concluded whether ADS is superior to ADP. Also, the difference in performance between ADS and ADP might be relevant to the kind of epistasis[59] associated to the test problem. For the test problems used here there is no knowledge about the degree or the pattern of epistasis of the test problems. More investigation should be done to clarify this point in the future.

Table 4.4: Knapsack test problems.

Problem				*Parameters*		
Name	n	m	*Max*	τ	$p_m^{(CM)}$	T
Petersen 3	15	10	4015	0.48	0.067	5×10^3
Petersen 4	20	10	6120	0.52	0.050	10^4
Petersen 5	28	10	12400	0.48	0.036	5×10^4
Petersen 6	39	5	10618	0.48	0.030	10^5
Petersen 7	50	5	16537	0.48	0.020	10^5
Sento 1	60	30	7772	0.52	0.017	10^5
Sento 2	60	30	8722	0.52	0.017	10^5
Weing 7	105	2	1095445	0.40	0.01	2×10^5

Table 4.5: Results for various knapsack test problems.

	cGA(100)			*GA(50,100)*			*GA-SRM(50,100)*					
							SRM-ADS			*SRM-ADP*		
Name	*N*	*Average*	*Stdev*	*N*	*Average*	*Stdev*	*N*	*Average*	*Stdev*	*N*	*Average*	*Stdev*
Petersen 3	48	4007.0	12.0	85	4013.4	3.8	100	4015.0	0.0	97	4014.7	1.2
Petersen 4	6	6031.1	50.8	35	6099.7	59.6	42	6112.5	8.4	54	6113.5	8.1
Petersen 5	2	12278.0	55.7	50	12375.1	67.5	94	12398.9	5.5	98	12399.8	1.2
Petersen 6	-	10454.5	36.5	4	10524.7	67.4	16	10588.2	37.6	16	10587.3	24.5
Petersen 7	-	16300.9	53.7	-	16367.8	93.6	23	16485.2	53.1	21	16474.2	50.0
Sento 1	-	7505.1	50.5	14	7712.7	57.8	85	7770.3	5.4	67	7765.1	9.0
Sento 2	-	8506.3	33.9	-	8682.1	31.7	55	8718.5	5.3	50	8717.7	6.6
Weing 7	-	1085421.8	1881.2	-	1092615.0	2843.4	26	1095345.5	337.17	11	1094908.3	1106.3

Table 4.6: Results by Khuri et al. [52].

Name	*N*	*Average*
Petersen 3	83	4012.7
Petersen 4	33	6102.3
Petersen 5	33	12374.7
Petersen 6	4	10536.9
Petersen 7	1	16378.0
Sento 1	5	7626
Sento 2	2	8685
Weing 7	-	1093897

Table 4.7: Results for Weing 7 using other mutation schedules [31].

Mutation Schedule	*Selection*	*N*	*Average*
Constant mutation rate $p_m = 1/n$	(15,100) selection	-	1091268
	proportional selection	-	1093924
Self-adaptive	(15,100) selection	-	1092743
	proportional selection	-	1094311
Time-dependent hyperbolic deterministic	(15,100) selection	3	1094711
	proportional selection	-	1094479

4.12 Results for Random 0/1 Multiple Knapsack Problems

In the previous sections the structure of GA-SRM has been deeply examined. The effect of extinctive selection, varying mutation parallel to crossover, and the interaction of adaptively varying mutations parallel to crossover on several small real-world knapsacks problems were studied too.

In this section, to obtain a broader perspective on performance and scalability, the algorithms used before are applied to classes of larger random knapsacks problems varying the difficulty of the problem. As mentioned before, factors related to the difficulty of the problem are the number of knapsacks m (constraints), the size of the search space (number of objects n), and the tightness ratio ϕ (ratio of the feasible region to the whole search space).

First, the performance of the algorithms on classes of problems with different ratios of the feasible region (ϕ) is observed. Figure 4.10 illustrates results by the algorithms on problems with $m = 30$ knapsacks, $n = 100$ objects, and ratio $\phi = \{0.75, 0.50, 0.25\}$. The main conclusions drawn from Figure 4.10 are as follows. (i) The quality of solutions found by the GAs decrease (larger Gap values) as the ratio ϕ is reduced. These results are quite intuitive since reductions on ϕ imply reductions on the fraction of possible subsets of objects that constitute feasible solutions. Consequently, the ratio between the feasible part of the search space and the whole search space gets smaller and the smaller this ratio is the harder it is to find feasible results. This was also observed for 0/1 single ($m = 1$) knapsack problems in [30]. (ii) The performance of GA(50,100) is by far superior to cGA(100) for all values of ϕ, indicating that extinctive selection is an important factor to increase the performance of GAs in problems with infeasible regions. (iii) The algorithms that combine extinctive selection with varying mutations give better results than a simple GA with extinctive selection. (iv) GA-SRM with ADS is superior to GA-SRM with ADP; the gain in quality of the solutions comes from the use of segment mutation strategy (ADS). Note that the difference in performance between GA-SRM with ADS and the other algorithms becomes more apparent for problems with larger infeasible regions (more difficult problems).

Second, the effect of the number of constraints (knapsacks m) is observed fixing the size of the search space (number of objects n) and the ratio of the feasible region to the whole search space (tightness ratio ϕ). Figure 4.11 illustrates results by the GAs on problems of $m = \{5, 10, 30\}$ capacities, $n = 100$ objects, and ratio $\phi = 0.25$.

Similar to the tightness ratio ϕ of the feasible region, from Figure 4.11 it can be seen that increasing the number of constraints (knapsacks m) also has an strong impact on the performance of the algorithms. The behavior of the algorithms is similar to that observed in Figure 4.10. Looking at Figure 4.10 and Figure 4.11, judging from the relative increase on the Gap values and from the slopes of the curves, reducing the ratio of the feasible region (ϕ) has an stronger impact than increasing the number of constraints (knapsacks m).

Third, the effect of the size of the search space (number of objects n) is observed fixing the number of constraints (knapsacks m) and the ratio of the feasible region (tightness ratio ϕ). Figure 4.12 illustrates results by GAs on problems of $m = 30$ knapsacks, $n = \{100, 250, 500\}$ objects, and ratio $\phi = 0.25$. From Figure 4.12 it can be seen that increasing the size of the search space also makes it harder for the algorithms to find high quality solutions. Looking at Figure 4.10, Figure 4.11, and Figure 4.12 and again judging from the slopes of the curves, it can be seen that (for the values of n, m, and ϕ used here) increasing the size of the search space (n) presents less difficulties to the GAs than reducing the feasible region (ϕ) or increasing the number of constraints (m).

An important observation is that changing the mutation strategy from ADP to ADS in SRM noticeably increases the quality of results achieved by GA-SRM. The better results achieved by

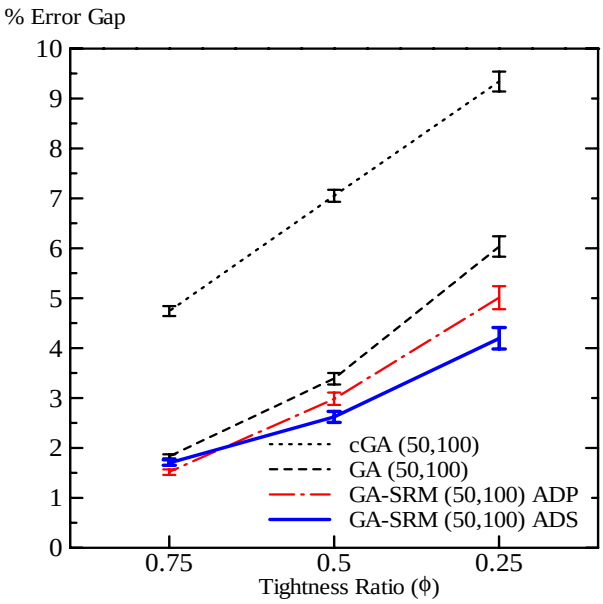


Figure 4.10: Reducing the feasible region $\phi = \{0.75, 0.50, 0.25\}$, $m = 30$, $n = 100$.

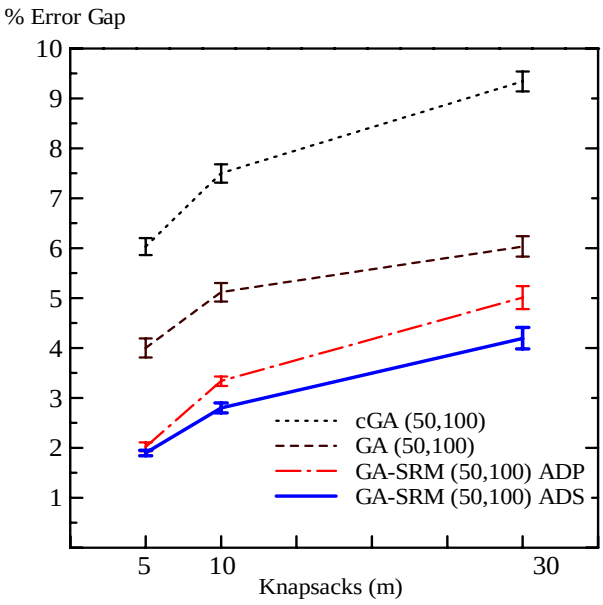


Figure 4.11: Increasing the number of constraints $m = \{5, 10, 30\}$, $n = 100$, $\phi = 0.25$.

GA-SRM with ADS[2] suggests that the design of parallel mutation could be an important factor to improve further the search performance of GAs.

The mutation strategies used by parallel mutation (SRM-ADS and SRM-ADP) is studied in detail in chapter 6 using NK-Landscapes[57] for problems having either nearest neighbor or random fitness epistatic patterns among bits. It is shown that SRM-ADS's convergence reliability

[2]GA-SRM with ADS has proven effective for other problems as well[71, 72]

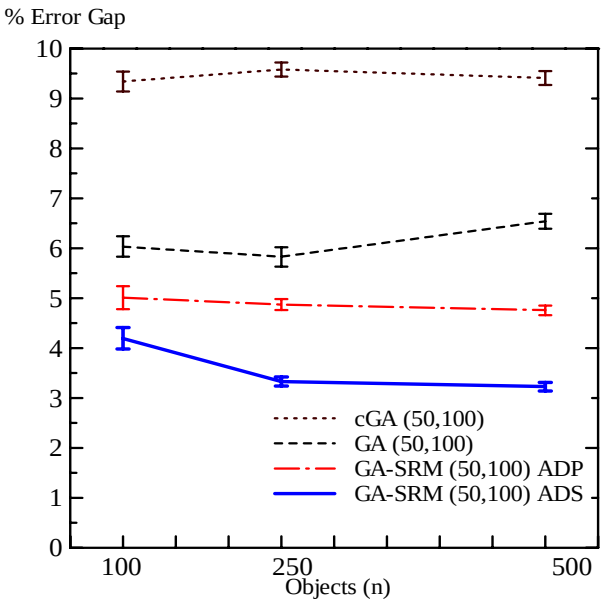


Figure 4.12: Increasing the search space $n = \{100, 250, 500\}$, $m = 30$, $\phi = 0.25$

(which applies mutation within a continuous segment of the chromosome) is higher than SRM-ADP's on problems with nearest neighbor epistatic patterns. When random epistatic patterns are present convergence reliability by SRM-ADS and SRM-ADP are similar[69, 73].

Since the object weights of the knapsack problems are drawn from a uniform generator, as mentioned in 5.2, random fitness epistatic patterns among bits would seem more likely to be present. However, the results obtained in this work by GA-SRM with ADS resemble those achieved in NK-Landscapes with nearest neighbor epistatic patterns (especially for medium and large values of K; that is, for highly multimodal problems).

It should be noticed that in NK-Landscapes the whole search space constitutes a feasible region whereas the knapsack problems are highly constrained. Since a penalty term is used in the fitness function, unfeasible solutions would map to low fitness values increasing the sharpness around the peaks. However, it is not clear whether and how the constraints (number of knapsacks m) and the restrictiveness of the capacities (ratio ϕ) could affect the fitness epistatic interactions among bits. This deserves further research.

Similar to the case of real-world knapsack problems, the contribution of parallel adaptive mutation SRM and the robustness of its adaptive mechanism in the larger random problems are also verified. Figure 4.13 plots results by an algorithm that makes the configuration transition from GA(50,100) to GA-SRM(50,100) at $\{0.10T, 0.20T, 0.5T\}$ evaluations, respectively. Every time a transition takes place, initial mutation probability for SRM is set to $p_m^{(SRM)} = 0.5$. As a reference it also includes results by GA(50,100) and GA-SRM(50,100). From Figure 4.13 it can be seen that as soon as SRM is included fitness starts to pick up increasing the convergence reliability of the algorithm. The almost instantaneous pick up on fitness after the transitions in Figure 4.13 is also an indication of the robustness of the adaptation mechanism. That is, it quickly finds the suitable mutation probability regardless of the stage of the search.

Figure 4.14 plots results by an algorithm that makes the configuration transition from GA-SRM(50,100) to a all CM regime with extinctive selection or to a all SRM regime with extinctive selection (in the latter case no further adaptations on SRM's mutation rate are done) as soon as

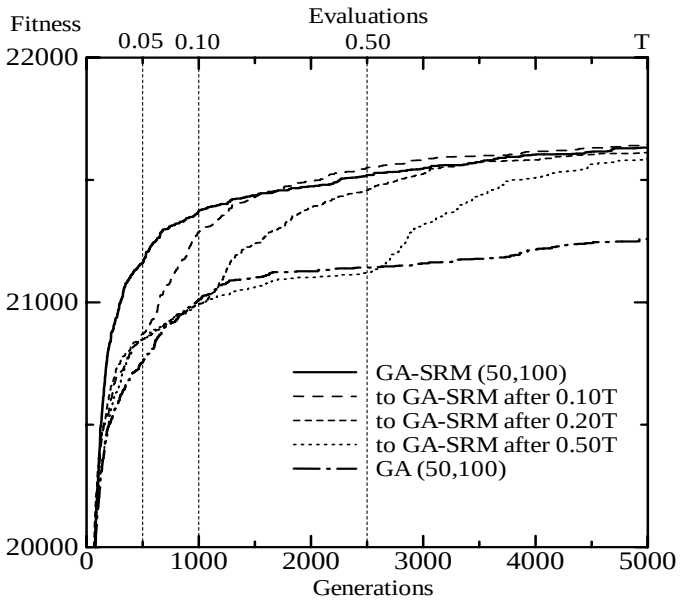


Figure 4.13: Transition from GA(50,100) to GA-SRM(50,100) at various fractions of T

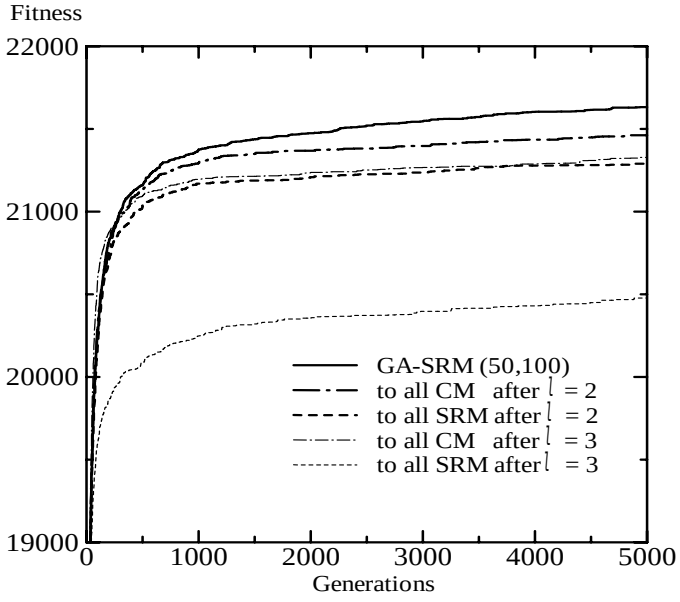


Figure 4.14: Transitions from GA-SRM(50,100) to either all CM(50,100) or all SRM(50,100) regimes at $\ell = \{2, 3\}$

the mutation segment length in SRM has reached $\ell = \{2, 3\}$. As a reference it also includes the results by GA-SRM(50,100). From Figure 4.14 it can be seen that neither CM nor SRM alone but the interaction of both CM and SRM leads to a higher convergence reliability. Figure 4.14 shows that CM alone (even though extinctive selection is on) has difficulties pulling the population to higher peaks. In the case of GA-SRM, the lost of effectiveness by CM during the final stages

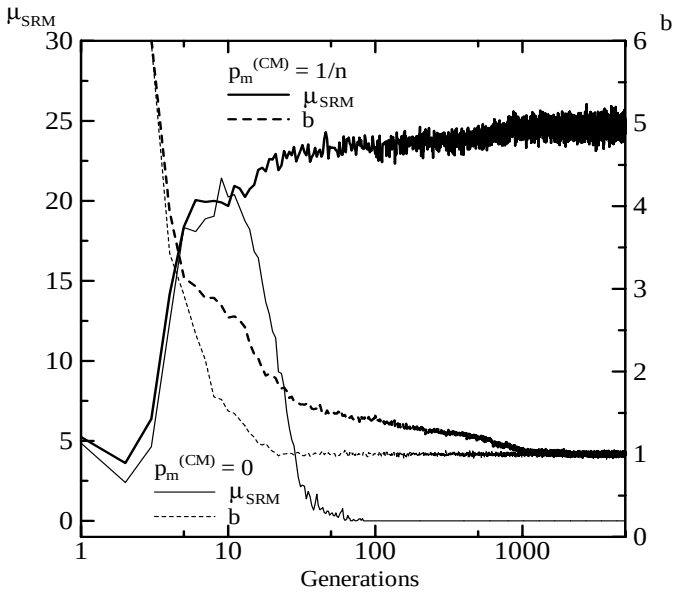


Figure 4.15: SRM's contribution to the parent population (μ_{SRM}) and number of bits actually flipped by SRM (b) for CM's mutation probabilities of $p_m^{(CM)} = \{0, 1/n\}$

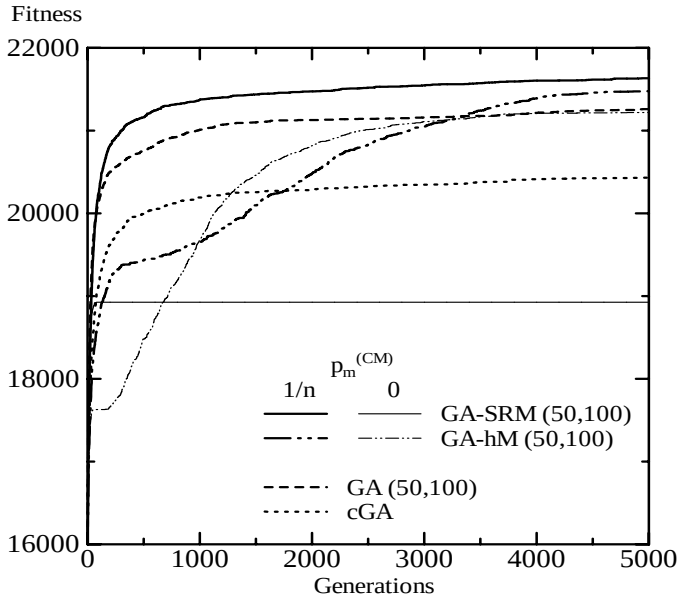


Figure 4.16: Fitness Transition for CM's mutation probabilities of $p_m^{(CM)} = \{0, 1/n\}$

of the search is supplemented by the augment of SRM's contribution conform mutation rates are reduced as shown in Figure 4.15 ($p_m^{(CM)} = 1/n$).

Furthermore, the behavior of GA-SRM when CM's mutation probability is either on or off is also verified for random problems. Figure 4.15 plots SRM's average contribution (in 50 runs) to the parent population (μ_{SRM}) and the average number of bits actually flipped by SRM (b) for

CM's mutation probabilities of $p_m^{CM} = \{0, 1/n\}$, respectively. From Figure 4.15 it can be seen that when $p_m^{(CM)} = 1/n$ the average number of SRM's offspring that survive selection (μ_{SRM}) increases as SRM's mutation rates are reduced. For $p_m^{(CM)} = 0$, however, it can be seen that after few generations SRM does not contribute to the parent population, causing a premature reduction of SRM's mutation rate, lost of diversity in the population and convergence to a local optimum. Note that in this case diversity is not being introduced at all neither by CM ($p_m^{(CM)} = 0$) nor by SRM ($\mu_{SRM} = 0$).

Finally, Figure 4.16 shows the fitness of the best so far individual by GA-SRM(50,100) and GA-hM(50,100) when CM's mutation probability is either on or off. GA-hM(50,100) is a parallel varying mutation algorithm that includes deterministic varying mutations instead of the adaptive approach. Results by cGA(100) and GA(50,100) are also included for comparison. The major conclusion from this figure is that the lack of "background" mutation in CM affects negatively the convergence reliability even if deterministic mutation is used, although in this case the mutation schedule is not affected because mutation rate depends on time.

Again, in the case that *fitness duplicates* are not eliminated, small mutation after crossover in CM helps to achieve a robust search performance on random problems by keeping an appropriate balance between CM and SRM.

4.13 Conclusions

This chapter has studied the internal structure of GA-SRM applying adapting varying mutations (SRM) parallel to crossover & background mutation (CM) putting the operators in a cooperative-competitive stand with each other by way of extinctive selection. Important structural issues such the balance between CM and SRM for offspring creation, the mutation probability in CM, the ratio between number of parents and number of offspring (extinctive selection pressure), and the threshold to trigger adaptation in SRM were analyzed. It was found that the sexual operator CM performs better than the asexual operator SRM during the initial stages of the search. On the other hand, SMR's contribution significantly increases as the search progresses, mutation rates are reduced, and the population approaches the global optimum. Also, in spite of CM's initial effectiveness, configurations favoring SRM (mutation in general) result into better performance than configurations favoring CM. Properly balancing the operators the cooperation expected from CM and SRM emerges producing a higher search velocity and higher search reliability in both real-world and random generated 0/1 multiple knapsacks problems. Small mutation after crossover in CM helps to achieve a robust search performance by keeping an appropriate balance between CM and SRM. The robustness of the adaptation mechanism in SRM was verified and it was also shown that neither CM nor SRM alone but the interaction of both CM and SRM leads to higher convergence reliability. Regarding mutation strategy, it was found that ADS performs better than ADP in these classes of problems. Comparisons were performed with canonical GAs and other enhanced GAs.

Chapter 5

Comparing Standard and Parallel Varying Mutation GAs: Effect of the Major Components

This chapter compares the proposed model of parallel varying mutation with the standard model of varying mutation across a broad range of problems. The statistical significance of the results is verified with ANOVA tests. First, the models are compared using deterministic varying mutations. Then, the models are compared using self-adaptive mutations. It is found that the proposed model is more effective and efficient than the standard model in both deterministic and self-adaptive varying mutation GAs. It is also found that the standard model of varying mutations affects negatively the (adaptive) self-adaptive mutation rate control. This strongly suggests that the standard model of varying mutation GAs may not be appropriate for combining forms of control.

5.1 Introduction

The standard model of varying mutation GAs raises several important questions such as: Is there interference between crossover and high mutation? If so, does it affect performance of the algorithm? In the case of (adaptive) self-adaptive varying mutation algorithms, does it affect the mutation rate control itself? And more generally, is it an appropriate model for combining forms of control (co-adaptation of parameters)?

This chapter tries to answer such questions by comparing the performance of standard and parallel models of varying mutation GAs using deterministic and self-adaptive mutation rate controls for varying mutation[74, 75, 76]. As explained in 2.3.3. The deterministic time-dependent schedule reduces mutation rates in a hyperbolic shape[31]. The self-adaptive approach uses a continuous representation for mutation rates and updates mutation probabilities using a learning rate[31, 33]. In the following varying mutation *serial to crossover* refers to the standard model of varying mutation and varying mutation *parallel to crossover* refers to the proposed model of varying mutation.

The effect on performance of extinctive selection and higher mutations is studied in both models of varying mutation GAs.

5.2 Experimental Setup

The following GAs are used in the simulations. A simple canonical GA that applies crossover followed by background mutation, denoted as cGA. A GA with deterministic varying mutation serial (parallel) to crossover, denoted as hGA (GA-hM). A GA with self-adaptive varying mutation serial (parallel) to crossover, denoted as sGA (GA-sM).

The GAs use either proportional selection or (μ,λ) proportional selection. This is indicated by appending to the name of the GA (μ) or (μ,λ), respectively[1]. All algorithms use fitness linear scaling and mating is restricted to $(\boldsymbol{x}_i, \boldsymbol{x}_j),\ i \neq j$, so a solution will not cross with itself. For cGA, hGA, and sGA $p_c = 0.6$ and for GA-hM and GA-sM the ratio for offspring creation is set to $\lambda_{CM} : \lambda_{SRM} = 1 : 1$. Background mutation is set to $p_m^{(CM)} = 1/n$. The learning rate for self-adaptation is set to $\gamma = 0.2$.

This chapter uses the difficult, large, and highly constrained, 0/1 multiple knapsack problems[2][53], created by the test problem generator as explained in 3.1, to observe the behavior of the varying mutation GAs in a broad range of classes of problems. Results are averaged over 10 problems for each of the seven classes of problems (50 runs per problem) and the number of generations is set to $T = 5000$. Vertical bars overlaying the mean curves represent 95% confidence intervals. The statistical significance of the results is verified using ANOVA tests.

5.3 Simple GA and Extinctive Selection

First, we observe the effect of extinctive selection on the performance of a simple GA. Figure 5.1 plots the fitness of the best-so-far individual over the generations by the canonical cGA(100) and a simple GA using $(\mu, \lambda) = \{(15, 100), (50, 100)\}$ extinctive ratios. From this figure we see that extinctive selection alone remarkable improves the solution quality reached by the cGA in this kind of problems.

[1] a simple GA with (μ,λ) Proportional Selection is denoted GA(μ,λ)

[2] http://mscmga.ms.ic.ac.uk/jeb/orlib/info.html

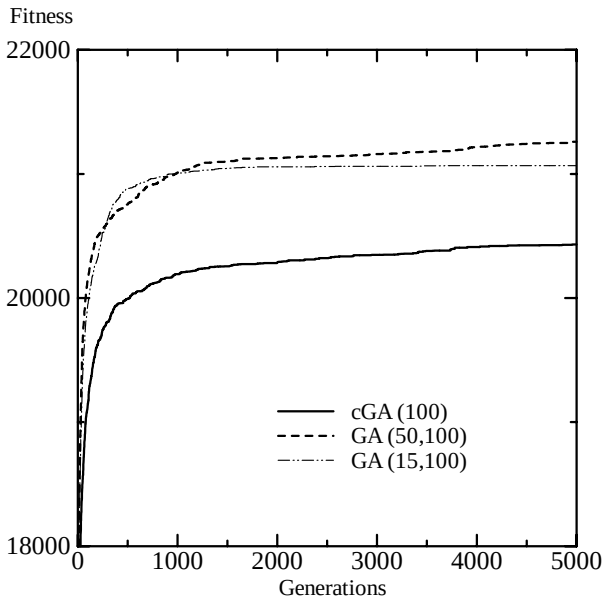


Figure 5.1: Effect of Extinctive Selection on a simple GA: Fitness transition over the generations by cGA and GA(μ, λ)

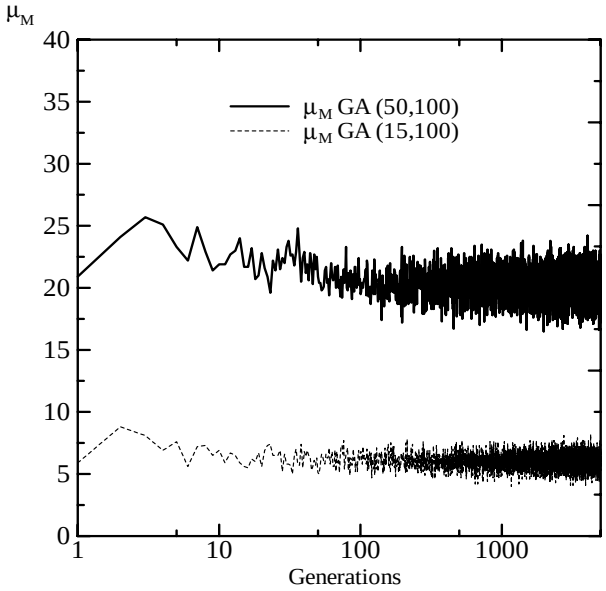


Figure 5.2: Effect of Extinctive Selection on a simple GA: M's contribution to the parent population (μ_M) in GA(μ, λ) ($p_c = 0.6$)

Figure 5.2 plots the number of individuals created by implicit parallel mutation M[3] present in the parent population after extinctive selection (μ_M). Looking at GA(50,100), for example, we can see that $\mu_M \sim 20$ throughout the run. That is, only about 50% of the offspring created by CM survive extinctive selection ($p_c = 0.6$). Since GA(50,100) exhibits higher convergence than cGA(100), this suggests that the crossover based operator is highly disruptive.

[3]Implicit parallel mutation M refers to mutation when is applied alone in the absence of crossover ($1 - p_c$), see 2.2

As mentioned before, the problems used in this study are highly constrained with sparse feasible regions where algorithms with penalty functions have a hard time finding feasible solutions[30, 53]. A higher selection pressure in these problems is helping the algorithm to focus the search around the feasible regions.

Extinctive selection has also another effect. It increases the convergence speed of the algorithm[24]. Both GA(15,100) and GA(50,100) are faster than cGA(100). However, between (15,100) and (50,100) we observe that the latter gives better final results than the former.

5.4 Varying Mutation without Extinctive Selection

Second, we observe the effect of deterministically varying mutations either serial or parallel to crossover when no extinctive selection is being used. Figure 5.3 plots results by hGA(100) and GA-hM(100). In both algorithms the range for either serial or parallel varying mutation is $[0.5, 1/n]$. As a reference for comparison, it also presents results obtained by cGA(100) and GA(50,100).

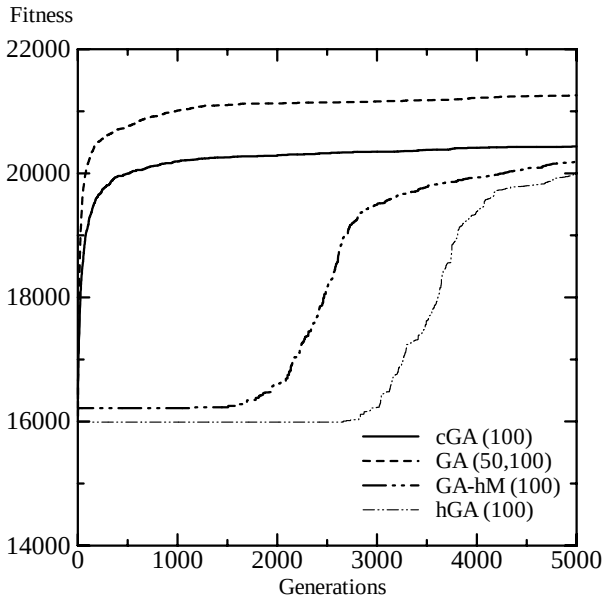


Figure 5.3: Deterministic Varying Mutation without Extinctive Selection

From Figure 5.3 we can observe that varying mutations either serial or parallel to crossover without extinctive selection undermines the reliability of the algorithm (final results are poorer than cGA's). Convergence velocity is considerably affected as well.

It should be remarked that there is an initial period consisting of a large number of generations in which the fitness of the best-so-far individual by hGA and GA-hM corresponds to the best feasible individual randomly created at generation 0 (horizontal lines). The mutation probability applied by these algorithms is too high at the beginning of the search. As mutation is reduced with the number of generations, the mutation probability becomes more suitable for these constrained problems and fitness picks up.

In the case of GA-hM this initial *flat* period can be attributed exclusively to a too high mutation probability used by the varying mutation operator. However, in the case of hGA the disruption

that mutation causes to crossover also contributes to extend this *flat* period. Note that between hGA and GA-hM, which apply the same determinist schedule for mutation rate control, GA-hM's fitness picks up much earlier than hGA's. This is a clear indication of the disruption caused by high mutation after crossover in the case of hGA.

5.5 Extinctive Selection and Varying Mutation

5.5.1 Deterministic Varying Mutation

Deterministic mutation varies mutation rates with exactly the same schedule whether it is applied serial (hGA) or parallel to crossover (GA-hM) and therefore is an ideal candidate to isolate and observe the impact of higher mutations in both models of GAs.

In order to observe in detail the effect of extinctive selection and deterministic varying mutation serial/parallel to crossover experiments are conducted using various populations $(\mu, \lambda) = \{(15, 100), (50, 100), (100, 100)\}$ and initial mutation probabilities $p_m^{(t=0)} = \{0.50, 0.10, 0.05\}$. Figure 5.4 and 5.5 plot the average fitness of the best-so-far individual over the generations illustrating the convergence behavior by hGA and GA-hM, respectively. Results by the simple GA with extinctive selection GA(50,100) are also included for comparison.

Looking at Figure 5.3 and Figure 5.4 we can see that extinctive selection, similar to the case of cGA, increases the performance of hGA. Moreover, the combination of extinctive selection and varying mutation produces results with better final quality than the simple GA with extinctive selection.

As we increase the extinctive pressure (reduce μ while keeping constant λ) the initial *flat* period is shorten. See for example hGA(50,100) and hGA(15,100) for $p_m^{(t=0)} = 0.5$. In these cases mutation would not be less harmful than it is in hGA(100) of Figure 5.3, but extinctive selection will discard much of the poor performing individuals. Note, however, that better final results are given by (μ,λ)=(50,100) rather than by (μ,λ)=(15,100).

Also, as expected, using lower values for the initial mutation probability helps to speed up convergence. See for example hGA(50,100) for $p_m^{(t=0)} = 0.5$ and $p_m^{(t=0)} = 0.05$. These smaller initial mutation values, however, do not help to increase the quality of the final results.

If convergence speed is to be considered, then from this figure it also becomes clear that deterministic varying mutations mechanisms, besides the number of generations, incorporates the initial mutation probability as an additional parameter, which must be set properly in order to achieve high performance.

From Figure 5.5 we can see that the inclusion of extinctive selection and the reductions of the initial mutation probability in GA-hM produce similar effects to those remarked for hGA.

Looking at both Figure 5.4 and Figure 5.5 becomes more apparent that varying mutation parallel to crossover is less disruptive than varying mutation serial to crossover. Contrary to hGA, in the case of GA-hM the initial *flat* periods have disappeared and in all cases GA-hM converges faster than hGA for similar values of (μ,λ) extinctive ratio and initial mutation probability $p_m^{(hM)(t=0)}$. Also, GA-hM(50,100)'s final quality is better than hGA(50,100)'s (the statistical significance is shown below).

As a consequence of the less disruptiveness of mutation parallel to crossover, the initial value $p_m^{(t=0)}$ set for varying mutation in GA-hM has a smaller impact on convergence speed than it does in hGA. See for example GA-hM(50,100) for $p_m^{(t=0)} = 0.5$ and $p_m^{(t=0)} = 0.05$ and compare it with hGA for similar settings in Figure 5.4. Thus, GA-hM can be considered as a more robust algorithm than hGA.

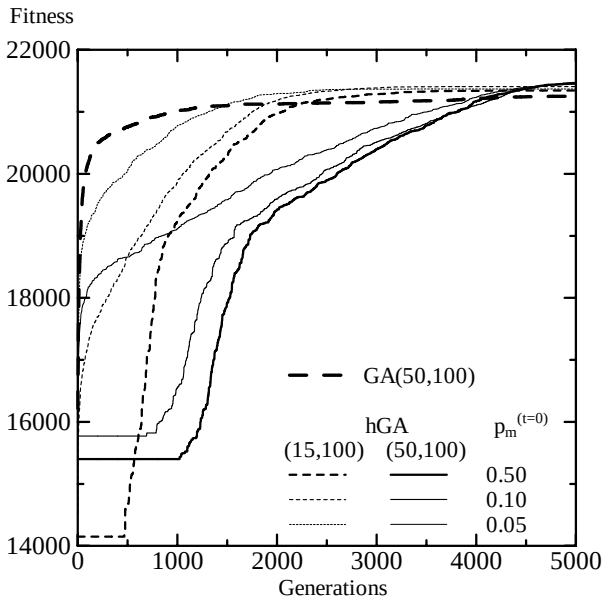


Figure 5.4: Deterministic Varying Mutation Serial to Crossover ($m = 30$, $n = 100$, $\phi = 0.25$)

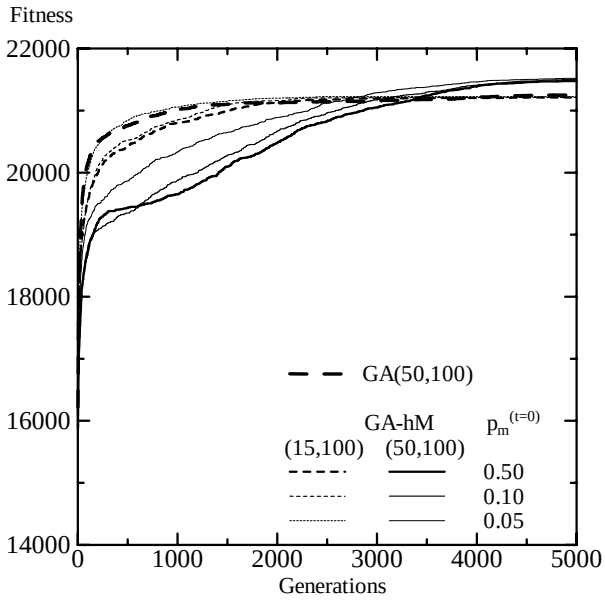


Figure 5.5: Deterministic Varying Mutation Parallel to Crossover ($m = 30$, $n = 100$, $\phi = 0.25$)

In GA-hM, similar to GA and hGA, a (μ, λ)=(50,100) extinctive ratio gives better final results than (μ, λ)=(15,100). In fact, notice that GA-hM(15,100)'s final quality is not better than GA(50,100)'s, which does not apply varying mutations. This is because the offspring created by varying parallel mutation hM within GA-hM have to compete with CM's offspring (that always applies M with background mutation) and a (15,100) extinctive selection turns out to be too strong. A less strong selection pressure, such (50,100), gives better chances to hM's offspring, which in turn would help to improve the search process.

Figure 5.6, Figure 5.7, and Figure 5.8 plot the average percentage error gap by hGA and

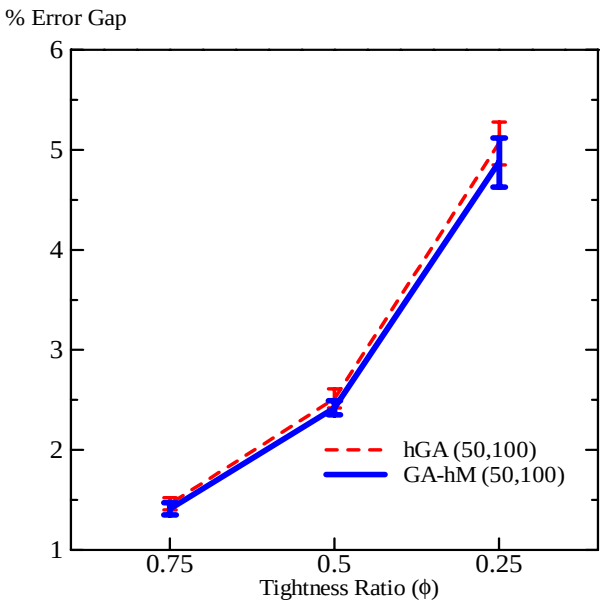


Figure 5.6: Convergence Reliability of Deterministic Varying Mutation GAs: Reducing the feasible region $\phi = \{0.75, 0.50, 0.25\}$, $m = 30$, $n = 100$.

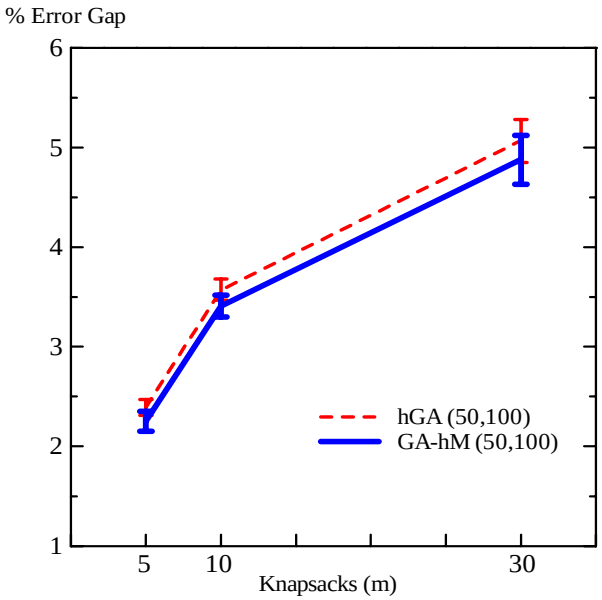


Figure 5.7: Convergence Reliability of Deterministic Varying Mutation GAs: Increasing the number of constraints $m = \{5, 10, 30\}$, $n = 100$, $\phi = 0.25$.

