An Analysis of Decentralized and Spatially Distributed Genetic Algorithms

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

by

Jayshree A. Sarma

M.Com. Bombay University, India
MS, Information Systems, George Mason University

Director: Kenneth A. De Jong, Associate Professor
Department of Computer Science

Spring Semester 1998
George Mason University
Fairfax, Virginia

DEDICATION

To my parents – Seetha and Ramaswamy

To my children – Ganesh and Ranjani

ACKNOWLEDGMENTS

First of all, I would like to thank my advisor Professor Kenneth De Jong for his continuous support, advice and constant encouragement throughout my dissertation work. I am grateful to him for guiding me through this endeavor. I would like to thank my committee members, Professors Kenneth Hintz, Eugene Norris and Pearl Wang, for their comments and suggestions regarding my research. I would also like to thank Dr. Murray Black for providing me with a SITE fellowship during the year 1994.

I would like to thank my graduate colleagues, Mitch Potter, Alan Schultz, Bill Spears, and Haleh Vafaie for their helpful discussions on the topics related to my research work. I would also like to thank Ron Morrison for reading my dissertation and for his comments.

This work was supported in part by a grant of HPC time from the DoD HPC Center, (Naval Research Laboratory, Washington, DC and CM-5).

I am forever grateful to my parents for their encouragement and support through the years. Their willingness to help me by taking care of my children so that I could continue to do my research was invaluable. Without the support and love of my husband, Prasad, and my children, Ganesh and Ranjani, I could never have completed my dissertation.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Abstract

AN ANALYSIS OF DECENTRALIZED AND SPATIALLY DISTRIBUTED GENETIC ALGORITHMS
Jayshree Sarma
George Mason University, 1998
Thesis Director: Dr. Kenneth A. De Jong

A number of interesting design issues arise when one wants to implement and use decentralized and spatially distributed genetic algorithms (GAs). The implications of these design decisions are not well understood and have only been tested empirically. The objective of this research is to understand and characterize the behavior of the decentralized and spatially distributed GAs with overlapping neighborhoods. To do so the control structure of the GA must be decentralized which alters the semantics of the selection algorithm used. This in turn usually produces changes in the GA's problem solving behavior. First, the implications of the various decentralized selection algorithms are analyzed by studying the changes produced on the characteristics of the selection pressure they induce on the entire population. This analysis shows that the emergent global selection pressure of a particular local selection algorithm is qualitatively similar to its centralized counterpart, but quantitatively weaker. Next, the techniques to formally analyze the selection methods in centralized sequential GAs are extended and applied to the local neighborhood GAs to understand the effects of the neighborhood sizes and shapes. This results in a quantitative model in which the coefficient of the growth rate of the best individual in the population is

expressed as a function of the ratio of the neighborhood radius to the grid radius.

This quantitative model is used to predict the expected behavior of the spatially distributed GAs. These predictions are empirically validated using the domain of function optimization. These validations help us gain valuable insights into the behavior of these GAs. Furthermore, it is also demonstrated how the selection pressure in a local neighborhood GA can be adjusted based on the analytical model. Finally, the analytical model is used to design effective spatially distributed GAs for solving problems in complex and dynamic environments.

The analytical model is also capable of explaining (in some cases conflicting) previously published empirical results. In addition, a developer can now make more effective choices when designing and implementing spatially distributed GAs.

Chapter 1

INTRODUCTION

1.1 Motivation

Genetic algorithms (GAs) are robust adaptive systems that can be used as general purpose problem solvers. Holland (1975) introduced and presented the theoretical framework for the GAs in the early 70's. Even though GAs are based on natural selection and biological evolution, they are only highly idealized models of natural evolution. GAs are general purpose models because the algorithm needs no *a priori* information about the problem being solved. This is one of its strong points. A genetic algorithm uses a population of individuals (structures) to explore the search space. Each structure in the population represents a potential solution in the search space. Since GAs search using a population, they can rapidly discover structures with high fitness values and exploit this information to locate a satisfactory solution. This ability to rapidly converge to a few regions in the search space is especially useful when GAs are used for function optimization where the goal is to find a single global optimum.

Nowadays GAs are being used to solve not only larger and more difficult function optimization problems but also more complex applications, such as multiobjective, multimodal, and covering problems, as well as problems in time-varying environments. In multimodal, multiobjective, and covering problems, the goal is not to find the best solution but to arrive at a set of solutions which satisfy the problem requirements. In a time-varying environment, GAs need to adapt quickly by locating the new set of structures with high fitness values as the environment changes. In all these problem classes, GAs need to maintain sufficient diversity in the population as long as possible to satisfy the problem requirements. This objective of sustaining population diversity cannot be met by the traditional GAs as their tendency to rapidly converge implies that diversity is lost fairly quickly. Hence, the rapid convergence property, which is a strength with respect to unimodal function optimization (single global optimum), is a weakness for the other classes of complex problems (multiple global optima) listed above.

Difficult function optimization problems have huge search spaces which require the use of large population sizes. But large populations are computationally expensive on serial machines and necessitate the use of parallel and distributed machines. In order to take advantage of distributed architectures, the traditional GAs will have to be changed semantically. When the semantics are changed the existing theoretical results may no longer hold and new theoretical results are required. Smaller population sizes have also been used to search large spaces on sequential machines, but to do so effectively, the sequential GA was considerably modified to solve these difficult problems. Researchers have pushed the limits of traditional GAs trying to solve these complex problems. So how does one enhance the traditional GA so that it is a better and more robust problem solver?

The traditional sequential GAs make a few implicit assumptions such as random mating (*i.e.*, each structure is equally likely to mate with any other structure in the population), availability of fitness information of all the individuals in the population at a central location (global calculations), sequential evaluation of the structures, and calculation of the fitness of an individual independent of other structures. These abstractions were made primarily due to the limited computational resources that were available at the time the GAs were introduced. The advances in computer hardware technology allow for the relaxation of most of these assumptions.

The general approach is to relax one or more of the assumptions made by the traditional GAs when solving the new classes of complex problems. Consider, for example, the idea of random mating which assumes that all individuals in the population are equally likely to mate with each other. In natural populations, there is no central controller to decide the mating pairs. Non-random mating is more likely than random mating, *i.e.*, individuals are more likely to mate with similar or nearby individuals. The notion of non-random mating is being investigated by researchers as it is perceived to be more robust. GAs with non-random mating are able to maintain diversity for longer periods by restricting mating to occur only within a certain group of individuals. Non-random mating aids in the formation of subpopulations within a larger population.

One method to perform non-random mating is to partition a large population into smaller nonoverlapping populations which evolve (run) concurrently. Even though each subpopulation is running a centralized GA with random mating, there is occasional migration of information among the subpopulations causing non-random mating. A second method to introduce non-random mating is to use the notion of tag bits to create subpopulations of similar individuals and restrict mating to occur between these individuals only. Yet a third method to do non-random mating is to introduce a distance metric so that the mating can be restricted to occur within a group of genetically similar individuals.

Metrics involving distance in genotype or phenotype spaces have been shown to be useful in solving multimodal problems. A distance metric imposed on the population which is laid out in some topology, has also been shown to be useful in solving a wide range of applications. In these cases a topology, such as a two-dimensional grid, is associated with the population. Each individual is assigned to a cell in the grid. This spatially distributed population can exploit parallelism much better than the sequential centralized GA. The assumptions which are relaxed here are the random mating and global calculation requirements, but these relaxations change the underlying semantics of the sequential evolutionary algorithm.

A decentralized and spatially distributed GA can exploit parallelism to a fuller extent than a centralized sequential GA. The decentralized and spatially distributed model is biologically plausible as natural populations are geographically separated. The model can take advantage of the available distributed hardware and a benefit of this is that larger population sizes can be effectively processed. Computational speed and efficiency are a must, especially when one attempts to use GAs for complex applications. The advantage of decentralized and spatially distributed GAs is fault tolerance. The algorithm is fault tolerant in the sense that even if a few processors fail, the algorithm can still work and continue with the search.

Both theoretical and experimental results are available for most of the commonly used sequential GAs, but these sequential GAs have fairly strong explicit or implicit forms of centralized control. When these centralized sequential GAs are turned into decentralized and spatially distributed GAs, the existing theoretical results are of little help in making the design decisions. In a decentralized and spatially distributed GA, local mating occurs concurrently on small overlapping neighborhoods. The dynamics of these overlapping local neighborhoods operating in parallel is extremely complex and difficult to analyze formally. There have been many ad-hoc empirical studies but no formal analysis of the decentralized and spatially distributed GAs. Empirical studies have shown that these systems can be used to solve complex problems, but are not of much help in designing effective decentralized and spatially distributed systems.

When one wants to implement a decentralized and spatially distributed GA, a number of interesting design issues arise such as what topology to use, what neighborhood size or shape to use, and which local selection algorithm to use. One does not understand the behavioral implications of choosing a particular topology. Also one does not understand the implications of choosing a neighborhood pattern and/or local selection algorithm within a topology. The implications of these design decisions are currently not well understood, but as a result of this dissertation research the effects of these design decisions will be better understood.

1.2 Research Objectives and Methodology

The goal of this research is to perform a systematic and careful study to understand and characterize the behavior of the decentralized and spatially distributed GAs. A deeper understanding of these systems will aid in making effective design decisions in a decentralized and spatially distributed GA.

First, the possibility of decentralizing the selection algorithm without changing the semantics will be explored. Decentralized selection implies that a global controller is no longer performing the process of selecting parents for reproduction. Note that as the semantics are unchanged, the theoretical results from the centralized GA will transfer to the decentralized GA. The existing theoretical results can be used to make predictions about the expected behavior of the decentralized system.

The next step in the study is to introduce a spatial topology on the population and define local neighborhood patterns. The process of selection, reproduction and evaluation is occurring concurrently in each of these local neighborhoods. Since these local neighborhoods overlap, there is an implicit migration of information through the spatial topology. Local behavior gives rise to a collective or *emergent* behavior. Here our goal is to understand and characterize the behavior of the selection algorithms when they operate in local neighborhoods. The effects of the different neighborhood sizes and shapes used within a population topology with respect to each local selection algorithm will also be investigated.

Both formal and empirical methods will be used to meet our objectives. We will first characterize the emergent selection pressure induced by the different selection algorithms in the overlapping local neighborhoods. We will then formally analyze the effects of the neighborhood patterns and local selection algorithms by extending the growth curve analysis used in sequential GAs. The growth curve analysis will help us understand the behavior and effect of a particular combination of neighborhood pattern and local selection algorithm in a given population topology. Such an understanding will help in designing effective decentralized and spatially distributed systems.

We will use regression analysis to formulate a mathematical model that shows the relationship between the neighborhood patterns and selection pressure for a given topology. We will then use this analytical model to predict the behavior of the decentralized and spatially distributed GAs. Then these predictions will be empirically tested in the domain of function optimization to validate our characterizations. Finally we will explore the additional interesting properties of these spatially distributed models by applying them to two classes of applications namely, multimodal and non-stationary problems.

1.3 Contributions

The major contributions that this research brings to the field of GAs are as follows:

1. We have developed a quantitative model characterizing the selection pressure induced by the local selection algorithms in terms of the ratio of the neighborhood radius to the grid radius for a two-dimensional toroidal grid. We have also shown how this model can be easily extended to dimensions greater than two. The emergent selection pressure induced by the selection algorithms in the decentralized spatially distributed models is qualitatively similar to the corresponding sequential and centralized GA, but quantitatively weaker.