GA-hM. These figures show the effect on convergence reliability of reducing the feasible region (ϕ), increasing the number of constraints (m), and increasing the search space (n), respectively. The vertical bars, overlaying the mean curves, represent 95% confidence intervals.

The statistical significance of the results achieved by hGA and GA-hM is verified. Table

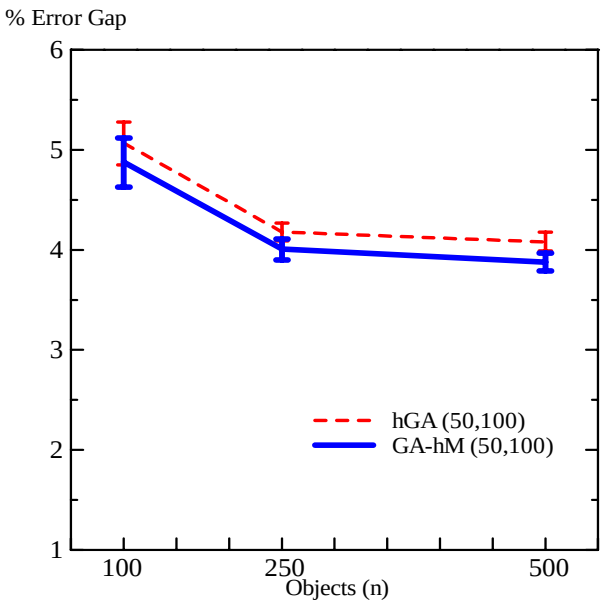


Figure 5.8: Convergence Reliability of Deterministic Varying Mutation GAs: Increasing the search space $n = \{100, 250, 500\}$, $m = 30$, $\phi = 0.25$.

5.1 summarizes the three two-factor factorial analysis of variance (ANOVA) corresponding to the plots presented in Figure 5.6, Figure 5.7, and Figure 5.8. In Table 5.1 *Source* indicates the source of variation, *df* the degrees of freedom, F is the ratio between the mean square treatment and the mean square error, and *Pval* is the *p value* (the smallest significant level α that would allow rejection of the null hypothesis). The treatment means are illustrated in Table 5.2, Table 5.3, and Table 5.4, respectively.

From Table 5.1, inspection of the p values reveals that in the case of reducing the feasible region (ϕ) there is *some indication* of an effect by the GA type factor (serial/parallel application of deterministic varying mutation). Note that *Pval* $= 0.0766$ is not much greater than $\alpha = 0.05$. In the cases of increasing the number of constraints (m) and the size of the search space (n) there are indications of a *strong main effect* by the GA type concluding that the parallel deterministic varying mutation (GA-hM) attains significantly smaller error than the standard deterministic varying mutation GA (hGA). Notice that the p values 0.0157 and 0.0047, respectively, are considerably less than 0.05. Furthermore, note that in all three cases there is indication of a main effect by the problem difficulty factor (ϕ, m, and n) but the interaction between GA type and problem difficulty factor was not significant.

Table 5.1: Factorial ANOVA for hGA and GA-hM

Source	SS	df	MS	F	*Pval*
GA	0.18371	1	0.18371	3.260	0.0766
ϕ	132.47870	2	66.23935	1175.553	0.0000
GA-ϕ	0.05120	2	0.02560	0.454	0.6373
Error	3.04276	54	0.05635		
Total	135.75637	59			
GA	0.40017	1	0.40017	6.219	0.0157
m	70.77277	2	35.38639	549.927	0.0000
GA-m	0.00654	2	0.00327	0.051	0.9505
Error	3.47476	54	0.06435		
Total	74.65424	59			
GA	0.54150	1	0.54150	8.685	0.0047
n	11.72973	2	5.86487	94.062	0.0000
GA-n	0.00121	2	0.00060	0.010	0.9903
Error	3.36696	54	0.06235		
Total	15.63940	59			

Table 5.2: Error Gap Means by hGA and GA-hM reducing the feasible region (ratio ϕ)

GA	Ratio ϕ			
	0.75 (ϕ_1)	0.50 (ϕ_2)	0.25 (ϕ_3)	
	1.49, 1.48	2.56, 2.38	4.92, 5.14	
hGA	1.37, 1.31	2.51, 2.63	5.22, 5.33	
(g_1)	1.48, 1.42	2.74, 2.7	4.96, 5.5	
	1.36, 1.65	2.32, 2.36	4.7, 5.39	
	1.45, 1.57	2.56, 2.37	5.17, 4.34	
	14.58	25.13	50.67	$T_{g_1} = 90.38$
	1.46	2.51	5.07	$\mu_{g_1} = 3.01$
	1.38, 1.42	2.46, 2.27	4.88, 5.02	
GA-hM	1.35, 1.3	2.38, 2.56	5.06, 5.13	
(g_2)	1.54, 1.36	2.56, 2.57	4.59, 5.33	
	1.31, 1.59	2.36, 2.26	4.43, 5.22	
	1.35, 1.47	2.46, 2.34	5.06, 4.05	
Totals	14.07	24.22	48.77	$T_{g_1} = 87.06$
Averages	1.41	2.42	4.88	$\mu_{g_2} = 2.90$
Totals	$T_{\phi_1} = 28.65$	$T_{\phi_2} = 49.35$	$T_{\phi_3} = 99.44$	
Averages	$\mu_{\phi_1} = 1.43$	$\mu_{\phi_2} = 2.47$	$\mu_{\phi_3} = 4.97$	

Table 5.3: Error Gap Means by hGA and GA-hM increasing the number of constraints (knapsacks m)

GA	Knapsacks m			
	5 (m_1)	10 (m_2)	30 (m_3)	
	2.27, 2.43	3.35, 3.44	4.92, 5.14	
hGA	2.53, 2.22	3.64, 3.43	5.22, 5.33	
(g_1)	2.50, 2.42	3.55, 3.71	4.96, 5.50	
	2.60, 2.24	3.44, 3.58	4.70, 5.39	
	2.30, 2.36	3.68, 3.91	5.17, 4.34	
Totals	23.87	=35.73	50.67	$T_{g_1} = 110.27$
Averages	2.39	3.57	5.07	$\mu_{g_1} = 3.68$
	2.10, 2.36	3.25, 3.36	4.88, 5.02	
GA-hM	2.35, 1.99	3.55, 3.11	5.06, 5.13	
(g_2)	2.24, 2.39	3.56, 3.35	4.59, 5.33	
	2.49, 2.25	3.29, 3.48	4.43, 5.22	
	2.02, 2.29	3.44, 3.73	5.06, 4.05	
Totals	22.48	34.12	48.77	$T_{g_2} = 105.37$
Averages	2.25	3.41	4.88	$\mu_{g_2} = 3.51$
Totals	$T_{m_1} = 46.35$	$T_{m_2} = 69.85$	$T_{m_3} = 99.44$	
Averages	$\mu_{m_1} = 2.32$	$\mu_{m_2} = 3.49$	$\mu_{m_3} = 4.97$	

Table 5.4: Error Gap Means by hGA and GA-hM increasing the search space (number of objects n)

GA	Objecs n			
	100 (n_1)	250 (n_2)	500 (n_3)	
	4.92, 5.14	4.34, 4.04	3.94, 4.34	
hGA	5.22, 5.33	4.21, 4.05	4.11, 4.12	
(g_1)	4.96, 5.5	3.89, 4.25	4.16, 3.76	
	4.7, 5.39	4.21, 4.27	4.19, 4.13	
	5.17, 4.34	4.31, 4.27	4, 4.09	
Totals	50.67	41.84	40.84	$T_{g_1} = 133.35$
Averages	5.07	4.18	4.08	$\mu_{g_1} = 4.45$
	4.88, 5.02	4.02, 3.86	3.98, 4.1	
GA-hM	5.06, 5.13	4, 3.95	3.93, 3.87	
(g_2)	4.59, 5.33	3.64, 4.1	3.91, 3.6	
	4.43, 5.22	4.17, 4.01	4.01, 3.91	
	5.06, 4.05	4.14, 4.16	3.73, 3.79	
Totals	48.77	40.05	38.83	$T_{g_2} = 127.65$
Averages	4.88	4.01	3.88	$\mu_{g_2} = 4.26$
Totals	$T_{n_1} = 99.44$	$T_{n_2} = 81.89$	$T_{n_3} = 79.67$	
Averages	$\mu_{n_1} = 4.97$	$\mu_{n_2} = 4.09$	$\mu_{n_3} = 3.98$	

5.5.2 Self-Adaptive Varying Mutation

A self-adaptive scheme uses one mutation rate per individual, which are usually set at $t = 0$ to random values in the range allowed for mutation. Two important ingredients of self-adaptation are the diversity of parameter settings and the capability of the method to adapt the parameters. It has been indicated that some of the implementations of self-adaptation exploit more the diversity of parameter settings rather than adapting them. However, it has also been argued that the key to the success of self-adaptation seems to consist in using at the same time both a reasonably fast adaptation and reasonably large diversity to achieve a good convergence velocity and a good convergence reliability, respectively[33].

To observe the influence that the serial/parallel application of varying mutations could have on the self-adaptive capability itself we avoid initial diversity of parameters. Experiments are conducted using populations $(\mu, \lambda) = \{(15, 100), (50, 100)\}$ and mutation ranges of $p_m = [p_m^{min}, p_m^{max}] = [1/n, \{0.50,\ 0.25, 0.10, 0.05\}]$. In all cases initial mutation for each individual is set to the maximum value allowed for the range, $p_m^{(t=0)} = p_m^{max}$. Figure 5.9 and Figure 5.10 plot the average fitness of the best-so-far individual over the generations illustrating the convergence behavior by sGA and GA-sM, respectively. Results by GA(50,100) are also included for comparison.

From these and previous figures it is worth noting the following. (i) Self-adaptive mutation increases convergence speed compared to deterministic mutation either serial or parallel to crossover. Looking at Figure 5.9 and Figure 5.4, note that in sGA the initial *flat* periods observed in hGA have disappeared completely. Also, looking at Figure 5.10 and Figure 5.5 we can see that GA-sM(50,100)'s fitness picks up much earlier than GA-hM(50,100)'s for similar values of $p_m^{(t=0)}$. Between sGA and GA-sM, however, looking at Figure 5.9 and Figure 5.10 note that sGA can match GA-sM's convergence velocity only for small values of $p_m^{(t=0)}$. This is an indication that even in the presence of adaptation the convergence velocity of a GA that applies varying mutation serial to crossover would depend heavily on initial mutation rates, which is not an issue if adaptive mutation is applied parallel to crossover. (ii) Contrary to deterministic varying mutation, convergence reliability of self-adaptive mutation serial to crossover could be severely affected, which becomes quite notorious if no initial diversity of parameters is allowed. Note in Figure 5.9 that only the configurations of sGA(50,100) having $p_m^{(t=0)} = \{0.10, 0.05\}$ achieved better final results than GA(50,100). On the other hand, the initial lack of diversity of parameters does not affect convergence reliability of GA-sM. Note in Figure 5.10 that for the same selection pressure convergence reliability of GA-sM is similar for all values of $p_m^{(t=0)}$. (iii) Similar to deterministic varying mutation, better results are achieved by $(\mu, \lambda) = (50, 100)$ rather than by $(\mu, \lambda) = (15, 100)$.

Next, we allow for initial diversity of parameters setting $p_m^{(t=0)}$ to a random value between the minimum and maximum value allowed for mutation. In this case, the disruption that higher serial mutation causes to crossover becomes less apparent due to the initial diversity of parameters and convergence speed is similar for both sGA and GA-sM. Convergence reliability of sGA also improves. However, the negative impact on reliability remains quite significant for sGA. Figure 5.11 and Figure 5.12 illustrate the fitness transition and the average flipped bits (Log scale) by sGA and GA-sM both with random initial mutation rates between [1/n,0.50]. Results for hGA and GA-hM are also included in Figure 5.11 for comparison. From these figures note that sGA converges to lower fitness and reduces mutation rates faster than GA-sM.

The self-adaptation principle tries to exploit the indirect link between favorable strategy parameters and objective function values. That is, appropriate parameters would lead to fitter individuals, which in turn are more likely to survive and hence propagate the parameter they carry with them to their offspring. A GA that applies varying mutation parallel to crossover as GA-sM can interpret better the self-adaptation principle and achieve higher performance because (i) inappro-

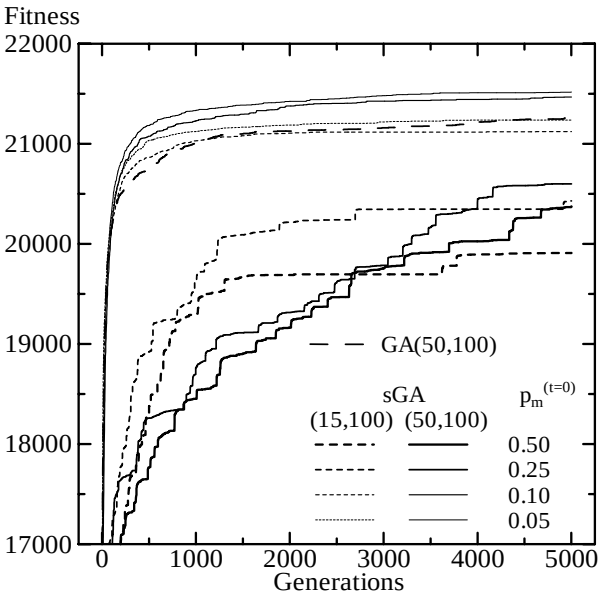


Figure 5.9: Self-Adaptive Varying Mutation Serial to Crossover , $p_m^{(t=0)}(i) = p_m^{max}$ ($m = 30$, $n = 100$, $\phi = 0.25$).

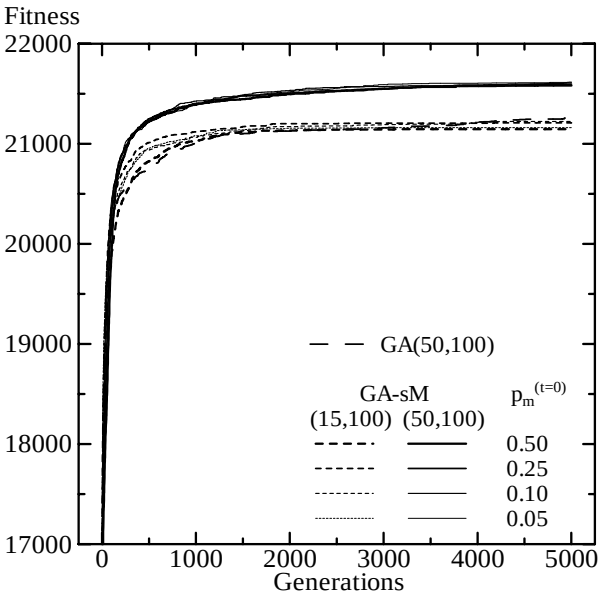


Figure 5.10: Self-Adaptive Varying Mutation Parallel to Crossover, $p_m^{(t=0)}(i) = p_m^{max}$ ($m = 30$, $n = 100$, $\phi = 0.25$).

priate mutation parameters do not disrupt crossover, and (ii) it preserves mutation rates (see Eq. (2.4)) that are being useful to the search. A GA that applies varying mutation serial to crossover as sGA, however, can mislead the mutation rate control because (i) appropriate parameters can be eliminated due to ineffective crossover operations, and (ii) in sGA an appropriate parameter

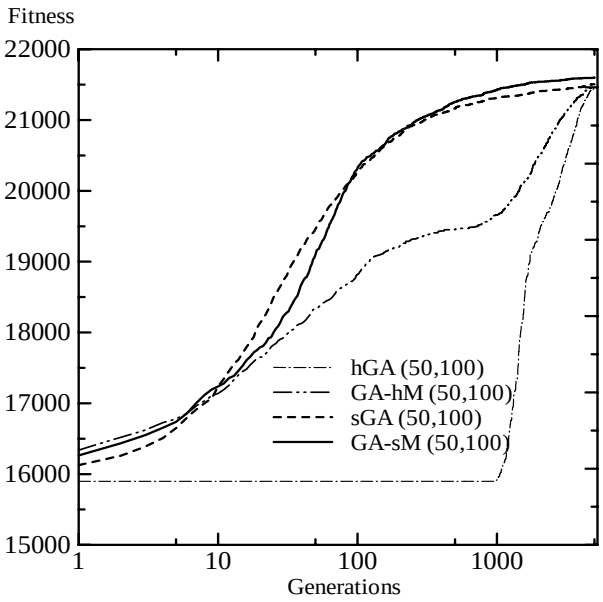


Figure 5.11: Convergence Velocity ($m = 30$, $n = 100$, $\phi = 0.25$). $p_m^{(t=0)} = 0.5$ for hGA and GA-hM. $p_m^{(t=0)}(i) = rand[1/n, 0.5]$ for sGA and GA-SM

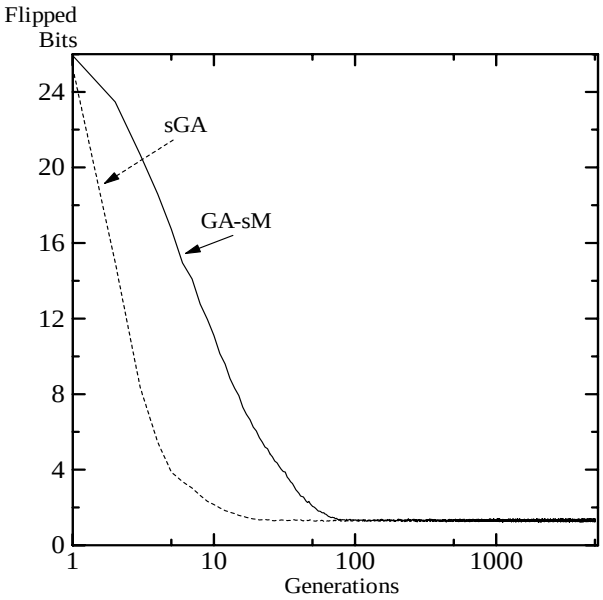


Figure 5.12: Average Number of Flipped Bits ($m = 30$, $n = 100$, $\phi = 0.25$). $p_m^{(t=0)} = 0.5$ for hGA and GA-hM. $p_m^{(t=0)}(i) = rand[1/n, 0.5]$ for sGA and GA-SM

implies parameters that would not affect greatly crossover. Thus, in sGA there is a selective bias towards smaller mutation rates.

Figure 5.13 , Figure 5.14, and Figure 5.15 plot the average percentage error gap by sGA and GA-sM showing the effect on performance of reducing the feasible region (ϕ), increasing the

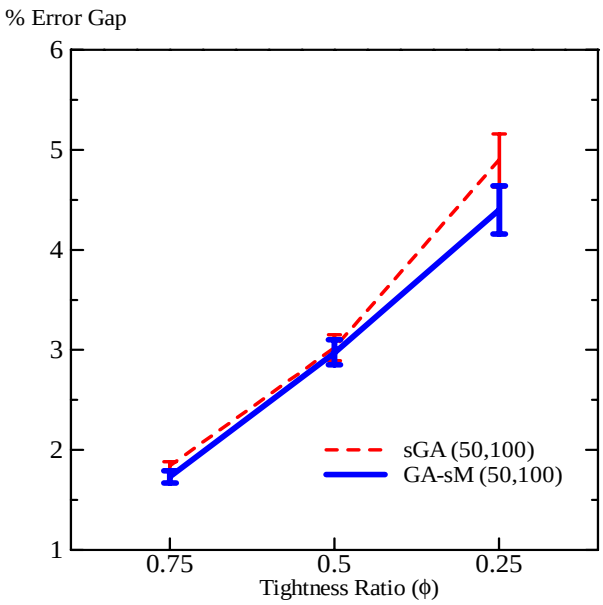


Figure 5.13: Convergence Reliability of Self-Adaptive Varying Mutation GAs: Reducing the feasible region $\phi = \{0.75, 0.50, 0.25\}$, $m = 30$, $n = 100$.

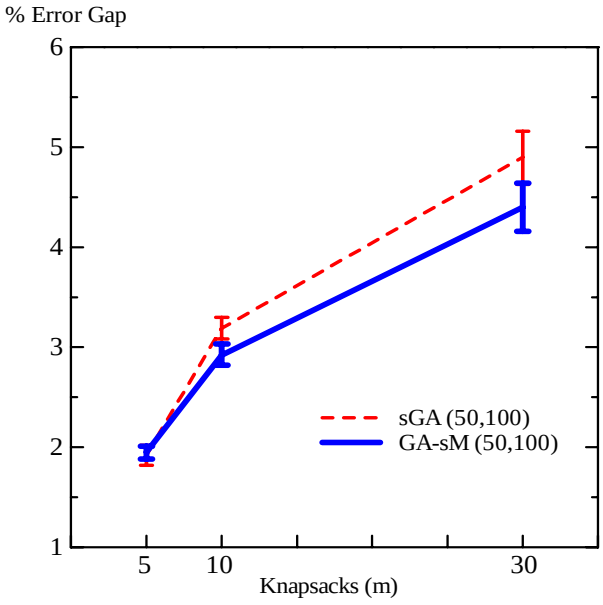


Figure 5.14: Convergence Reliability of Self-Adaptive Varying Mutation GAs: Increasing the number of constraints $m = \{5, 10, 30\}$, $n = 100$, $\phi = 0.25$.

number of constraints (m), and increasing the search space (n). Table 5.5 summarizes the three two-factor factorial ANOVA corresponding to the plots presented in Figure 5.13, Figure 5.14, and Figure 5.15. The treatment means are illustrated in Table 5.6, Table 5.7, and Table 5.8, respectively.

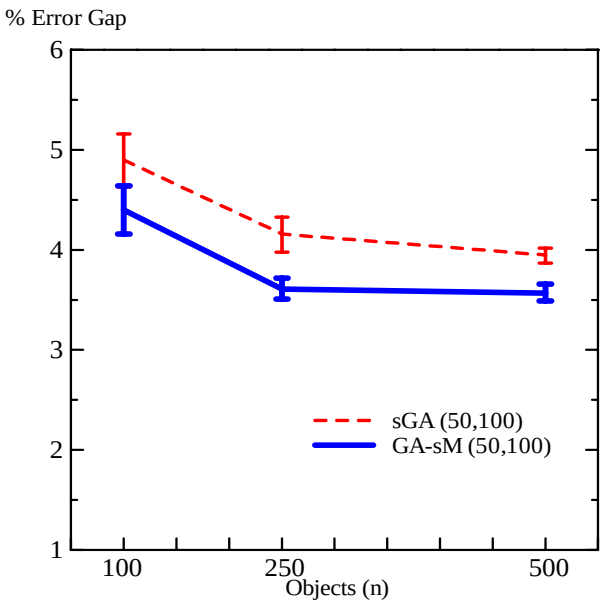


Figure 5.15: Convergence Reliability of Self-Adaptive Varying Mutation GAs: Increasing the search space $n = \{100, 250, 500\}$, $m = 30$, $\phi = 0.25$.

From Table 5.5, inspection of the p values reveals that in all three cases (reducing the feasible region, increasing the number of constraints, and increasing the size of the search space) there are indications of a *strong main effect* by the GA type concluding that the parallel self-adaptive varying mutation GA (GA-sM) attains significantly smaller error than the standard self-adaptive varying mutation GA (sGA). Notice that the p values 0.0029, 0.0009, and 0.0000, respectively, are considerably less than 0.05. Furthermore, note that in all three cases there are indications of a main effect by the problem difficulty factor (ϕ, m, and n). In other words, increasing the difficulty of the problem makes it more difficult for the self-adaptive GA to find good solutions. In addition, note that the interaction between GA type and problem difficulty factor was significant for ratio ϕ and number of constraints m, which means that the self-adaptive parallel varying mutation GA (GA-sM) not only performs better but also scales up better than the standard self-adaptive varying mutation GA (sGA) as the difficulty of the problem increases.

5.6 Conclusions

We have studied the application of varying mutation either serial or parallel to crossover and discussed its effect on the performance of deterministic and self-adaptive varying mutation GAs. Experiments were conducted with several classes of 0/1 multiple knapsacks problems. We found that mutation parallel to crossover is more effective and efficient than mutation serial to crossover.

In deterministic varying mutation GAs, a GA with varying mutation parallel to crossover showed faster convergence and higher robustness to initial settings of mutation rate than a GA with varying mutation serial to crossover. Reducing the feasible region (ϕ) an ANOVA gave some indication of higher convergence reliability by the parallel application of deterministic varying mutation. Increasing the number of constraints (m) and the size of the search space (n) there are indications of a *strong main effect* concluding that the parallel deterministic varying mutation

Table 5.5: Factorial ANOVA for sGA and GA-sM

Source	SS	df	MS	F	*Pval*
GA	0.70634	1	0.70634	9.731	0.0029
ϕ	82.77031	2	41.38516	570.167	0.0000
GA-ϕ	0.59193	2	0.29596	4.078	0.0224
Error	3.91955	54	0.07258		
Total	87.98813	59			
GA	0.89060	1	0.89060	12.418	0.0009
m	74.48359	2	37.24180	519.263	0.0000
GA-m	0.69806	2	0.34903	4.867	0.0114
Error	3.87291	54	0.07172		
Total	79.94516	59			
GA	3.33233	1	3.33233	42.053	0.0000
n	9.25077	2	4.62539	58.371	0.0000
GA-n	0.08025	2	0.04013	0.506	0.6055
Error	4.27902	54	0.07924		
Total	16.94237	59			

Table 5.6: Error Gap Means by sGA and GA-sM reducing the feasible region (ratio ϕ)

GA	Ratio ϕ			
	0.75 (ϕ_1)	0.50 (ϕ_2)	0.25 (ϕ_3)	
	1.73, 1.8	2.97, 2.95	4.75, 4.92	
sGA	1.84, 1.85	3, 3.2	4.96, 5.05	
(g_1)	1.86, 1.82	3.36, 3.22	4.8, 5.73	
	1.77, 1.93	2.76, 3.01	4.49, 5.13	
	1.76, 1.99	3.09, 2.66	5.05, 4.08	
Totals	18.35	30.22	48.96	$T_{g_1} = 97.53$
Averages	1.84	3.02	4.90	$\mu_{g_1} = 3.25$
	1.78, 1.72	2.78, 2.93	4.33, 4.44	
GA-sM	1.61, 1.58	2.93, 3.22	4.48, 4.71	
(g_2)	1.75, 1.78	3.2, 3.24	4.22, 4.82	
	1.63, 1.93	2.73, 2.93	3.95, 4.81	
	1.74, 1.77	3.06, 2.71	4.62, 3.62	
Totals	17.29	29.73	44.00	$T_{g_2} = 91.02$
Averages	1.73	2.97	4.40	$\mu_{g_2} = 3.03$
Totals	$T_{\phi_1} = 35.64$	$T_{\phi_2} = 59.95$	$T_{\phi_3} = 92.96$	
Averages	$\mu_{\phi_1} = 1.78$	$\mu_{\phi_2} = 3.00$	$\mu_{\phi_3} = 4.65$	

Table 5.7: Error Gap Means by sGA and GA-sM increasing the number of constraints (knapsacks m)

GA	Knapsacks m			
	5 (m_1)	10 (m_2)	30 (m_3)	
	1.73, 1.99	3.08, 3.21	4.75, 4.92	
sGA	2.09, 1.66	3.3, 2.93	4.96, 5.05	
(g_1)	1.98, 2.02	3.39, 2.95	4.8, 5.73	
	2.12, 1.74	3.22, 3.13	4.49, 5.13	
	1.86, 1.98	3.18, 3.48	5.05, 4.08	
Totals	19.17	31.87	48.96	$T_{g_1} = 100.00$
Averages	1.92	3.19	4.90	$\mu_{g_1} = 3.33$
	1.79, 1.99	2.76, 2.9	4.33, 4.44	
GA-sM	2.1, 1.78	3.05, 2.67	4.48, 4.71	
(g_2)	2.04, 1.95	2.9, 2.9	4.22, 4.82	
	2.01, 1.89	2.83, 2.96	3.95, 4.81	
	1.97, 1.96	2.94, 3.3	4.62, 3.62	
Totals	19.48	29.21	44.00	$T_{g_2} = 92.69$
Averages	1.95	2.92	4.40	$\mu_{g_2} = 3.09$
Totals	$T_{m_1} = 38.65$	$T_{m_2} = 61.08$	$T_{m_3} = 92.96$	
Averages	$\mu_{m_1} = 1.93$	$\mu_{m_2} = 3.05$	$\mu_{m_3} = 4.65$	

Table 5.8: Error Gap Means by sGA and GA-sM increasing the search space (number of objects n)

GA	Objecs n			
	100 (n_1)	250 (n_2)	500 (n_3)	
	4.75, 4.92	4.01, 4.04	4, 4.11	
sGA	4.96, 5.05	4.09, 3.86	4.02, 3.93	
(g_1)	4.8, 5.73	3.76, 4.12	4.06, 3.7	
	4.49, 5.13	4.33, 4.75	3.94, 3.98	
	5.05, 4.08	4.28, 4.33	3.76, 3.96	
Totals	48.96	41.57	39.46	$T_{g_1} = 129.99$
Averages	4.90	4.16	3.95	$\mu_{g_1}4.33$
	4.33, 4.44	3.56, 3.47	3.59, 3.74	
GA-sM	4.48, 4.71	3.57, 3.54	3.52, 3.57	
(g_2)	4.22, 4.82	3.28, 3.65	3.77, 3.29	
	3.95, 4.81	3.7, 3.88	3.62, 3.57	
	4.62, 3.62	3.67, 3.79	3.52, 3.55	
Totals	44.00	36.11	35.74	$T_{g_2} = 115.85$
Averages	4.40	3.61	3.57	$\mu_{g_2} = 3.86$
Totals	$T_{n_1} = 92.96$	$T_{n_2} = 77.68$	$T_{n_3} = 75.20$	
Averages	$\mu_{n_1} = 4.65$	$\mu_{n_2} = 3.88$	$\mu_{n_3} = 3.76$	

attains significantly smaller error than the standard deterministic varying mutation GA.

In self-adaptive varying mutation GAs, the convergence velocity of a parallel self-adaptive mutation GA was matched by a serial self-adaptive mutation GA only when initial diversity of parameters was allowed. Convergence reliability was higher for the parallel varying self-adaptive mutation GA with or without initial diversity of parameters. An ANOVA gave a strong indication in this direction weather the feasible region (ϕ) is reduced, the number of constraints (m) are increases, or the size of the search space (n) is enlarge. We also found that the standard model of varying mutations in fact affects negatively the (adaptive) self-adaptive mutation rate control. This strongly suggests that the standard model of varying mutation GAs may not be appropriate for combining forms of control.

Among deterministic and self-adaptive varying mutation GAs, best performance was achieved by a parallel varying mutation self-adaptive GA.

Chapter 6

Studying the Behavior of GA-SRM on Epistatic Problems

This chapter examines the behavior of the parallel varying mutation GA-SRM on epistatic problems using NK-Landscapes. We discuss properties of NK-Landscapes and analyze the major components of GA-SRM. Comparisons are made with a canonical GA, a simple GA with extinctive selection, a mutation only evolutionary algorithm, and a random bit climber RBC+. We show that avoiding unwanted selective bias by eliminating fitness duplicates and using an appropriate selection pressure make GAs quite robust in NK-Landscapes. We also show that parallel varying mutation improves further reliability of the GA and that mutation strategy is an important factor to improve the performance of GAs on problems with nearest neighbor patterns of epistasis. Similar to recent works, relatively larger landscapes than previous studies are used in order to be a step closer to problems found in real world applications.

6.1 Introduction

Test function generators[64] for broad classes of problems are seen as the correct approach for testing the performance of genetic algorithms (GAs). Kauffman's NK-Landscapes model of epistatic interactions[57, 58] is a well known example of a class of test function generator and has been the center of several theoretical and empirical studies both for the statistical properties of the generated landscapes and for their *GA-hardness*[60, 64, 65, 66, 67]. Previous works that investigate properties of GAs with NK-Landscapes have mostly limited their study to small landscapes, typically $10 \leq N \leq 48$, and observed the behavior of GAs only for few generations. Recently, studies are being conducted on larger landscapes expending more evaluations[61, 68, 69, 70] and the performance of GAs is being benchmarked against hill climbers[61, 68, 70].

Heckendorn et al.[61] analyzed the epistatic features of *embedded landscapes* showing that for NK-Landscapes all the schema information is random if K is sufficiently large predicting that, "since genetic algorithms theoretically only perform well when the algorithm can effectively exploit relationships between schemata"[61], a standard GA would have no advantage over a strictly local search algorithm. To verify this, the authors empirically compared the performance of a random bit climber (RBC+), a simple GA, and an enhanced GA (CHC)[77] known to be robust in a wide variety of problems. Experiments were conducted for $N = 100$ varying K from 0 to 65. A striking result of this study was the overall better performance of the random bit climber RBC+. The authors encourage test generators for broad classes of problems but suggest that NK-Landscapes (and kSAT) seem not to be appropriate for testing the performance of genetic algorithms. Motivated by [61], Mathias et al.[68] provided an exhaustive experimental examination of the performance of similar algorithms including also Davis' RBC[78]. A main conclusion of this study is that over the range $19 \leq N \leq 100$ there is a niche for the enhanced CHC in the region of $N > 30$ for $K = 1$ and $N > 60$ for $1 \leq K < 12$. Yet, this niche is very narrow compared to the broad region where RBC and RBC+ show superiority.

Adaptive evolution is a search process driven by mutation, recombination, drift, and selection over fitness landscapes[58]. All of these are important components in a GA and are carefully considered within GA-SRM. (i) GA-SRM applies varying mutation (SRM) parallel to crossover & background mutation (CM) using extinctive selection to put higher mutations and crossover in a cooperative-competitive stand with each other. (ii) It relies in adaptation and mutation strategy to increase effectiveness of the parallel varying mutation operator. (iii) It eliminates fitness duplicates enhancing selection by removing an unwanted source of selective bias, postponing drift, and providing a more fair competition between CM and SRM.

This chapter examines the behavior of the parallel varying mutation GA-SRM on epistatic problems using NK-Landscapes[70]. Properties of NK-Landscapes are discussed and the effect on performance of the four major processes mentioned above (mutation, recombination, drift, and selection) is verified. Mutation strategy for the varying mutation operator is also studied in detail. Experiments are conducted with NK-Landscapes for $N = 48$ and $N = 96$ varying K from 0 to $N - 1$ in increments of 4. Comparisons are made with a canonical GA, a simple GA with extinctive selection, a mutation only EA, and the random bit climber RBC+.

It is shown that avoiding unwanted selective bias by eliminating fitness duplicates and using an appropriate selection pressure make GAs quite robust in NK-Landscapes. With regards to strictly local search algorithms, different to previous works, it is shown that even simple GAs with these two features perform better than RBC+ for a broad range of classes of problems ($K \geq 4$). It is also shown that the interaction of parallel varying mutation with crossover improves further the reliability of the GA for $12 < K < 32$. Contrary to intuition, it is found that for small K a mutation only EA is very effective and crossover may be omitted; but the relative importance

of crossover interacting with varying mutation increases with K performing better than mutation alone for $K > 12$. Finally, Mutation strategy for parallel varying mutation turns out to be an important factor to improve the performance of GAs on problems with nearest neighbor patterns of epistasis.

6.2 RBC+

RBC+[61] is a variant of a random bit climber (RBC) defined by Davis[78]. Both are local search algorithms that use a single bit neighborhood. We implement RBC+ following indications given in [61, 68, 78].

RBC+ begins with a random string of length N. A random permutation of the string positions is created and bits are complemented (i.e. flipped) one at the time following the order indicated by the random permutation. Each time a bit is complemented the string is re-evaluated. All changes that results in equally good or better solutions are kept and accounted as an accepted change. After testing all N positions indicated by the random permutation, if accepted changes were detected a new permutation of string positions is generated and testing continues. If no accepted changes were detected a local optimum has been found, in which case RBC+ opts for a "soft-restart". That is, a random bit is complemented, the change is accepted regardless of the resulting fitness, a new permutation of string positions is generated, and testing continues. These soft-restarts are allowed until $5 \times N$ changes are accepted (including the bit changes that constituted the soft restarts). When RBC+ has exhausted the possibility of a soft-restart it opts for a "hard-restart" generating a new random bit string. This process continues until a given total number of evaluations have been expended. The difference between RBC and RBC+ is the inclusion of soft-restarts in the latter.

6.3 Experimental Setup

The study is conducted on NK Landscapes with $N = 48$ and $N = 96$ bits varying the number of epistatic interactions from $K = 0$ to $K = N-1$ in increments of 4. Two kinds of epistatic patterns are investigated for the distribution of the K bits among the N: (i) *nearest neighbor*, in which each bit interacts with its $K/2$ left and right adjacent bits, and (ii) *random*, in which each bit interacts with K other randomly chosen bits in the chromosome. Circular genotypes are considered to avoid boundary effects. The landscapes created with *random* espistatic pattern correspond to Kauffman's random neighborhood whereas the landscapes created with *nearest neighbor* epistatic pattern are of two types: Kauffman's adjacent neighborhood and generalized NK-Landscapes with $P = N$. Unless indicated otherwise, results are reported for landscapes with *random* epistatic pattern

In order to observe and compare the effect on performance of selection pressure, parallel varying mutation, and recombination, GA-SRM is decomposed to create the four following algorithms: (i) A simple canonical GA with proportional selection and crossover & background mutation (CM), denoted as cGA; (ii) a simple GA with (μ, λ) proportional selection and CM, denoted as GA(μ, λ); (iii) a GA that uses (μ, λ) proportional selection, CM, and parallel varying mutation SRM, denoted as GA-SRM(μ, λ); and (iv) a version of GA-SRM(μ, λ) with no crossover ($p_c = 0.0$), denoted as M-SRM(μ, λ). To observe the effect of drift the algorithms are used with the fitness duplicates elimination feature either on or off. The superscript ^{ed} attached to the name of a GA indicates that the elimination of duplicates feature is on, i.e cGA^{ed}, GA$^{ed}(\mu, \lambda)$, GA-SRM$^{ed}(\mu, \lambda)$, and M-SRM$^{ed}(\mu, \lambda)$. GA-SRM's adaptive mutation strategies, ADS and ADP, are used to study the relevance of mutation strategy to epistatic pattern.

Table 6.1: GAs Parameters

Parameter	*GA*			
			GA-SRM(μ, λ)	
	cGA	*GA*(μ, λ)	*ADS*	*ADP*
Selection	*Proportional*	(μ, λ) *Prop.*	(μ, λ) *Prop.*	(μ, λ) *Prop.*
p_c	0.6	0.6	0.6	0.6
$p_m^{(CM)}$	$1/N$	$1/N$	$1/N$	$1/N$
$p_m^{(SRM)}$	–	–	$\alpha = 0.5$	$\alpha = [0.5, 1/N]$
			$\ell = [N, 1/\alpha]$	$\ell = N$
$\lambda_{CM} : \lambda_{SRM}$	–	–	$1:1$	$1:1$

Similar to [64], for CM two-point crossover is used with probability p_c as the recombination operator and the standard bit-flipping method with probability $p_m^{(CM)}$ as the mutation operator after crossover. In the case of GA-SRM, SRM is used with ADS or ADP with mutation rate $p_m^{(SRM)}$. Also, since adaptation is used for SRM the threshold τ that triggers adaptation is sampled for all combinations of mutation strategy, N and K. Results presented here for ADS and ADP are for its best sampled τ. The number of evaluations is set to 2×10^6 for all GAs and RBC+. The offspring population λ is set to 200. Several parent populations (μ) are tried for the GAs that use extinctive selection. All GAs use linear scaling. The parameters used for the GAs are summarized in Table 6.1. The parameters for M-SRM(μ, λ) are the same as GA-SRM(μ, λ) with ADP, except that $p_c = 0.0$ (no recombination).

Results presented here are the mean value of the best solution observed over 50 different randomly generated problems starting each time with the same initial population. The vertical bars overlaying the mean curves in the plots represent 95% confidence intervals. Note that the problems are maximized.

6.4 Selection Pressure

First, the effect of a higher selection pressure on the performance of a simple GA is observed. Figure 6.1 plots results by cGA(200) and GA with (μ, λ)=$\{$(100,200), (60,200), (30,200)$\}$. Results by the random bit climber RBC+ are also included for comparison.

$K = 0$ and $K = N - 1$

In the limits, $K = 0$ corresponds to an additive genetic model in which there are no epistatic interactions and yields a single peaked smooth fitness landscape. On the opposite extreme, $K = N - 1$ corresponds to a maximally rugged fully random fitness landscape in which each gene is epistatically affected by all the remaining genes. In the extremes the performance of the algorithms is expected to be similar. This can be seen for $K = 0$ in Figure 6.1.