2. We introduced an appropriate distance measure from spatial analysis techniques for computing the effective radius of the neighborhoods and the grids. This distance measure discriminates the internal layout of the neighborhood patterns.

3. This quantitative model can be used as a tool for predicting the expected behavior of the decentralized and spatially distributed GAs. These predictions were further validated in the function optimization domain. We showed how

the quantitative model can be used to adjust the selection pressure to improve performance.

4. We have also demonstrated how this analytical model can be used to make effective decisions about selection algorithms and neighborhood patterns when designing decentralized and spatially distributed GAs for complex problem classes. We now have a better understanding of the behavior of these decentralized and spatially distributed GAs.

1.4 Dissertation Outline

This dissertation is organized as follows. Chapter 2 gives a brief overview of sequential and decentralized genetic algorithms and the research pertaining to decentralized GAs. An overview of the open issues is also presented. Chapter 3 discusses the research issues that are addressed and the framework for this research. Chapter 4 presents the effects of decentralizing the selection algorithm and compares the emergent selection pressure induced by the selection algorithms in the decentralized spatially distributed GAs to the global selection pressure induced in the centralized sequential GAs. Chapter 5 provides a quantitative overview of the different selection algorithms used in decentralized spatially structured GAs. Chapter 6 validates the predictions made from the theoretical analysis by empirically testing these predictions. In Chapter 7 the spatially distributed model is applied to multimodal problems and non-stationary or time-varying environments. Chapter 8 summarizes the contributions of this dissertation and provides suggestions for future research.

Chapter 2

BACKGROUND

This chapter will provide a brief description of the canonical sequential genetic algorithms and the parallel genetic algorithms and their variants. It will also give an overview of the current state of research in decentralized and spatially distributed GAs and the open issues in that area.

2.1 Sequential Genetic Algorithms

Genetic algorithms (GAs) were first introduced by Holland as robust adaptive search techniques that could be applied to different fields, such as economic planning, and artificial intelligence (Holland 1975). They form a highly idealized abstraction of the natural biological evolution. GAs have been used for function optimization, evolving neural networks, scheduling problems, image processing, and learning classification rules. GAs are part of a larger category of evolutionary algorithms called *evolutionary computation*. The other branches in evolutionary computation are *evolutionary programming* (Fogel, Owens, and Walsh 1966) and *evolution strategies* (Rechenberg 1973; Schwefel 1981). An overview of evolutionary computation can be found in (De Jong and Spears 1993; Spears, De Jong, Bäck, Fogel, and de Garis 1993). Bäck performed a comparison of the three paradigms using the function optimization domain (1993).

One of the strong points of GAs is that they are general purpose algorithms and do not require domain-specific knowledge (De Jong 1988). Since GAs search by using a population of candidate solutions, it can quickly discover solutions with high fitness values and exploit this information to arrive at a satisfactory solution. This property of swiftly converging to a high-performance solution makes them attractive for those classes of problems which have a single global optimum. For other classes of problems, where the goal is to find multiple final solutions, one would like to maintain population diversity longer so that the different solutions can be found and the rapid convergence property can be a drawback.

A genetic algorithm consists of a population of interacting structures. Each structure represents a point or solution in the space that is being searched. Generally these structures are formulated as binary string representations that encode information about the search space. A fitness value which is problem dependent is associated with each of these individuals (structures). The individuals compete with each other for survival – a concept analogous to the Darwinian notion of survival of the fittest. The fittest are selected to reproduce offspring through the genetic operators and thus evolve better solutions. The main genetic operator is *crossover* which is used to recombine two or more selected strings to form offspring. The *mutation* operator is generally used as a background operator to prevent the permanent loss of allele values. Each such iteration consisting of selection, recombination, and evaluation steps is called a generation.

Structures are usually referred to as *genotypes*, which in biology represent the overall genetic makeup of an individual. Each structure can be made up of one or more strings, called *chromosomes* after their biological counterparts. A chromosome consists of genes which represent the variables of the problem environment. Each variable (gene) has a specific value or allele, that can be binary (0 or 1) or real valued. Even though binary strings are most commonly used, there have been many studies that have used representations like real-valued vectors, high-level computer programs (Lisp expressions), and decision trees. In biology the chromosomes are *diploid* (pairs of chromosomes), but in GAs the chromosomes are mainly *haploid* (single chromosome). The structure when transformed represents a solution or a point in the search space. This converted structure corresponds to the *phenotype* which represents the observable characteristics of an individual. In nature, these characteristics are usually determined by the underlying genetic makeup (genotype).

For any problem to be solved using a GA, one has to find the appropriate mapping of the space to be searched onto a representation domain. The representation defines the admissible structures that can be explored by the GA. Designing a mapping for the problem environment onto a suitable representation space is an important part of the design process (De Jong 1985). The next step is to define a metric for evaluating the fitness of the structures in the problem environment. An appropriate fitness measure is desirable so that a good feedback from the environment can be obtained. The usual assumption is that the fitness of a structure is independent of the fitness of other individuals in the population.

Traditionally GAs have been implemented on sequential machines and in most cases used small population sizes $(30 - 200)$ due to limited computational capability. The canonical sequential GA algorithm is described using pseudo-code in Figure 2.1. From this pseudo-code we can see that there are essentially three phases in a sequential GA, namely, evaluation, recombination (applying crossover and mutation operators), and selection for reproduction as well as deletion. A generation is an iteration consisting of these three phases. The algorithm is usually run until a stopping criterion is met, which is domain dependent. The population is generally initialized at random, though it is possible to seed the initial population with domain specific knowledge. Unlike natural populations, the GA population is usually of constant size and does not vary with generations.

Of the three phases in a sequential GA, the evaluation phase is inherently parallel, if each individual can be evaluated independently of the others. Hence it is easy to parallelize this phase without any change in the semantics of the algorithm. Of course, parallelization makes sense only if the evaluation phase takes significantly more time (computationally) than the rest of the phases. But if the evaluation function is itself

```
Randomly initialize population
Evaluate fitness of each individual
While (stopping condition not true)
        Begin
                Select parents
                Apply crossover and mutation
                Select survivors
                Evaluate fitness of each individual
        End
```

Figure 2.1: Pseudo code for the sequential GA.

trivial (requiring very short computation time) then parallelizing the evaluation phase may actually increase the computation time due to the communication overhead.

In the recombination phase, crossover is applied to the two parents chosen by the selection algorithm, and generally two offspring are created. Segments of the chromosomes are exchanged between the two parents to create new offspring. The most commonly used are the *one-point* and the *two-point* crossover operators. In a one-point crossover a location in the string is selected at random and the segments after that location in the two parents are exchanged to form two offspring. But, in a two-point crossover, two locations are selected at random from 1 to $l-1$, where l is the length of the string. The portions between the two crossover sites in the two parents are then exchanged to form two new offspring (See Figure 2.2). Mutation alters the value of the allele and is mainly used to ensure that no allele value is permanently lost. The crossover operator only recombines the existing genetic material but the mutation operator creates new genetic material. Like the evaluation phase, the recombination phase also can be parallelized without any change in semantics. Again one has to consider the tradeoffs between communication and computation time.

The selection phase consists of selecting individuals for reproduction using a selection algorithm and selecting individuals for deletion after the offspring are generated.

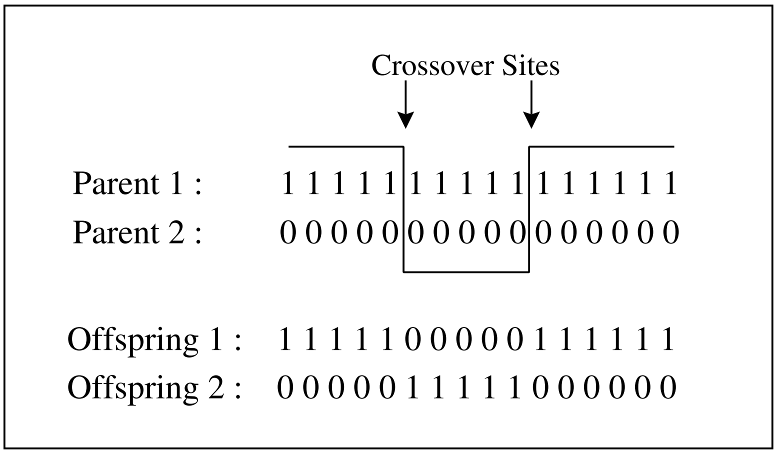

Figure 2.2: An example of two-point crossover.

The selection algorithm involves two elements: a selection pool, and a selection probability distribution over that pool. A selection pool is required for both reproduction selection as well as for deletion selection. The key issue in both these cases is: "what does the pool contain when parents are selected and when survivors are selected?". The selection distribution assigns each individual in the pool a probability for being selected.

The selection pool can be either a global or a local pool. The global pool consists of the entire population while the local pool consists of subsets of the population and these subsets may or may not overlap. An individual in a population can potentially mate with any other individual in the global selection pool. This approach is referred to as random mating. In biology, randomly mating populations are called *panmictic* populations. When the mating is restricted by some means then the mating is non-random, and the selection pool is local.

Generally two parents are selected to produce offspring based on their fitness. In most centralized sequential GAs the selection pool is a global pool consisting of the entire population. The selection probability distributions are typically defined in terms of an individual's fitness relative to the fitness of the other individuals in this pool. Traditionally used selection algorithms such as fitness proportional, rank proportional, and truncation selection, need to keep global statistics for computing the selection probability distribution. The basic underlying assumption with these selection algorithms is that information about the fitness of all the individuals in the population is available at a central location and calculations can be done based on that. This requirement of global statistics makes the selection phase difficult to decentralize.

A decision has to be made as to which individuals should be chosen for deletion to make room for the new offspring. When the entire parent population is always replaced by the offspring population, the selection pool for deletion is a global pool consisting of the current parent population only. Such GA models are called *generational* or *non-overlapping* systems. In *overlapping* GAs, the selection pool for deletion consists of both parents and their offspring. The survivors for the next generation are selected from this combined set. This again requires that the information be available centrally. The main criteria for overlapping systems is that both the parents and their offspring can coexist. The probability distribution is usually uniform but there exist a few non-uniform deletion strategies like the delete-least-fit, and the first-in-first-out (Syswerda 1991; De Jong and Sarma 1993).

Thus the selection phase for reproduction is hard to decentralize because of the traditional selection strategies used to select individuals which need globals knowledge. Since global calculations (centralized control) are required in each of these selection strategies, it is hard to decentralize this phase without changing the semantics. A change in the semantics alters the search behavior of the algorithm.

An important point to note is that the selection phase has a great influence on the progress of the evolutionary algorithm, *i.e.*, the search can be biased towards certain regions of the fitness landscape. This bias is usually referred to as *selection pressure* and is affected by how parents are selected to reproduce, how many offspring are produced, and who survives or dies. Strong selection pressure implies that higher fitness individuals are preferred (selected) for reproduction and survival at the expense of the lower fitness individuals. Selecting above average fitness individuals for reproduction leads to higher exploitation of the information gathered. On the other hand, weak selection pressure means that there is little or no bias towards any group of individuals, either high- or low-fitness. In general, selection should be broad-based so that the new offspring created will allow exploration of the search space. Decentralizing the selection strategies can alter the induced selection pressure and change the balance between exploration and exploitation.