$0 < K < N - 1$

In [57, 58] sampling the landscapes by one-mutant adaptive walks, it was observed that low levels of epistatic interactions seem to bend the landscape and yield higher optima than the $K = 0$ landscape. Increasing K, however, would both cause a *complexity catastrophe* (the attainable

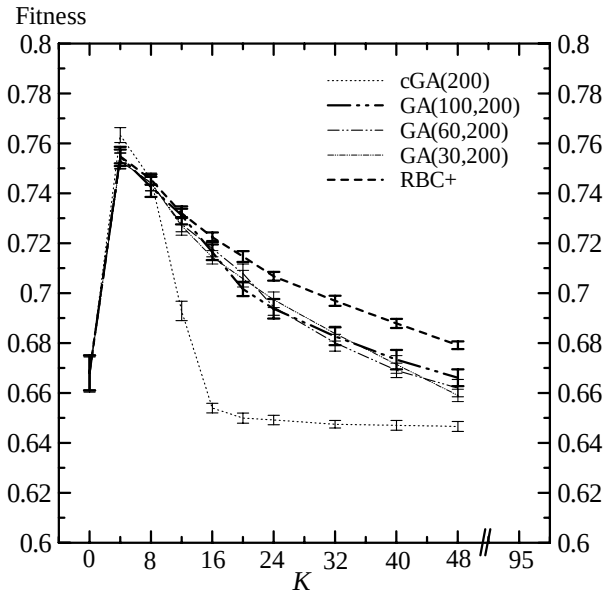


Figure 6.1: Higher selection pressure. Landscapes with random patterns of epistasis.

optima falls toward the mean fitness of the landscape) and make fitness landscapes more multi-peaked. Recent works, using exhaustive search for $N \leq 25$, confirm that there is a positive correlation between K and the number of peaks[62] and that the global optima of the landscape increase for small K[68]. Note from Figure 6.1 that the highest optima is for small K. Contrary to [57, 58], however, [62] and [68] show that the global optima is not a function of K for a given N as K increases from small to large values. Thus, the decreasing best values found by the algorithms when K increases indicates that the search performance is in fact worsening, and not that the values of the optima in the NK-Landscapes are decreasing.

For $0 < K < N - 1$, different from $K = 0$ and $K = N - 1$, differences in performance by the GAs can be seen. Some important observations are as follows.

1. cGA does relatively well for very low epistasis ($K \leq 8$) but its performance falls sharply for medium and high epistasis ($K \geq 12$). Similar behavior by another simple GA has been observed in [61] and [68].

2. The behavior of GA(μ,λ) is revealing. GA(μ, λ) that includes an stronger selection pressure performs worse than cGA for low epistasis but it outperforms cGA(200) for medium and high epistasis ($K \geq 12$). The behavior of GA(100,200), GA(60,200), and GA(30,200) are similar. Results by cGA and GA(μ, λ) indicate the importance of an appropriate selection pressure to pull the population to fittest regions of the search space (note that the genetic operators are the same in both algorithms). The selection pressure induced by proportional selection seems to be appropriate only for very small K. As K increases a stronger selection pressure works better. It is worth noting that a similar behavior by extinctive selection has been observed in large, difficult, and highly constrained combinatorial optimization problems[79].

3. The overall performance of RBC+ is better than both cGA and GA(μ, λ).

6.5 Parallel Varying Mutation, Mutation Strategy, and Patterns of Epistasis

Second, the effect of the mutation strategy used in parallel varying mutation is investigated on landscapes with different patterns of epistasis. Figure 6.2 and Figure 6.3 present the mean fitness of the best solutions achieved by GA-SRM(100,200) with ADS, and GA-SRM(100,200) with ADP on landscapes with *nearest neighbor* and *random* epistatic patterns for $N = 48$ and $K = 0, \cdots, N-1$. Similarly, Figure 6.4 and Figure 6.5 present results for $N = 96$. Results by cGA and GA(100,200) are also included for comparison.

Kauffman[57, 58] has observed that (i) the actual fitness of optima are largely insensitive to the distribution of K among N. That is, for the same values of N and K, the expected attainable optima is similar for nearest neighbor and random epistatic patterns. Also, he showed that (ii) for small values of K and two alleles either for random or nearest neighbor epistatic patterns there is a global structure to the fitness landscape. That is, the global optima are not distributed randomly in genotype space but instead are near one another. As K increases this correlation drops, more rapidly for random than for nearest neighbor epistatic pattern. The correlation of K to the dispersion of sub optima, its relative fitness and Hamming distance from the global optimum, is corroborated in [62].

Looking at Figure 6.2 and Figure 6.3 it can be seen that for same values of K ($N = 48$) higher optima are achieved by the GAs when nearest neighbor epistatic interactions exist ($4 < K < N-1$). Similar behavior can be observed for $N = 96$ in Figure 6.4 and Figure 6.5. If Kauffman's observation that the achievable optima is similar for random and nearest neighbor epistatic patterns holds, this indicates that the GAs perform better on the presence of nearest neighbor epistatic patterns. Otherwise, this suggests that a nearest neighbor epistatic pattern does not only impose a stronger structure to the landscape as showed by Kauffman but also that it is capable of bending the landscape more than a random epistatic pattern does, which yields higher achievable optima. Either case, the epistatic pattern should be taken into account during the design of representations for problems to be solved by GAs. Representations that induce nearest neighbor epistatic interactions should be preferred over those that produce random interactions.

From Figure 6.2 and Figure 6.4 looking at the results by ADS and ADP it is observed that in the case of a nearest neighbor epistatic pattern the strategy that mutates within a continue mutation segment, i.e. ADS, performs significantly better than the strategy that mutates any bit of the chromosome, i.e. ADP. Bigger differences are observed for $N = 96$ than $N = 48$, which suggest that mutation strategy could be a more significant factor as the search space increases. ADP performs similar to ADS only for low epistatic levels, $K \leq 8$ and $K \leq 4$ for $N = 48$ and $N = 96$, respectively.

In the case of a random epistatic pattern, from figure Figure 6.3 it can be seen that for $N = 48$ there is no difference in performance by ADS and ADP for any value of K. From Figure 6.5, however, it can be observed that when $N = 96$ ADS performs better than ADP as K increases from medium to high levels of epistatis ($K \geq 20$).

ADS's better performance is explained from Kaufmman's findings that epistatic interactions, even a random epistatic pattern, inflict a global structure to the fitness landscape in which high peaks are close in genotype space. In such case a segment mutation strategy as ADS can take advantage of this underlying structure to perform a more effective search. The global structure of the landscape is stronger for a nearest neighbor epistatic pattern and the difference in performance between ADS and ADP can be clearly noticed as indicated above. In the case of a random epistatic pattern this structure is weaker and its effect would remain hidden in small search spaces as $N = 48$, but would become more evident for medium and high K as N increases.

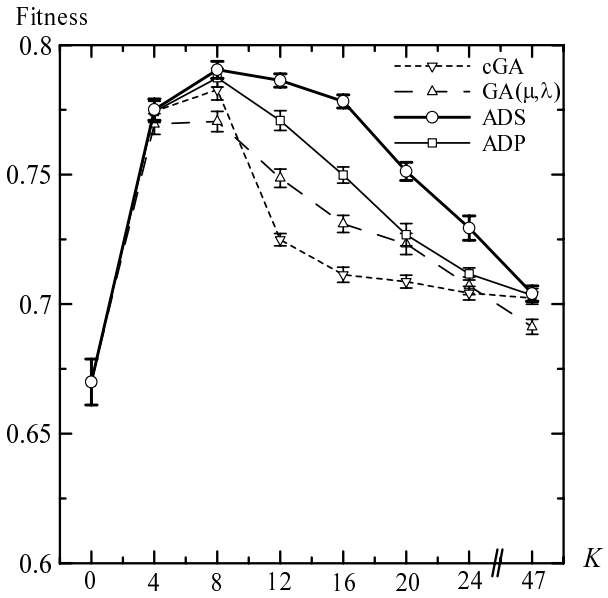


Figure 6.2: Nearest neighbor patterns of epistasis: Mean fitness after 10^4 generations for $N = 48$, $K = 0, \cdots, N-1$

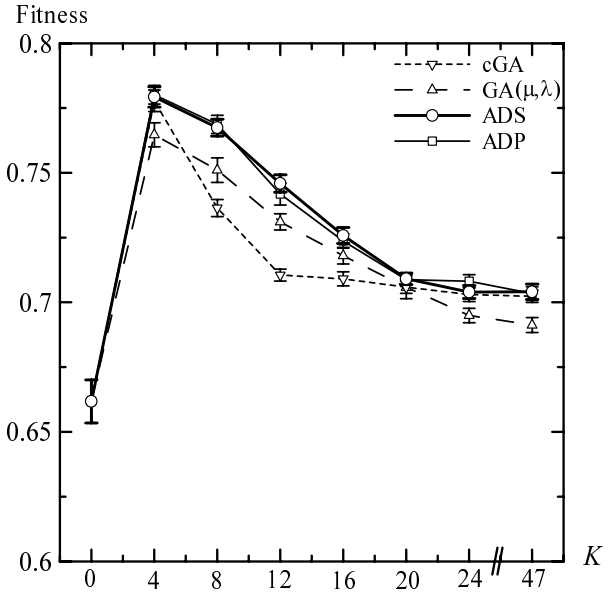


Figure 6.3: Random patterns of epistasis: Mean fitness after 10^4 generations for $N = 48$, $K = 0, \cdots, N-1$

Figure 6.6 and Figure 6.7 illustrate typical fitness transitions of the best so far individual on landscapes with nearest neighbor and random epistatic patterns for $N = 48$ and medium epistasis $K = 12$. Similarly, Figure 6.8 and Figure 6.9 present results for $N = 96$ and $K = 24$. Note that the plots for $N = 96$ are for 10^5 rather than 10^4 generations so the behavior of the algorithms can be observed in the long run.

From these figures it can be seen that GA(μ, λ) and GA-SRM(μ, λ) that include extinctive

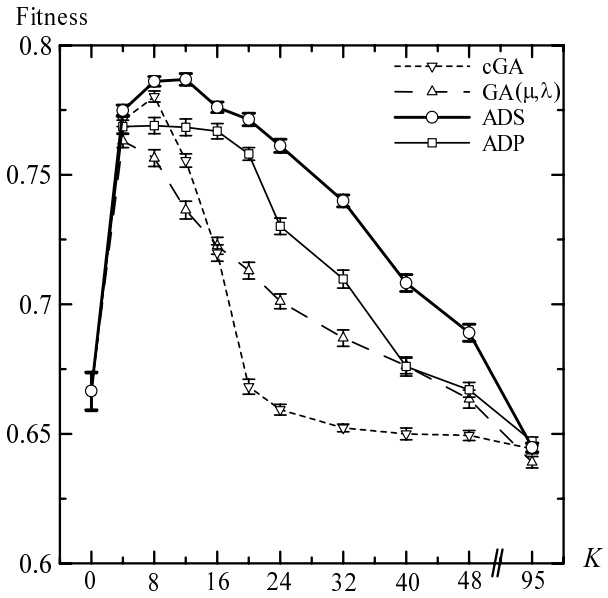


Figure 6.4: Nearest neighbor patterns of epistasis: Mean fitness after 10^4 generations for $N = 96$, $K = 0, \cdots, N-1$

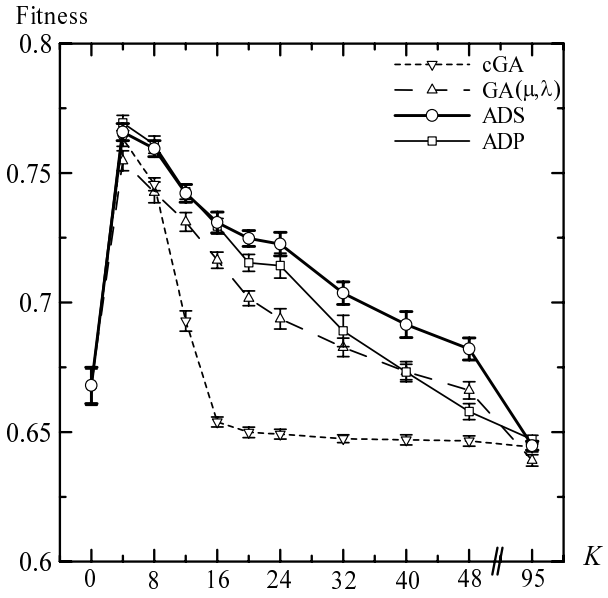


Figure 6.5: Random patterns of epistasis: Mean fitness after 10^4 generations for $N = 96$, $K = 0, \cdots, N-1$

selection converge faster than cGA. As mentioned above, because of the higher selection pressure of extinctive selection, GA(μ, λ) also converges to higher optima than cGA (for medium and higher epistasis). However, it stagnates because of lack of diversity. GA-SRM(μ, λ), either with ADS or ADP, can improve further the fitness achieved by GA(μ, λ) because of its better balance between higher selection pressure and diversity introduced by SRM.

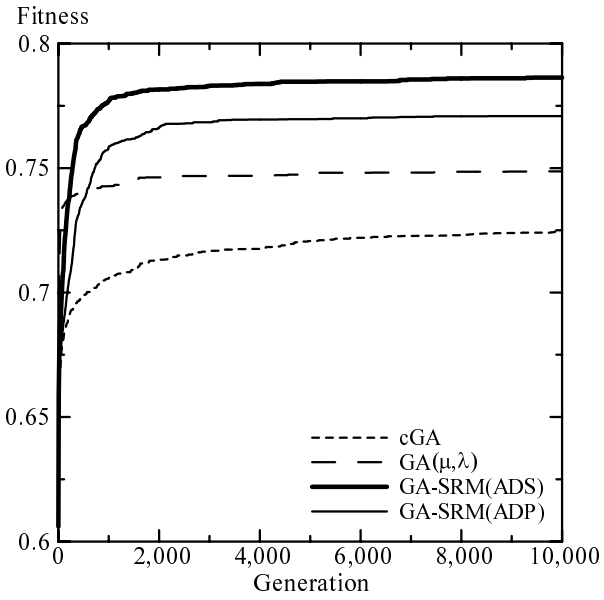


Figure 6.6: Nearest neighbor patterns of epistasis: Fitness transition of best so far, $N = 48$, $K = 12$, 10^4 generations

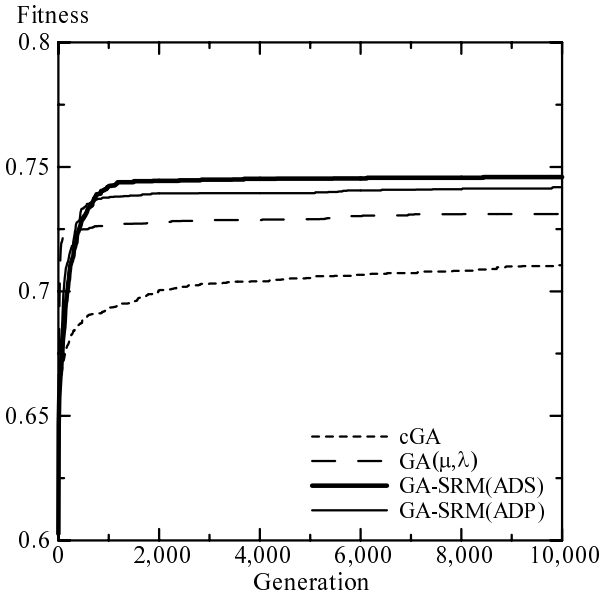


Figure 6.7: Random patterns of epistasis: Fitness transition of best so far, $N = 48$, $K = 12$, 10^4 generations

6.6 Duplicates Elimination

Third, the effect of genetic drift is observed by setting on the fitness duplicates elimination feature. Figure 6.10 plots results by cGA^{ed}(200) and GA$^{ed}(\mu, \lambda)$ with (μ, λ)= {(100,200), (60,200), (30,200)}. Results by cGA(200) and RBC+ are also included for comparison. From this figure it can seen that eliminating duplicates affects differently the performance of the GAs. (i) It

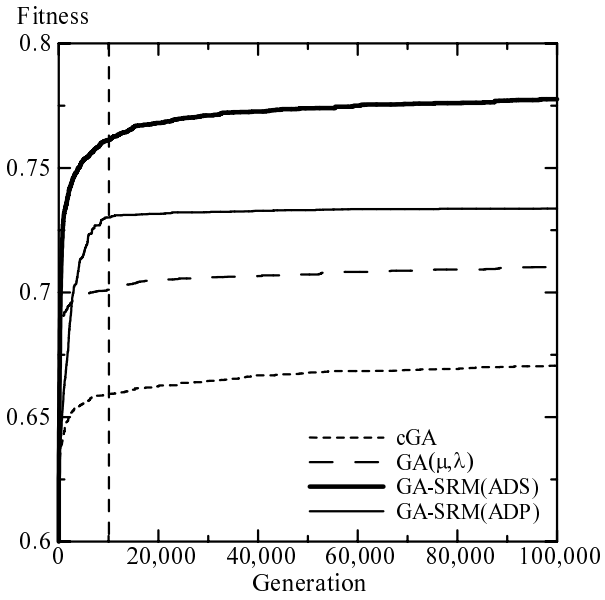


Figure 6.8: Nearest neighbor patterns of epistasis: Fitness transition of best so far, $N = 96$, $K = 24$, 10^5 generations

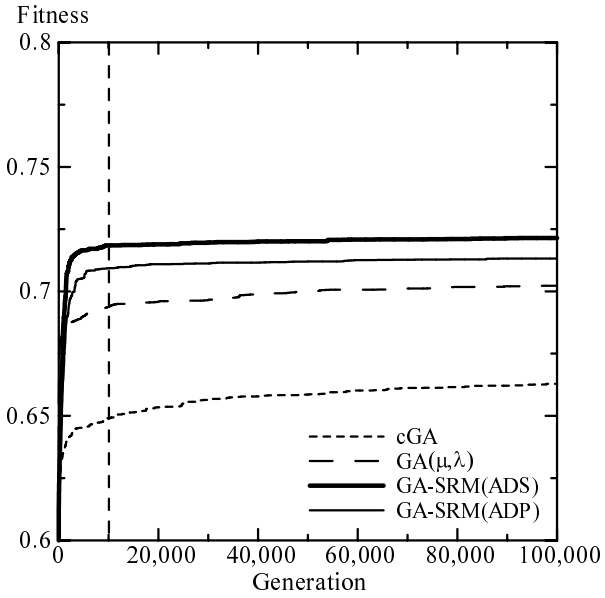


Figure 6.9: Random patterns of epistasis: Fitness transition of best so far, $N = 96$, $K = 24$, 10^5 generations

deteriorates even more the performance of cGA. (ii) Conversely, the combination of higher selection pressure with duplicates elimination produces a striking increase on performance. Note that all GA$^{ed}(\mu, \lambda)$ algorithms achieved higher optima than RBC+ for $4 \leq K \leq 40$. The optima achieved by GA^{ed}(100,200) is lower than RBC+ for $K = 48$. However, note that GA^{ed}(60,200) and GA^{ed}(30,200) achieved higher optima than RBC+. This suggest that for $K = 48$ even the pressure imposed by (μ, λ)=(100,200) is not strong enough.

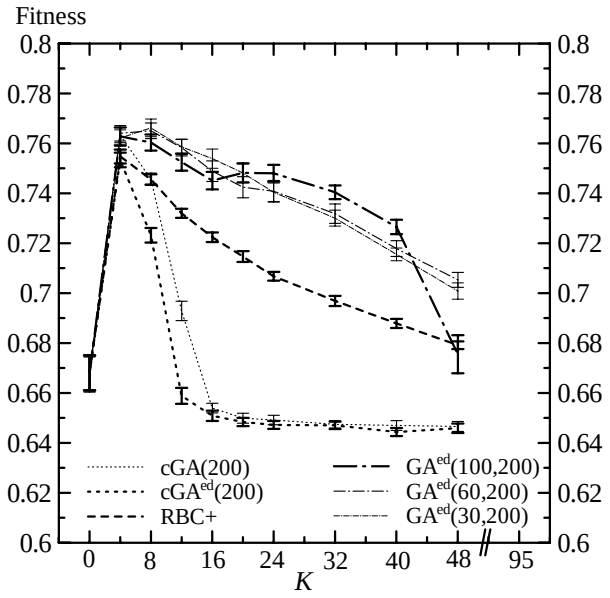


Figure 6.10: Duplicates elimination: Higher selection pressure.

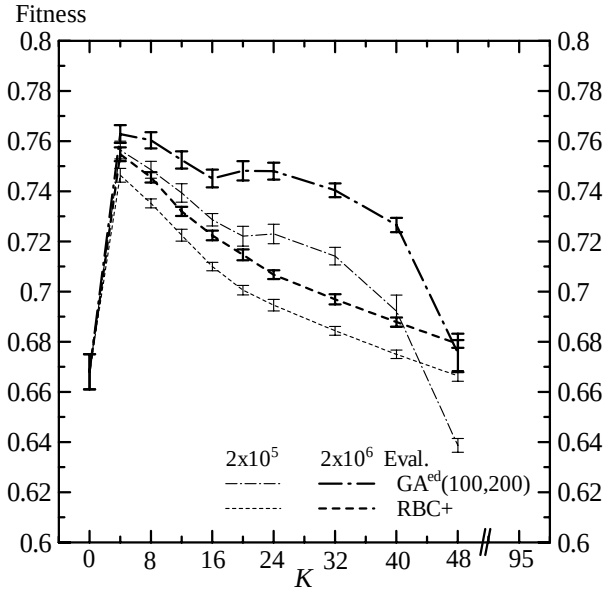


Figure 6.11: Duplicates elimination: Effect of number of evaluations (mean fitness after 2×10^5 and 2×10^6 evaluations).

In this study the algorithms are run for 2×10^6 evaluations whereas only 2×10^5 evaluations have been used in previous studies[61, 68]. Figure 6.11 illustrates the optima achieved by GA^{ed}(100,200) and RBC+ after 2×10^5 and 2×10^6 evaluations. From this figure it can be seen that allocating more evaluations allows both algorithms to find higher optima, being the rate of improvement by the GA greater than the random bit climber RBC+. Note that for most values of K, even after 2×10^6 evaluations, RBC+ still does not reach the optima achieved by GA^{ed}(100,200) after 2×10^5 evaluations.

As shown in 6.4, as K increases a higher selection pressure improves the performance of the simple GA. However, it would also increase the likelihood of duplicates. Preventing duplicates distributes more fairly selective pressure in the population, removes an unwanted source of selective bias in the algorithm[41] and postpones genetic drift[80, 81]. GA$^{ed}(\mu, \lambda)$ takes advantage of the higher selection pressure avoiding the unwanted selective bias. The drop in performance by cGA^{ed} compared to cGA suggests that the latter uses the duplicates as a way to increase its selection pressure. In the following the effects of no eliminating duplicates are elaborated further.

Figure 6.12 illustrates with the thicker lines the ranking of the λ offspring by the GA(100,200), which does not eliminates duplicates, for generations 10, 100 and 2000 during one run of the algorithm ($K = 12$). The offspring with highest fitness is giving rank 1. The horizontal segments in those lines indicate the presence of duplicates. This figure also illustrates with the thinner lines what the rank of the $\lambda - d$ not duplicates offspring would be, where d is the number of duplicates. From this figure it can be seen that if duplicates are not eliminated they accumulate rapidly. In this example, the number of duplicates at generation 10 is 7, which increases to 62 at generation 100 and to 80 at generation 2000. Figure 6.13 presents a similar plot for GA^{ed}(100,200), which does eliminate duplicates. In this case the thicker (thinner) lines indicate the ranking of the offspring after (before) duplicates elimination. From this figure it can seen that, compared to GA(100,200), eliminating duplicates at each generation prevent them from increasing their number. Note that at generation 100 there are only 13 duplicates and that at generation 2000 their number remains similar. This effect, that in the case of GA^{ed} (μ,λ) the number of fitness duplicates remains relatively constant throughout the generations for a given K, can be observed with more detail in Figure 6.15 for various values of K. An important conclusion from both Figure 6.12 and Figure 6.13 is that by eliminating duplicates the likelihood that the algorithm will explore a larger number of different candidate solutions increases substantially, augmenting the possibility of finding higher optima.

Another important aspect of duplicates is related to selection. In the case of algorithms that do not eliminate duplicates, such GA(100,200), selection of parents will be based on the ranking of the all λ offspring as shown by the thicker lines in Figure 6.12, which contains ensembles of clones (i.e. individuals with the same genotype besides having the same fitness). From a genotype uniqueness point of view, each ensemble of clones represents one individual. However, the selection mechanism will assign a selection probability to each clone, the same for all clones within an ensemble, as if they were different individuals. As a consequence, the chances of selecting a given genotype are multiplied by the number of clones of that genotype present in the offspring population. To illustrate this better, Figure 6.14 plots the fitness of unique genotypes when duplicates are not eliminated. Here, an ensemble of clones is treated as one individual and its fitness is the accumulated fitness of the clones in the ensemble. In the figure, ensembles of clones can be clearly recognized by the peaks in the curves. This figure clearly indicates that low fitness genotypes (due to the duplicates effect) can end up with higher selective advantage than high fitness unique genotypes. It should be noted that this unwanted selective bias, which is not based in actual fitness, cannot be avoided by fitness scaling mechanisms, ranking procedures, or even extinctive (truncated) deterministic selection schemes (such (μ, λ) Selection).

In the case of eliminating duplicates, selection of parents will be based on the ranking of the $\lambda - d$ unique genotype offspring as shown by the thicker lines in Figure 6.13. In this case, selection will depend exclusively on the fitness of the individuals.

Figure 6.15 illustrates the number of fitness duplicates[1] over the generations eliminated in

[1]Fitness duplicates are counted and eliminated while the offspring population is being truncated from λ to μ. Thus, the number of duplicates of Figure 6.15 indicate only those counted until the best μ are found. The number of duplicates in λ can be a little higher.

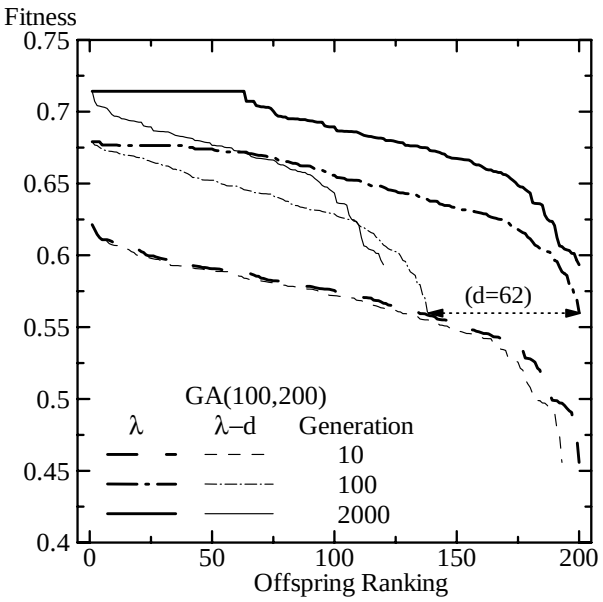


Figure 6.12: Offspring ranking by GA(μ, λ) ($K = 12$).

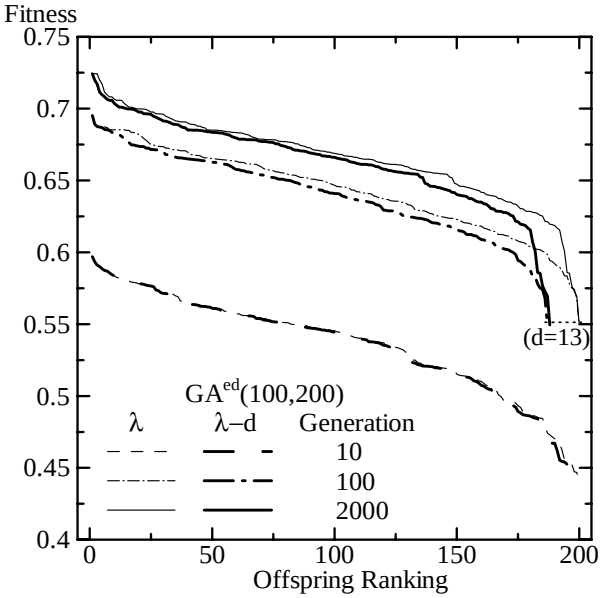


Figure 6.13: Offspring ranking by GA^{ed}(μ, λ) ($K = 12$).

GA^{ed}(100,200) for landscapes with values of $K = \{4, 12, 24, 48\}$ and the number of duplicates eliminated in GA^{ed}(60,200) for $K = 48$. Most of this fitness duplicates were actual clones. For example for $K = 4$, 148993 fitness duplicates were created during the 10000 generations (average in the 50 runs). Out of these, 99.88% corresponded to actual clones. Similar percentages were observed for other values of K. This indicates that the approach is quite effective eliminating clones while being computationally efficient (there is no need to calculate and check hamming distances). Note that (μ,λ) proportional selection is a kind of truncation selection and sorting of the whole population is necessary. Once sorting has been done, the non-duplicates policy requires

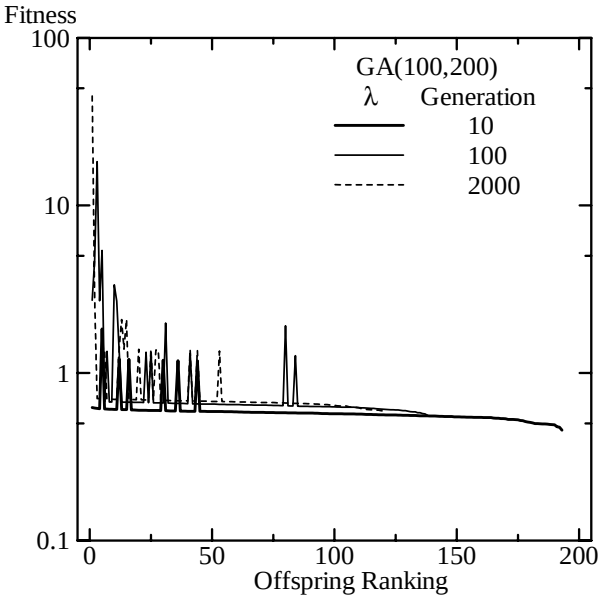


Figure 6.14: Accumulated fitness of the unique genotype offspring, which is reflected in the selection probabilities of GA(μ, λ) ($K = 12$). An ensemble of clones is treated as one individual and its fitness is the accumulated fitness of the clones in the ensemble.

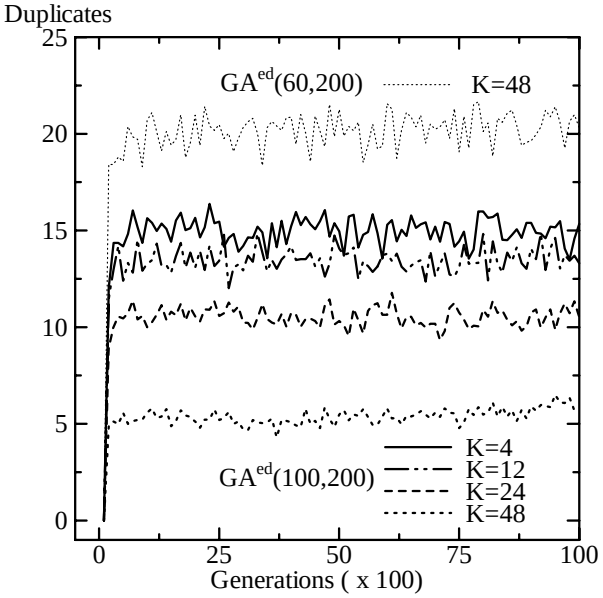


Figure 6.15: Number of eliminated duplicates.

at most $\mathcal{O}(\lambda)$ to eliminate fitness duplicates.

6.7 Eliminating Fitness Duplicates and Parallel Varying Mutation

Four, the effect of the mutation strategy used in parallel varying mutation is investigated on landscapes with different patterns of epistasis when the fitness duplicates feature is on. Figure 6.16

plots results for NK-Landscapes with *random* pattern of epistasis by GA-SRM^{ed}(100,200) with either ADS or ADP. Results by cGA(200), GA^{ed}(100,200) and RBC+ are also included for comparison. From this figure the following observations are relevant. (i) The inclusion of parallel varying mutation can improve further convergence reliability. Note that GA-SRM^{ed}(100,200) with ADS or ADP achieves higher optima than GA^{ed}(100,200) for $4 \leq K < 32$. For $K = 32$ and $K = 40$, however, GA-SRM^{ed}(100,200) is not better than GA^{ed}(100,200), which indicates that varying mutation is not working properly at this values of K. (ii) Mutation strategy seems not to have effect in landscapes with *random* pattern of epistasis when the fitness duplicates feature is on. Note that performance by GA-SRM^{ed}(100,200) with ADS or ADP is very similar for any value of K. It should be remembered from 6.5 that when the duplicates elimination feature is off ADS shows better performance than ADP. This suggests that enhancements in selection could hide the lack of effectiveness of genetic operators, in this case of ADP. It would be interesting to observe whether differences between ADP and ADS appear when N is increased. (iii) The optima achieved by GA^{ed}(100,200) is lower than RBC+ for $K = 48$, which seems to be caused by a lack of appropriate selection pressure as mentioned in 6.6. However, for the same selection pressure, GA-SRM^{ed}(100,200) achieved higher optima than RBC+. This nicely shows that the effectiveness of a given selection pressure is not only correlated to the complexity of the landscape, but also to the effectiveness of the operators searching in that landscape. In the case of GA-SRM^{ed}(100,200), the inclusion of varying mutation increases the effectiveness of the operators, hiding selection deficiencies. It would be better to correct selection pressure and try to use mutation to improve further the search.

Figure 6.17 plots results for NKP-Landscapes with *nearest neighbor* pattern of epistasis. Similar to landscapes with *random* pattern of epistasis, parallel varying mutation improves further convergence reliability. Note that GA-SRM^{ed}(100,200) with ADS achieves higher optima than GA^{ed}(100,200)for $4 \leq K \leq 48$, being more evident for $K < 40$. In this case, contrary to *random* pattern of epistasis, it is observed that for $4 \leq K < 32$ the strategy that mutates within a continue mutation segment, i.e. ADS, performs better than the strategy that mutates any bit of the chromosome, i.e. ADP. As mentioned before in 6.5, ADS's better performance is explained from Kaufmman's[57, 58] findings that epistatic interactions for small K, even a random epistatic pattern, inflict a global structure to the fitness landscape in which high peaks are close in genotype space. The global structure of the landscape falls as K increases and is stronger for *nearest neighbor* than *random* epistatic patterns. In such case a segment mutation strategy as ADS can take advantage of this underlying structure to perform a more effective search.

Looking at Figure 6.16 and Figure 6.17 it can be seen that the behavior of RBC+ is similar in both kinds of landscapes with the exception of $K = 4$. Note also that the behavior of cGA is more robust in landscapes with *nearest neighbor* than *random* pattern of epistasis.

6.8 No Crossover

Finally, the effect of (not) using recombination is observed on NK-Landscapes with random patterns of epistasis. Figure 6.18 plots results by M-SRM^{ed}(100,200), which is a GA-SRM^{ed}(100,200) using ADP with crossover turned off. Results by GA-SRM^{ed}(100,200) with ADP, GA^{ed}(100,200), and RBC+ are also included for comparison. From this figure the following observations are important. (i) For $K \leq 12$ the mutation only algorithm M-SRM^{ed}(100,200) performs similar or better than GA-SRM^{ed}(100,200) that includes crossover. For some instances of other combinatorial optimization problems it has also been shown that a mutation only evolutionary algorithm can produce similar or better results with higher efficiency than a GA that includes crossover, see for

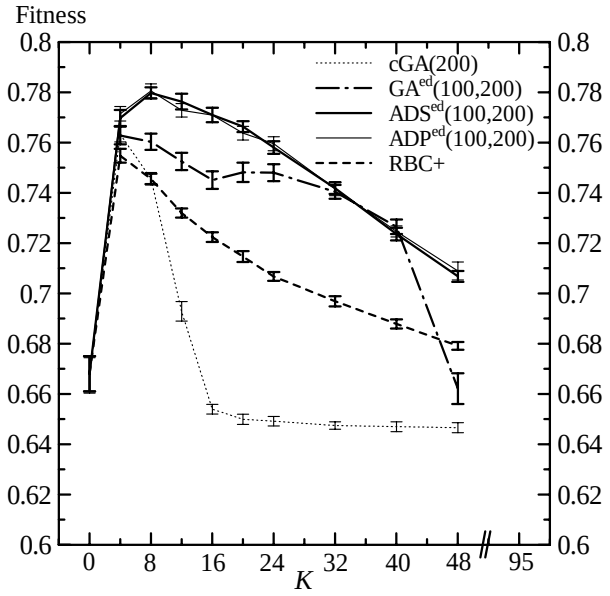


Figure 6.16: Parallel varying mutation and mutation strategy on NK-Landscapes with $N = 96$ and *random* epistatic pattern . Duplicates elimination feature is turned on

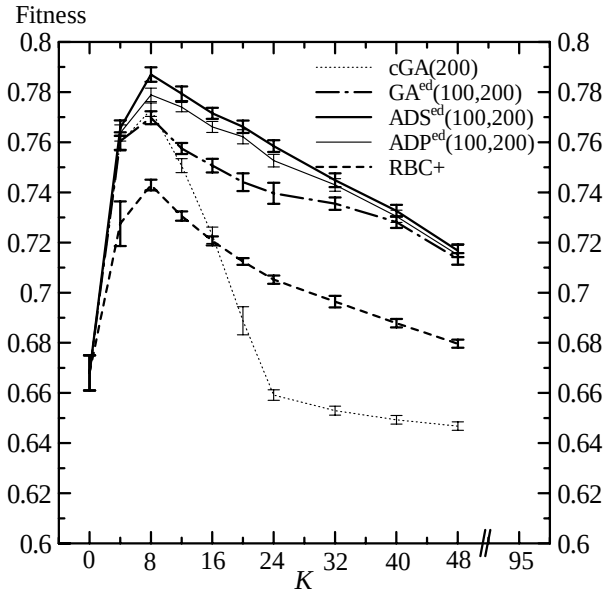


Figure 6.17: Parallel varying mutation and mutation strategy on NKP-Landscapes with $N = P = 96$ and *nearest neighbor* epistatic pattern. Duplicates elimination feature is turned on

example [82]. For $K \geq 16$, GA-SRM that includes both crossover and parallel varying mutation achieves higher optima; note that the difference between GA-SRM and M-SRM increases with K. (ii) Similar to GA-SRM, the mutation only algorithm M-SRM achieves higher optima than RBC+ for $K \geq 4$, which illustrates the potential of evolutionary algorithms, population based, with or without recombination, over strictly local search algorithms in a broad range of classes of problems. This is in accordance with theoretical studies of first hitting time of population-based

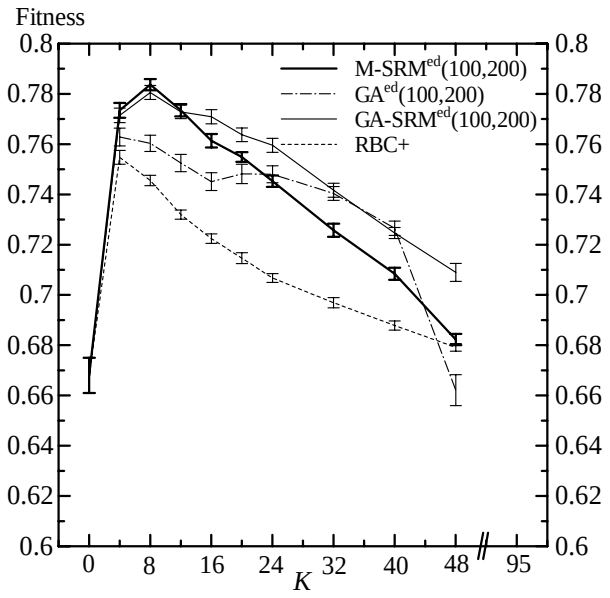


Figure 6.18: No recombination, M-SRM^{ed}(100,200)

evolutionary algorithms. He and Yao[83] have shown that in some cases the average computation time and the first hitting probability of an evolutionary algorithm can be improved by introducing a population.

The behavior of M-SRM compared with GA-SRM is at first glance counterintuitive and deserves further explanation. Results by Heckendorn et al.[61] imply that the usefulness of recombination would decrease with K. However, Figure 6.18 seems to imply exactly the opposite. A sensible explanation for these apparently opposing results comes from the structure of the underlying fitness landscape. As indicated above, Kauffman[57, 58] has shown that clustering of high peaks can arise as a feature of landscapes structure without obvious modularity, as is the case of NK-Landscapes with K epistatic inputs to each site chosen at random among sites. In this case, recombination would be a useful search strategy because the location of high peaks carry information about the location of the other high peaks[57, 58].

In the case of small K the problems are the easier and a mutation only EA proves to be very effective, although the landscape is more structured than for high K. As K increases the structure of the landscape fades away decreasing also the effectiveness of recombination. However, what Figure 6.18 is showing is that the decay of mutation alone seems to be faster than the decay of recombination interacting with varying mutation as the complexity of the landscape increases with K and its structure fades. In other words, the relative importance of recombination interacting with varying mutation increases with K respect to mutation alone.

It should be mentioned that recent developments and discussions on the status of the schema theorem[84] might give new insights to better understand the behavior of GAs than the traditional interpretation of the theorem.

6.9 Conclusions

This chapter has examined the behavior of the parallel varying mutation GA-SRM on epistatic problems using NK-Landscapes. Properties of NK-Landscapes were discussed and the effect on performance of selection, drift, mutation, and recombination was verified. Mutation strategy for the varying mutation operator was also studied in detail. Experiments were conducted with NK-Landscapes with nearest neighbor and random patterns of epistasis for $N = 48$ and $N = 96$ varying K from 0 to $N - 1$ in increments of 4. Comparisons were made with a canonical GA, a simple GA with extinctive selection, a mutation only EA, and a random bit climber RBC+.

It was shown that GAs can be robust algorithms on NK-Landscapes postponing drift by eliminating fitness duplicates and using selection pressure higher than a canonical GA. Different to previous works, even simple GAs with these two features performed better than the single bit climber RBC+ for a broad range of classes of problems ($K \geq 4$). It was also shown that the interaction of parallel varying mutation with crossover (GA-SRM) improves further the reliability of the GA for $12 < K < 32$. Contrary to intuition it was found that a mutation only EA can perform as well as GA-SRM that includes crossover for small values of K, where crossover is supposed to be advantageous; but the relative importance of crossover interacting with varying mutation increased with K performing better than mutation alone for $K > 12$. Better overall performance by population based mutation only evolutionary algorithms over random bit climbers was also observed. With regards to mutation strategy for parallel varying mutation, it was found that a dynamic segment mutation strategy improves the performance of GAs on problems with nearest neighbor patterns of epistasis.

It is concluded that NK-Landscapes are useful for testing the overall behavior and performance of GAs on a broad range of classes of problems and for testing each one of the major processes involved in a GA, which gives valuable insights to improve GAs by understanding better the non-linear, complex, and interesting behavior that arises from the interaction of such processes.

In the future the relationship among selection pressure, the range for varying mutation, and K should be studied deeper. This could give important insights to incorporate heuristics that can increase the adaptability of GAs according to the complexity of the problem.

Chapter 7

Distributed GA with Parallel Varying Mutation

In this chapter we extend GA-SRM that incorporates in its core parallel cooperative-competitive genetic operators to a parallel distributed GA (DGA-SRM). Crossover and mutation, from a processing time stand, are usually simple and their parallelization has been mostly overlooked precisely because any gain we might expect reducing the overall time to completion of the algorithm would seem minor. We argue that rather than as a hardware accelerator, the more significant gains from the parallel application of crossover and higher mutations within parallel GAs could come from exploiting their interaction in a better way. Experiments are conducted using real world 0/1 multiple knapsack problems and various instances of large and difficult 0/1 multiple knapsack problems generated by the test problem generator. Comparisons are made between DGA-SRM and a canonical distributed GA. In our study we observe the effectiveness of extinctive selection and high mutation parallel to crossover in distributed GAs. Furthermore, we examine the influence of the problem difficulty, subpopulation size, and migration rate on the robustness of the distributed GAs.

7.1 Introduction

The development of parallel implementations of algorithms has been mainly motivated by the desire to reduce the overall time to completion of a task by distributing the work implied by a given algorithm to processing elements working in parallel[85]. An alternative approach explores parallel computational models that can exploit interactions among primitive components inducing emergent synergetic behaviors for the entire system[86].

There are a variety of models for parallelizing Genetic Algorithms (GAs) in the evolutionary algorithms literature. They have been separated in four main categories: global master-slave, island, cellular, and hierarchical parallel GAs[87, 88, 89]. In a global master-slave GA there is a single population and the evaluation of fitness is distributed among several processors. Selection, crossover and mutation consider the entire population[90]. A cellular or fine-grained GA consists of one spatially structured population. Selection and mating are restricted to a small neighborhood. The neighborhoods are allowed to overlap permitting some interaction among individuals[91, 92]. An island GA, also known as coarse-grained or distributed GA, consists on several subpopulations evolving separately with occasional migration of individuals between subpopulations [85, 93, 94]. Finally, a hierarchical parallel GA combines an island model with either a master-slave or cellular GA[89].

The global master-slave GA does not affect the behavior of the algorithm and can be considered only as a hardware accelerator. However, the other parallel formulations of GAs are very different from canonical GAs[11, 16], especially with regards to population structure and selection mechanisms. These modifications change the way the GA works affecting its dynamic and the trajectory of evolution. For example, the subpopulation size, migration rate, and migration frequency are crucial to the performance of island models. Cellular, island and hierarchical models perform as well as or better than canonical versions and have the potential of being more than just hardware accelerators[87, 88, 89].

Another aspect of GAs that can be parallelized is the application of crossover and mutation. These operators, from a processing time stand, are usually simple and any gain we might expect reducing the overall time to completion could seem minor. However, the processing time viewpoint alone misses the dynamics that can arise from operators with complementary roles acting in parallel. The balance between crossover and mutation is crucial to the performance of GAs. One way to pursue better balances, and therefore better performance, is to combine crossover with higher mutation rates. Higher mutations parallel to crossover can give an efficient framework towards this goal, in which the strengths of the individual operators can be kept without interfering one with the other. Rather than as a hardware accelerator, the more significant gains from the parallel application of operators within parallel GAs could come from exploiting in a better way the interaction between them.

In this chapter we focus on distributed GAs and study the performance of a distributed GA that incorporates in its core parallel cooperative-competitive genetic operators. We conduct a series of controlled experiments using various large and difficult 0/1 multiple knapsack problems to test the robustness of the distributed GA. Simulation results verify that the proposed distributed GA compared with a canonical distributed GA significantly gains in search speed and convergence reliability with less communication cost for migration.

7.2 Distributed GA (Island Model)

The island model GA consists on several subpopulations evolving separately and concurrently with occasional migration of individuals between subpopulations[85]. Selection, recombination

and mutation are applied within each subpopulation. The basic island model uses the same values for crossover and mutation rates in all subpopulations. However, different values for these parameters can be chosen for each subpopulation[94]. Migration of individuals is controlled by several parameters such as: (i) the communication topology that defines the connections between subpopulations, (ii) a migration rate that controls how many individuals migrate, and (iii) a migration interval that affects the frequency of migration. Also, migration must include strategies for migrant selection and for their inclusion in their new subpopulations.

The communication topology can be defined as a graph in which the subpopulations P_i ($i = 0, 1, ..., K-1$) are the vertices and each defined edge $L_{j,k}$ specifies a communication link between the incident vertices P_j and P_k (neighbor subpopulations). In general, assuming a directed graph, for each defined link $L_{j,k}$ we can indicate the number of individuals $R_{j,k}$ that will migrate from P_j to P_k (migration rate) and the number of generations M between migration events (migration interval). The communication topology and migration rates could be static or dynamic and migration could be asynchronous or synchronous. Various strategies for choosing migrants, such as selection of the best and random selection, have been applied.

The basic island model considers an overall population of λ_{total} individuals that is partitioned into K subpopulations. For an even partition each subpopulation has $\lambda = \lambda_{total}/K$ individuals. It also considers a static topology that is specified at the beginning of the run and synchronous migration occurring every M generations with a constant migration rate R for each defined link $L_{j,k}$.

Distributed GAs are more complex than single population GAs. The subpopulation size, the communication topology (its degree of connectivity), migration rate, and migration frequency are important factors related to the performance of distributed GAs[89]. There is some experimental evidence that distributed GAs can produce solutions with similar or better quality than single population GAs while reducing the overall time to completion in an factor that is almost in reciprocal proportion to the number of processors.

7.3 DGA-SRM

7.3.1 Communication Topology

To create a distributed GA here we use a $+1+...+L$ communication topology[95] in which each subpopulation P_i ($i = 0, 1, ..., K-1$) is linked to the next L subpopulations. The neighbor populations are defined by the directed links $L_{j,k}$ where

$$k = \{j+1, ..., j+L\} \bmod K. \tag{7.1}$$

Figure 7.1 illustrates a $+1+2$ island model in which each subpopulation is linked to two neighbors ($L = 2$). In this example, for instance, subpopulation P_0 can only send individuals to P_1 and P_2 and receive migrants from P_4 and P_5.

7.3.2 CM and SRM in DGA

In the above setting, the extension of GA-SRM to a distributed GA[55, 79, 96] is straightforward. The basic components of the single population GA-SRM are mostly preserved in each subpopulation $P_i(t)$ ($i = 0, 1, ..., K-1$) at the $t-$th generation. CM creates offspring by conventional one-point crossover and successive background mutation operator[11, 16] in each $P_i(t)$. The same crossover rate p_c is used in all $P_i(t)$. The mutation probability $p_m^{(CM)}$ is set to a constant small value and is also the same in all $P_i(t)$. CM creates λ_{CM} offspring within $P_i(t)$ and is expected

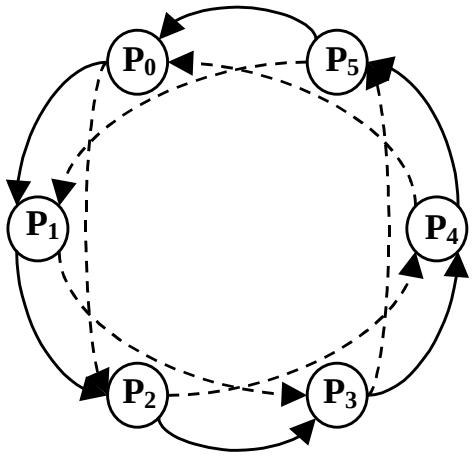


Figure 7.1: $+1+2$ communication topology.

to propagate beneficial genetic information into the subpopulation by combining segments from parent individuals.

On the other hand, SRM creates offspring by an adaptive mutation operators called Adaptive Dynamic Segment (ADS)[34, 35, 55], which directs mutation only to a segment of the chromosome using constant high mutation probabilities per bit in all $P_i(t)$,

$$p_m^{(SRM)} = \begin{cases} \alpha \ (if \ the \ bit \ is \ in \ the \ segment) \\ 0 \ (otherwise) \end{cases}$$

However, the mutation segment size ℓ_i $(i = 0, 1, ..., K-1)$ is independently adjusted in each $P_i(t)$ based on a normalized mutants survival ratio specified by

$$\gamma_i = \frac{\mu_{SRMi}}{\lambda_{SRM}} \cdot \frac{\lambda}{\mu}, \tag{7.2}$$

where μ_{SRMi} is the number of SRM's offspring that survive extinctive selection, λ_{SRM} is the offspring number created by SRM, λ is the total offspring number ($\lambda_{CM} + \lambda_{SRM}$), and μ is the number of parent individuals in $P_i(t)$.

In each $P_i(t)$, ℓ_i varies dynamically from n_0 (initial segment mutation size) to $1/\alpha$ by reducing it to ℓ_i/β ($\beta > 1$) every time γ_i falls under a predetermined threshold τ ($\gamma_i < \tau$). Hence, the expected average number of flipped bits goes down from $n_0\alpha$ to 1. Also, the segment initial position, for each chromosome, is chosen at random. SRM is expected to introduce diversity into each subpopulation and its adaptation mechanism to provide better balances for mutation and crossover throughout the course of a run.

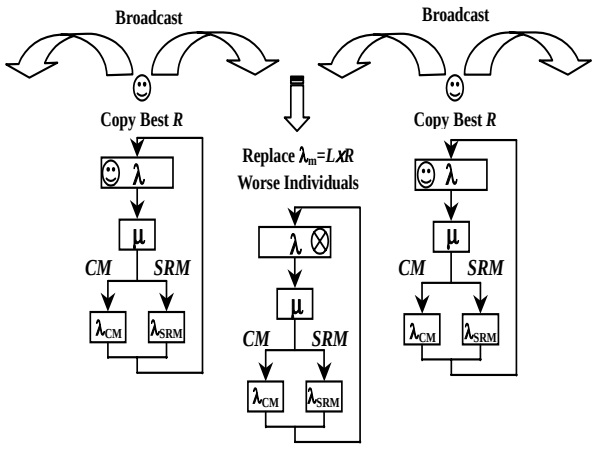


Figure 7.2: Migration policy and extinctive selection.

7.3.3 (μ, λ) Proportional Selection in DGA

In order to implement the extinctive selection mechanism, we use (μ, λ) Proportional Selection[24]. Selection probabilities within each subpopulation $P_i(t)$ are computed by

$$Prob(\boldsymbol{x}_j^{(t)}) = \begin{cases} \dfrac{f(\boldsymbol{x}_j^{(t)})}{\sum_{k=1}^{\mu} f(\boldsymbol{x}_k^{(t)})} & (1 \leq j \leq \mu) \\ 0 & (\mu < j \leq \lambda) \end{cases} \tag{7.3}$$

where $\boldsymbol{x}_j^{(t)}$ is an individual at generation t which has the j-th highest fitness value $f(\boldsymbol{x}_j^{(t)})$ in $P_i(t)$.

7.3.4 Migration Policy

Migration implements a synchronous elitist broadcast strategy[97] occurring every M generations. Each subpopulation broadcasts a copy of its R best individuals to all its neighbor subpopulations. Hence, every subpopulation in every migration event receives $\lambda_m = L \times R$ migrants. In the target subpopulations, the arriving λ_m migrants replace the same number of worst performing individuals. Replacement occurs before extinctive selection. Thus, λ_m migrants also compete to survive with the best $\lambda - \lambda_m$ offspring produced by SRM and CM inside $P_i(t)$. In the following the migration rate is calculated as $100 \times \lambda_m/\lambda$. Figure 7.2 illustrates the migration process to a given subpopulation from its two neighbors (assuming a $+1+2$ communication topology). As mentioned in 7.3.2, SRM's adaptation occurs locally in each subpopulation $P_i(t)$ but it is not realized at the generations in which migration is performed.

7.3.5 Algorithm of DGA-SRM

The algorithm of DGA-SRM is presented in Figure 7.3.

Procedure DGA-SRM;
Concurrently for each subpopulation P_i **do**
begin
$t := 0$
initialize $(P_i(0))$
evaluate $(P_i(0))$
while (**not** termination condition) **do**
begin
$P_i'(t)$ = CM $(P_i(t))$
$P_i''(t)$ = SRM $(P_i(t))$
evaluate $(P_i'(t) \cup P_i''(t))$
if $(t + 1$ **mod** $M) == 0$ **then**
for each neighbor j of i **do**
migration $(P_i'(t) \cup P_i''(t), P_j'(t) \cup P_j''(t))$
$P_i(t + 1)$ = (μ, λ) proportional selection
$(P_i'(t) \cup P_i''(t))$
if $(\gamma < \tau)$ **then**
adapt SRM's mutation rate
$t := t + 1$
end
end

Figure 7.3: Algorithm of DGA-SRM

7.4 Experimental Setup

We test two kinds of distributed GAs in our simulations. (i) A distributed canonical GA (denoted as DGA), and (ii) the proposed distributed GA-SRM (denoted as DGA-SRM). Table 7.1 details the parameters used within each subpopulation by DGA and DGA-SRM. DGA implements the same $+1+...+L$ communication topology and migration policy used by DGA-SRM described in 7.3.1 and 7.3.4.

The problems we conduct experiments with are the 0/1 multiple knapsack problems described in 3.1. The large, difficult, and highly constrained[1] [53] problems generated by the problem test generator are used to study performance and scalability of the algorithms in a broad range of classes of problems. Instances of real-world problems with known global optimum, which from previous efforts seem to be fairly difficult for GAs[52],[31], are also used to show the ability of the algorithm for global optimization.

7.5 Results and Discussion for Large Random 0/1 Multiple Knapsack Problems

As a point of reference for the quality of solutions, Table 7.2 shows results for some of the test problems obtained by the single population versions of the distributed GAs, denoted as cGA and GA-SRM, respectively, set with a population size of $\lambda = 100$ individuals, $T = 10^6$ function evaluations, and SRM's adaptation threshold $\tau = 0.64$. In Table 7.2 column *Problem* identifies the

[1] http://mscmga.ms.ic.ac.uk/jeb/orlib/info.html

Table 7.1: Genetic algorithms parameters

Parameter	*DGA*	*DGA-SRM*
Selection	Proport.	(μ, λ) Proport.
Scaling	Linear	Linear
Mating	$(\boldsymbol{x}_i, \boldsymbol{x}_j),\ i \neq j$	$(\boldsymbol{x}_i, \boldsymbol{x}_j),\ i \neq j$
p_c	0.6	1.0
$p_m^{(CM)}$	$1/n$	$1/n$
$p_m^{(SRM)}$	-	$\alpha = 0.5,\ \ell = [n, 2]$
β	-	2
$\mu : \lambda$	-	$1 : 2$
$\lambda_{CM} : \lambda_{SRM}$	-	$1 : 1$

Table 7.2: Results by single population cGA and GA-SRM

Problem				*cGA*			*GA-SRM*		
$Name$	m	n	ϕ	$Average$	$Stdev$	Gap	$Average$	$Stdev$	Gap
$5.100-00$	5	100	0.25	23271.3	125.75	5.35	24242.8	33.78	1.40
$10.100-00$	10	100	0.25	21846.3	198.94	6.96	22855.4	67.3	2.66
$30.100-00$	30	100	0.25	20494.2	180.49	9.23	21711.8	116.76	3.84
$30.100-10$	30	100	0.50	38509.1	186.49	6.70	40241.9	144.31	2.51
$30.100-20$	30	100	0.75	55348.4	244.34	4.55	57046.1	241.98	1.62
$30.250-00$	30	250	0.25	51920.6	180.11	9.59	55703.9	117.55	3.01
$30.500-00$	30	500	0.25	106023.5	395.15	9.09	113135.3	280.52	2.99

problem instance. *Name* is the name of the problem, m the number of knapsacks, n the number of objects, and ϕ the tightness ratio between knapsack capacities and object weights (restrictiveness of the capacities). *Average* is the average of the best solutions in 10 runs, *Stdev* is the standard deviation around *Average*, and *Gap* indicates the percentage gap between *Average* and the optimal value given by the linear programming relaxation[53] (the optimal integer 0/1 solutions for the test problems are not known).

In our study we observe the influence of the problem difficulty, the subpopulation size, and the migration rate on the robustness of the distributed GAs. The distributed GAs use a $\lambda_{total} = 800$ individuals and the same $T = 10^6$ function evaluations. Also, unless indicated otherwise, the distributed GAs use a configuration of $K = 16$ subpopulations ($\lambda = 50$), SRM's adaptation threshold $\tau = 0.56$, and a 10% migration rate.

7.5.1 Problem Difficulty

Factors related to the difficulty of the problem are the tightness ratio ϕ, the number of objects n, and the number of knapsacks m.

First we observe the performance of the distributed GAs on problems with different ratio ϕ. Figure 7.4 illustrates results by DGA and DGA-SRM on problems of $m = 30$ capacities, $n = 100$ objects, and ratio $\phi = \{0.25, 0.50, 0.75\}$. To present a broader picture this and the subsequent figures plot the error Gap for migration intervals of $M = \{2, 5, 10, 20, 40, 100\}$ generations as well as results when no migration is used and the subpopulations evolve in total isolation (indicated

by NM). Results obtained by the single population GAs are also indicated on the left Y axis.

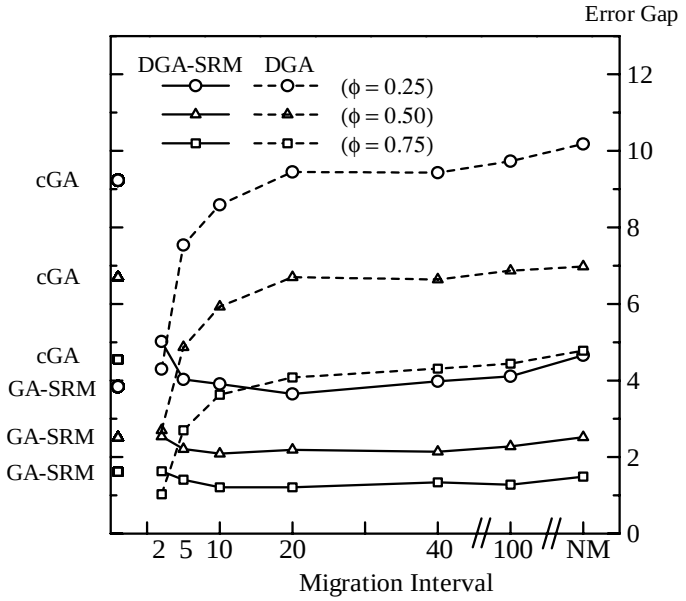


Figure 7.4: Tightness of the Capacities ϕ ($K = 16$, 10% migration, $m = 30$, $n = 100$).

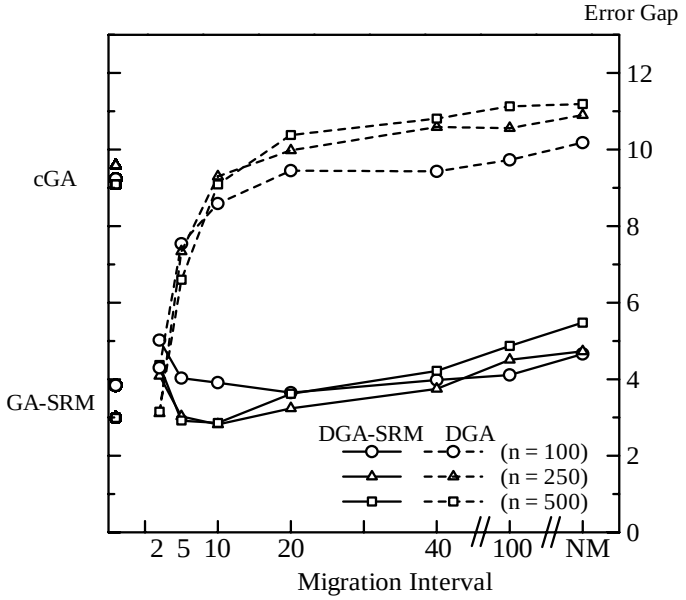


Figure 7.5: Objects n ($K = 16$, 10% migration, $m = 30$, $\phi = 0.25$).

The main conclusions drawn from Figure 7.4 are as follows. (i) The quality of the solutions found by both DGA and DGA-SRM decreases (larger Gap values) as the ratio ϕ is reduced. These results are quite intuitive since reductions on ϕ imply reductions on the fraction of possible subsets of objects that constitute feasible solutions. Consequently, the ratio between the feasible part of the search space and the whole search space gets smaller and the smaller this ratio is the harder

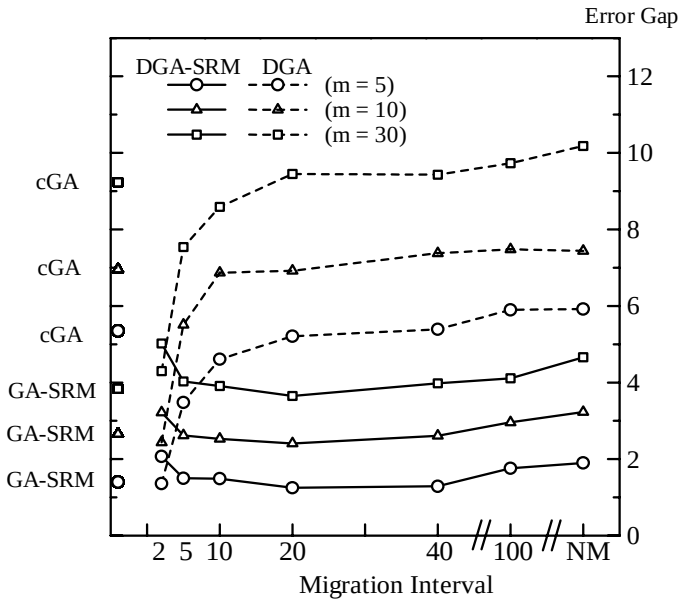


Figure 7.6: Knapsacks m ($K = 16$, 10% migration, $n = 100$, $\phi = 0.25$).

it is to find feasible results. This was also observed for 0/1 single ($m = 1$) knapsack problems in [30]. (ii) The performance of DGA-SRM without migration is by far superior to DGA without migration indicating that a better search is being carried out within each subpopulation in DGA-SRM. (iii) DGA is far away from DGA-SRM unless DGA uses very short migration intervals (2 generations). DGA-SRM achieves high performance with less communication cost for migration, which could be a big advantage for implementation (note that DGA-SRM even without migration performs better than DGA with migration intervals of 5 generations). It seems that DGA-SRM has an optimum range of migration interval (around $10 \sim 20$, for a 10% migration rate) that attains the minimum error gap. Different from DGA, very short migration intervals (less than 10 generation) deteriorates the performance of DGA-SRM. (iv) DGA and DGA-SRM can achieve better results than its single population versions if migration is included.

Second we fix the number of capacities m and tightness ratio ϕ and observe the effect of the number of objects n. Figure 7.5 illustrates results by DGA and DGA-SRM on problems of $m = 30$ capacities, $n = \{100, 250, 500\}$ objects, and ratio $\phi = 0.25$. From Figure 7.5 we can see that increasing the number of objects n also makes it harder for the algorithms to find high quality solutions. Also, we can clearly see the DGA-SRM's optimum migration interval in this figure. The general behavior by DGA-SRM and DGA are similar to that observed in Figure 7.4.

Third, we observe the effect of the number of knapsacks m fixing the number of objects n and tightness ratio ϕ. Figure 7.6 illustrates results by DGA and DGA-SRM on problems of $m = \{5, 10, 30\}$ capacities, $n = 100$ objects, and ratio $\phi = 0.25$. Similar to ϕ and n, from Figure 7.6 we can see that increasing the number of knapsacks m has an strong impact on the performance of the algorithms and that between DGA and DGA-SRM the latter exhibits higher robustness. Also, looking at Figure 7.5 and Figure 7.6, judging from the relative increase on the Gap values, increasing the number of knapsacks (constraints) has an stronger impact than increasing the number of objects (search space).

7.5.2 Subpopulation Size

Fourth, we choose one problem ($m = 30$, $n = 100$, and $\phi = 0.25$) and observe the effect of reducing the subpopulation size while increasing the number of subpopulations (the overall number of offspring and the total number of function evaluations are kept constant to $\lambda_{total} = 800$ individuals and $T = 10^6$ evaluations). Figure 7.7 illustrates results by DGA and DGA-SRM using $K = \{8, 16, 32\}$ subpopulations with subpopulations sizes of $\lambda = \{100, 50, 25^2\}$ and $\tau = \{0.64, 0.56, 0.50\}$, respectively.

DGA-SRM tolerates population reductions better than DGA (see NM for both algorithms) and can still approach GA-SRM's performance relaying on migration. We could not recognize big performance differences between $K = 8$ and $K = 16$ while smaller subpopulations ($K = 32$) tend to require shorter migration intervals.

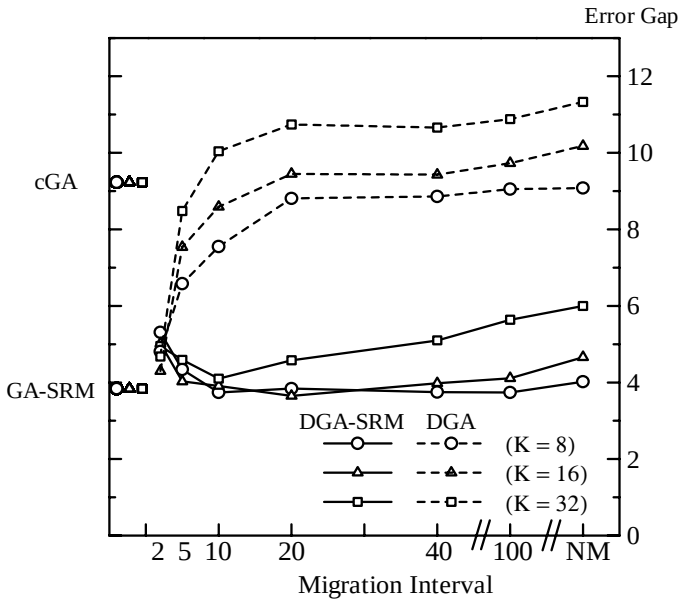


Figure 7.7: Subpopulation size λ, K (10% migration, $m = 30$, $n = 100$, $\phi = 0.25$).

7.5.3 Migration Rate

Fifth, the effect of the migration rate is also observed. Figure 7.8 illustrates results by DGA and DGA-SRM on one of the problems using migration rates of $\{5\%, 10\%, 15\%\}$.

In DGA-SRM smaller migration rates need shorter migration intervals and vice versa. To reduce communication cost, it may be better to use larger migration intervals with higher migration rates in DGA-SRM.

7.5.4 Adaptation

Figure 7.9 illustrates the adaptation of mutations rates in DGA-SRM. The figure shows the average number of the actual bits flipped by SRM over the generations for some of the subpopulations. From Figure 7.9 we can see that adaptation of mutation rates follow similar trajectories and that

[2]When $K = 32$ DGA-SRM uses only $\lambda = 24$ to keep a $1 : 1$ balance for offspring creation between CM and SRM.

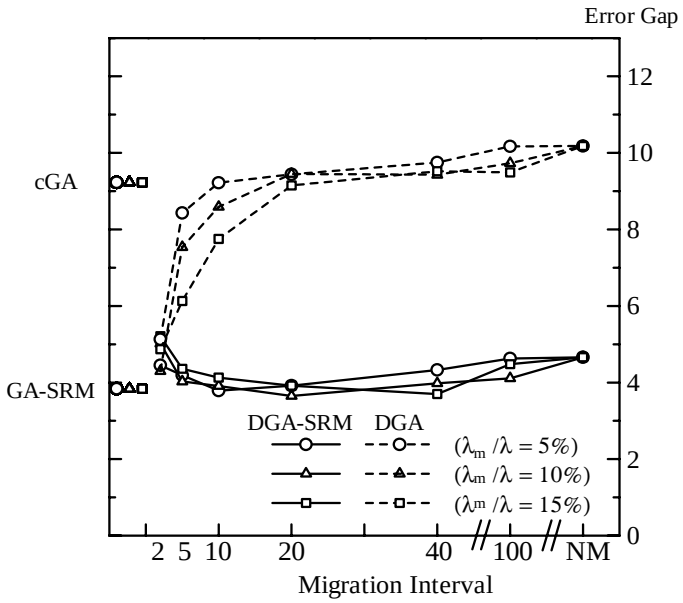


Figure 7.8: Migration Rate λ_m/λ ($K = 16$, $m = 30$, $n = 100$, $\phi = 0.25$).

for most of the run the mutation rates are higher than the usual expected 1 flipped bit of canonical algorithms. It should also be noticed that the instantaneous averages differ, which is a consequence of the local adaptation within each subpopulation. The local adaptation, besides varying mutation rates, also induces different mutation rates for each subpopulation $P_i(t)$.

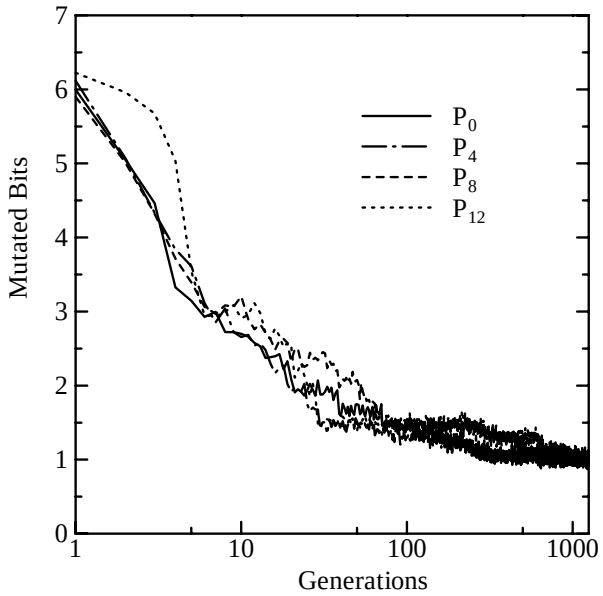


Figure 7.9: SRM's adaptation in DGA-SRM ($K = 16$, $m = 30$, $n = 100$, $\phi = 0.25$).

7.5.5 Extinctive Selection

The remarkable increase in solution quality by the canonical DGA when very short migration intervals are used, i.e. 2 generations, seems at first glance rather counterintuitive (with such migrations intervals one might expect faster convergence but not higher solution quality). However, this is explained by the nature of the test problems and the additional selection intensity caused by migration.

As mentioned above, the problems used in this study are highly constrained with sparse feasible regions where algorithms with penalty functions have a hard time finding feasible solutions[53, 30]. A higher selection pressure in these problems is helping the algorithms to focus the search around the feasible regions (this point has been previously verified in [35] for single population GAs).

The strategies chosen in this work for migrants selection and replacement (selection of the best and replacement of the worst) causes an increase in the overall selection intensity[97]. These strategies combined with very short migration intervals are capable of producing significant selection pressures, which are being used by the DGA. In the case of DGA-SRM, the higher selection pressure is incorporated within the selection mechanism.

Figure 7.10 illustrates the effect of extinctive selection in the distributed algorithms and clarifies the contributions of extinctive selection and adaptive mutation SRM in DGA-SRM. We show results by the canonical DGA with $\lambda = 50$ individuals (DGA(50)) in each subpopulation, a DGA using (μ, λ) Proportional Selection with $\mu = 25$ parents and $\lambda = 50$ offspring in each subpopulation (DGA(25,50)), and the DGA-SRM with similar population sizes (DGA-SRM(25,50)). From this figure we see that extinctive selection alone increases the reliability of the distributed GA in this kind of problems. However, when adaptive parallel mutation, SRM, is used the robustness of the algorithm is improved further.

Figure 7.11 plots the average fitness in the 10 runs of the best solution over the generations by DGA-SRM and DGA. From this figure it can be observed that DGA-SRM has not only higher convergence reliability due to SRM but also a higher search speed caused by extinctive selection.

7.6 Results and Discussion for Real-World 0/1 multiple knapsack problems

In this section we show results on real world instances of 0/1 multiple knapsack problems for which the global optima are known. We especially show results for *Weing7*[98], which is a well-known problem in which GAs have had problems finding the global optimum. For this problems the fitness function used is f_1 as described in 7.4 similar to to [35],[52]. Every experiment presented here consists of 100 independent runs. Each run uses a different seed for the random initial population. The parameters used by GAs are the same used in the previous sections and are indicated in Table 7.1. The results achieved in real world problems confirm our conclusions obtained using random problems generated by the test problem generator.

First Table 7.3 presents results for problem $Weing7$ by the single population cGA and GA-SRM running for $T = 8 \times 10^5$ function evaluations in each run and using offspring populations of $\{800, 400, 200, 100, 50, 25^3\}$ individuals. In Table 7.3 μ and λ indicate the parent and offspring population sizes, G is the number of generations, N is the number of times the global optimum

[3]For GA-SRM we use only 24 individuals instead of 25 to keep a $1:1$ balance for offspring creation by CM and SMR. In this case, the number of evaluations performed by GA-SRM is less than T. The exact number can be calculated by $G \times \lambda$.

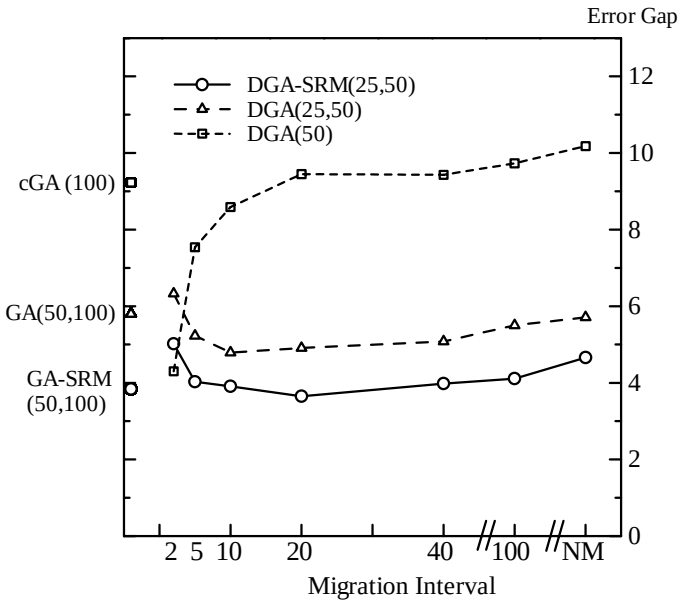


Figure 7.10: Effect of Extinctive Selection ($K = 16$, $m = 30$, $n = 100$, $\phi = 0.25$).

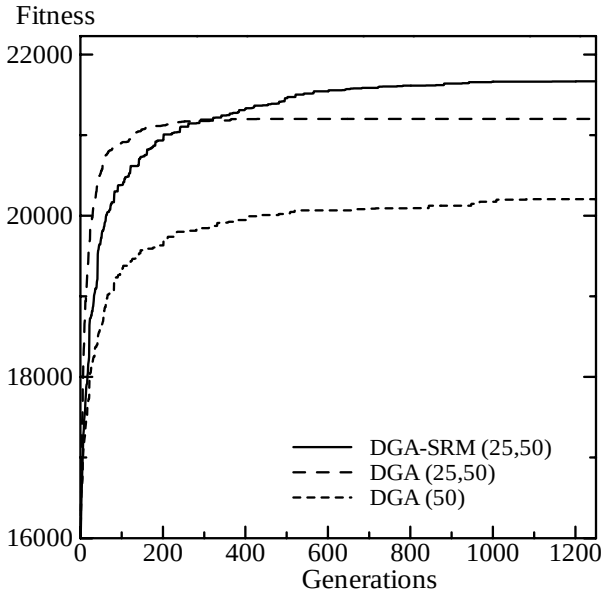


Figure 7.11: Fitness Transition ($K = 16$, $m = 30$, $n = 100$, $\phi = 0.25$, $M = 20$).

was found in the 100 runs, *Average* is the average of the best solutions, and *Stdev* is the standard deviation around *Average*, respectively.

Table 7.3 shows that the inclusion of parallel adaptive mutation, SRM, and extinctive selection significantly increases the reliability of the single population GA in all configurations of population size. Since the population size itself is an important parameter, especially in distributed genetic algorithms, a robust performance in various population sizes is desirable. GA-SRM produces high values for N and *Average* while reducing *Stdev*. cGA, however, fails to locate the global optimum in all occasions and *Average* is very poor. It should be noticed that GA-SRM using

Table 7.3: Results for *Weing7* (105,2) by cGA and GA-SRM using different population sizes ($T = 8 \times 10^5$)

	cGA				*GA-SRM*					
G	λ	N	*Average*	*Stdev*	μ	λ	τ	N	*Average*	*Stdev*
250	800	0	1086807.61	1584.24	400	800	0.58	35	1095401.42	34.55
500	400	0	1087025.10	1401.86	200	400	0.52	39	1095406.57	30.73
1000	200	0	1087515.92	1377.19	100	200	0.46	59	1095418.01	34.34
2000	100	0	1087311.95	1555.19	50	100	0.4	77	1095430.51	26.51
4000	50	0	1086579.32	1304.48	25	50	0.3	60	1095417.55	40.30
8000	25	0	1085391.20	1964.27	12	24	0.2	21	1095302.72	435.54

Table 7.4: Results for *Weing7* (105,2) by DGA and DGA-SRM ($\lambda_{total} = 800, T = 8 \times 10^5$).

			DGA				*DGA-SRM*			
K	λ_m/λ	L R	M	N	*Average*	*Stdev*	M	N	*Average*	*Stdev*
2	10%	1 40	5	0	1092188.75	819.12	100	41	1095401.69	43.94
2	5%	1 20	5	0	1091514.5	963.76	100	45	1095408.92	34.74
2	1%	1 4	5	0	1089631.68	1239.06	80	55	1095415.47	34.67
2	NM	-	-	0	1086895.44	1397.73	-	44	1095409.42	31.68
4	10%	2 10	5	0	1093376.30	602.03	60	48	1095411.94	31.90
4	5%	2 5	5	0	1092294.43	752.08	60	57	1095417.66	31.57
4	1%	2 1	5	0	1090015.36	1208.44	80	61	1095420.43	30.73
4	NM	-	-	0	1087401.30	1342.98	-	52	1095414.46	31.93
8	10%	5 2	5	0	1094423.4	433.38	80	63	1095421.44	30.84
8	5%	5 1	5	0	1093284.95	733.24	100	66	1095423.58	29.84
8	1%	1 1	5	0	1089452.96	1082.41	80	77	1095430.51	26.51
8	NM	-	-	0	1087385.56	1729.4	-	60	1095419.80	30.86
16	10%	5 1	5	0	1094266.87	449.20	40	67	1095424.21	29.62
16	5%	3 1	5	0	1092943.14	686.05	30	66	1095423.33	30.29
16	1%	1 1	5	0	1090143.21	1114.66	10	62	1095421.06	30.58
16	NM	-	-	0	1086404.14	1818.83	-	39	1095404.52	35.44
32	10%	3 1	5	0	1093783.26	514.6	20	51	1095414.13	31.49
32	5%	1 1	5	0	1090507.10	1063.99	10	49	1095412.62	31.84
32	NM	-	-	0	1084922.07	1817.16	-	9	1095284.11	119.23

only a $(\mu, \lambda) = (12, 24)$ population configuration performs better than any cGA. For the same number of function evaluations there is an appropriate combination of population size and number of generations to achieve better results. Intermediate population sizes running for intermediate number of generations perform better than (i) bigger populations running for a small number of generations and (ii) smaller populations running for a higher number of generations.