One of the recurring themes in Holland's work is the importance of a proper balance between exploration of the search space and exploitation of the information gleaned, when adaptively searching an unknown space for high performance solutions (Holland 1975). Many aspects of a GA affect this balance including representation, selection for reproduction and deletion mechanisms, genetic operators, and population size. When the balance is not maintained, the GA can potentially converge to a non-optimal solution. Convergence to a non-optimal solution is called *premature convergence*. Premature convergence implies that the diversity in the population is lost.

Converging prematurely adversely affects the performance of the GA in applications where maintaining diversity is important. For example, in multimodal problems (having many final solutions), one would like to locate all the global optima. Diversity is not only required to explore new areas in the search space to find all optimum solutions but also to delay convergence to a single optimum solution especially when solving a multimodal problem. But, due to many factors such as stochastic sampling errors, and high selection pressure, the GA may converge to one of the peaks (final solutions) and may not be able to locate the other peaks in the same run. Different runs might find different peaks but in any one run of a GA the population may converge to a single peak. Premature convergence and maintaining diversity in the population, especially when complex problems are attempted, have been studied extensively. Some of these studies are reviewed in the next section. A general approach when attempting to solve such classes of problems is to relax some of the assumptions made by the traditional GAs.

2.1.1 Modifications to Sequential Genetic Algorithms

Crowding was introduced as an alternative to increasing the mutation rate and as a method for overcoming the problem of premature convergence in multimodal functions when using small populations (De Jong 1975). In the crowding scheme, only a few offspring were produced every generation making it an overlapping GA. The assumption that every individual in the population has a uniform probability of being selected for deletion was modified in the crowding scheme. After an offspring was produced, the deletion pool was formed by selecting a few individuals from the parent population specified by a crowding factor. The individual most similar to the offspring in the genotypic space is selected from the deletion pool to be replaced by the offspring. Later studies showed that the crowding scheme could maintain subpopulations only on a few peaks in multimodal problems (Deb and Goldberg 1989).

Baker (1989) studied the premature convergence problem from a sampling point of view and introduced two selection algorithms called the remainder stochastic independent sampling and the stochastic universal sampling. Stochastic universal sampling has no bias (statistical) and a minimum spread with respect to the number of offspring each individual receives. Since individuals are selected for reproduction in the selection phase, reducing the bias in that phase helps in controlling the convergence rate. Other means of controlling the selection pressure is to use linear ranking selection strategy instead of fitness proportional selection (Baker 1989; Whitley 1989).

Most GAs assume random or global mating within a population. This assumption has been relaxed when GAs are used to solve multipeaked problems. The sharing function was introduced as a method for maintaining diversity while solving multimodal functions (Goldberg and Richardson 1987). The idea was to maintain a population at the various peaks in a multimodal function thus creating niches and forming new species. The population is divided into subpopulations using a distance metric which can be defined either in the genotype or decoded parameter space. The fitness of an individual is then reduced based on the number of individuals in its subpopulation. This process is called fitness sharing. Additionally, mating was also restricted to occur within the subpopulations only which was made up of similar individuals (Deb and Goldberg 1989). Phenotypic sharing was shown to maintain subpopulations at the different peaks while genotypic sharing was unable to do the same. Mating restriction in addition to phenotypic fitness sharing improved the online performance measure. Mating restriction is a computationally intensive method because each individual has to be compared with all the other members in the population.

Spears's study (1994) showed that adding tag bits (labels) to the bit-string representation is another method for creating niches and forming subpopulations when solving multipeaked functions. Global knowledge of the labels are required to group the individuals into different subpopulations. The fitness of the individuals in each subpopulation was normalized by the number of members in that group. Also mating was restricted to occur only between individuals with the same labels. These different forms of non-random mating were introduced in the sequential GA in order to solve multimodal problems with many final solutions. But these non-random mating schemes still need global knowledge and thus are not easily decentralized.

GAs used in the immune system model relax the assumption that each individual is evaluated independently of others (Smith, Forrest, and Perelson 1993). In the immune system problem, the goal is to evolve a population of antibodies that will recognize a particular set of antigens. Each antibody is now evaluated on the basis of how well it performs with respect to a set of other antibodies in recognizing antigens. Hence the fitness of an antibody is dependent on the set of other antibodies with which it is evaluated.

Another interesting class of problems are the time-varying or non-stationary environments, where the fitness landscape changes over time. Cobb and Grefenstette (1993) applied GAs to time-varying landscapes. They introduced two new mutation strategies called *random immigrants* and *triggered hypermutation* to enable the GA to track the optimum in a changing environment.

As the class of problems get more complex, the simple GA has trouble evolving satisfactory solutions. Researchers have tried relaxing and/or extending the simple GAs to solve these problems. Most of these complex problems require large population sizes and can be computationally intensive on sequential machines. Hence these simple GAs have been implemented on parallel and distributed systems, so that larger population sizes can be processed in reasonable time. But because of inherent sequentiality of certain parts of the algorithm, the decentralization has involved semantically changing the algorithm itself.

2.2 Decentralized Genetic Algorithms

Recent advances in distributed computing have provided a strong motivation for decentralizing the canonical GA. Availability of more sophisticated computer hardware has spawned many studies in which the sequential GA has been modified so that it is better suited for these distributed architectures. As discussed in Section 2.1, another motivation has been the desire to solve harder and more complex problems; usually these problems require larger population (> 1000) sizes. The computational time for large population sizes on a serial computer can be quite high and hence researchers have turned to parallel and distributed computer systems to reduce computation time.