Next, we observe the effect that the parallel formulation of genetic operators tied to extinctive selection has on the performance of distributed GAs. We fix the overall population size of the distributed algorithms to $\lambda_{total} = 800$ and the overall function evaluations to T, and conduct several experiments partitioning λ_{total} into $K = \{2, 4, 8, 16, 32\}$ populations. For each subpopulation configuration in our experiments we keep the same ratio of the number of arriving

Table 7.5: Results for other problems by DGA and DGA-SRM ($K = 16, \lambda_{total} = 800, T = 4 \times 10^5$)

				DGA				*DGA-SRM* ($\tau = 0.35$)			
Problem (n,m)	K	λ_m/λ	L R	M	N	*Average*	*Stdev*	M	N	*Average*	*Stdev*
Petersen6 (39,5)	16	10%	5 1	5	0	10552.96	19.46	80	54	10611.63	6.91
	16	1%	1 1	5	0	10506.90	26.11	140	77	10614.82	5.82
	16	NM	-	-	0	10490.91	30.19	-	71	10614.03	6.22
Petersen7 (50,5)	16	10%	5 1	5	0	16442.39	21.52	40	89	16535	5.94
	16	NM	-	-	0	16347.02	32.03	-	60	16530.18	9.28
Sento1 (60,30)	16	10%	5 1	5	0	7694.22	9.97	40	98	7771.78	1.54
	16	NM	-	-	0	7563.32	43.62	-	89	7770.79	3.44
Sento2 (60,30)	16	10%	5 1	5	0	8661.23	19.18	40	84	8721.32	2.11
	16	NM	-	-	0	8559.52	25.4	-	70	8720.89	2.31

migrants to the offspring subpopulation size $\lambda_m/\lambda = L \times R/\lambda$ and vary the migration interval $M = \{5, 10, 20, 30, 40, 60, 100\}$.

Table 7.4 presents the results by DGA and DGA-SRM for migration rates of $100 \times \lambda_m/\lambda = \{10, 5, 1\}$ (values for number of links L and the number of migrants R are chosen accordingly). Only the best results achieved in the migration intervals M mentioned above are shown. It also presents the results when there is no migration and the subpopulations evolve in total isolation (denoted NM in the table).

Table 7.4 shows that DGA-SRM outperforms DGA for any configuration of number of subpopulations and subpopulation sizes. It should be noticed that the performance by DGA-SRM even without migration is higher than DGA. Similarly, DGA-SRM also exhibits higher convergence reliability (higher values of N and *Average* with smaller *Stdev*) than its corresponding single population GA-SRM (800 offspring and T evaluations in Table 7.3). Actually, except for $K = 32$, DGA-SRM without migration produce better results than GA-SRM. If low migration rates are included then DGA-SRM further improves GA-SRM results. A 10% migration rate turns out to be too high for DGA-SRM in configurations of $K = \{2, 4, 8\}$ subpopulations. Especially for $K = \{2, 4\}$ DGA-SRM without migrations performs better than DGA-SRM with migration. For these population sizes a 1% migration rate is enough. As we increase the number of subpopulations and reduce the subpopulation sizes migration is required with more frequency. Something similar happens when migration rate is reduced. Better results are obtained by DGA-SRM with 10% migration rate. However, results by 1% migration rate are still pretty high.

DGA also presents convergence reliability higher than cGA. However, significant improvements by DGA are achieved only if migration is performed very frequently (small values of M). For higher migration intervals the performance of DGA approaches to that of cGA. Also, It should be noticed that convergence reliability of the single population GA-SRM is higher than the distributed canonical GA (DGA).

Table 7.5 shows the results for other real world 0/1 multiple knapsack problems by DGA and DGA-SRM running for $T = 4 \times 10^5$ using $K = 16$ subpopulations and a 10% migration rate. The distributed algorithms present a similar behavior to that observed in *Weing7* problem.

The migration rate that leads to higher performance could be different for each distributed configuration of K and λ. Thus, values for N, *Average*, and *Stdev* presented in Table 7.4 and Table 7.5 do not necessarily show the best results that can be achieved by the distributed algorithms

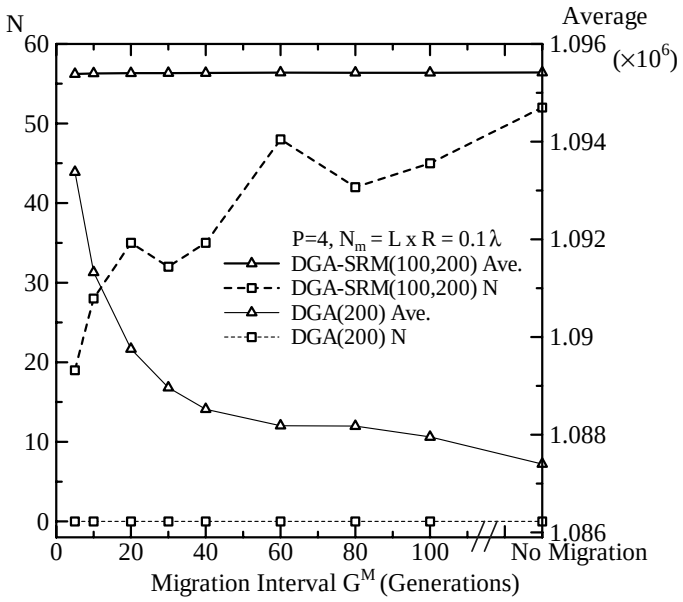


Figure 7.12: Effect of the Migration Interval: Four subpopulations ($K = 4$), migration rate $\lambda_m/\lambda = 10\%$.

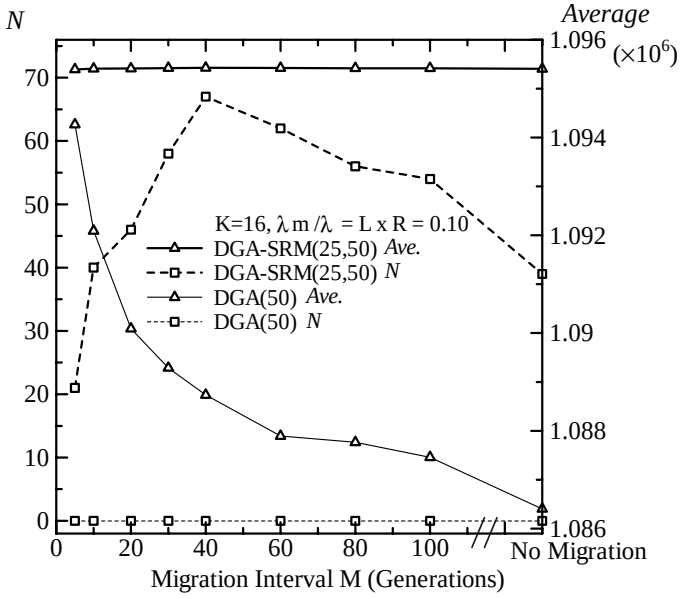


Figure 7.13: Effect of the Migration Interval: Sixteen subpopulations ($K = 16$), migration rate $\lambda_m/\lambda = 10\%$.

but rather a more general tendency when we scale down subpopulation sizes and increase the number of subpopulations using a unified migration rate criteria.

Figure 7.12 and Figure 7.13 illustrate the effect of the migration interval M in the distributed GAs using configurations of $K = 4$ and $K = 16$ subpopulations and a 10% migration rate. From this figures we should notice that DGA-SRM maintains high *Average* regardless of the migration interval while DGA achieves its better results for very small values of M (high migration

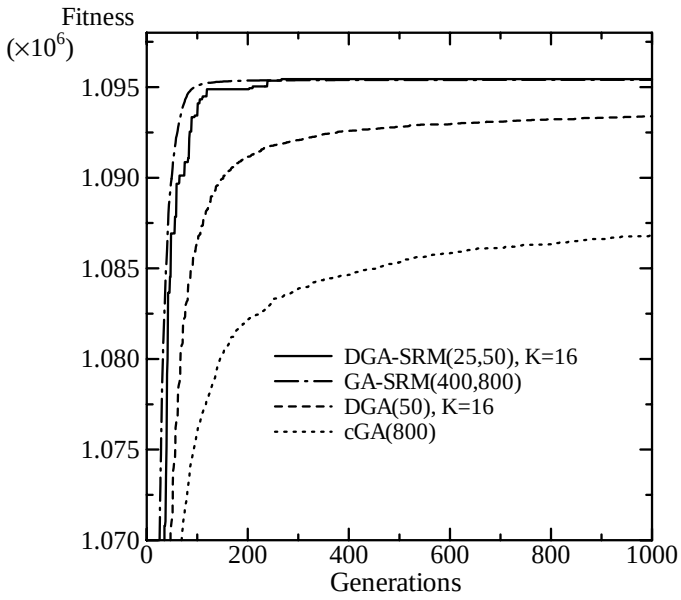


Figure 7.14: Fitness transition over the generations ($K = 16$)

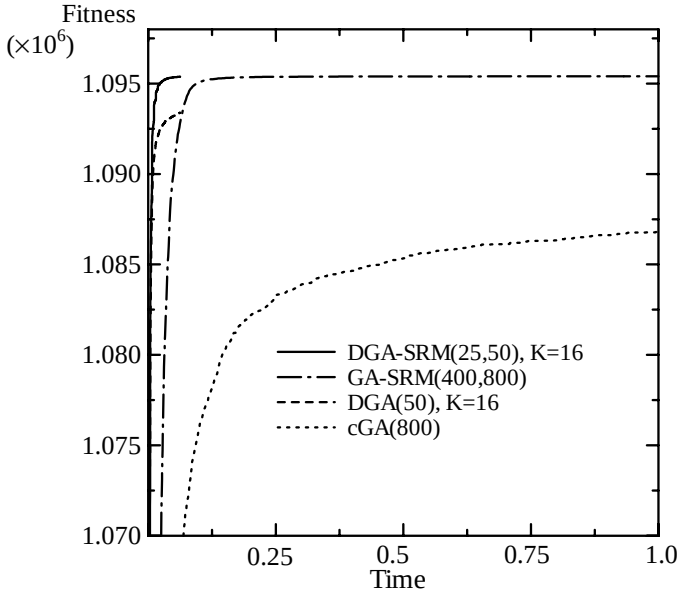


Figure 7.15: Fitness transition over time ($K = 16$)

frequency). Smaller migration cost without deterioration of performance is also an important feature of DGA-SRM. DGA performance decreases notoriously as M is increased. DGA-SRM for $K = 16$ subpopulations (50 offspring individuals each) with intermediate migration intervals (around 40 generations) produces best results. Also, if M is too small the performance by DGA-SRM is worse than DGA-SRM without migration. This is because the selection pressure induced by migration becomes too high for DGA-SRM as the migration interval M reduces. It is interesting to note that for $K = 4$ subpopulations best results are achieved by DGA-SRM without migration rather than by DGA-SRM with migration. In this case, the selection pressure induced

by a 10% migration rate in $K = 4$ large subpopulations (200 offspring individuals each) is already too high for DGA-SRM.

Figure 7.14 plots the average in 100 runs of the best so far solutions over the generations by the single population algorithms and the distributed algorithms for $K = 16$. Similarly, Figure 7.15 illustrates the solution quality over actual time by the same algorithms. These figures clearly show the higher convergence reliability and higher search speed by the single and distributed algorithms using parallel genetic operators. Also, it should be noted that using the distributed version of the parallel formulation of genetic operators we could remarkably reduce the overall time to completion while keeping high solution quality.

7.7 Conclusions

In this chapter we have studied the performance of a distributed GA that incorporates parallel cooperative-competitive genetic operators (DGA-SRM). A series of controlled experiments using various large and difficult 0/1 multiple knapsack problems were conducted to test the robustness of DGA-SRM. Comparisons were made between DGA-SRM and a canonical distributed GA (DGA).

We observed that high selection intensity helps to perform a better search in this kind of combinatorial problems. The DGA-SRM incorporates a higher selection pressure within its selection mechanism. The canonical DGA, however, has to rely in the higher selection intensity introduced by migration and can achieve high results only at the expense of very high communication cost.

The inclusion of high mutation parallel to crossover within DGA-SRM improves further the convergence reliability of the canonical DGA regardless of the difficulty of the problem. Also, due to the high selection intensity within each subpopulation and its built-in source of diversity by SRM, the search speed of the algorithm is increased without sacrificing the quality of solutions. DGA-SRM even without migration produces very high results compared to canonical DGA with small migration intervals.

As future works, the effect that other communication topologies and migration policies have on the performance of this kind of hierarchical GA should be investigated. Also, we would like to extend the concept of GA-SRM to cellular models.

Chapter 8

Real World Application: Halftone Image Generation

This chapter shows that GA-SRM can be successfully applied to real world problems in which efficiency in processing time and computer memory is a major issue. The improved GA-SRM is extended to the two dimensional image halftoning problem and an accelerated image halftoning technique with tiny populations is presented. Simulation results show that the proposed scheme is impressively efficient reducing computer memory and processing time required to obtain high quality halftone images, making the improved scheme appealing for practical implementations of the image halftoning technique using GAs.

8.1 Introduction

Evolutionary computation (EC) is a multidisciplinary growing field that simulates evolution for problem solving. Global search abilities, adaptation to the task in hand, and robust performance are favorable characteristics of evolutionary algorithms that have made them successful to solve various kinds of complex optimization problems[15, 99]. In particular, Genetic Algorithms (GAs) have vigorously been developed and analyzed since Holland's[11] and Goldberg's[16] contributions and employed in different kinds of application domains. In the signal processing area, a number of application methods using GA are also being increasingly developed[100]. Though evolutionary algorithms are having great success, it is known that evolutionary algorithms have not been able to comply especially with efficiency requirements in several applications that require intensive processing time. Research towards improving the effectiveness and efficiency of evolutionary algorithms is needed in order to broaden their field of application in industry and real world problems. With this motivation, in previous chapters we have proposed, analyzed, and showed that GA-SRM, a GA that applies crossover and varying mutation in a parallel cooperative-competitive manner, greatly improves the performance of GAs for global optimization problems[101, 34]. Acceleration of the search process and robustness even with small populations are also remarkable characteristics observed in the improved GA-SRM.

In this chapter we show that GA-SRM can be successfully applied to real world problems in which efficiency in processing time and computer memory is a major issue. In our work, we especially focus on the image halftone technique using GA. Kobayashi et al[102, 103] use a GA to generates bi-level halftone images with quality higher than conventional techniques such as ordered dithering, error diffusion and so on[104]. Both high gray level and high spatial resolution are attained in the halftone images generated by this scheme. However, this scheme uses a substantial amount of computer memory and processing time[102, 103] that deprive it from practical implementations.

We extend the improved GA[101, 34] to the two dimensional image halftoning problem with the objective to obtain a new efficient GA based image halftoning scheme that can be suitable for practical implementations[105, 106]. That is, a scheme that can generate high quality images, such as those obtained by the conventional scheme using GA[102, 103], but minimize computer memory and processing time simultaneously. The proposed scheme is applied to SIDBA's benchmark images in our simulation. Simulation results show that our scheme impressively reduces computer memory and processing time required to generate high quality images. For example, compared to the conventional halftoning technique with GA[102, 103], our scheme using only a 2% population size[1] requires about 15% of the objective function evaluations (processing time) to generate high quality images. The results make our scheme appealing for practical implementations of the image halftoning technique using GA.

8.2 Conventional Image Halftoning Technique Using GA

In the conventional image halftoning technique using GA an input gray scale image is first divided into non-overlapped blocks of $n \times n$ pixels. Then, the two dimensional optimum binary pattern for each image block is searched using a GA[102, 103]. The GA uses a $n \times n$ two dimensional binary representation for the chromosomes. Crossover is implemented to interchange either sets of adjacent rows or columns between two chromosomes. Mutation inverts bits with a very small

[1] A 2 parents and 4 offspring configuration in our method against a 200 parents and 200 offspring configuration used in [102, 103].

probability per bit and it is applied after crossover similar to canonical GA[11, 16].

Individuals are evaluated for two factors required to obtain visually high quality halftone images. (i) One is high gray level resolution (local mean gray levels close to the original image), and (ii) the other is high spatial resolution (appropriate contrast near edges)[102, 103]. The function that measures the individuals' error to simultaneously satisfy these conditions is expressed by

$$e(\boldsymbol{x}_i^{(t)}) = \omega_m E_m(\boldsymbol{x}_i^{(t)}) + \omega_c E_c(\boldsymbol{x}_i^{(t)}) \tag{1}$$

where $\boldsymbol{x}_i^{(t)}$ is i-th individual at t-th generation, E_m and E_c are the errors for gray level factor and contrast one, and ω_m and ω_c are their weighting parameters, respectively. Then individuals' fitness is assigned by

$$f(\boldsymbol{x}_i^{(t)}) = e(\boldsymbol{x}_W^{(t)}) - e(\boldsymbol{x}_i^{(t)}) \tag{2}$$

where $e(\boldsymbol{x}_W^{(t)})$ is the error associated with the worst individual at t-th generation. GA is used to find the optimum compromise between (i) and (ii) with the above fitness function. High quality, visually satisfactory, halftone images are obtained with 200 individuals and 200 generations (totally 40,000 objective function evaluations per block)[102, 103].

8.3 Accelerated Halftoning Scheme Using GA-SRM

8.3.1 Extension to Two Dimensional Image Halftoning Problem

In this section, the improved GA[101, 34] is extended to the two dimentional image halftoning problem. The nature of the problem must be taken into account and reflected in the representation used for the chromosome and in the implementation of two kinds of genetic operators as well. The extended scheme is described in detail as follows.

Two Dimensional Representation

A chromosome is represented as a $n \times n$ 2-dimensional binary structure[102, 103]. That is, the chromosome is interpreted as having n rows and n columns. Thus, let us generally denote a bit in the individual $\boldsymbol{x}_i^{(t)}$ as $b(u, v)$ $(0 \leq u \leq n-1, 0 \leq v \leq n-1)$ from now on.

Two Dimensional CM

Crossover is implemented for two dimentional chromosomes similar to [102, 103]. Two random numbers, c_t and c_p, define its method of operation. First, $c_t = N[0, 1]$, is sampled to decide whether to interchange chromosomes' rows or columns from two previously selected parents, say (i) if $c_t = 0$, interchange rows and (ii) if $c_t = 1$, interchange columns. Then, $c_p = N[0, n]$ indicates the crossing point. Both c_t and c_p are sampled anew for each individual created by CM. An example[2] of the 2-dimensional crossover is shown in Figure 8.1. Although crossover can potentially create two offspring at a time, only one of them is kept being this decision also made at random in our scheme.

After crossover, mutation inverts[3] bits with a small probability per bit, $p_m^{(CM)}$, analogous to canonical GA[11, 16]. Thus, mutation in CM is of a quantitative nature after which the number of 0s and 1s in the chromosome may change.

[2]Only one of the two possible offspring is shown for both types of interchange.

[3]For every bit actually selected for mutation a 0 becomes 1 and vice versa.

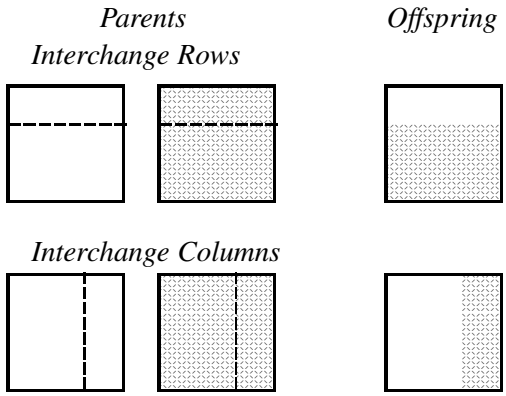


Figure 8.1: Illustration of 2D Crossover

Tow Dimensional SRM

In order to produce offspring with SRM, individuals are selected from the parent population $P(t)$, an exact copy is created and then mutation is applied only to the bits inside a mutation block. SRM is provided with an Adaptive Dynamic-Block (ADB) mutation schedule similar to Adaptive Dynamic-Segment mutation (ADS)[101, 34].

With ADB mutation is directed only to a block (square region) of the chromosome and the mutation block area $\ell \times \ell$ is dynamically adjusted every time a normalized mutants survival ratio falls under a threshold, $\gamma < \tau$. The normalized mutant survival ratio is specified by

$$\gamma = \frac{\mu_{SRM}}{\lambda_{SRM}} \cdot \frac{\lambda}{\mu} \tag{3}$$

where μ is the number of individuals in the parent population $P(t)$, μ_{SRM} is the number of individuals created by SRM present in $P(t)$ after selection, λ_{SRM} is the offspring number created by SRM and λ is the total offspring number, $\lambda_{CM} + \lambda_{SRM}$. The block's side length reduction is summarized below:

$\ell = n, (t = 0)$
if $(\gamma < \tau)$ **and** $(\ell > 2)$
  $\ell = \ell/2$

where the block's side length ℓ varies from n to 2, $[n, 2]$ following a decreasing approach as shown in Figure 8.2. The offset position of the mutation block, $\Delta_s = (\phi_s, \psi_s)$, for each chromosome is chosen at random. Thus, $\phi_s = N[0, n - \ell]$ and $\psi_s = N[0, n - \ell]$. The final position is calculated by $\Delta_f = (\phi_f, \psi_f) = (\phi_s + \ell - 1, \psi_s + \ell - 1)$.

Two kinds of mutation schemes are investigated for ADB: (i) quantitative and (ii) qualitative mutation. Quantitative mutation in ADB is implemented as the standard bit flipping process, i.e. 0 becomes 1 and vice versa. Mutation probability for the bits inside the segment is $p_m^{(SRM)} = \alpha$. After this kind of mutation has been applied, the contrast near edges and the local mean average might change in an individual affecting both E_c and E_m in Eq. (1). Quantitative mutation would allow observing the general effect of parallel mutation in this problem. We call this mutation scheme quantitative mutation from now on.

On the other hand, qualitative mutation in ADB is implemented as a bit swapping process. First, a set B is initialized with every bit in the mutation block. A pair of bits are randomly

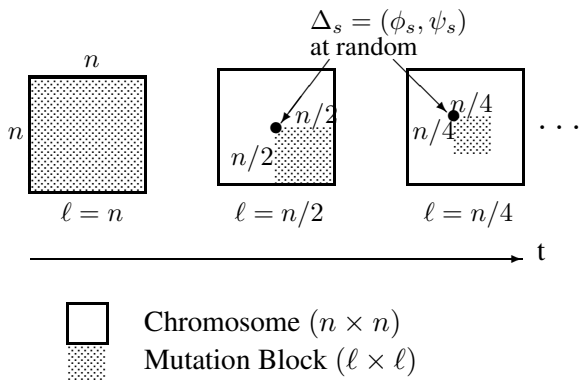


Figure 8.2: Adaptive Dynamic-Block mutation (ADB)

marked[4] and then swapped. The marked bits are removed from B and the process is repeated until there are no remaining bits in B. The whole process is summarized in Figure 8.3 and Figure 8.4 enlarges a 4×4 mutation block to illustrate the bit swapping process for two of the bits. Note that it is not necessary to set a mutation probability in qualitative mutation since all pairs of bits within the mutation block are simply swapped. We call this mutation scheme qualitative mutation from now on.

Note that after qualitative mutation the number of 0s and 1s remains unchanged. In other words, qualitative mutation has an impact only on the evaluation of the spatial resolution's error, E_c, but not on the gray level resolution's error, E_m, in Eq.(1). This kind of mutation could take better advantage of the high correlation among contiguous pixels in an image[107], and contribute to a more effective search.

The adaptive mechanism in SRM is designed to control the required exploration-exploitation balance during the search process.

Selection

(μ, λ) Proportional Selection[108] implements the required extinctive selection mechanism. Selection probabilities are computed by

$$p(\boldsymbol{x}_i^{(t)}) = \begin{cases} \dfrac{f(\boldsymbol{x}_i^{(t)})}{\sum_{j=1}^{\mu} f(\boldsymbol{x}_j^{(t)})} & (1 \leq i \leq \mu) \\ 0 & (\mu < i \leq \lambda) \end{cases} \tag{5}$$

where $\boldsymbol{x}_i^{(t)}$ is an individual at t-th generation which has the i-th highest fitness value $f(\boldsymbol{x}_i^{(t)})$, μ is the number of parents and λ is the number of offspring. This kind of selection has been characterized as dynamic, extinctive pure selection[108]. Also, selection is reinforced to assure that the two parents selected for crossover are different avoiding that an individual crosses with itself,[5] i.e. the parents for crossover are $\boldsymbol{x}_i^{(t)}$ and $\boldsymbol{x}_j^{(t)}$ $(i \neq j)$.The extinctive nature of this selection mechanism subjects SRM's and CM's offspring to compete for survival.

[4]The bits' coordinates are different but the bit values could be the same; for example, $b(0,0) = 0$ and $b(3,2) = 0$.

[5]Note that for parent selection we do not check whether two individuals have identical genetic information.

$B = \{b(u,v); \phi_s \leq u \leq \phi_f, \psi_s \leq v \leq \psi_f\}$
while B is not empty
$mark(B)$; randomly mark $b', b'' \in B(b' \neq b'')$
$sawp(b', b'')$; swap the bits b' and b'',
then $B = B - \{b', b''\}$

Figure 8.3: Bit swapping

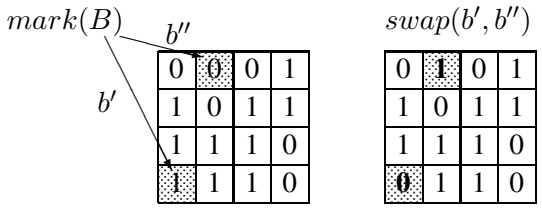


Figure 8.4: Illustration of bit swapping process for qualitative mutation (4×4 mutation block)

8.4 Experimental Results and Discussion

8.4.1 Experimental Setup

The improved GA extended to the two dimensional image halftoning problem is applied to SIDBA's benchmark images in our simulation. The size of the original image is 256×256 pixels with 256 gray levels. The evaluation function is the same used in [102, 103] and the weighting parameters are set to $w_m = 0.2$ and $w_c = 0.8$. An image is divided into 256 blocks, each one of size 16×16 pixels. For each block, the algorithm was set with different seeds for the random initial population and ended after the same $T = 4 \times 10^4$ evaluations used in [102, 103] were performed (the number of generations is calculated as T/λ in this scheme). Mutation probability for CM is set accordingly to $p_m^{(CM)} = 0.001$. Unless stated otherwise, in every experiment in the following subsections we use "Girl" as the experimental image, $\lambda_{CM} : \lambda_{SRM} = 1 : 1$ for offspring creation, and $\mu : \lambda = 1 : 2$ (extinctive pressure) which in [101, 34] proved to be the best parameters' balance for a robust and reliable search. Also, we use a $\tau = 0.40$ as a threshold for the normalized mutant survival ratio specified by Eq. (3). Mutation probability for ADB when it is implemented with quantitative mutation is set to $p_m^{(SRM)} = 0.125$. Again, note that it is not necessary to set a mutation probability in qualitative mutation as mentioned in 3.2.3.

8.4.2 Performance Observation with Same Size Offspring Population Used by Conventional Scheme

First, in order to observe the performance by the proposed algorithm (GA-SRM), i.e. the evolution of image quality and its convergence speed, we set the population sizes to $\mu = \lambda_{CM} = \lambda_{SRM} = 100$. With this values our scheme creates the same number of offspring (200 offspring from 100 parents) as the conventional scheme with GA (cGA) does (200 offspring from 200 parents)[102, 103] to attain high quality images. Figure 8.5 shows the image's average-error

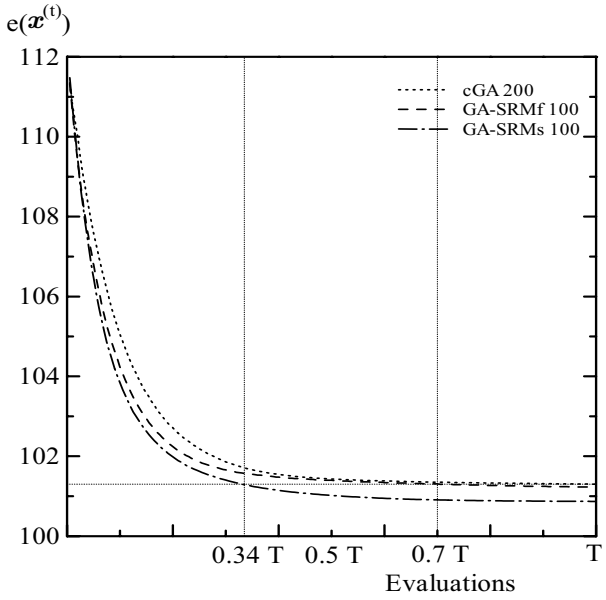


Figure 8.5: cGA and GA-SRM's performance using same size offspring population

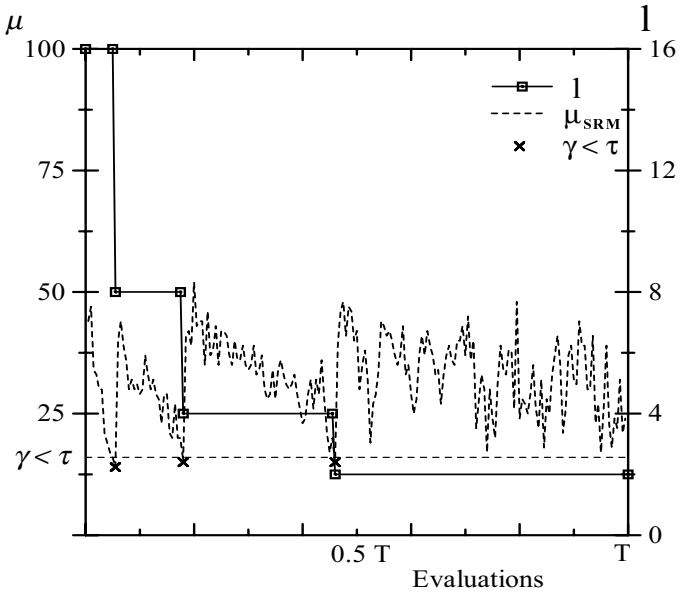


Figure 8.6: Mutation block's side length reduction and SRM-ADB offspring that survive selection

transition, calculated as the average of the best individuals' error in the 256 image blocks, by the two schemes. From Figure 8.5 it can be seen that GA-SRM converges faster and reaches better quality levels than cGA. Also, as expected, qualitative mutation performs better than quantitative mutation. Under this population configuration, GA-SRM needs only 0.34 T evaluations to surpass the final image quality levels obtained by cGA when qualitative mutation is used (GA-SRMs) whereas 0.7 T evaluations are needed in the case of quantitative mutation (GA-SRMf).

SRM's behavior can be observed from Figure 8.6, which presents the block's side length reduction, ℓ, and the number of individuals produced by SRM-ADB that survive selection, μ_{SRM},

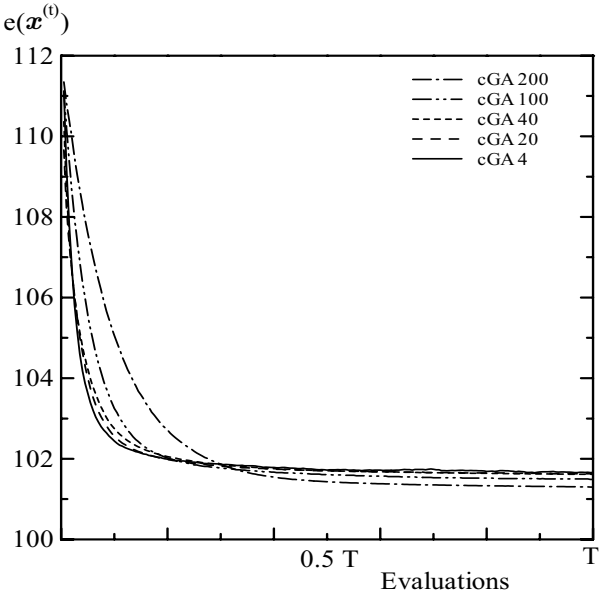


Figure 8.7: Performance by cGA with different populations sizes

for one image block. From this figure it is clear that (i) SRM contributes with beneficial mutations (carried by mutants that survive selection) in every generation of the search process, and (ii) the key factor for SRM to be an effective operator lies in its own regulation mechanism: mutation block adjusted every time the number of mutants that survive selection falls under a minimum level τ.

8.4.3 Effect of Population Size Reduction

Next, since GA-SRM introduce higher levels of diversity than cGA we also want to observe the performance of the algorithms with smaller populations where diversity is even a more important issue. Figure 8.7 show results by cGA using {200,100,40,20,4} population configurations. Figure 8.8 and Figure 8.9 present results for equivalent configurations $\mu = \lambda_{CM} = \lambda_{SRM} = \{100,\ 50,\ 20,\ 10,\ 2\}$ by GA-SRMf and GA-SRMs, respectively, along with those obtained by cGA using a 200 population. From Figure 8.7 we can see that the 200 population size leads to the best image quality in cGA. As the population size is reduced the final image quality is also deteriorated. Figure 8.8 shows that the introduction of quantitave mutation allows us to considerably reduce population sizes from 100 to 10 and still obtain a gain on search speed to generate images of quality similar or a little better compared to cGA. However, a further reduction in population sizes from 10 to 2 is not effective. In this minimum configuration the levels of mutation introduce by GA-SRMf are too high, which does not allow SRM's offspring to compete properly against CM's offspring. We should also mention that a population size of 2 individuals was tried for cGA obtaining a final image's average-error above 112.

In Figure 8.9 we observe that GA-SMRs using qualitative mutation with bigger populations eventually achieve a higher image quality (this trend is similar in cGA and GA-SRMf as well). However, smaller populations converge faster and always produce a better image quality that the one obtained by cGA. In this case, qualitative mutation not only allows to reach higher levels of image quality but also to reduce the population configuration to its minimum level. This is because

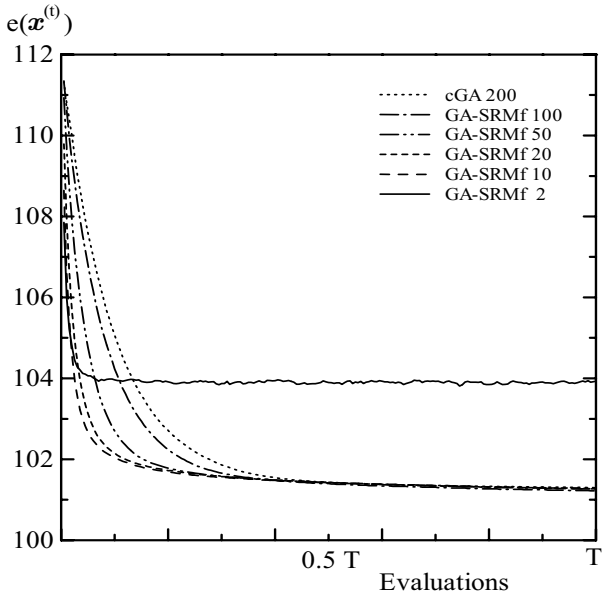


Figure 8.8: Performance by GA-SRMf (quantitative mutation) with different population sizes

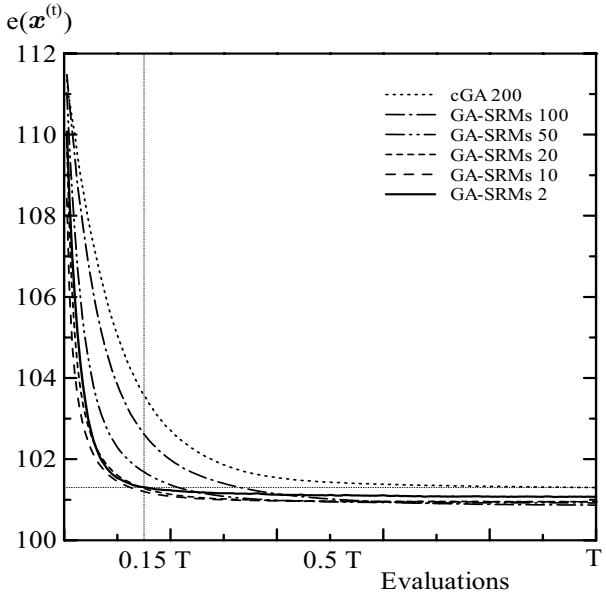


Figure 8.9: Performance by GA-SRMs (qualitative mutation) with different population sizes

SRM with this kind of mutation always contributes to introduce diversity in levels such that SRM could be competitive with CM regardless of the population size, avoiding premature convergence,[6] which is an important concern in cGA[11, 16]. It should be noticed that the probability of cloning with this operator is higher when the mutation block's lenght has reached its minimum lenght. In this way qualitative mutation also introduces a kind of implict elitism. These characteristic

[6] All the individuals are trapped on local optima in earlier generations.

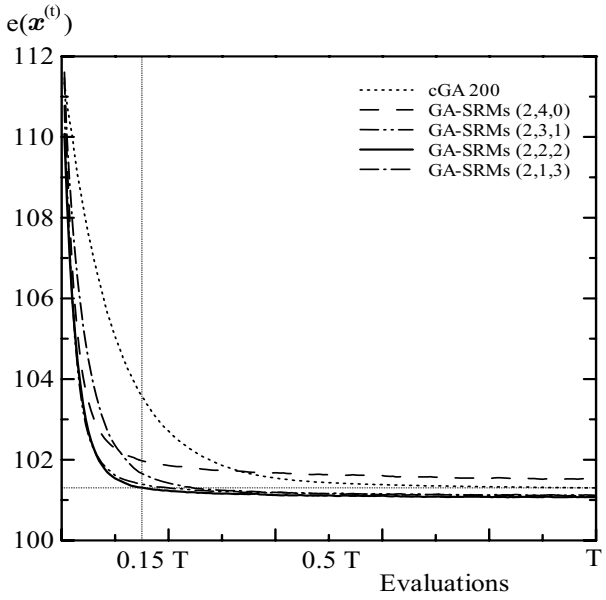


Figure 8.10: Performance with different parameter balance for offspring creation ($\mu = 2$ and $\lambda = 4$ configuration)

explains the GA-SRM's robust performance even with tiny populations and allows us to choose the smallest memory configuration to generate halftone images without compromising the image quality. In fact, the improved GA-SRM using qualitative mutation with $\mu = 2$ and $\lambda = 4$ configuration (merely 2% of the population size used in [102, 103]) attained after only $0.15T$ evaluations the same image quality obtained by cGA after T evaluations, especially for "Girl" image.

8.4.4 Effect of Parameters' Balance for Offspring Creation

Furthermore, the parameter's balance for offspring creation is studied for the smallest configuration $\mu = 2$ and $\lambda = 4$. Results obtained by GA-SRMs using configurations of $(\mu, \lambda_{CM}, \lambda_{SRM}) = \{(2,4,0),\ (2,3,1),\ (2,2,2),\ (2,1,3)\}$ are shown in Figure 8.10. This figure clearly shows that the diversity required to find the global optimum comes from SRM. That is, when no individuals are created by SRM, $(\mu, \lambda_{CM}, \lambda_{SRM}) = (2,4,0)$, the algorithm converges prematurely with worse performance than cGA whereas even a one SRM's offspring configuration, $(\mu, \lambda_{CM}, \lambda_{SRM}) = (2,3,1)$, outperforms cGA.

Figure 8.10 also highlights the importance of a proper balance between operators. A configuration that emphasizes CM's over SRM's offspring number will tends to converge faster. For example, see $(\mu, \lambda_{CM}, \lambda_{SRM}) = (2,3,1)$ and $(\mu, \lambda_{CM}, \lambda_{SRM}) = (2,1,3)$. On the other hand, the lack of enough SRM's offspring may result in a lower optimum. In this work, analogous to [101, 34], a $1:1$ operators' balance has proved again to be the best choice for offspring creation that leads to a better optimum in the shortest time. This is because the operators are properly balanced and SRM is implemented to be competitive to CM, and consequently the cooperation expected from them emerges producing a higher convergence velocity and reliability. Although the same operators' balance is used for all image blocks and it is kept constant through the entire evolution in this work, it may be worth assigning dynamic configurations based on individual block's characteristics. For example, blocks that not include contrast near edges may be favored by

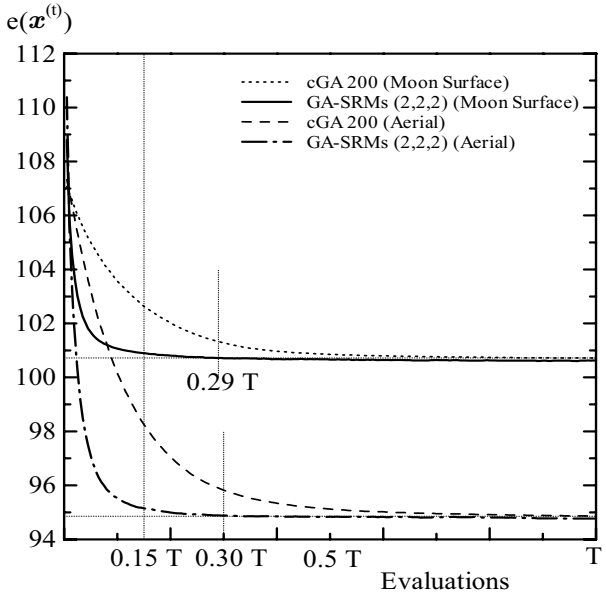


Figure 8.11: cGA and GA-SRMs's performance for "Moon Surface" and "Aerial" ($(\mu, \lambda_{CM}, \lambda_{SRM}) = (2, 2, 2)$ configuration in GA-SRMs)

a configuration that either emphasizes CM's over SRM's offspring or starting with a $1:1$ balance later on switches to a CM intensive regime. This point deserves future research.