The assumption of centralized control no longer exists. Instead the computation is now performed locally, and there is no global random mating. Decentralized GAs are labeled either as coarsely-grained or finely-grained systems depending on how the algorithm was changed. In the coarsely-grained approach, a single large population is divided into many smaller nonoverlapping populations. Each subpopulation does the process of selection, recombination and evaluation and communicates with the other subpopulations through the migration of individuals. This maps well onto coarsely-grained parallel architectures, which have fewer but computationally powerful processing elements. In finely-grained GAs, each individual in the population is now responsible for the process of selection, recombination and evaluation. This framework fits well onto the finely-grained parallel computers which have numerous processing elements each of which may be computationally less powerful than the coarse-grained computers.

2.2.1 Coarsely-Grained Genetic Algorithms

The biological motivation for these coarsely grained models comes from the work by Wright (1932) who showed that isolated populations with infrequent migration were able to attain higher levels of differentiation and fitness than a panmictic population. Hence, the conjecture is that if many small populations are searching the solution space simultaneously, the final solution will be much better than if one single large population was being used.

In coarsely-grained parallel GAs, the large population is subdivided into many smaller subpopulations. Each subpopulation is evolving in parallel on different processing elements of a parallel computer or on a network of workstations. Thus each subpopulation can evolve to a different point in the search space and diversity is maintained. Diffusion of genetic material between the subpopulations is achieved through migration of individuals from one subpopulation to another. Hence a communication pattern for the different subpopulations is required. If there is no communication (migration) between the different subpopulations then it is equivalent to running several sequential GAs simultaneously. The pseudo-code for the coarsely-grained GA is given in Figure 2.3. We can see from the pseudo-code that the algorithm is essentially the same as the sequential GA described in Section 2.1 except for the addition of the communication step ("Time for Migration").

The user has to decide upon the number of subpopulations to be used and the

```
Randomly initialize population
Evaluate fitness of each individual
For each subpopulation
While (stopping condition not true)
        Begin
                if (Time for Migration)
                   Send Migrants
                   Receive Migrants
                   Amalgamate Migrants
                endif
                Select parents
                Apply crossover and mutation
                Select survivors
                Evaluate fitness of each individual
        End
```

Figure 2.3: Pseudo code for the coarsely-grained GA.

pattern of migration among them before utilizing this form of parallel GA. There are no heuristics available for making this decision and it is generally dictated by the interconnection topology of the underlying architecture. When the pattern of migration between the different subpopulations is arbitrary then it is referred to as the *island model*. In the *stepping stone* model, the subpopulations are arranged either in a ring or in a two-dimensional grid format and the migration of individuals from one subpopulation to another is confined only to their immediate neighbors.

There are three new (additional) parameters that have to be specified before a coarsely-grained parallel GA can be executed: 1) the frequency of migration, 2) the percentage of migrating individuals and 3) the amalgamation strategy. The first parameter specifies how often information should be exchanged between the subpopulations. Exchanging information frequently can be disruptive to the evolutionary process while infrequent exchange may not contribute anything. The second parameter specifies the number of migrating individuals for the subpopulations. The greater the number of individuals exchanged the larger will be the communication time required. The third parameter specifies how the immigrants in each subpopulation should be amalgamated with the existing population so that the population size remains constant. Choosing these parameters is critical to the performance of the GA. The selection pool within each subpopulation is global consisting of all the individuals in the current generation and parents are selected based on their fitness from this global pool. This coarsely-grained GA model has been studied extensively (Grosso 1985; Tanese 1989; Pettey and Leuze 1989; Whitley and Starkweather 1990; Cohoon, Martin, and Richards 1991).

2.2.2 Finely-Grained Genetic Algorithms

The main motivation for the finely-grained decentralized GA comes from Wright's *Isolation by distance* model in population genetics (Wright 1943). In that model, there is a continuity of the population which is either one-dimensional or two-dimensional. The area from which parents of the central individuals are drawn at random is termed a *neighborhood* (Wright 1969). Thus neighborhoods are defined by geographical distances and can be overlapping. Wright showed that the local differentiation in a linear continuum was higher than in an area continuum (Wright 1969). This model is similar to natural populations, where individuals live in a spatial continuum and interact with only the nearby individuals.

In the finely-grained approach an explicit spatial topology has to be defined on the population. There is no centralized controller deciding, on behalf of the entire population, who gets to reproduce and survive. Instead these decisions are made at the local level. In this decentralized approach, an individual in the population interacts only with the nearby individuals and this forms the local selection pool for that individual. These local selection pools are called neighborhoods which generally overlap thus causing implicit migration of information through the topology. The finely-grained GA pseudo-code is given in Figure 2.4.

```
Randomly initialize population
Evaluate fitness of each individual
For each grid location
    While (stopping condition not true)
    Begin
            Select parents from neighborhood
            Apply crossover and mutation
            Select a survivor
            Evaluate fitness of survivor
    End
```

Figure 2.4: Pseudo code for the finely-grained GA.

In the finely-grained model which uses local selection pools, two additional decisions have to be made once the population size is known. First, the spatial topology to be used for the population has to be decided. For example, a two-dimensional grid can be used as the topology for the population with each individual occupying a grid location. Second, the neighborhood size and shape has to be decided which will determine the neighborhood for each grid location; this decides the set of individuals with whom the individual at each grid location can interact.

Each cell in the two-dimensional topology contains one individual. Each individual can be thought of as running a GA. The process of evaluation, selection, reproduction and replacement is done in each cell. A neighborhood is defined for each individual and this forms the local pool for selection of parents for reproduction. Two parents are selected based on their fitness from this local pool to reproduce. One offspring is produced using the genetic operators and the individual at that grid location can be potentially replaced by the offspring. When the neighborhood of an individual is the entire grid, then the local pool degenerates to a global pool and is similar to the sequential GA. This finely-grained GA model can be run either in a synchronous or in an asynchronous mode.