8.4.5 Other Benchmark Images

It should be mentioned that similar behavior was also observed for other benchmark images and weighting parameters in Eq. (1). Main results for "Moon Surface" and "Aerial" images obtained by the proposed GA-SRM using qualitative mutation with a $(\mu, \lambda_{CM}, \lambda_{SRM}) = (2, 2, 2)$ configuration are shown in Figure 8.11, where the weighting parameters are the same used for "Girl", $w_m = 0.2$ and $w_c = 0.8$. Analogous to the case of "Girl", the evolution is accelerated by GA-SRMs even with the smallest memory configuration, and the evaluation times necessary to reach the final result obtained by cGA are remarkably reduced. The exact reduction of the number of evaluations would depend on the characteristics of the input image. For example, as shown in Figure 8.11, 0.29 T and 0.30 T evaluations were requiered for "Moon Surface" and "Aerial", respectively.

8.4.6 Generated Images

Figure 8.12 shows the original image "Girl", the generated halftone image by cGA after T and $0.15\ T$ evaluations and those generated by GA-SRM using qualitative mutation after T and 0.15 T evaluations for visual comparison. There is a notorious difference between (c) and (d). On the other hand, we cannot visually recognize the difference between (b) and (d).

Figure 8.13 shows the original image and the generated halftone images for "Moon Surface" and "Aerial". Although in the plots of Figure 8.11 we can observe that GA-SRM obtains better quality images than cGA, from the generated images in Figure 8.13 we cannot visually recognize the difference between (b) and (c) in the case of "Moon Surface" at 0.29 T, and also between (g)

and (h) in the case of "Aerial" at 0.30 T, which differs from the case of "Girl". However there is a notorious visual difference at 0.15 T evaluations between (d) and (e) for "Moon" and between (i) and (j) for "Aerial", analogous to the case of "Girl". It should be noticed that our scheme still keeps enough image quality even at 0.15 T evaluations, and thus almost no deterioration from (c) to (e), and from (h) to (j) is visually recognized. Summarizing our study, with the proposed scheme almost the same results as the conventional scheme can be surely obtained for various images if we spend at most about 0.3 T evaluations. In addition, from a practical point of view, we can obtain halftone images with enough high quality if we spend only about 0.15 T evaluations (6,000 evaluations).

It should be mentioned that recently the bi-level halftoning technique has been extended to generate multi-level halftone images. In the multi-level problem the search space becomes significantly larger than the bi-level problem. Similar to the bi-level case, GA-SRM proved to be very effective[109, 110].

8.5 Conclusions

In this chapter, based on a new cooperative model for genetic operators, we have extended an improved GA to the 2-dimensional image halftoning problem. For this kind of problem, the proposed scheme manages to introduce diversity regardless of population size avoiding premature convergence even when tiny populations are used. This allows us to choose the smallest population configuration to generate bi-level images without compromising the higher image quality. The simulation results show that our scheme impressively reduces computer memory and processing time required to generate high quality bi-level halftone images. For example, compared to the conventional halftoning technique with GA[102, 103], our scheme using only a 2% population size (2 parents and 4 offspring configuration) requires about 15% evaluations to generate high quality images. The results make our scheme appealing for practical implementations of the image halftoning technique using GA.

As future works, higher levels of adaptation for the improved GA and its application to other imaging problems should be investigated.

(a) Original image

(b) T evaluations

(c) $0.15\ T$ evaluations

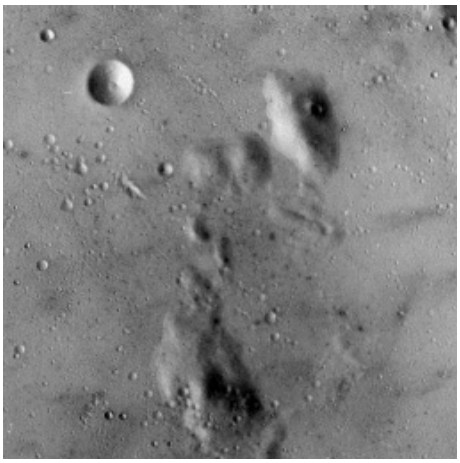
Halftone by cGA (200 individuals)

(d) $0.15\ T$ evaluations

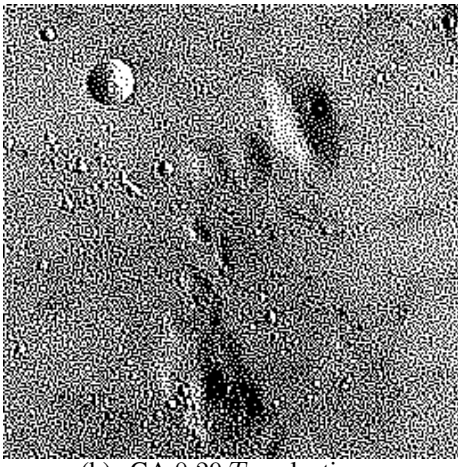
(e) T evaluations

Halftone by GA-SRMs (2 individuals)

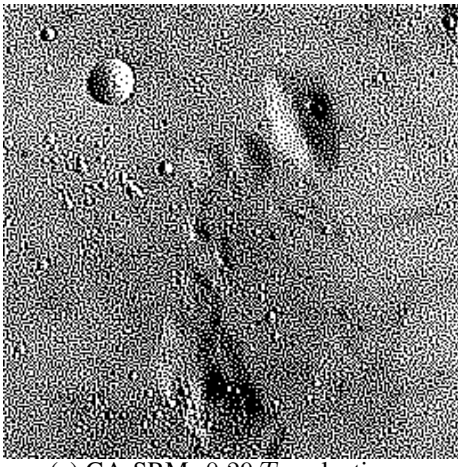
Figure 8.12: Original and generated halftone images (“Girl”)



(a) Original image “Moon Surface”

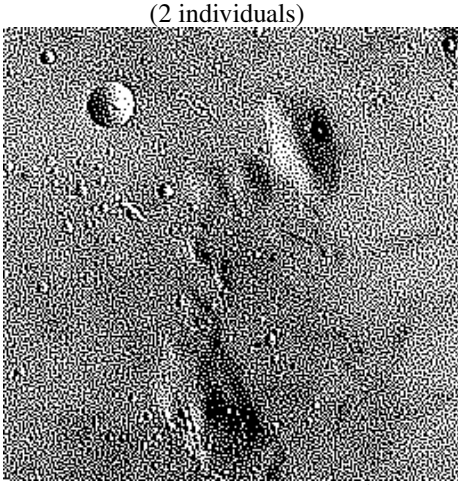


(b) cGA $0.29\,T$ evaluations (200 individuals)



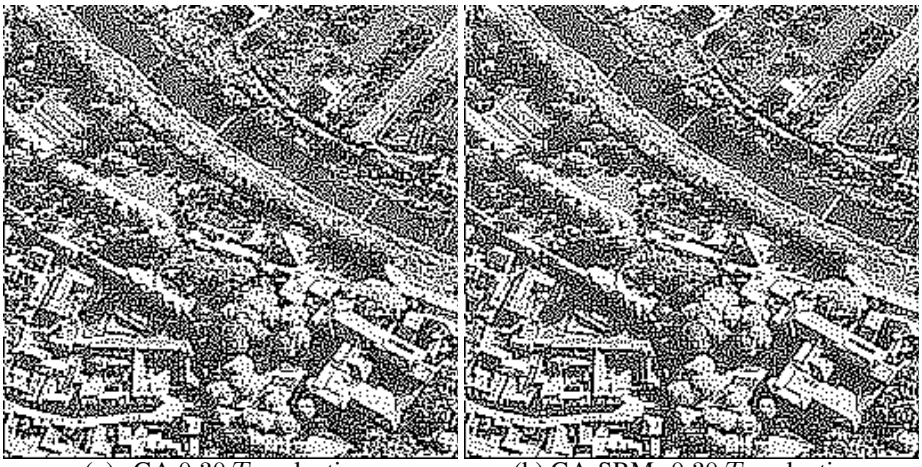
(c) GA-SRMs $0.29\,T$ evaluations (2 individuals)

(d) cGA $0.15\,T$ evaluations (200 individuals)



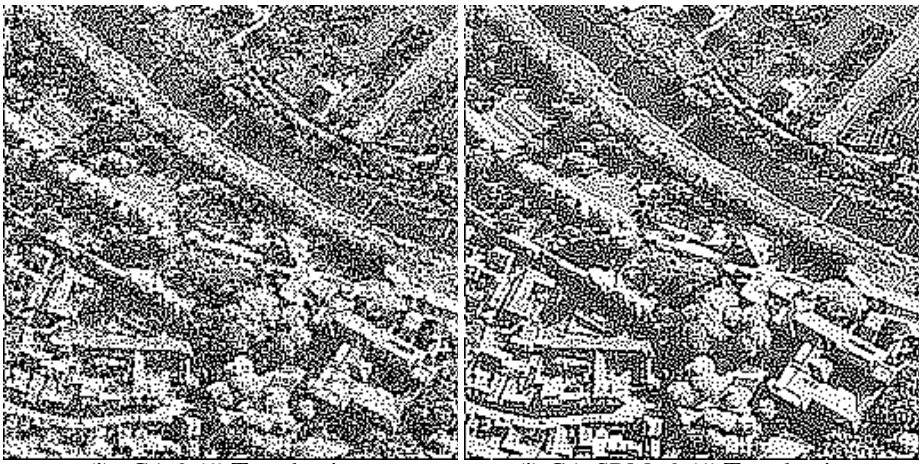
(e) GA-SRMs $0.15\,T$ evaluations (2 individuals)

(f) Original image “Aerial”



(g) cGA $0.30\,T$ evaluations (200 individuals)

(h) GA-SRMs $0.30\,T$ evaluations (2 individuals)



(i) cGA $0.15\,T$ evaluations (200 individuals)

(j) GA-SRMs $0.15\,T$ evaluations (2 individuals)

Figure 8.13: Original and generated halftone images (“Moon Surface” and “Aerial”)

Chapter 9

Simultaneous Halftone Image Generation with Multiobjective GA-SRM

The multiobjective nature of most real-world problems makes multiobjective optimization a very important research topic. In this chapter we show that the concept of GA-SRM can also be effective for multi-objective optimization of real world applications. The halftoning problem studied in the previous chapter is in fact a true multiobjective optimization problem. So far, however, the GA based halftoning techniques have treated the problem as a single objective optimization problem. Here, the improved GA-SRM is extended to a multiobjective optimization GA to simultaneously generate halftone images with various combinations of gray level precision and spatial resolution. Simulation results verify that the proposed scheme can effectively generate several high quality images simultaneously in a single run reducing even further the overall processing time.

9.1 Introduction

The multiobjective nature of most real-world problems makes multiobjective optimization (MO) a very important research topic. Evolutionary algorithms (EAs) seem particularly desirable to solve MO problems because they evolve simultaneously a population of potential solutions to the problem in hand, which allows to search for a set of Pareto optimal solutions concurrently in a single run of the algorithm. Many authors have been increasingly investigating MO using EAs in recent years and the number of applications has been rapidly growing [111, 112, 113, 114]. In the signal processing area, application methods using EAs, especially genetic algorithms (GAs), are also steadily being developed[100].

In this chapter, we especially focus on the image halftoning technique using GAs. Kobayashi et al.[102, 103] use a simple GA to generate bi-level halftone images with quality higher than conventional techniques such as ordered dithering, error diffusion and so on[104]. However, it uses a substantial amount of computer memory and processing time[102, 103]. Recently, Aguirre et al.[106, 71] have proposed an improved GA-SRM to overcome these two drawbacks of the conventional halftoning technique with GAs. GA-SRM applies varying mutation parallel to standard crossover & background mutation, putting the operators in a cooperative-competitive stand with each other[101, 34, 35, 115]. The improved GA-SRM, extended to the halftoning problem, can generate high quality images achieving a 98% reduction in the population size and an 85%-70% reduction in processing time.

The halftoning problem is a true MO problem in which high gray level precision and high spatial resolution must be sought to achieve visually high quality images. The appropriate combination of these two factors is not only device but also application dependent. Moreover, a combination that is appropriate for one image may not be the best for other, depending on the characteristics of the individual images. Hence, it is desirable to have a set of generated images where to choose from the images that best suit an application. The GA based halftoning techniques mentioned above, however, treat the problem as a single objective optimization problem and can generate only one image at a time. Thus, to generate a set of images these techniques must do it sequentially, one at the time.

In this chapter, we extend the improved GA-SRM[106, 71] to a multiobjective optimization GA and study its behavior and applicability generating simultaneously halftone images with various combinations of gray level precision and spatial resolution. We especially pay attention to the dynamics produced by the information sharing in concurrent processes and show how evolutionary multiobjective techniques, besides finding multiple solutions, can also be used as a way of accelerating the search, which is a significant issue in the halftoning problem. Another important aspect we look at is the relevance of genetic operators with complementary roles acting in parallel and the importance of using different configurations of the algorithm to search on the various landscapes of a multiobjective problem. The simulations results show that the proposed scheme can effectively generate several images in a single run reducing even further the overall processing time.

9.2 Halftoning Problem with GAs

Kobayashi et al.[102, 103] use a GA to generates bi-level halftone images with quality higher than traditional techniques such as ordered dithering, error diffusion and so on[104]. An input gray tone image of R gray levels is divided into non-overlapping blocks of $r \times r$ pixels, and then the 2-dimensional optimum binary pattern for each image block is searched using a GA[102, 103]. The GA uses an $r \times r$ 2-dimensional binary representation for the individuals. Crossover interchanges

either sets of adjacent rows or columns between two individuals and mutation inverts bits with a very small probability per bit after crossover similar to canonical GA[11, 16]. Individuals are evaluated for two factors required to obtain visually high quality halftone images. (i) One is high gray level precision (local mean gray levels close to the original image), and (ii) the other is high spatial resolution (appropriate contrast near edges)[102, 103]. The gray level precision error is calculated by

$$E_m(\boldsymbol{x}_i^{(t)}) = \frac{1}{r^2} \sum_{(j,k)\in block} \mid p(j,k) - p_b(j,k) \mid \tag{9.1}$$

where $\boldsymbol{x}_i^{(t)}$ is i-th individual at t-th generation, $p(j,k)$ is the gray level of the (j,k)-th pixel in the original image block, and $p_b(j,k)$ is the estimated gray level associated to the (j,k)-th pixel from the generated binary block. To obtain $p_b(j,k)$, a reference region around the (j,k)-th binary pixel (for example 5×5 pixels) is convoluted by a gaussian filter that models the correlation among pixels. On the other hand, the spatial resolution error is calculated by

$$E_c(\boldsymbol{x}_i^{(t)}) = \frac{1}{r^2} \sum_{(j,k)\in block} \mid \hat{p}(j,k) - \hat{q}(j,k) \mid \tag{9.2}$$

$$\hat{p}(j,k) = p(j,k) - \bar{p}(j,k) \tag{9.3}$$

$$\hat{q}(j,k) = (q(j,k) - \frac{1}{2})R \tag{9.4}$$

where $\bar{p}(j,k)$ is the local mean gray level around the (j,k)-th pixel (within a reference region) in the original image block, and $q(j,k)$ is the binary level of the (j,k)-th pixel in the generated image block. These two errors are combined into one single objective function as

$$e(\boldsymbol{x}_i^{(t)}) = \omega_m E_m(\boldsymbol{x}_i^{(t)}) + \omega_c E_c(\boldsymbol{x}_i^{(t)}) \tag{9.5}$$

where ω_m and ω_c are the weighting parameters for gray level precision and spatial resolution errors, respectively. The individuals' fitness is assigned by

$$f(\boldsymbol{x}_i^{(t)}) = e(\boldsymbol{x}_W^{(t)}) - e(\boldsymbol{x}_i^{(t)}) \tag{9.6}$$

where $e(\boldsymbol{x}_W^{(t)})$ is the combined error associated with the worst individual at t-th generation. The high image quality that can be achieved is the method's major strength. However, it uses a substantial amount of computer memory and processing time. High quality, visually satisfactory, halftone images are obtained with 200 individuals and 200 generations (totally 40,000 evaluations) per image block[102, 103].

Recently, Aguirre et al.[106, 71] have proposed an improved GA (GA-SRM) to overcome these two drawbacks of the conventional halftoning technique with GAs. GA-SRM is based on an empirical model of GA that applies genetic operators in parallel putting them in a cooperative-competitive stand with each other[101, 34, 35, 115]. GA-SRM is applied to the halftoning image problem using genetic operators properly modified for this kind of problem(see 9.4.1). GA-SRM with parallel adaptive dynamic block (ADB) mutation impressively reduces processing time and computer memory to generate high quality images. For example, GA-SRM with qualitative ADB mutation using a 2 parent 4 offspring configuration needs about 6,000-12,000 evaluations per image block, depending on the input image, to obtain results of similar quality to those achieved by the conventional image halftoning technique using GAs. These data represent a 98% reduction in the population size and an 85%-70% reduction in processing time.

9.3 Multiobjective Optimization (MO)

MO methods deal with finding optimal solutions to problems having multiple objectives. Let us consider, without loss of generality, a minimization multiobjective problem with *M* objectives:

$$minimize \ \boldsymbol{g}(\boldsymbol{x}) = (g_1(\boldsymbol{x}), \cdots, g_M(\boldsymbol{x})) \tag{9.7}$$

where $\boldsymbol{x} \in \boldsymbol{X}$ is a solution vector in the solution space $\boldsymbol{X}$, and $g_1(\cdot), \cdots, g_M(\cdot)$ the *M* objectives to be minimized. Key concepts used in determining a set of solutions for multiobjective problems are dominance, Pareto optimality, Pareto set, and Pareto front. These concepts can be defined as follows.

A solution vector $\boldsymbol{y} \in \boldsymbol{X}$ is said to *dominate* a solution vector $\boldsymbol{z} \in \boldsymbol{X}$, denoted by $\boldsymbol{g}(\boldsymbol{y}) \preceq \boldsymbol{g}(\boldsymbol{z})$, if and only if $\boldsymbol{y}$ is partially less than $\boldsymbol{z}$, i.e., $\forall m \in \{1, \cdots, M\},\ g_m(\boldsymbol{y}) \leq g_m(\boldsymbol{z}) \wedge \exists m \in \{1, \cdots, M\} : g_m(\boldsymbol{y}) < g_m(\boldsymbol{z})$.

A solution vector $\boldsymbol{x} \in \boldsymbol{X}$ is said to be *Pareto optimal* with respect to $\boldsymbol{X}$ if it is not dominated by any other solution vector, i.e., $\neg\exists\, \boldsymbol{x'} \in \boldsymbol{X} : \boldsymbol{g}(\boldsymbol{x'}) \preceq \boldsymbol{g}(\boldsymbol{x})$. The presence of multiple objectives, usually conflicting among them, gives rise to a set of optimal solutions. The Pareto optimal set is defined as:

$$P = \{\boldsymbol{x} \in \boldsymbol{X} | \neg\exists\, \boldsymbol{x'} \in \boldsymbol{X} : \boldsymbol{g}(\boldsymbol{x'}) \preceq \boldsymbol{g}(\boldsymbol{x})\} \tag{9.8}$$

and the Pareto front is defined as:

$$PF = \{\boldsymbol{g}(\boldsymbol{x}) = (g_1(\boldsymbol{x}), \cdots, g_M(\boldsymbol{x})) \, | \boldsymbol{x} \in P\} \tag{9.9}$$

The multiobjective nature of most real-world problems makes MO a very important research topic. The presence of various objectives, however, implies trade-off solutions and makes these problems complex and difficult to solve. EAs seem particularly desirable to solve MO problems because they evolve simultaneously a population of potential solutions to the problem in hand, which allows to search for a set of Pareto optimal solutions concurrently in a single run of the algorithm.

Many authors have been increasingly investigating MO using EAs (MOEA) and the number of applications has been rapidly growing. The list of contributors to the field is extensive and comprehensive reviews can be found in [111, 112, 113, 114]. Fonseca and Fleming[111] and Horn[112] examined major MOEA techniques, Coello [113] presented a MOEA review classifying implementations from a detailed algorithmic standpoint, discussing the strengths and weaknesses of each technique. Recently, Van Veldhuizen and Lamont[114] expanded upon these reviews.

9.4 Multiobjective GA-SRM for Halftoning Problem

In order to extend GA-SRM to MO for halftoning image generation[116, 117, 118] we follow a cooperative population search with aggregation selection[112, 119, 120, 121, 122]. The population is monitored for non-dominated solutions; however, Pareto based fitness assignment is not directly used. A predetermined set of weights $\boldsymbol{W}$, which ponder the multiple objectives, defines the directions that the algorithm will search simultaneously in the combined space of the multiple objectives. $\boldsymbol{W}$ is specified by

$$\boldsymbol{W} = \{\boldsymbol{\omega}^1, \boldsymbol{\omega}^2, \cdots, \boldsymbol{\omega}^N\} \tag{9.10}$$

where *N* indicates the number of search directions. The n-th search direction $\boldsymbol{\omega}^n$ is a vector of nonnegative weights specified by

$$\boldsymbol{\omega}^n = (\omega_1^n, \cdots, \omega_M^n) \tag{9.11}$$

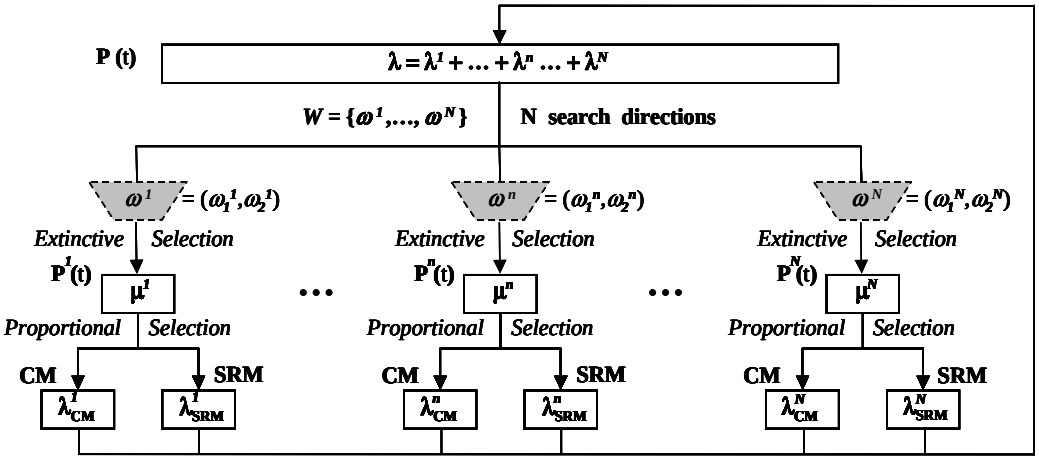


Figure 9.1: Block diagram of the extended multiobjective GA-SRM

where M indicates the number of objectives and its components satisfy the following conditions

$$\omega_m^n \geq 0 \; (m = 1, \cdots, M), \tag{9.12}$$

$$\sum_{m=1}^{M} \omega_m^n = 1. \tag{9.13}$$

We evaluate individuals for the same two factors indicated in 9.2, (number of objectives $M = 2$): (i) high gray level precision and, (ii) high spatial resolution. Here we use the same evaluation functions E_m and E_c, respectively, proposed in [102, 103] to calculate objective values and assign its normalized values to each individual as indicated by

$$g_1(\boldsymbol{x}_i^{(t)}) = \frac{100 \times (E_m(\boldsymbol{x}_i^{(t)}) - E_m^{min})}{E_m^{max} - E_m^{min}}, \tag{9.14}$$

$$g_2(\boldsymbol{x}_i^{(t)}) = \frac{100 \times (E_c(\boldsymbol{x}_i^{(t)}) - E_c^{min})}{E_c^{max} - E_c^{min}} \tag{9.15}$$

where E_m^{max}, E_m^{min}, E_c^{max}, and E_c^{min} are maximum and minimum values for E_m and E_c, respectively, obtained experimentally using various test images.

The objective values are calculated once for each individual in the offspring population. However, we keep as many fitness values as search directions have been defined. A combined objective value is calculated for each search direction $\boldsymbol{\omega}^n$ by

$$\begin{aligned} g^n(\boldsymbol{x}_i^{(t)}) &= \sum_{m=1}^{M} \omega_m^n g_m(\boldsymbol{x}_i^{(t)}) \\ &= \omega_1^n g_1(\boldsymbol{x}_i^{(t)}) + \omega_2^n g_2(\boldsymbol{x}_i^{(t)}) \end{aligned} \tag{9.16}$$

and the individuals' fitness in the *n*-th search direction is assigned by

$$f^n(\boldsymbol{x}_i^{(t)}) = g^n(\boldsymbol{x}_W^{(t)}) - g^n(\boldsymbol{x}_i^{(t)}) \tag{9.17}$$

where $g^n(\boldsymbol{x}_W^{(t)})$ is the combined objective value associated with the worse individual in the *n*-th search direction at the *t*-th generation.

For each search direction $\boldsymbol{\omega}^n$, CM creates a corresponding λ_{CM}^n number of offspring. Similarly, SRM creates λ_{SRM}^n offspring (see detailed information about CM and SRM implementation for halftoning problem in 9.4.1). Thus, the total offspring number for each search direction is

$$\lambda^n = \lambda_{CM}^n + \lambda_{SRM}^n. \tag{9.18}$$

The offspring created for all *N* search directions coexist within one single offspring population. Hence the overall offspring number is

$$\lambda = \sum_{n=1}^{N} \lambda^n. \tag{9.19}$$

SRM's mutation rates are adapted based on a normalized mutants survival ratio. The normalized mutant survival ratio used in [106, 71] is extended to

$$\gamma = \frac{\sum_{n=1}^{N} \mu_{SRM}^n}{\sum_{n=1}^{N} \lambda_{SRM}^n} \cdot \frac{\lambda}{\sum_{n=1}^{N} \mu^n} \tag{9.20}$$

where μ^n is the number of individuals in the parent population of the *n*-th search direction $P^n(t)$, μ_{SRM}^n is the number of individuals created by SRM present in $P^n(t)$ after extinctive selection, λ_{SRM}^n is the offspring number created by SRM and λ is the overall offspring number as indicated in Eq. (9.19).

We chose (μ, λ) Proportional Selection[24] to implement the extinctive selection mechanism. Since we want to search simultaneously in various directions, selection to choose the parent individuals that will reproduce either with CM or SRM is accordingly applied for each one of the predetermined search directions. Thus, selection probabilities for each search direction $\boldsymbol{\omega}^n$ are computed by

$$Prob^n(\boldsymbol{x}_i^{(t)}) = \begin{cases} \dfrac{f^n(\boldsymbol{x}_i^{(t)})}{\sum_{j=1}^{\mu^n} f^n(\boldsymbol{x}_j^{(t)})} & (1 \le i \le \mu^n \le \lambda^n) \\ 0 & (\mu^n < i \le \lambda) \end{cases} \tag{9.21}$$

where $\boldsymbol{x}_i^{(t)}$ is an individual at generation t which has the i-th highest fitness value in the n-th search direction $f^n(\boldsymbol{x}_i^{(t)})$, μ^n is the number of parents and λ^n is the number of offspring in the *n*-th search direction, and λ is the overall number of offspring.

Note that for each search direction only $\lambda^n < \lambda$ individuals are created. However, the parent population μ^n is chosen among the overall λ offspring population. In this way information sharing is encouraged among individuals created for neighboring search directions provided that the neighbors' fitness are competitive with the locals'. Figure 9.1 presents the block diagram of the extended multiobjective GA-SRM for the image halftoning problem.

Once the offspring has been evaluated, a set of non-dominated solutions is sought for each search direction, i.e. for the *n*-th search direction non-domination is checked only among the offspring created for that search direction. Two secondary populations keep the non-dominated

solutions. $P_{cur}(t)$ keeps the non-dominated solution obtained from the offspring population at generation *t* and P_{nds} keeps the set of the non-dominated solutions found through the generations. P_{nds} is updated at each generation with $P_{cur}(t)$. In the halftoning problem an image is divided into blocks and the GA is applied to each image block. Hence, the GA would generate a set of non-dominated solutions for each image block. Since we are interested in generating simultaneously various Pareto optimal "whole" images, a decision making process is integrated to choose only one solution for each search direction in each image block. Thus, among the various non-dominated solutions found for a given search direction, we choose the one that minimizes the combined error E_m and E_c in that particular direction. Figure 9.2 illustrates the algorithm to simultaneously generate N halftone images with the extended multiobjective GA-SRM.

begin
split original image in blocks
set *N* search directions $\boldsymbol{W}$=$\{(\boldsymbol{\omega}^1, \cdots, \boldsymbol{\omega}^N)\}$
for (each image block $\boldsymbol{B}_u$)
$t = 0$
initialize ($P(0)$)
mo_evaluation ($P(0)$)
while (**not** termination condition)
for (each search direction $\boldsymbol{\omega}^n$)
$P^n(t)$ = (μ, λ) proportional selection ($P(t)$)
$P(t+1)$ += *CM*($P^n(t)$)
$P(t+1)$ += *SRM* ($P^n(t)$)
done
mo_evaluation ($P(t+1)$)
get $P_{cur}(t+1)$ from $P(t+1)$
update P_{nds} with $P_{cur}(t+1)$
$t = t + 1$
done
$\boldsymbol{G}_u$= P_{nds}, keep N generated block images from $\boldsymbol{B}_u$
done
generate *N* images ($\boldsymbol{G}_u$)
end

Figure 9.2: Algorithm to simultaneously generate N halftone images with the extended multiobjective GA-SRM

9.4.1 CM and SRM for Halftoning Problem

In the halftoning problem an individual is represented as an $r \times r$ two-dimensional structure. In this work we use the same two-dimensional operators, CM (Crossover and Mutation) and SRM-ADB (Self Reproduction with Mutation - Adaptive Dynamic Block), presented in [106, 71] to create offspring.

CM first crosses over two previously selected parents interchanging either their rows or columns, similar to [102, 103], and then it applies standard mutation inverting bits with a small mutation probability per bit, $p_m^{(CM)}$, analogous to canonical GAs.

SRM, on the other hand, first creates an exact copy of a previously selected individual from the

Table 9.1: Genetic algorithms parameters

Parameter	*cGA*	*moGA*	*GA-SRM*	*moGA-SRM*
Selection	Proport.	(μ, λ) Proport.	(μ, λ) Proport.	(μ, λ) Proport.
Mating	$(\boldsymbol{x}_i, \boldsymbol{x}_j),\ i \neq j$	$(\boldsymbol{x}_i, \boldsymbol{x}_j),\ i \neq j$	$(\boldsymbol{x}_i, \boldsymbol{x}_j),\ i \neq j$	$(\boldsymbol{x}_i, \boldsymbol{x}_j),\ i \neq j$
p_c	0.6	0.6	1.0	1.0
$p_m^{(CM)}$	0.001	0.001	0.001	0.001
$\mu^n : \lambda^n$	-	$1:1$	$1:2$	$1:2$
$\lambda_{CM}^n : \lambda_{SRM}^n$	-	-	$1:1$	$1:1$

parent population and then applies mutation only to the bits inside a mutation block. SRM is provided with an Adaptive Dynamic-Block (ADB) qualitative mutation schedule similar to Adaptive Dynamic-Segment mutation (ADS)[34, 115]. This kind of mutation could take better advantage of the high correlation among contiguous pixels in an image[107], and contribute to a more effective search. See [106, 71] in detail.

9.5 Experimental Results and Discussion

9.5.1 Experimental Setup

We observe and compare the performance of four kinds of GAs generating halftone images: (i) a simple GA that uses CM and proportional selection, similar to that used in [102, 103], (denoted as cGA) (ii) an extended cGA using the same multiobjective technique described in 9.4 (denoted as moGA), (iii) a GA with SRM that uses CM, SRM and (μ, λ) proportional selection[106, 71] (denoted as GA-SRM), and (iv) the proposed extended multiobjective GA-SRM (denoted as moGA-SRM).

The GAs are applied to SIDBA's benchmark images in our simulation. The size of the original image is 256×256 pixels with 256 gray levels. An image is divided into 256 non-overlapping blocks, each one of size 16×16 pixels. For each block, the algorithms were set with different seeds for the random initial population.

We define 11 search directions, $N = 11$, setting $\boldsymbol{W} = \{\boldsymbol{\omega}^1, \boldsymbol{\omega}^2, \cdots, \boldsymbol{\omega}^{11}\} = \{(0.0, 1.0), (0.1, 0.9), \cdots, (1.0, 0.0)\}$ between E_m (gray level precision) and E_c (spatial resolution). With $\boldsymbol{\omega}^1 = (0.0, 1.0)$ the search focuses exclusively in E_c's space and with $\boldsymbol{\omega}^{11} = (1.0, 0.0)$ in E_m's; whereas with $\boldsymbol{\omega}^n$, $2 \leq n \leq 10$, the search focuses in the combined space of E_c and E_m. moGA and moGA-SRM generate simultaneously 11 images, one image for each direction, in a single run. On the other hand, to generate the 11 images with either cGA or GA-SRM an equal number of separate runs are carried out, each one using a different $\boldsymbol{\omega}^n$ as weighting parameter. Unless stated otherwise, the GAs are set with the parameters detailed in Table 9.1[1]. The values set for crossover and mutation probabilities in cGA are the same used in [102, 103]. The image quality attained by the cGA with a 200 parent population and the same $T = 4 \times 10^4$ evaluations used in [102, 103] are taken as a reference for comparison in our study. The number of generations performed for each algorithm is calculated as T/λ.

[1]GA-SRM search only in one direction at a time and the population related parameters μ^n, λ^n, λ_{CM}^n, and λ_{SRM}^n should be read without the index *n*

Table 9.2: Evaluations needed to generate high quality images by cGA(200)

(a) Lenna

Algorithm	$\boldsymbol{W} = \{\boldsymbol{\omega}^1, \boldsymbol{\omega}^2, \cdots, \boldsymbol{\omega}^{11}\}$											$T^{\mathbf{W}}$
	$\boldsymbol{\omega}^1$	$\boldsymbol{\omega}^2$	$\boldsymbol{\omega}^3$	$\boldsymbol{\omega}^4$	$\boldsymbol{\omega}^5$	$\boldsymbol{\omega}^6$	$\boldsymbol{\omega}^7$	$\boldsymbol{\omega}^8$	$\boldsymbol{\omega}^9$	$\boldsymbol{\omega}^{10}$	$\boldsymbol{\omega}^{11}$	
combined error	121.0	111.4	100.6	89.5	78.2	66.9	55.5	44.2	32.8	21.5	10.1	–
$cGA(200)$	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	$11T^2$
$moGA(18, 198)$	1.43	2.43	1.65	1.27	1.21	1.00	0.86	0.76	0.70	0.65	0.72	$2.43T^3$
$moGA(4, 44)$	1.12	2.30	1.44	1.36	1.20	1.02	0.85	0.79	0.73	0.66	0.79	$2.30T^3$
$GA\text{-}SRM(2, 4)$	0.40	0.23	0.15	0.13	0.12	0.11	0.10	0, 09	0.09	0.08	0.08	$1.58T^2$
$moGA\text{-}SRM(9, 198)$	1.12	1.07	0.58	0.44	0.30	0.27	0.24	0.23	0.22	0.21	0.21	$1.12T^3$
$moGA\text{-}SRM(2, 44)$	1.56	1.03	0.50	0.30	0.20	0.16	0.15	0.13	0.12	0.12	0.12	$1.56T^3$

(b) Girl

Algorithm	$\boldsymbol{W} = \{\boldsymbol{\omega}^1, \boldsymbol{\omega}^2, \cdots, \boldsymbol{\omega}^{11}\}$											$T^{\mathbf{W}}$
	$\boldsymbol{\omega}^1$	$\boldsymbol{\omega}^2$	$\boldsymbol{\omega}^3$	$\boldsymbol{\omega}^4$	$\boldsymbol{\omega}^5$	$\boldsymbol{\omega}^6$	$\boldsymbol{\omega}^7$	$\boldsymbol{\omega}^8$	$\boldsymbol{\omega}^9$	$\boldsymbol{\omega}^{10}$	$\boldsymbol{\omega}^{11}$	
combined error	121.9	112.7	101.7	90.3	78.8	67.3	55.8	44.3	32.8	21.3	9.7	–
$cGA(200)$	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	$11T^2$
$moGA(18, 198)$	1.84	2.14	1.54	1.15	1.04	0.85	0.78	0.68	0.62	0.64	0.64	$2.14T^3$
$moGA(4, 44)$	1.69	2.08	1.18	0.86	0.90	0.75	0.63	0.58	0.51	0.54	0.53	$2.08T^3$
$GA\text{-}SRM(2, 4)$	0.48	0.24	0.16	0.13	0.12	0.11	0.10	0.10	0.10	0.10	0.14	$1.78T^2$
$moGA\text{-}SRM(18, 198)$	1.58	1.25	0.61	0.43	0.37	0.30	0.28	0.26	0.26	0.26	0.26	$1.58T^3$
$moGA\text{-}SRM(2, 44)$	3.81	1.23	0.54	0.29	0.20	0.17	0.16	0.15	0.15	0.15	0.15	$3.81T^3$

(c) Aerial

Algorithm	$\boldsymbol{W} = \{\boldsymbol{\omega}^1, \boldsymbol{\omega}^2, \cdots, \boldsymbol{\omega}^{11}\}$											$T^{\mathbf{W}}$
	$\boldsymbol{\omega}^1$	$\boldsymbol{\omega}^2$	$\boldsymbol{\omega}^3$	$\boldsymbol{\omega}^4$	$\boldsymbol{\omega}^5$	$\boldsymbol{\omega}^6$	$\boldsymbol{\omega}^7$	$\boldsymbol{\omega}^8$	$\boldsymbol{\omega}^9$	$\boldsymbol{\omega}^{10}$	$\boldsymbol{\omega}^{11}$	
combined error	113.0	104.9	95.5	85.4	74.9	64.3	53.6	42.9	32.3	21.7	11.1	–
$cGA(200)$	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	$11T^2$
$moGA(18, 198)$	1.29	1.30	1.29	1.30	1.11	0.92	0.80	0.69	0.59	0.52	0.47	$1.30T^3$
$moGA(4, 44)$	1.15	1.37	1.02	1.18	1.14	0.85	0.69	0.59	0.50	0.42	0.37	$1.37T^3$
$GA\text{-}SRM(2, 4)$	0.37	0.26	0.20	0.17	0.15	0.13	0.11	0.10	0.09	0.09	0.14	$1.82T^2$
$moGA\text{-}SRM(18, 198)$	1.49	0.97	0.91	0.82	0.61	0.50	0.40	0.33	0.29	0.26	0.24	$1.49T^3$
$moGA\text{-}SRM(2, 44)$	1.88	1.14	1.02	0.58	0.50	0.37	0.27	0.20	0.17	0.16	0.15	$1.88T^3$

(d) Moon

Algorithm	$\boldsymbol{W} = \{\boldsymbol{\omega}^1, \boldsymbol{\omega}^2, \cdots, \boldsymbol{\omega}^{11}\}$											$T^{\mathbf{W}}$
	$\boldsymbol{\omega}^1$	$\boldsymbol{\omega}^2$	$\boldsymbol{\omega}^3$	$\boldsymbol{\omega}^4$	$\boldsymbol{\omega}^5$	$\boldsymbol{\omega}^6$	$\boldsymbol{\omega}^7$	$\boldsymbol{\omega}^8$	$\boldsymbol{\omega}^9$	$\boldsymbol{\omega}^{10}$	$\boldsymbol{\omega}^{11}$	
combined error	121.5	111.7	100.8	89.7	78.4	67.1	55.7	44.3	32.8	21.3	9.9	–
$cGA(200)$	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	$11T^2$
$moGA(18, 198)$	1.62	1.72	1.29	1.27	1.04	0.95	0.90	0.79	0.68	0.70	0.66	$1.72T^3$
$moGA(4, 44)$	1.65	1.29	1.07	1.04	0.93	0.88	0.83	0.82	0.72	0.76	0.76	$1.65T^3$
$GA\text{-}SRM(2, 4)$	0.83	0.33	0.19	0.15	0.12	0.10	0.10	0.09	0.09	0.08	0.17	$2.24T^2$
$moGA\text{-}SRM(18, 198)$	1.47	1.14	0.71	0.50	0.41	0.33	0.26	0.23	0.21	0.21	0.21	$1.47T^3$
$moGA\text{-}SRM(2, 44)$	3.74	1.57	0.54	0.40	0.26	0.21	0.17	0.14	0.13	0.13	0.12	$3.74T^3$

9.5.2 Comparison Between Simple and Multiobjective GAs

Table 9.2 presents results for four typical benchmark images: "Lenna","Girl", "Aerial", and "Moon". It shows under column $\boldsymbol{W}$ the average in all image blocks of the non-normalized combined errors $e^n(\boldsymbol{x}) = \omega_1^n E_m(\boldsymbol{x}) + \omega_2^n E_c(\boldsymbol{x})$ by cGA(200) after T evaluations for each search direction $\boldsymbol{\omega}^n$, $1 \leq n \leq 11$. For the other algorithms under $\boldsymbol{W}$ we present the fraction of T at which the algorithms reach similar image quality (for cGA(200) these values are all 1.00 and are shown right below the combined error). Column $T^{\mathbf{W}}$ indicates the overall evaluations needed to generate the 11 images. Since the cGA generates one image at a time, it needs $11T^2$ evaluations to generate all 11 images.

The first moGA row show results by the multiobjective simple GA with a $\mu^n = 18$ parents

[2]The entire number of evaluations required by the single objective GAs to generate all 11 images are given by the sum of the evaluations expended in each direction

and a $\lambda^n = 18$, $\lambda = 198$ offspring configuration. moGA simultaneously generates the 11 images and needs, depending on the input image, about $1.30T$ to $2.43T$[3] to guarantee that the images in all search direction have at least the same quality as cGA(200).

moGA's second row show results by moGA with a $\mu^n = 4$ parents and a $\lambda^n = 4$, $\lambda = 44$ offspring configuration. In this case population size reduction in moGA accelerates a little bit more the overall convergence in most of the benchmark images (only in "Aerial" convergence is slightly delayed in $\boldsymbol{\omega}^2$ search direction) needing about $1.37T$ to $2.30T$ to produce better images than cGA(200). It should be noticed that population reductions in cGA accelerates convergence but it is affected by a loss of diversity and the final image quality is inferior than cGA(200)'s[102, 103]. moGA benefits from the information sharing induced by selection (see explanation below for Figure 9.3 and Figure 9.4) and can tolerate population reductions. Compared with cGA, the results by moGA represents an enormous reduction in processing time and illustrates the benefits that can be achieved by including multiobjective techniques within GAs.

Row GA-SRM(2,4) presents results by GA-SRM with a 2 parents and 4 offspring configuration. GA-SRM even with a very scaled down population configuration considerably reduces processing time to sequentially generate high quality images for all combinations of weighting parameters in all benchmark images. Compared with the $11T$ needed by cGA, GA-SRM needs only about $1.58T$ to $2.24T$. Also, note that taking the slower overall generation time among all benchmark images, GA-SRM is faster than moGA.

The first moGA-SRM row show results by the proposed multiobjective GA-SRM with a $\mu^n = 9$ parents and a $\lambda^n = 18$, $\lambda = 198$ offspring configuration. Compared with moGA we can see that the inclusion of SRM notoriously improves the multiobjective algorithm's performance needing no more than $1.12T$ to $1.58T$[3] to generate the 11 images in all the benchmark cases, being faster than both GA-SRM and moGA. From GA-SRM and moGA-SRM results we see that parallel mutation SRM can greatly improve the performance of single objective as well as multiobjective genetic algorithms in the halftoning problem.

Results by a scaled down population configuration is shown in row moGA-SRM(2,44) that represents a $\mu^n = 2$ parents and a $\lambda^n = 4$, $\lambda = 44$ offspring configuration. The population size reduction in moGA-SRM notoriously accelerates convergence in almost all the search directions. However it delays convergence in $\boldsymbol{\omega}^1$ direction in all benchmark images making the overall evaluation time to be slower than GA-SRM and moGA (in the case of "Aerial" this effect extends to $\boldsymbol{\omega}^2$ and $\boldsymbol{\omega}^3$ and in the case of "Moon" to $\boldsymbol{\omega}^2$). This behavior deserves further analysis and we shall return to this point in 9.5.5.

9.5.3 Non-dominated Pareto Solutions

Throughout this work Pareto optimal solutions refer to strongly non-dominated solutions. Our objective is to generate a set of strongly non-dominated images (in this case, one image for each one of the $N = 11$ predefined search directions). The generation of a set of images implies a three step processes: (i) generation of non-dominated solutions, (ii) clustering the solutions around the N search directions, and (iii) selection of the preferred solution for each search direction.

In the halftoning problem the input image is divided into blocks and processed one at a time. So the above three steps process is translated to the block level. Steps (i) and (ii) are intertwined and can be accomplished following two approaches. (a) One is to check non-dominance among the overall offspring produced for all search directions, which will give us a set of non-dominated solutions per image block. However, the number of these solutions will vary from block to block

[3] In the case of multiple objective GAs, due to the concurrent search, the maximum number of the evaluations among all search directions determines the overall number of evaluations needed to generate all 11 images

Table 9.3: Obtained Pareto front (Lenna)

	Typical Image Blocks					
	B_a		B_b		*Whole Images*	
$\boldsymbol{W}$	E_m	E_c	E_m	E_c	$\bar{E}_m$	$\bar{E}_c$
$\boldsymbol{\omega^1}$	33.22	113.61	43.06	123.97	43.4	121.0
$\boldsymbol{\omega^2}$	26.67	113.71	16.48	124.35	21.0	121.3
$\boldsymbol{\omega^3}$	23.95	113.86	14.43	124.43	16.9	121.5
$\boldsymbol{\omega^4}$	16.22	114.87	13.65	124.58	12.3	122.1
$\boldsymbol{\omega^5}$	13.20	115.37	8.12	125.57	11.4	122.2
$\boldsymbol{\omega^6}$	13.19	115.41	7.86	125.72	9.8	122.7
$\boldsymbol{\omega^7}$	13.08	115.46	7.75	125.75	9.6	122.8
$\boldsymbol{\omega^8}$	10.11	118.36	7.53	125.93	9.4	123.5
$\boldsymbol{\omega^9}$	9.61	118.90	7.53	125.93	9.3	123.7
$\boldsymbol{\omega^{10}}$	9.52	119.04	7.05	126.06	9.2	123.8
$\boldsymbol{\omega^{11}}$	9.49	119.18	7.05	126.06	9.1	124.0

depending on the individual characteristics of the blocks. We observe that for some of the blocks the number of generated solutions are greater than N but for others it could be smaller than N. Thus, with this approach we obtain the non-dominated solutions but their clustering around N search directions is still unsolved and could become a problem especially on those blocks where the generated images are smaller than N. (b) Another approach is to constraint non-dominance to the offspring generated for each search direction. By doing so N non-dominated solution sets are generated for each image block (one per search direction). With this approach the problem of clustering the solutions around the N search directions is also implicitly solved. Each set covers the subspace corresponding to each search direction and it is possible that two sets overlap (that will depend on the characteristics of the images block). In other words, although within each set there are only non-dominated solutions a solution may be present in more than one set.

We use approach (b) to generate solutions. For step (iii), only one solution is chosen from each one of the N sets of non-dominated solutions. The preferred solution in a given search direction is the solution in the corresponding set of non-dominated solutions that has the minimum combined error for that search direction. Thus, we select N preferred solutions, one per search direction, in each image block. Once we have processed all image blocks, N images are output by assembling the blocks in the corresponding search directions.

In Table 9.3, under columns B_a and B_b we present the preferred solutions obtained for each search direction in two typical image blocks. Column B_a illustrates a block in which the clusters are separated one from each other and the preferred solutions also form a strongly non-dominated Pareto front. On the other hand, column B_b illustrates a block in which some clusters are very close one to another and the final preferred solution is the same in more than one search direction (see for example ω^8 and ω^9, or ω^{10} and ω^{11}). Also, from these two columns we can see that the errors' ranges vary depending on the characteristics of the image block. Under *Whole Images* we present the mean errors $\bar{E}_m$ and $\bar{E}_c$ on all image blocks of the assembled images for each search direction. We can see that in the average the proposed method induces a strongly non-dominated Pareto front for the generated images.

The closeness of the solutions is also relative to the characteristics of the individual blocks. For instance for block B_a the minimum and maximum values found for E_m and E_c are $(9.49, 40.11)$ and $(113.61, 126.71)$, respectively. Similarly, the values for B_b are $(7.05, 54.91)$ and $(123.97, 127.92)$.

Table 9.4: Actual percentage of evaluations expended in each direction by the GAs (Lenna)

	$\boldsymbol{W} = \{\boldsymbol{\omega}^1, \boldsymbol{\omega}^2, \cdots, \boldsymbol{\omega}^{11}\}$										
Algorithm	$\boldsymbol{\omega}^1$	$\boldsymbol{\omega}^2$	$\boldsymbol{\omega}^3$	$\boldsymbol{\omega}^4$	$\boldsymbol{\omega}^5$	$\boldsymbol{\omega}^6$	$\boldsymbol{\omega}^7$	$\boldsymbol{\omega}^8$	$\boldsymbol{\omega}^9$	$\boldsymbol{\omega}^{10}$	$\boldsymbol{\omega}^{11}$
$cGA(200)$	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
$moGA(18, 198)$	13.0	22.1	15.0	11.5	11.0	9.1	7.8	6.9	6.4	5.9	6.5
$moGA(4, 44)$	10.2	20.9	13.1	12.4	10.9	9.3	7.7	7.2	6.6	6.0	7.2
$GA\text{-}SRM(2, 4)$	40.0	23.0	15.0	13.0	12.0	11.0	10.0	9.0	9.0	8.0	8.0
$moGA\text{-}SRM(9, 198)$	10.2	9.7	5.3	4.0	2.7	2.5	2.2	2.1	2.0	1.9	1.9
$moGA\text{-}SRM(2, 44)$	14.2	9.4	4.5	2.7	1.8	1.5	1.4	1.2	1.1	1.1	1.1

9.5.4 Effect of Information Sharing

It should be noticed that in Table 9.2 moGA's and moGA-SRM's rows show the evaluations expended by the algorithm in all search directions. The actual percentage of the evaluations expended in each search direction is shown for "Lenna" image in Table 9.4. From this table it can be seen that with the multiobjective algorithms there is a substantial reduction of the actual number evaluations for each search direction. These reductions are explained by the effect of information sharing induced by the selection process. As mentioned in sub 9.4 and indicated by Eq. (9.21), the individuals with higher fitness in an specific direction are selected as parents. Thus, the individuals chosen to be parents for the *n*-th search direction at generation t may have been created for neighboring directions at generation t-1. In order to verify this point we also observe the composition of the parent population for each search direction. Figure 9.3 and Figure 9.4 show the average distribution for some of the $\boldsymbol{\omega}^n$ directions after $0.1T$ and T evaluations for "Lenna" image, respectively. For example, in Figure 9.3, the parent population of $\boldsymbol{\omega}^4$ is in average composed by 18% of individuals coming from $\boldsymbol{\omega}^3$, 30% from $\boldsymbol{\omega}^4$ itself, and 13% from $\boldsymbol{\omega}^5$. From these figures we can see that each search direction benefits from individuals that initially were meant for other neighboring directions. This information sharing pushes forward the search reducing convergence times. Looking at Figure 9.3 and Figure 9.4 we can see that the effect of information sharing is higher during the initial stages of the search.

Figure 9.5 illustrates typical transitions of the non-normalized combined error $e(\boldsymbol{x})$ over the number of evaluations for some of the search directions by the GAs. The plots are for "Lenna" and are cut after T evaluations. From this figures it can be visually appreciated the higher convergence velocity and higher convergence reliability (lower errors) by the algorithms that include SRM, GA-SRM and moGA-SRM. In general, moGA is faster than the cGA, but their final image quality tends to be the same. Also, it should be noticed that results by moGA and moGA-SRM are achieved simultaneously in one run (thus, T for these algorithms indicates the evaluations expended in all search directions).

9.5.5 Dynamic Configurations for Further Improvement

In this subsection we carefully study the behavior of moGA-SRM(2,44) and propose dynamic configurations for the algorithm in order to improve its robustness in all search directions with less overall evaluation time for the various kinds of test images.

First, we observe that moGA(2,44), which uses CM but not SRM, only for $\boldsymbol{\omega}^1$ produces faster convergence than moGA-SRM(2,44) ($e^1 = 0.0E_m + 1.0E_c$) in the four test images (only in "Moon" this is also true for $\boldsymbol{\omega}^2$). It seems that CM alone is particularly useful for searching in E_c's search space. However, when the search involves both E_m's and E_c's spaces the interaction of CM and SRM produces by far better results. To confirm this point we conduct an experiment in which we favor CM's offspring over SRM's only in the $\boldsymbol{\omega}^1$ direction, i.e. $\lambda^1_{CM} = 4$, $\lambda^1_{SRM} = 0$

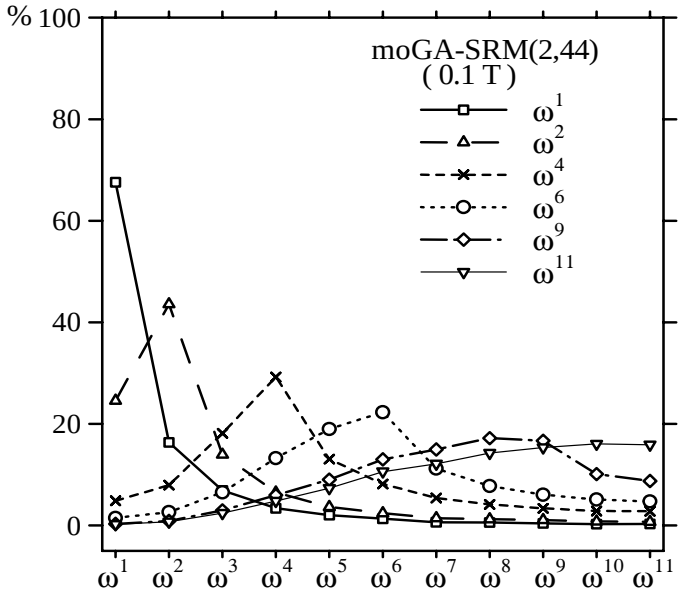


Figure 9.3: moGA-SRM's average parent population distribution after $0.1T$ (Lenna)

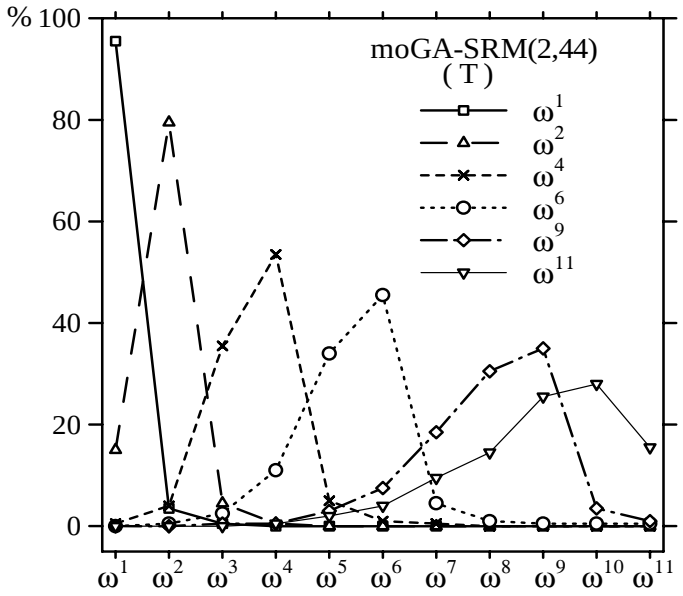


Figure 9.4: moGA-SRM's average parent population distribution after T evaluations (Lenna)

and $\lambda^n_{CM} = 2$, $\lambda^n_{SRM} = 2$ for $2 \leq n \leq 11$. Results by this configuration are shown in row moGA-SRM*(2,44) of Table 9.5 (for an easy comparison we also include results by moGA-SRM(2,44)). As expected, this has the effect of accelerating convergence in $\boldsymbol{\omega}^1$. But, it also has a negative impact in the neighboring $\boldsymbol{\omega}^2$ search direction. As mentioned above, the interaction of CM and SRM performs very well in the combined search space of E_c and E_m and by using CM alone in $\boldsymbol{\omega}^1$ we are also reducing the amount of information that can be effectively shared with the neighboring $\boldsymbol{\omega}^2$ search direction, which is located precisely in the combined space of E_c and E_m.

Second, as a general tendency we see that for directions closer to E_m the algorithms need

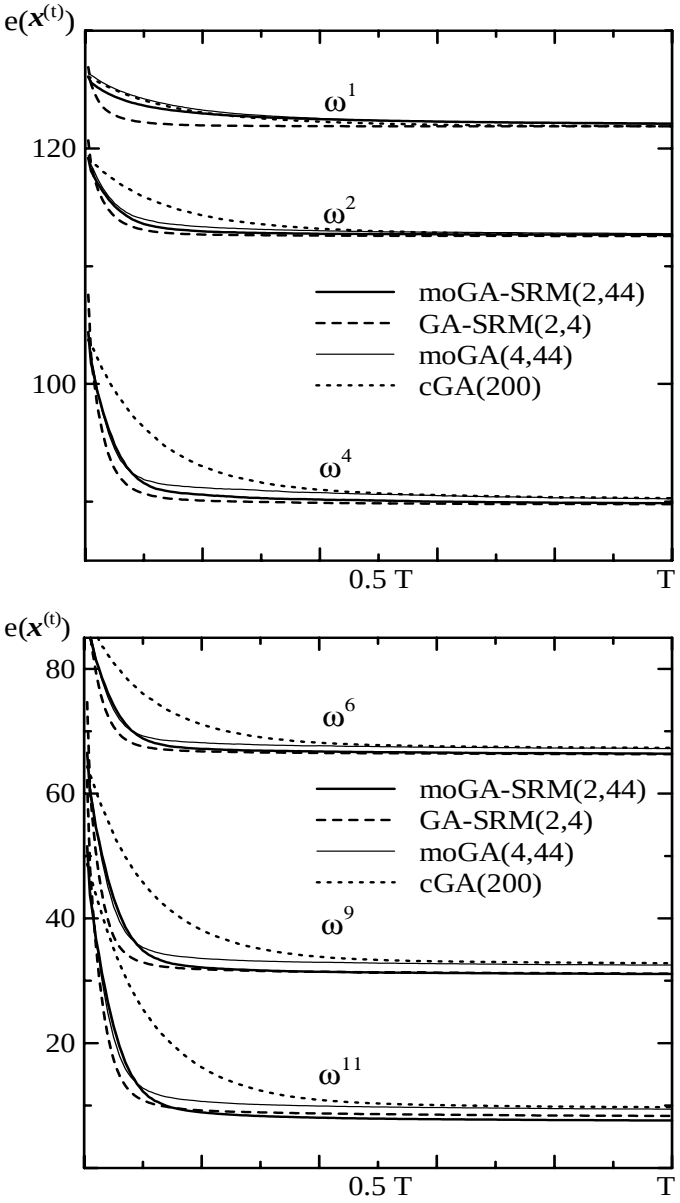


Figure 9.5: Error transition for various ω^n (Lenna)

less time to converge. It should be especially noticed that for $\boldsymbol{\omega}^n$ with $n \geq 8$, $0.2T$ is enough to find high quality images by moGA-SRM. This suggests that dynamic configurations that assign more evaluations to those directions that require more time to converge could reduce the overall time to convergence. Row moGA-SRM^D(2,44) in Table 9.5 shows results by an algorithm that at $0.2T$ stops searching for $\boldsymbol{\omega}^8, ..., \boldsymbol{\omega}^{11}$, at $0.35T$ stops searching for $\boldsymbol{\omega}^6, \boldsymbol{\omega}^7$, and from $0.50T$ focuses only in $\boldsymbol{\omega}^1, \boldsymbol{\omega}^2, \boldsymbol{\omega}^3$ search directions. That is, moGA-SRM^D(2,44) starting with a moGA-SRM(2,44) ($n \leq 11$) configuration gradually reconfigures itself to moGA-SRM(2,28) ($n \leq 7$), moGA-SRM(2,20) ($n \leq 5$), and moGA-SRM(2,12) ($n \leq 3$). As expected, a better assignment of evaluations helps to significantly reduce the overall time to convergence.

Finally, we combine these two explored approaches to set a robust algorithm that can work

Table 9.5: Reduced evaluations to generate high quality images by moGA-SRM(2,44) with dynamic configurations

(a) Lenna

Algorithm $moGA\text{-}SRM$	$\boldsymbol{W} = \{\boldsymbol{\omega}^1, \boldsymbol{\omega}^2, \cdots, \boldsymbol{\omega}^{11}\}$											$T^{\mathbf{W}}$
	$\boldsymbol{\omega}^1$	$\boldsymbol{\omega}^2$	$\boldsymbol{\omega}^3$	$\boldsymbol{\omega}^4$	$\boldsymbol{\omega}^5$	$\boldsymbol{\omega}^6$	$\boldsymbol{\omega}^7$	$\boldsymbol{\omega}^8$	$\boldsymbol{\omega}^9$	$\boldsymbol{\omega}^{10}$	$\boldsymbol{\omega}^{11}$	
$(2, 44)$	1.56	1.03	0.50	0.30	0.20	0.16	0.15	0.13	0.12	0.12	0.12	$1.56T^3$
$^*(2, 44)$	0.88	1.53	0.39	0.30	0.22	0.17	0.14	0.13	0.12	0.12	0.12	$1.53T^3$
$^D(2, 44)$	0.72	0.55	0.36	0.26	0.21	0.17	0.14	0.13	0.12	0.12	0.12	$0.72T^3$
$^{D*}(2, 44)$	0.59	0.57	0.40	0.28	0.21	0.17	0.14	0.13	0.13	0.12	0.12	$0.59T^3$

(b) Girl

Algorithm $moGA\text{-}SRM$	$\boldsymbol{W} = \{\boldsymbol{\omega}^1, \boldsymbol{\omega}^2, \cdots, \boldsymbol{\omega}^{11}\}$											$T^{\mathbf{W}}$
	$\boldsymbol{\omega}^1$	$\boldsymbol{\omega}^2$	$\boldsymbol{\omega}^3$	$\boldsymbol{\omega}^4$	$\boldsymbol{\omega}^5$	$\boldsymbol{\omega}^6$	$\boldsymbol{\omega}^7$	$\boldsymbol{\omega}^8$	$\boldsymbol{\omega}^9$	$\boldsymbol{\omega}^{10}$	$\boldsymbol{\omega}^{11}$	
$(2, 44)$	3.81	1.23	0.54	0.29	0.20	0.17	0.16	0.15	0.15	0.15	0.15	$3.81T^3$
$^*(2, 44)$	1.24	1.76	0.46	0.33	0.24	0.17	0.16	0.15	0.15	0.15	0.15	$1.76T^3$
$^D(2, 44)$	1.53	0.61	0.31	0.28	0.21	0.17	0.16	0.15	0.14	0.14	0.14	$1.53T^3$
$^{D*}(2, 44)$	0.68	0.62	0.39	0.26	0.21	0.18	0.16	0.15	0.15	0.15	0.15	$0.68T^3$

(c) Aerial

Algorithm $moGA\text{-}SRM$	$\boldsymbol{W} = \{\boldsymbol{\omega}^1, \boldsymbol{\omega}^2, \cdots, \boldsymbol{\omega}^{11}\}$											$T^{\mathbf{W}}$
	$\boldsymbol{\omega}^1$	$\boldsymbol{\omega}^2$	$\boldsymbol{\omega}^3$	$\boldsymbol{\omega}^4$	$\boldsymbol{\omega}^5$	$\boldsymbol{\omega}^6$	$\boldsymbol{\omega}^7$	$\boldsymbol{\omega}^8$	$\boldsymbol{\omega}^9$	$\boldsymbol{\omega}^{10}$	$\boldsymbol{\omega}^{11}$	
$(2, 44)$	1.88	1.14	1.02	0.58	0.50	0.37	0.27	0.20	0.17	0.16	0.15	$1.88T^3$
$^*(2, 44)$	1.01	1.94	0.84	0.66	0.50	0.37	0.26	0.21	0.17	0.16	0.15	$1.94T^3$
$^D(2, 44)$	0.98	0.53	0.51	0.43	0.36	0.30	0.32	0.20	0.17	0.15	0.14	$0.98T^3$
$^{D*}(2, 44)$	0.63	0.58	0.50	0.43	0.32	0.29	0.31	0.20	0.17	0.15	0.14	$0.63T^3$

(d) Moon

Algorithm $moGA\text{-}SRM$	$\boldsymbol{W} = \{\boldsymbol{\omega}^1, \boldsymbol{\omega}^2, \cdots, \boldsymbol{\omega}^{11}\}$											$T^{\mathbf{W}}$
	$\boldsymbol{\omega}^1$	$\boldsymbol{\omega}^2$	$\boldsymbol{\omega}^3$	$\boldsymbol{\omega}^4$	$\boldsymbol{\omega}^5$	$\boldsymbol{\omega}^6$	$\boldsymbol{\omega}^7$	$\boldsymbol{\omega}^8$	$\boldsymbol{\omega}^9$	$\boldsymbol{\omega}^{10}$	$\boldsymbol{\omega}^{11}$	
$(2, 44)$	3.74	1.57	0.54	0.40	0.26	0.21	0.17	0.14	0.13	0.13	0.12	$3.74T^3$
$^*(2, 44)$	1.02	2.00	0.51	0.39	0.28	0.21	0.16	0.13	0.12	0.12	0.12	$2.00T^3$
$^D(2, 44)$	1.95	0.60	0.41	0.31	0.24	0.19	0.16	0.14	0.12	0.12	0.12	$1.95T^3$
$^{D*}(2, 44)$	0.64	0.61	0.39	0.31	0.25	0.19	0.16	0.14	0.12	0.12	0.12	$0.64T^3$

effectively in all search directions for different kinds of test images. moGA-SRM^{D*}(2,44) row in Table 9.5 shows results by an algorithm that: (i) reconfigures itself as moGA-SRM^{D}(2,44) and (ii) from $0.50T$ it searches in $\boldsymbol{\omega}^1$ direction using only CM, i.e. $\lambda^1_{CM} = 4$, $\lambda^1_{SRM} = 0$. In other words, at the beginning we keep CM and SRM in $\boldsymbol{\omega}^1$ direction so that neighboring directions could take advantage of the effect of information sharing, which is more intense at the beginning of the search, and later we use only CM to improve $\boldsymbol{\omega}^1$ convergence. With moGA-SRM^{D*}(2,44) at most $0.70T$ evaluations are needed to simultaneously generate 11 images. Figure 9.6, similar to Figure 9.5, illustrates typical transitions of the non-normalized combined error $e(\boldsymbol{x})$ for some of the search directions by moGA-SRM^{D*}(2,44) running for $0.70T$ and include results by the other GAs.

In this study we take as a reference the image quality obtained by a canonical GA (cGA) and the number of evaluations it expends. We decide about T based on these results. The values of the fraction of T at which the algorithms moGA-SRM^{D}(2,44) and moGA-SRM^{D*}(2,44) reconfigure themselves ($0.2T$, $0.35T$ and $0.5T$) are upper bounds obtained experimentally for which high performance (at least that achieved by cGA) was observed in all test images. Lower values work fine and lead to better results in some of the images, such as "Lenna" or "Girl" in the examples presented here, but those lower values are not effective for other images. Thus these values of T are not optimum values and although several kinds of test images were tried its generality is

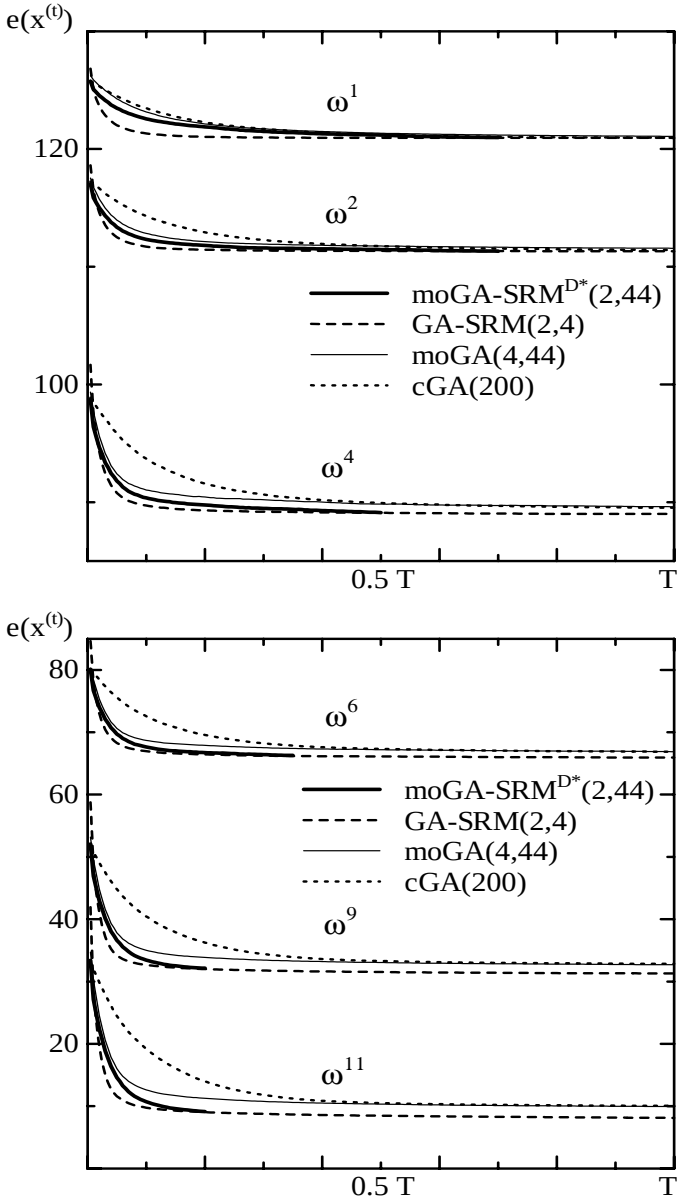


Figure 9.6: Error transition for various ω^n using $GA\text{-}SRM^{D*}(2,44)$ (Lenna) running for $0.70T$(Lenna)

restricted. The optimality of T is a very important issue and should be investigated further. For example, to achieve more optimum and robust T it might be worth pursuing an approach that would decide the stopping time based on the progress that the algorithm achieves reducing the errors. However, this implies taking adaptation at the block level and rises other issues that will need to be addressed. Also, the canonical GA cannot be a reference any more and we will need another way of comparison. We are planning to undertake these topics in future works.

Figure 9.7, Figure 9.8, Figure 9.9, and Figure 9.10 show some of the original benchmark images and some of the simultaneously generated images by moGA-SRM^{D*}(2,44) after $0.70T$. As can be observed, the images for each search direction are high quality images and the difference

in contrast and gray level precision can be visually appreciated.

9.6 Conclusions

In this chapter we have extended the improved GA-SRM to a multiobjective optimization GA (moGA-SRM) for the image halftoning problem with the aim of simultaneously generating halftone images with various combinations of gray level precision and spatial resolution.

GA-SRM is based on an empirical model of GA that puts parallel genetic operators in a cooperative-competitive stand with each other. To extend GA-SRM we followed a cooperative population search with aggregation selection preserving the fundamental features of the cooperative-competitive model. We compared the performance of four genetic algorithms generating halftone images: (i) a single objective simple GA (cGA), (ii) a single objective GA-SRM, (iii) a multiobjective simple GA (moGA), (iv) the proposed multiobjective GA-SRM (moGA-SRM).

From our experimental results we have observed that multiobjective techniques benefit from the effect of information sharing and can greatly reduce processing time to simultaneously generate high quality images. To generate 11 images moGA requires only about 21% of the evaluations used by cGA. The cooperative-competitive model for parallel operators helps to improve the performance of single and multiobjective GAs in this problem reducing even further processing time. GA-SRM requires about 20% and moGA-SRM using a simple dynamic configuration needs at most 6.4% of the evaluations used by cGA.

The optimality and generality of the number of evaluations needed to generate high quality images is a very important issue and should be investigated further. As future works, it might be worth pursuing higher levels of adaptations in GA-SRM. For example, an algorithm that adapt its configuration and decides the stopping time based on the characteristics of each image block. Also, the application of GA-SRM to other real-world problems should be investigated.

original

ω^1

ω^2

ω^3

ω^4

ω^6

$\boldsymbol{\omega^9}$ $\boldsymbol{\omega^{11}}$

Figure 9.7: Lennas's original and simultaneously generated halftone images by moGA-SRM^{D*}(2,44) after $0.70T$

original

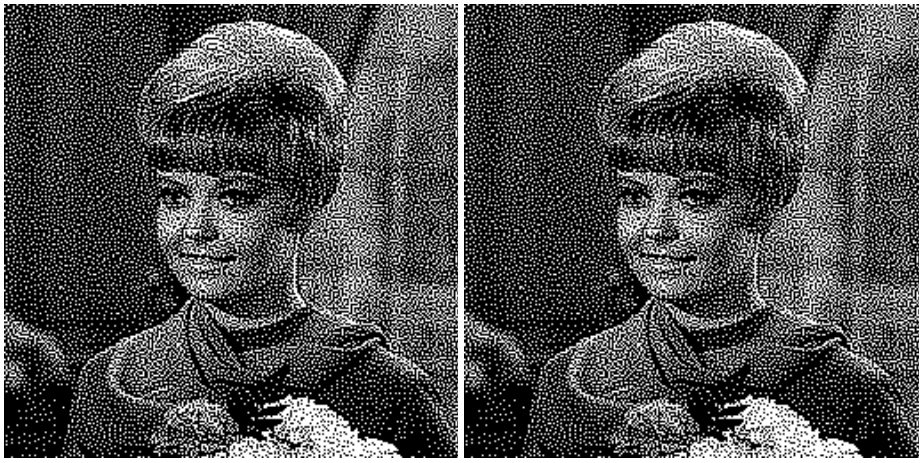
ω^1

ω^2

ω^3

ω^4

ω^6



$\boldsymbol{\omega^9}$ $\boldsymbol{\omega^{11}}$

Figure 9.8: Girls's original and simultaneously generated halftone images by moGA-SRM^{D*}(2,44) after $0.70T$

original

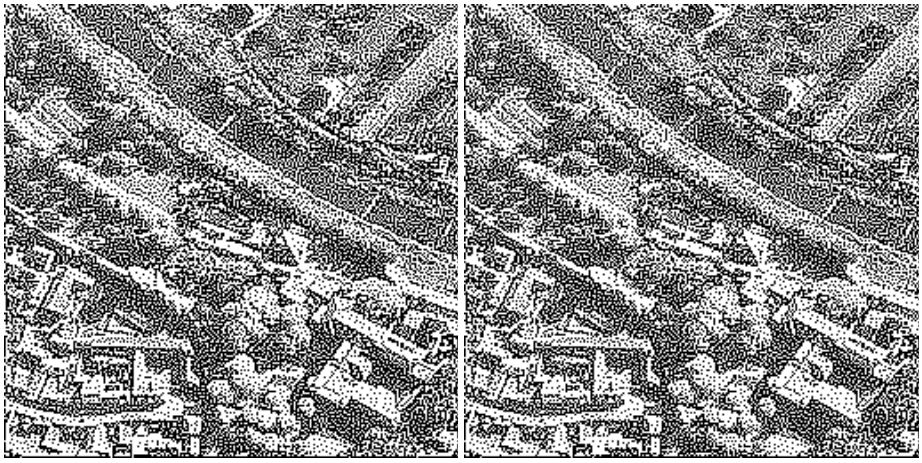
ω^1

ω^2

ω^3

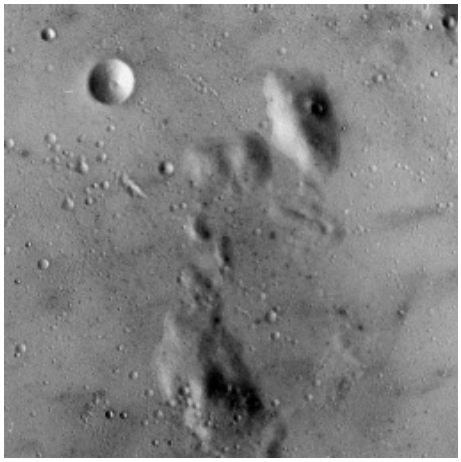
ω^4

ω^6

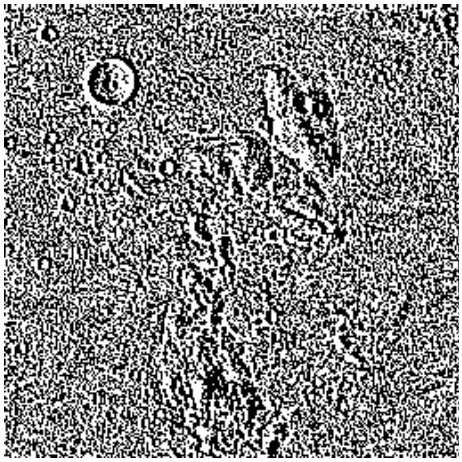


$\boldsymbol{\omega^9}$ $\boldsymbol{\omega^{11}}$

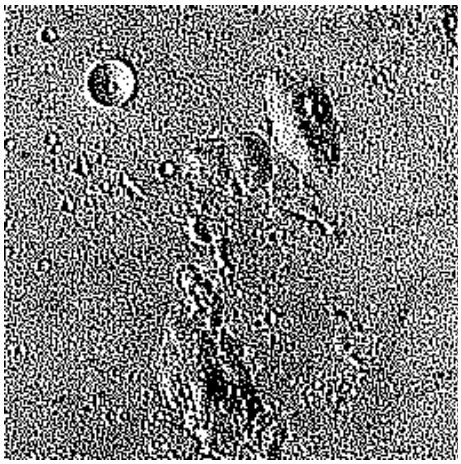
Figure 9.9: Aerial's original and simultaneously generated halftone images by moGA-SRM^{D*}(2,44) after $0.70T$



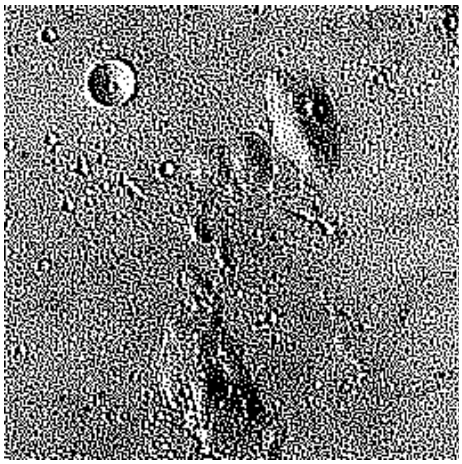
original



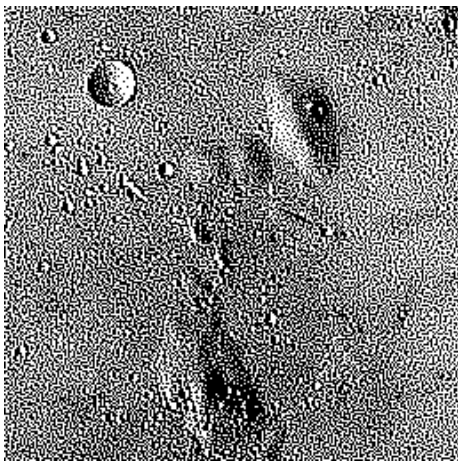
ω^1



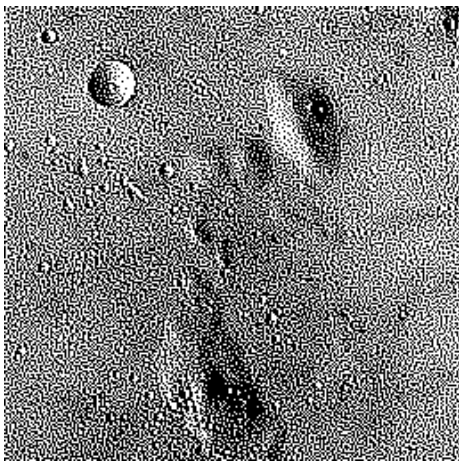
ω^2



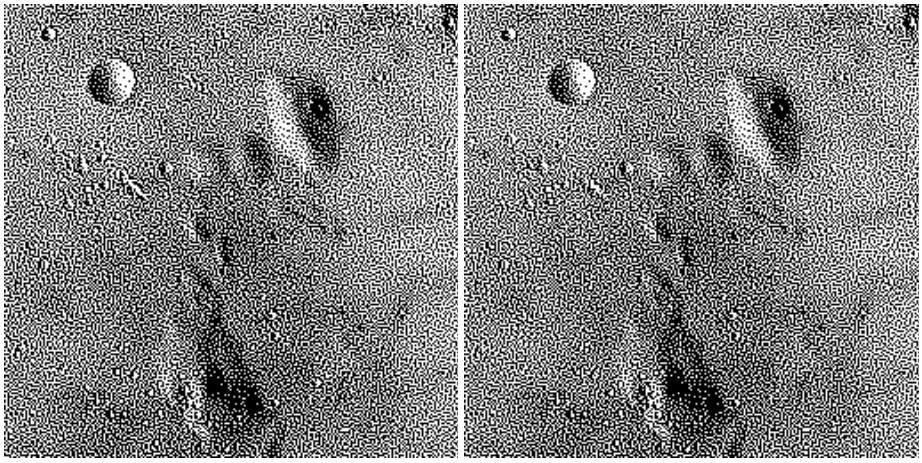
ω^3



ω^4



ω^6



$\boldsymbol{\omega^9}$ $\boldsymbol{\omega^{11}}$

Figure 9.10: Moon's original and simultaneously generated halftone images by moGA-SRM^{D*}(2,44) after $0.70T$

Chapter 10

Conclusions

The standard model of varying mutation GAs raises several important questions regarding the interference between crossover and high mutation, how this affects performance of the algorithm, whether this affect the mutation rate control itself in the case of (adaptive) self-adaptive varying mutation algorithms, and more generally whether this is an appropriate model for combining forms of control (co-adaptation of strategy parameters).