Since the neighborhoods can be overlapping, there is an implicit diffusion of genetic material through the entire population. If the defined neighborhoods are non-overlapping then it becomes similar to the coarsely-grained GA model and an explicit migration scheme has to be specified. The finely-grained GA model has been implemented on massively parallel machines as well as other parallel computers. This model has been interchangeably called the *diffusion model*, the *continuous model*, and the *neighborhood model*.

Introduction of a topology and a neighborhood for mating is a semantic change to the sequential GA. Decentralizing the selection algorithms and using local pools produce different selection pressures than the centralized sequential versions and can thus result in a different evolutionary behavior. Recall that the selection pressure is affected by the choice of individuals selected for reproduction, how many offspring are produced and how individuals are selected for deletion. In a decentralized and spatially distributed GA, an individual competes only with the individuals nearby. The number of individuals in the local selection pool is dependent on the defined size of the neighborhood. The various neighborhood sizes can change the selection bias and thus produce different selection pressures and alter the trajectory of the evolutionary search process. Therefore a clear understanding of the characteristics of the decentralized and spatially distributed GA is important for the effective use of these systems.

2.3 Related Research

Many of the studies using the finely-grained GAs have been motivated by the desire to take advantage of the available parallel hardware. This involves changing the traditional GA so that parallelism can be exploited, but it also changes the behavior of the algorithm. This section reviews the work done in the finely-grained parallel GA as it is the main topic of interest in this dissertation.

As mentioned earlier, the finely-grained GA involves the imposition of a topology on the population and the restriction of mating within a prespecified neighborhood. The two new parameters introduced are the topology for the population and the size of the neighborhood. Different population topologies used in the studies reviewed here are the ladder topology, the ring or one-dimensional grid, the two-dimensional grid which can be either a planar grid or a torus, and grids with dimensions greater than two. Within each of these topologies different neighborhood sizes have been used. The neighborhoods always overlap. Selection of parents is done locally from within the specified neighborhood. The individual at the grid is usually selected for deletion although other schemes for deletion selection have also been suggested and studied.

Most of these are empirical studies with several parameters changed simultaneously. Therefore it is hard to generalize the results, and quantify changes in performance with respect to modification in or addition of the various parameters. None of these studies help in making effective design decision like what shape and size neighborhood to use and what local selection algorithm to choose.

2.3.1 Population Topology and Neighborhood Sizes

One of the earliest studies that proposed a parallelization of GAs was done by Suh and Van Gucht (1987). They called their approach the *distributed* GA and proposed that one individual be allotted to each processor. The algorithm was implemented on BBN's Butterfly machine with 14 parallel processors. Selection was done on a local basis by each individual, and there was no central controller deciding who could reproduce. They had suggested and used five different forms of tournament selection schemes. No explicit topology was associated with the selection process and the selection for reproduction consisted of the global pool. Each processor (individual) was selected as a parent and then probabilistically chose another processor to be the other parent. This method can lead to greater communication costs as the number of processors increase. The test problem used was the traveling salesman problem with

20, 60, and 100 cities. They also implemented local improvement or hill-climbing by the individuals in the population. They were not able to allocate one individual to each processor as the number of available processors was smaller than the population size. They tested their algorithm by allocating a few individuals to each processor. Each processor then sequentially simulated the process of selection, recombination, local hill-climbing and evaluation. Their results, with respect to the best solution found in a certain number of trials, showed that the performance of the distributed GA was as good as the performance of the centralized sequential GA.

Gorges-Schleuter (1989) introduced a parallel GA model called the Asynchronous Parallel Genetic Optimization Strategy (ASPARAGOS). The algorithm used the ladder topology connected at the ends. One individual lives in each grid location. The individual at each grid location is selected as a parent and a mate is found from the neighborhood of that particular individual. The selection pool does not include the individual for whom the mate is being selected. Thus the same individual cannot be selected twice as a parent at the same grid location. This parallel model was called asynchronous because each individual in a cell could be in a different generation. The motivation for the asynchronous communication scheme was to overcome the problem of load balancing in the parallel machine.

Unlike the canonical sequential GAs which do not ensure that every individual can become a parent, in the ASPARAGOS model every individual can participate in the reproduction phase. The population sizes used in their studies were fairly small (16, 32, and 64) and the grid sizes were 2×8, 2×16 and 2×32 respectively. The two neighborhood sizes used in each of these cases were 5 and 8. The test suite consisted of two traveling salesman problems (442 and 532 cities). The representation, mutation, and crossover operators used were all application specific. Her results showed that a better solution was reached faster by the decentralized model than the sequential model. However, it is unclear from the empirical results whether the improved performance is due to introduction of the topology and local selection or due to the usage of application specific representation and operators. The study recommends that one use a relatively small neighborhood size and a large diameter for population topology. It is not clear how to quantify the relatively small neighborhood size or the large diameter when one is trying to design these systems.

Mühlenbein (1989) extended the ASPARAGOS model by introducing a local hill-climber. Each individual performed hill-climbing to improve its fitness before crossover. He has used the spatially distributed GA to solve complex optimization problems like the traveling salesman, graph partitioning, and quadratic assignment. The crossover operator was application specific. The results reported shows that better solutions were obtained using a spatially distributed GA. What is not certain is whether the spatial distribution or the individual hill-climbing helped in the success of solving hard optimization problems.