This work has proposed a model of parallel varying mutation GA that addresses the questions raised by the standard model of varying mutation GAs. The proposed model detaches varying mutation from crossover, applies "background" mutation (or none at all) after crossover (CM) and varying mutations (SRM) only parallel to crossover, putting the operators CM and SRM in a cooperative-competitive stand with each other by subjecting their offspring to extinctive selection. The model relies in adaptive (self-adaptive) mutation schedules to increase the effectiveness of SRM and enhance selection by eliminating fitness duplicates, which postpones genetic drift and creates a fair competition between the offspring created by both operators.

It was argued that there are several advantages in the proposed model. First, it gives an efficient framework to achieve better balances for varying mutation and crossover in which the strengths of the operators can be kept without interfering one with the other. Second, since varying mutation is detached from crossover, the instantaneous effectiveness of the varying mutation operator depends only upon itself and its relative success can be directly tied to the mutation rate to create adaptive (self-adaptive) schemes for mutation rate control. The same can be said for crossover, especially if no "background" mutation is applied after it. Third, parallel varying mutation can be studied on its own seeking to increase the performance of GAs. Fourth, the individual roles and the interaction of crossover and varying mutation throughout the run of the algorithm can be better understood, which could be important for co-adaptation studies.

The model was studied using two test problem generators. One of generators was for 0/1 multiple knapsack problems, which allowed testing the model on a broad range of classes of constrained problems by varying the feasible region of the search space, number of constraints, and the size of the search space. Real-world 0/1 multiple knapsack problems with known global optimum were also used. These latter problems allowed studying the global search abilities of the algorithms. The second generator was the well known Kauffman's NK-Landscapes, which allowed testing the model in a broad range of classes of epistatic non-linear problems.

First, the internal structure of the proposed model (GA-SRM) was studied in depth using and adaptive schedule for mutation. Important structural issues studied were the balance for offspring creation between CM and SRM, the ratio between number of parents and number of offspring (extinctive selection pressure), "background" mutation probability in CM, and the threshold to trigger adaptation in SRM. The effect of population size and number of evaluations was observed,

too. Two mutation strategies to select the bits that will undergo mutation were investigated. In addition, the importance and effect on performance of extinctive selection and the interaction of varying mutation parallel to crossover was assessed. We found that that GA-SRM greatly improves the performance of GAs for global optimization in constrained problems. Extinctive selection accelerated the search process and parallel varying mutation increased the convergence reliability of the algorithm. Robustness even with small populations was also a remarkable characteristics observed in the improved GA-SRM. Mutation strategy for parallel varying mutation turned out to be an important issue to improve the performance of parallel varying mutation.

Second, the proposed model was compared with the standard model of varying mutations GAs across a broad range of problems. The statistical significance of the results was verified with ANOVA tests. It was found that the proposed model is more effective and efficient than the standard model. In deterministic varying mutation GAs, a GA with varying mutation parallel to crossover showed faster convergence and higher robustness to initial settings of mutation rate than a GA with varying mutation serial to crossover. In self-adaptive varying mutation GAs, the convergence velocity of a parallel self-adaptive mutation GA was matched by a serial self-adaptive mutation GA only when initial diversity of parameters was allowed. Convergence reliability was higher for the parallel varying self-adaptive mutation GA in both deterministic and self-adaptive GAs. It was also found that the standard model of varying mutations in fact affects negatively the (adaptive) self-adaptive mutation rate control. This strongly suggested that the standard model of varying mutation GAs may not be appropriate for combining forms of control.

Then, the behavior of the parallel varying mutation GA-SRM was examined on epistatic problems using NK-Landscapes. Properties of NK-Landscapes were discussed and the effect on performance of selection, drift, mutation, and recombination was verified. Mutation strategy for the varying mutation operator was also studied in detail. Experiments were conducted using NK-Landscapes with nearest neighbor and random patterns of epistasis. Comparisons were made with a canonical GA, a simple GA with extinctive selection, a mutation only EA, and a random bit climber RBC+. It was shown that GAs can be robust algorithms on NK-Landscapes postponing drift by eliminating fitness duplicates and using selection pressure higher than a canonical GA. Different to previous works, even simple GAs with these two features performed better than the single bit climber RBC+ for a broad range of classes of problems. It was also shown that the interaction of parallel varying mutation with crossover (GA-SRM) improves further the reliability of the GA. Contrary to intuition it was found that a mutation only EA can perform as well as GA-SRM that includes crossover for small values of K, where crossover is supposed to be advantageous; but the relative importance of crossover interacting with varying mutation increased with K performing better than mutation alone for medium and high K. Better overall performance by population based mutation only evolutionary algorithms over random bit climbers was also observed. With regards to mutation strategy for parallel varying mutation, it was found that a dynamic segment mutation strategy improves the performance of GAs on problems with nearest neighbor patterns of epistasis.

After analyzing the model and comparing with standard model of varying mutation GAs, it was shown that the fundamental concept of the model can be successfully extended to other important classes of GAs and that can be effectively applied to real-world problems. An important area of research is the parallelization of GAs. Evolutionary algorithms are population based methods and it is considered that its full potential would come from implementing the algorithm in parallel architectures. It was shown that the proposed model extended to a parallel distributed GA (DGA-SRM) achieves higher search speed, higher convergence reliability, and less communication cost for migration than a canonical distributed GA. It was also shown that DGA-SRM scales up better as the difficulty of the problem increases and tolerates population reductions better than a canonical

distributed GA.

Next, it was verified that GA-SRM can be successfully applied to real world problems in which efficiency in processing time and computer memory is a major issue. The improved GA-SRM was extended to the two dimensional image halftoning problem and an accelerated image halftoning technique with tiny populations was presented. It was shown that the proposed scheme impressively reduced computer memory and processing time to generate high quality images, making the improved approach appealing for practical implementation.

Finally, the multiobjective nature of most real-world problems makes multiobjective optimization a very important research topic. It was shown that the concept of GA-SRM can also be effective for multi-objective optimization of real world applications. The improved GA-SRM was extended to a multiobjective optimization GA to simultaneously generate halftone images with various combinations of gray level precision and spatial resolution. Simulation results verified that the proposed scheme can effectively generate several high quality images simultaneously in a single run reducing even further the overall processing time.

As future works, it would be worth pursuing co-adaptation in GA-SRM and investigating forms to increase its performance in problems that exhibit strong epistasis. Also, we would like to extend the concept of GA-SRM to cellular GAs, which is another important class of parallel GAs. The application of the improved GA-SRM to other imaging problems, and to domains that present serious challenges to optimization techniques, such as bioinformatics, should be investigated. In addition, in this work we have shown that the concept of parallel varying mutation can be successfully extended to parallel distributed GAs and multiobjective GAs as well. A next step would be to compare the standard model with the parallel model of varying mutation in these classes of GAs, in order to asses the impact or benefit of one model over the other. This is important because parallel and multiobjective GAs are very different from single population GAs and what is known for single population GAs can not be directly inferred for parallel and multiobjective GAs.

Acknowledgements

A large number of people contributed to the completion of this work. I thank my supervisors Prof. Kiyoshi Tanaka and Prof. Shinjiro Oshita, Prof. Tatsuo Sugimura, the thesis committee members Prof. Kiyohito Yamasawa, Prof. Yasunari Shidama, Prof. Masayuki Okamoto, and Prof. Masayuki Nakamura, and the Faculty at Shinshu University for their support and advice. Prof. Kiyoshi Tanaka's guidance and comments have contributed significantly to the thesis. Naturally, this work has been influenced by many other researchers as well. I am especially grateful to the anonymous reviewers for their valuable comments and insights. My gratitude to César Esquetini and Vinicio Baquero who have always supported and encouraged my curiosity for science and engineering, to the SSB, and to Diego Andrade, Rafael Melgarejo, and Alfredo von Reckow at Pontifícia Universidad Católica del Ecuador who kindly help me to keep a tight bond with Ecuador. I also thank the Takemasa family, my host family in Tokyo, and Mr. Kimiaki Takizawa at Shinshu University for their kind support throughout these years. I gratefully acknowledge the financial support provided by the Government of Japan through a MONBUSHO Research Scholarship from April 1997 to March 2003. Finally, thanks to my family for being just the way they are.

Bibliography

[1] G. E. P. Box. Evolutionary operation: A method of increasing industrial productivity. *Applied Statistics*, 6:81–101, 1957.

[2] A. S. Fraser. Simulation of genetic systems by automatic digital computers I: Introduction. *Australian Journal of Biological Science*, 10:484–491, 1957.

[3] G. J. Friedman. Digital simulation of an evolutionary process. *General Systems Yearbook*, 4:171–184, 1959.

[4] W. W. Bledsoe. The use of biological concepts in the analytical study of systems. In *Paper presented at ORSA-TIMS National Meeting*, San Francisco, 1961.

[5] H. J. Bremermann. Optimization through evolution and recombination. In M.C Yovits, G. T. Jacobi, and G. D. Goldstein, editors, *Self-Organizing Systems*. Spartan Books, 1962.

[6] J. Reed, R. Toombs, and N. A. Barricelli. Simulation of biological evolution and machine learning. *Journal of Theoretical Biology*, (17):319–342, 1967.

[7] N. A. Barricelli. Symbiogenetic evolution process realized by artificial methods. *Methodos*, (35-36):143–182, 1957.

[8] N. A. Barricelli. Numerical testing of evolution theories. *ACTA Biotheorica*, (16):69–126, 1962.

[9] A. S. Fraser. Simulation of genetic systems by automatic digital computers II: Effects of linkage on rates of advance under selection. *Australian Journal of Biological Science*, 10:492–499, 1957.

[10] F. G. Martin and C. C. Cockerham. High speed selection studies. In O. Kempthorne, editor, *Biometrical Genetics*. Pergamon, 1960.

[11] John H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.

[12] I. Rechenberg. *Evolutionsstrategie: Optimierung Technischer Systeme nach Prinzipien der Biologischen Evolution*. Frommann-Holzboog, 1973.

[13] H.-P. Schwefel. *Numerical Optimization of Computer Models*. Wiley, Chichester, UK, 1981.

[14] L. Fogel, A. Owens, and M. Walsh. *Artificial Intelligence Through Simulated Evolution*. Wiley, New York, 1966.

[15] T. Bäck, D. B. Fogel, and Z. Michalewicz, editors. *Handbook of Evolutionary Computation*. Institute of Physics Publishing and Oxford University Press, New York and Bristol, 1997.

[16] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading. Addison-Wesley, 1989.

[17] K. A. De Jong. *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*. PhD thesis, University of Michigan, 1975.

[18] J. J. Grefenstette. Optimization of control parameters for genetic algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*, (16):122–128, 1986.

[19] J. Schaffer, R. A. Caruana, L. J. Eshelman, and R. Das. A study of control parameters affecting online performance of genetic algorithms for function optimization. In Schaffer [123], pages 51–60.

[20] H. Muhlenbein. How genetic algorithms really work: Mutation and hillclimbing. In *Proc. 2nd Int. Conf. on Parallel Problem Solving From Nature*, pages 15–25, 1992.

[21] T. Bäck. The interaction of mutation rate, selection, and self-adaptationwithin a genetic algorithm. In *Proc. 2nd Int. Conf. on Parallel Problem Solving From Nature*, pages 85–94, 1992.

[22] T. Bäck. Optimal mutation rates in genetic search. In Forrest [124], pages 2–8.

[23] M. Yanagiya. A simple mutation-dependent genetic algorithm. In Forrest [124], page 659.

[24] T. Bäck. *Evolutionary Algorithms in Theory and Practice*. Oxford University Press, 1996.

[25] A. E. Eiben, R. Hinterding, and Z. Michalewicz. Parameter control in evolutionary algorithms. *IEEE Transactions on Evolutionary Algorithms*, 3(2):121–141, July 1996.

[26] L. Davis. Adapting operators probabilities in genetic algorithms. In Schaffer [123], pages 61–69.

[27] B. A. Julstrom. What have you done for me lately? Adapting operator probabilities in a steady state genetic algorithms. In L. J. Eshelman, editor, *Proc. 6th Int'l Conf. on Genetic Algorithms*. Morgan Kaufmann, 1995.

[28] W. Spears. Adapting crossover in evolutionary algorithms. In *Proc. of 4th Annu. Conf. Evolutionary Programming*, pages 367–384, Cambridge, 1995. MA:MIT Press.

[29] T. C. Fogarty. Varying the probability of mutation in the genetic algorithm. In Schaffer [123], pages 104–109.

[30] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, third revised and extended edition, 1996.

[31] T. Bäck and M. Schutz. Intelligent mutation rate control in canonical genetic algorithms. In *Lecture Notes on Artificial Intelligence*, volume 1079, pages 158–167. Springer, 1996.

[32] J. Smith and T. C. Fogarty. Self adaptation of mutation rates in a steady state genetic algorithm. In Banzhaf and Reeves [125].

[33] Thomas Bäck. *Handbook of Evolutionary Computation*, chapter Self-adaptation, pages C7.1:1–13. In Bäck et al. [15], 1997.

[34] H. Aguirre, K. Tanaka, and T. Sugimura. Cooperative model for genetic operators to improve GAs. In *Proc. IEEE Int'l Conf. on Information, Intelligence, and Systems*, pages 98–106, 1999.

[35] H. Aguirre, K. Tanaka, T. Sugimura, and S. Oshita. Cooperative-competitive model for genetic operators: Contributions of extinctive selection and parallel genetic operators. In *Proc. Late Breaking Papers Genetic and Evolutionary Computation Conference*, pages 6–14. Morgan Kaufmann, 2000.

[36] W. Spears. Crossover or mutation? In L. Darrell Whitley, editor, *Foundations of Genetic Algorithms 2*, pages 221–237. Morgan Kaufmann, 1993.

[37] A. Wu, R. Lindsay, and R. Riolo. Empirical observations on the roles of crossover and mutation. In Bäck [126], pages 362–369.

[38] W. Spears. *The Role of Mutation and Recombination in Evolutionary Algorithms*. PhD thesis, George Mason University, 1998.

[39] W. Spears. *Evolutionary Algorithms: The Role of Mutation and Recombination*. Springer-Verlag, 2000.

[40] H. Aguirre, K. Tanaka, and S. Oshita. Parallel cooperative-competitive self-adaptive mutation in genetic algorithms. In *Proc. IEEE International Conference on Systems, Man, and Cybernetics (SMC2001)*, pages 2343–2348, 2001.

[41] D. Whitley. The genitor algorithm and selection pressure: Why rank-based allocation of reproductive trials is best. In Schaffer [123], pages 116–121.

[42] S. Martello and P. Toth. *Knapsack Problems: Algorithms and Computer Implementations*. John Wiley & Sons, Chichester, West Sussex, England, 1990.

[43] M.R. Garey and S.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, San Francisco, 1979.

[44] A. Lokketangen and F. Glover. *Surrogate Constraint Analysis - New Heuristics and Learning Schemes for Satisfiability Problems*, volume 35 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, chapter Satisfiability Problem: Theory and Applications. 1977.

[45] B. Gavish and H. Pirkul. Allocation of databases and processors in a distributed computing system. *Management of Distributed Data Processing*, pages 215–231, 1982.

[46] W. Shih. A branch and bound method for the multiconstraint zero-one knapsack problem. *Journal of the Operational Research Society*, (30):369–378, 1979.

[47] P.C. Gilmore and R.E. Gomory. The theory and computation of knapsack functions. *Operations Research*, (14):1045–1075, 1966.

[48] K. Bertet, C. Chaudet, I. Guérin Lassous, and L. Viennot. Impact of interferences on bandwidth reservation for ad hoc networks: a first theoretical study. In *Proc. The IEEE Symposium on Ad-Hoc Wireless Networks (GLOBECOM SAWN'2001)*, volume 5. IEEE, 2001.

[49] C. C. Petersen. Computational experience with variants of the balas algorithm applied to the selection of r&d projects. *Managment Science*, (13):736–750, 1967.

[50] S. Senyu and Y. Toyoda. An approach with linear programming with 0-1 variables. *Managment Science*, (15):B196–B207, 1967.

[51] H. M. Weingartner and D. N. Ness. Methos for the solution of the multidimensional 0/1 knapsack problem. *Operations Research*, (15):83–103, 1967.

[52] S. Khuri, T. Bäck, and J. Heitkotter. The zero/one multiple knapsack problem and genetic algorithms. In *Proc. 1994 ACM Symp. on Applied Computing*, pages 188–193. ACM Press, 1994.

[53] P.C. Chu and J. E. Beasley. A genetic algorithm for the multidimensional knapsack problem. *Journal of Heuristics*, 4:63–86, 1998.

[54] A. Freville and G. Plateau. An efficient preprocessing procedure for the multidimensional 0-1 knapsack problem. *Discrete Applied Mathemathics*, (49):189–212, 1994.

[55] H. Aguirre, K. Tanaka, T. Sugimura, and S. Oshita. Improved distributed genetic algorithm with cooperative-competitive genetic operators. In *Proc. IEEE Int'l Conf. on Systems, Man, and Cybernetics (SMC 2000)*, pages 3816–3822, 2000.

[56] G. Liepins J. T. Richardson, M. R. Palmer and M. Hilliard. Some guidelines for genetic algorithms with penalty functions. In Schaffer [123], pages 191–197.

[57] S. A. Kauffman. *Lectures in the Sciences of Complexity*, volume 1, chapter Adaptation on Rugged Fitness Landscapes, pages 527–618. Addison Wesley, 1989.

[58] S. A. Kauffman. *The Origins of Order: Self-Organization and Selection in Evolution*. Oxford University Press, 1993.

[59] L. Altenberg. *Handbook of Evolutionary Computation*, chapter Fitness Landscapes: NK Landscapes, pages B2.7:5–10. In Bäck et al. [15], 1997.

[60] L. Altenberg. Evolving better representations through selective genome growth. In *Proc. 1st IEEE Conf. on Evolutionary Computation*, pages 182–187, NJ:IEEE, 1994. Pistcaway.

[61] R. Heckendorn, S. Rana, and D. Whitley. Test function generators as embedded landscapes. In Banzhaf and Reeves [125], pages 183–198.

[62] R. E. Smith and J. E. Smith. An examination of tunable, random search landscapes. In Banzhaf and Reeves [125], pages 165–182.

[63] R. E. Smith and J. E. Smith. New methods for tunable, random landscapes. In Martin and Spears [127], pages 47–67.

[64] K. A. De Jong, M. A. Potter, and W. M. Spears. Using problem generators to explore the effects of epistasis. In Bäck [126], pages 338–345.

[65] Y. Davidor. Epistasis variance: A viewpoint of GA-hardness. In Rawlins [128], pages 23–35.

[66] B. Manderick, M. de Weger, and P. Spiessens. The genetic algorithm and the structure of the fitness landscape. In Belew and Booker [129], pages 143–150.

[67] J. E. Smith. *Self Adaptation in Evolutionary Algorithms*. PhD thesis, University of the West of England, Bristol, 1998.

[68] K. E. Mathias, L. J. Eshelman, and D. Schaffer. Niches in NK-landscapes. In Martin and Spears [127], pages 27–46.

[69] M. Shinkai, H. Aguirre, and K. Tanaka. Performance study of improved GA on epistatic problems with NK-landscapes. In *Proc. IASTED Int'l. Conf. on Artificial Intelligence and Soft Computing (ASC 2001)*, pages 237–242, 2001.

[70] H. Aguirre and K. Tanaka. Genetic algorithms on NK-landscapes: Effects of selection, drift, mutation, and recombination. In *Proc. Third European Workshop on Evolutionary Computation (EvoCOP 2003)*, volume 2611 of *Lecture Notes in Computer Science*. Springer-Verlag, 2003. to appear.

[71] H. Aguirre, K. Tanaka, and T. Sugimura. Accelerated image halftoning technique using improved genetic algorithm. *IEICE Trans.*, (8):1566–1574, August 2000.

[72] H. Ishibashi, H. Aguirre, K. Tanaka, and T. Sugimura. Multi-objective optimization with improved genetic algorithm. In *Proc. IEEE Int'l Conf. on Systems, Man, and Cybernetics (SMC 2000)*, pages 3852–3857, 2000.

[73] M. Shinkai, H. Aguirre, and K. Tanaka. Mutation strategy improves GA's performance on epistatic problems. In *Proc. 2002 IEEE World Congress on Computational Intelligence*, pages 795–800, 2002.

[74] H. Aguirre and K. Tanaka. Parallel varying mutation genetic algorithms. In *Proc. 2002 IEEE World Congress on Computational Intelligence*, pages 795–800, 2002.

[75] H. Aguirre and K. Tanaka. Modeling efficient parallel varying mutation genetic algorithms. In *Proc. Workshop Program 2002 Genetic and Evolutionary Computation Conference*, pages 256–259, 2002.

[76] H. Aguirre and K. Tanaka. Parallel varying mutation in deterministic and self-adaptive GAs. In *Proc. Seventh International Conference on Parallel Problem Solving from Nature (PPSN 2002)*, volume 2439 of *Lecture Notes in Computer Science*, pages 111–121. Springer-Verlag, 2002.

[77] L. J. Eshelman. The CHC adaptive search algorithm: How to have a save search when engaging in nontraditional genetic recombination. In Rawlins [128], pages 265–283.

[78] L. Davis. Bit-climbing, representational bias, and test suite design. In Belew and Booker [129], pages 18–23.

[79] H. Aguirre, K. Tanaka, T. Sugimura, and S. Oshita. Increasing the robustness of distributed genetic algorithms by parallel cooperative-competitive genetic operators. In L. Spector, E. D. Goodman, A. Wu, W.B. Langdon, H-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. H. Garzon, and E. Bruke, editors, *Proc. Genetic and Evolutionary Computation Conference*, pages 195–202. Morgan Kaufmann, 2001.

[80] L. Eshelman and D. Schaffer. Preventing premature convergence in genetic algorithms by preventing incest. In Belew and Booker [129], pages 115–122.

[81] D. Schaffer, M. Mani, L. Eshelman, and K. Mathias. The effect of incest prevention on genetic drift. In Banzhaf and Reeves [125], pages 235–243.

[82] K. H. Liang, X. Yao, C. Newton, and D. Hoffman. Solving cutting stock problems by evolutionary programming. In *Evolutionary Programming VII: Proc. of the Seventh Annual Conference on Evolutionary Programming (EP98)*, volume 1447 of *Lecture Notes in Computer Science*, pages 291–300. Springer-Verlag, 1998.

[83] J. He and X. Yao. From an individual to a population: An analysis of the first hitting time of population-based evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 6(5):495–511, 2002.

[84] C.R. Reeves and J.E. Rowe. *Genetic Algorithms - Principles and Perspectives*. Kluwer, Norwell, MA, 2002.

[85] W. N. Martin, J. Lienig, and J. P. Cohoon. *Handbook of Evolutionary Computation*, chapter Island (migration) models: Evolutionary algorithms based on punctuated equilibria, pages C.6.3:1–16. In Bäck et al. [15], 1997.

[86] S. Forrest. Emergent computation: Self-organizing, collective, and cooperative phenomena in natural and artificial computing networks. *Physica*, D42:1–11, 1991.

[87] V. S. Gordon and D. Whitley. Serial and parallel genetic algorithms as function optimizers. In Forrest [124], pages 177–183.

[88] S. C. Lin, W. Punch, and E. Goodman. Coarse-grain parallel genetic algorithms: Categorization and new approach. In *6th IEEE Symposium on Parallel and Distributed Processing*. IEEE Computer Society Press, 1994.

[89] E. Cantú-Paz. A survey of parallel genetic algorithms. *Calculateurs Paralleles, Reseaux et Systems Repartis*, 10(2):141–171, 1998.

[90] R. Hausser and R. Manner. Implementation of standard genetic algorithm on mimd machines. In *Parallel Problem Solving from Nature III*, pages 504–513. Springer-Verlag, 1994.

[91] B. Manderick and P. Spiessens. Fine-grained parallel genetic algorithms. In Schaffer [123], pages 428–433.

[92] H. Muhlenbein. Parallel genetic algorithms, population genetics and combinatorial optimization. In Schaffer [123], pages 416–421.

[93] C. B. Pettey, M. R. Leuze, and J. J. Grefenstette. A parallel genetic algorithm. In Grefenstette [130], pages 155–161.

[94] R. Tanese. Parallel genetic algorithms for a hypercube. In Grefenstette [130], pages 177–183.

[95] E. Cantú-Paz. Topologies, migration rates, and multi-population parallel genetic algorithms. In Banzhaf et al. [131], pages 91–98.

[96] H. Aguirre, K. Tanaka, and S. Oshita. Performance study of a distributed genetic algorithm with parallel cooperative-competitive genetic operators. *IEICE Trans.*, (9):2083–2088, September 2002.

[97] E. Cantú-Paz. Migration policies, selection pressure, and parallel evolutionary algorithms. In S. Brave and A. S. Wu, editors, *Proc. Late Breaking Papers Genetic and Evolutionary Computation Conference*, pages 65–73. Morgan Kaufmann, 1999.

[98] J. E. Beasley. Obtaining test problems via internet. *Journal of Global Optimization*, 8:429–433, 1996.

[99] D. B. Fogel. *Evolutionary Computation : Toward a New Philosophy of Machine Intelligence*. IEEE Press, 1995.

[100] K. S. Tang, K. F. Man, S. Kwong, and Q. He. Genetic algorithms and their applications. *IEEE Signal Processing Magazine*, 13(6):22–37, June 1996.

[101] H. Aguirre, K. Tanaka, and T. Sugimura. Cooperative crossover and mutation operators in genetic algorithms. In Banzhaf et al. [131], page 772.

[102] N. Kobayashi and H. Saito. Halftoning technique using genetic algorithm. In *Proc. IEEE ICASSP'94*, volume 5, pages 105–108, April 1994.

[103] N. Kobayashi and H. Saito. Halftone algorithm using genetic algorithm. *IEICE Trans.*, J78-D-II(10):1450–1459, October 1995. (in Japanese).

[104] R. Ulichney. *Digital Halftoning*. MIT Press, Cambridge, 1987.

[105] H. Aguirre, K. Tanaka, and T. Sugimura. Efficient halftoning scheme using improved genetic algorithm. In *Proc. IMPS'99*, pages 73–74, 1999.

[106] H. Aguirre, K. Tanaka, and T. Sugimura. Accelerated halftoning technique using improved genetic algorithm with tiny populations. In *Proc. IEEE Int'l Conf. on Systems, Man, and Cybernetics (SMC 1999)*, volume IV, pages 905–910, October 1999.

[107] R. C. Gonzalez and R. E. Wood. *Digital Image Processing*. Addison-Wesley, 1992.

[108] T. Bäck and F. Hoffmeister. Extended selection mechanism in genetic algorithms. In Belew and Booker [129], pages 92–99.

[109] T. Umemura, H. Aguirre, and K. Tanaka. Multi-level image generation with genetic algorithm. In *Proc. Second WSEAS International Conference on Applied and Theoretical Mathematics and Computer Science*, pages 202–207, 2001.

[110] T. Umemura, H. Aguirre, and K. Tanaka. Multi-level image halftoning technique with genetic algorithm. *IEICE Trans.*, (8):1892–1897, August 2002.

[111] C. Fonseca and P. Fleming. An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary Computation*, 3(1):1–16, 1995.

[112] J. Horn. *Handbook of Evolutionary Computation*, chapter Multicriterion Decision Making, pages F1.9:1–F1.9:15. In Bäck et al. [15], 1997.

[113] C. Coello. A comprehensive survey of evolutionary-based multiobjective optimization techniques. *Knowledge and Information Systems*, 1(3):269–308, 1999.

[114] D. Van Veldhuizen and G. Lamont. Multiobjective evolutionary algorithms: Analyzing the state-of-the-art. *Evolutionary Computation*, 8(2):125–147, 2000.

[115] H. Aguirre, K. Tanaka, and T. Sugimura. Empirical model with cooperative-competitive genetic operators to improve GAs: Performance investigation with 0/1 multiple knapsack problems. *IPSJ Journal*, 41(10):2837–2851, October 2000.

[116] H. Aguirre, K. Tanaka, T. Sugimura, and S. Oshita. Simultaneous halftone image generation with improved multiobjective genetic algorithm. *IEICE Trans.*, (8):1869–1882, August 2001.

[117] H. Aguirre, K. Tanaka, T. Sugimura, and S. Oshita. Halftone image generation with improved muiltiobjective genetic algorithm. In *Proc. First Int'l Conf. on Evolutionary Multi-Criterion Optimization*, volume 1993 of *Lecture Notes in Computer Science*, pages 501–515. Springer-Verlag, 2001.

[118] H. Aguirre, K. Tanaka, T. Sugimura, and S. Oshita. Dynamic configurations for simultaneous halftone image generation with multiobjective genetic algorithm. In *Proc. 2001 IEEE EURASIP Workshop on Nonlinear Signal and Image Processing*, 2001.

[119] J. Schaffer and J. Grefenstette. Multiobjective optimization with vector evaluated genetic algorithms. In Grefenstette [132], pages 93–100.

[120] M. Foruman. Compaction of symbolic layout using genetic algorithms. In Grefenstette [132], pages 141–153.

[121] F. Kurwase. A variant of evolution strategies for vector optimization. In *Lecture Notes in Computer Science*, volume 496, pages 193–197. Springer-Verlag, 1991.

[122] T. Murata and H. Ishibuchi. MOGA: Multi-objective genetic algorithms. In *Proc. 2nd IEEE Int. Conf. on Evolutionary Computation*, pages 289–294, 1995.

[123] J. D. Schaffer, editor. *Proc. 3th Int'l Conf. on Genetic Algorithms*. Morgan Kaufmann, 1989.

[124] S. Forrest, editor. *Proc. 5th Int'l Conf. on Genetic Algorithms*. Morgan Kaufmann, 1993.

[125] Wolfggang Banzhaf and Colin Reeves, editors. *Foundations of Genetic Algorithms 5*. Morgan Kaufmann, 1999.

[126] T. Bäck, editor. *Proc. 7th Int'l Conf. on Genetic Algorithms*.

[127] Worthy N. Martin and William Spears, editors. *Foundations of Genetic Algorithms 6*. Morgan Kaufmann, 2001.

[128] G. J. E. Rawlins, editor. *Foundations of Genetic Algorithms*. Morgan Kaufmann, 1991.

[129] R. K. Belew and L. Booker, editors. *Proc. 4th Int'l Conf. on Genetic Algorithms*. Morgan Kaufmann, 1991.

[130] J. J. Grefenstette, editor. *Proc. 2th Int'l Conf. on Genetic Algorithms*. Lawrence Erlbaum Assoc., 1987.

[131] W. Banzhaf, J. Daida, A. E. Eiben, M. Garzon, V. Honavar, M. Jakiela, and R. E. Smith, editors. *Proc. Genetic and Evolutionary Computation Conference*. Morgan Kaufmann, 1999.

[132] J. J. Grefenstette, editor. *Proc. 1st Int'l Conf. on Genetic Algorithms*. Lawrence Erlbaum Assoc., 1985.

Publications

Journals

1. H. Aguirre, K. Tanaka and S. Oshita, "Performance Study of a Distributed Genetic Algorithm with Parallel Cooperative-Competitive Genetic Operators", *IEICE Trans.*, vol.E85-A, no.9, pp.2083-2088, Sep. 2002.

2. T. Umemura, H. Aguirre and K. Tanaka, "Multi-level Image Halftoning Technique with Genetic Algorithm", *IEICE Trans.*, vol.E85-A, no.8, pp.1892-1897, Aug. 2002.

3. H. Aguirre, K. Tanaka, T. Sugimura and S. Oshita, "Simultaneous Halftone Image Generation with Improved Multiobjective Genetic Algorithm", *IEICE Trans.*, vol.E84-A, no. 8, pp.1869-1882, Aug. 2001.

4. H. Aguirre, K. Tanaka and T. Sugimura, "Accelerated Image Halftoning Technique Using Improved Genetic Algorithm", *IEICE Trans.*, vol.E83-A, no. 8, pp.1566-1574,Aug. 2000.

5. H. Aguirre, K. Tanaka and T. Sugimura, "Empirical Model with Cooperative-Competitive Genetic Operators to Improve GAs: Performance Investigation with 0/1 Multiple Knapsack Problems", *IPSJ Journal*, vol.41, no.10, pp.2837-2851, Oct. 2000.

International Conferences

1. H. Aguirre and K. Tanaka, "Genetic Algorithms on NK-landscapes: Effects of Selection, Drift, Mutation, and Recombination", *Proc. Third European Workshop on Evolutionary Computation (EvoCOP 2003)*, Lecture Notes in Computer Science (Springer), vol. 2611, to appear April 2003.

2. H. Aguirre and K. Tanaka, "Parallel Varying Mutation in Deterministic and Self-Adaptive GAs", *Proc. Seventh International Conference on Parallel Problem Solving from Nature*, Lecture Notes in Computer Science (Springer), vol.2439, pp.111-121, Sep. 2002.

3. H. Aguirre and K. Tanaka, "Modeling Efficient Parallel Varying Mutation Genetic Algorithms", *Proc. Workshop Program 2002 Genetic and Evolutionary Computation Conference*, pp. 256-259, July 2002.

4. H. Aguirre and K. Tanaka, "Parallel Varying Mutation Genetic Algorithms", *Proc. 2002 IEEE World Congress on Computational Intelligence*, pp.795-800, May 2002.

5. M. Shinkai, H. Aguirre and K. Tanaka, "Mutation Strategy Improves GA's Performance on Epistatic Problems", *Proc. 2002 IEEE World Congress on Computational Intelligence*, pp.968-973, May 2002.

6. T. Umemura, H. Aguirre, and K. Tanaka, "Multi-level Image Generation with Genetic Algorithm", *Proc. Second WSEAS International Conference on Applied and Theoretical Mathematics and Computer Science*, pp.202-207, Dec. 2001.

7. H. Aguirre, K. Tanaka and S. Oshita, "Parallel Cooperative-Competitive Self-Adaptive Mutation in Genetic Algorithms", *Proc. IEEE International Conference on Systems, Man, and Cybernetics (SMC2001)*, pp.2343-2348, Oct. 2001.

8. H. Aguirre, K. Tanaka, T. Sugimura and S. Oshita, "Increasing the Robustness of Distributed Genetic Algorithms by Parallel Cooperative-Competitive Genetic Operators", *Proc. 2001 Genetic and Evolutionary Computation Conference*, pp.195-202, July 2001.

9. H. Aguirre, K. Tanaka, T. Sugimura and S. Oshita, "Dynamic Configurations for Simultaneous Halftone Image Generation with Multiobjective Genetic Algorithm", *Proc. 2001 IEEE-EURASIP International Workshop on Nonlinear Signal and Image Processing*, June 2001.

10. M. Shinkai, H. Aguirre and K. Tanaka, "Performance Study of Improved GA on Epistatic Problems with NK-Landscapes", *Proc. IASTED International Conference on Artificial Intelligence and Soft Computing (ASC 2001)*, pp.237-242, May 2001.

11. H. Aguirre, K. Tanaka, T. Sugimura and S. Oshita, "Halftone Image Generation with Improved Multiobjective Genetic Algorithm", *Proc. First International Conference on Evolutionary Multi-Criterion Optimization (EMO'01)*, Lecture Notes in Computer Science (Springer), vol.1993, pp.501-515, March 2001.

12. H. Aguirre, K. Tanaka, T. Sugimura and S. Oshita, "Improved Distributed Genetic Algorithm with Cooperative-Competitive Genetic Operators", *Proc. IEEE International Conference on Systems, Man, and Cybernetics (SMC2000)*, pp. 3816-3822, Oct. 2000.

13. H. Ishibashi, H. Aguirre, K. Tanaka and T. Sugimura, "Multi-objective Optimization with Improved Genetic Algorithm", *Proc. IEEE International Conference on Systems, Man, and Cybernetics (SMC2000)*, pp.3852-3857, Oct. 2000.

14. H. Aguirre, K. Tanaka, T. Sugimura and S. Oshita, "Cooperative-Competitive Model for Genetic Operators: Contributions of Extinctive Selection and Parallel Genetic Operators", *Proc. Late Braking Papers at 2000 Genetic and Evolutionary Computation Conference*, pp.6-14, July 2000.

15. H. Aguirre, K. Tanaka, and T. Sugimura, "Cooperative Model for Genetic Operators to Improve GAs", *Proc. IEEE International Conference on Information, Intelligence, and Systems*, pp. 98-106, Nov. 1999.

16. H. Aguirre, K. Tanaka, and T. Sugimura, "Accelerated Halftoning Technique Using Improved Genetic Algorithm with Tiny Populations", *Proc. IEEE International Conference on Systems, Man, and Cybernetics (SMC99)*, vol. IV, pp. 905-910, Oct. 1999.

17. H. Aguirre, K. Tanaka, and T. Sugimura, "Cooperative Crossover and Mutation Operators in Genetic Algorithms", *Proc. Genetic and Evolutionary Computation Conference*, p.772, July 1999.

Domestic Conferences (in Japan)

1. N. Kogawa, H. Aguirre and K. Tanaka, "A Study on Performance Improvement of Cellular-Type EC", *Proc. 12th Fuzzy, Artificial Intelligence, Neural Networks and Soft Computing*, pp.7-10, Nov. 2002.

2. E. Myodo, H. Aguirre and K. Tanaka, "Performance Improvement of an Image Halftoning Technique Using GAs by a Modified Evaluation Method", *Proc. 12th Fuzzy, Artificial Intelligence, Neural Networks and Soft Computing*, pp.311-316, Nov. 2002.

3. T. Umemura, H. Aguirre and K. Tanaka, "A Study on the Extension of Halftone Image Generation with GA to Multi-Level One", *Proc. IEICE Image Media Processing Symposium*, pp.33-34, Nov. 2001.

4. H. Aguirre and K. Tanaka, "The Effect of Parallel Cooperative-Competitive Genetic Operators in the Performance of Distributed GAs", *Proc. Technical Report of IEICE*, AI2001-4, May 2001.

5. M. Shinkai, H. Aguirre and K. Tanaka, "Analytic Study on the Behavior of GAs for Generalized NK-Landscape Problems", *Proc. Technical Report of IEICE*, AI2001-3, May 2001.

6. H. Aguirre, K. Tanaka, T. Sugimura and S. Oshita, "Performance Study of Improved Distributed Genetic Algorithm in 0/1 Multiple Knapsack Problem", *Proc. 10th Fuzzy, Artificial Intelligence, Neural Networks and Soft Computing*, pp.447-450, Oct. 2000.

7. M. Shinkai, K. Tanaka, H. Aguirre and T. Sugimura, "Epistasis Effect in the Performance of Improved GA with NK-Landscape Problem", *Proc. 10th Fuzzy, Artificial Intelligence, Neural Networks and Soft Computing*, pp.451-454, Oct. 2000.

8. H. Aguirre, K. Tanaka and T. Sugimura, "Performance Study of Improved Genetic Algorithm with Cooperative Genetic Operators", *Proc. Technical Report of IEICE*, AI2000-13, May 2000.

9. H. Ishibashi, H. Aguirre, K. Tanaka and T. Sugimura, "Solving Multi-Objective Flowshop Scheduling Problem by Improved Genetic Algorithm", *Proc. Technical Report of IEICE*, AI2000-14, May 2000.

10. H. Aguirre, K. Tanaka, and T. Sugimura, "Efficient Halftoning Scheme Using Improved Genetic Algorithm", *Proc. IEICE Image Media Processing Symposium*, pp.73-74, Sep. 1999.

11. H. Aguirre, K. Tanaka, and T. Sugimura, "Genetic Algorithm Using SRM and its Application to the 0/1 Multiple Knapsack Problem", *Proc. Fuzzy, 8th Artificial Intelligence, Neural Networks and Soft Computing*, pp.273-278, Oct. 1998.