All the other studies reviewed in this section have used the one- and the two-dimensional grids and the population sizes have been fairly large (≥ 4000). Spiessens and Manderick (1991) introduced a finely-grained model and used a square two-dimensional grid. Population size used was 4096 (64×64 grid) and the neighborhood sizes were 5, 9, and 13. The algorithm was implemented on the Distributed Array of Processors (DAP) from Active Memory Technology Ltd. which is a massively parallel computer. They showed that the time complexity of the finely-grained parallel GA is a linear function of both the size of the neighborhood and the string length. The only test problem used was the deceptive order-4 problem. The two local selection algorithms compared were tournament and fitness proportional selection. They concluded that the neighborhood size depended on the local selection algorithm used.

Collins and Jefferson (1991) used one- and two-dimensional toroidal grids with population sizes ranging from 8192 to $524,288$. Selection of parents was based on a new method called random walk. In their study there was no explicit specification of neighborhood size as it depended on the length of the random walk. The test problem was a graph partitioning problem with two global optima. They compared the random walk selection using local pools with fitness proportional and linear rank selections using global pools. The results indicated that the decentralized GA was able to locate and maintain both global optima while the centralized GA with global pools converged to one of the optimal solutions and could not locate both optima in the same run. The local selection strategy used for reproduction and deletion changes the selection pressure. Hence comparing a decentralized local selection GA with a sequential GA, both of which induce different selection pressures, may not be a fair comparison.

McInerney (1992) also studied the effects of using very large population sizes (8192 to $524,288$) when the underlying spatial topology ranged from two- to five-dimensional toroidal grids. His study showed that for the same population size using dimensions greater than two, the diversity of the population is lost fairly quickly. Hence it may be better not to use a topology with greater than two-dimensions. His study, like Collins and Jefferson's study (1991), concluded that random walk selection should be used for solving complex problems.

The study done by Baluja (1993) used one- and two-dimensional toroidal grids. The main difference between his study and the others is that each grid location had two individuals instead of one. Thus even though the grid sizes were 4096 the population size was 8192. The motivation as to why two individuals were assigned to each processor instead of one is not clear. Also how that affected the performance of the GA is not obvious. The subpopulations were formed by including the two individuals at each grid point and one individual picked at random from the specified neighboring grid points. The study showed that the two-dimensional torus performed better than the one-dimensional grid in terms of the average number of generations required to find the optimal solution.

Davidor's ECO GA (1991) uses a two-dimensional toroidal grid as the topology for the population. Unlike the other models, all grid points do not update their content simultaneously. Instead in each generation one grid location is chosen at random and a subpopulation is formed using its immediate eight neighbors. Using fitness proportional selection strategy, two parents are selected to produce two offspring. The two offspring replace the worst two individuals in the subpopulation. The main reason for designing this model was to create niches and form species. Since the information is spatially distributed, information disperses slowly through the population and one can observe genetically similar regions forming as the search progresses. But when the ECO GA is run for a large number of generations, these different niches disappear and only one dominates. This is similar to the overlapping sequential model where only one or two offspring are produced each generation (Whitley 1989; Syswerda 1989). The main difference is the introduction of a spatial topology for the population. This model will be quite fast on a sequential machine, but is a waste of resources on a parallel machine as all processors except the one that is reproducing will be idle.

The performance of different topologies was studied by Schwehm (1992, 1994). The finely grained GA was implemented on the MasPar MP-1, a massively parallel mesh connected array processor with 1024 processing elements. He simulated different topologies like ring, torus, three-dimensional, five-dimensional and ten-dimensional grids. The population size was fixed at 1024 for all the topologies and the benchmark was the graph partitioning problem. The performance of a topology was measured by the number of generations needed to converge. The selection algorithm used was local tournament selection. He found that torus performed better than the higher or lower dimension topologies used. He also showed that the reliability of the algorithm was high, i.e., the algorithm was able to converge even when a few of the processing elements failed.

Studies comparing one-dimensional (ring) and two-dimensional torus topologies using large population sizes (Collins and Jefferson 1991; Baluja 1993) show that the one-dimensional parallel GA needs greater number of generations to find the optimum compared to the two-dimensional torus parallel GA for the same test problems. On the other hand diversity is maintained longer in the one-dimensional parallel GA.

This is consistent with the observation made by Wright (1969).

It is also claimed that the best or optimal topology for a population is one with the largest diameter as it will take longer for information to propagate through the topology (Gorges-Schleuter 1990; Mühlenbein 1991). For example, in a $N \times N$ torus it takes $O(N)$ iterations for genetic material to diffuse while in a ladder topology of the same size it takes $O(N^2)$ iterations. The diffusion of genetic material is related to the neighborhood size used. While the claim is true that it takes longer in a ladder topology for genetic material to disperse, it is not clear how the performance is related to the choices of topology, neighborhood pattern and local selection algorithm. For example for a population size of 8192, the ladder topology would be 2×4096. Then one may have to wait a very long time to see a satisfactory solution and some other topology which allows faster diffusion may be required. While it is true that the diameter of the topology imposed on the population controls the diffusion of genetic material, there has to be some relationship between the topology, the neighborhood size, and the selection strategy used and the performance of the finely-grained GA. What we need is a better understanding of the relationship between the behavior of the GA and the diffusion of genetic material controlled by the neighborhood size, and selection strategy.

2.3.2 Local Selection Strategies

The strategies used for both selection for reproduction as well as selection for deletion affect the balance between exploration and exploitation. A proper balance has to be maintained for the search to be efficient. The local decisions for selecting individuals for reproduction or deletion give rise to an emergent global decision which is hard to predict. The selection for reproduction strategies (like fitness proportional, ranking, and tournament selections) used in sequential GAs can essentially be used in a decentralized GA. An additional selection strategy that can be used is the *random walk selection*. Random walk is a well studied phenomenon in statistics and physics.

A notion of topology is needed to implement the random walk selection. A neighborhood has to be specified for all the selection schemes except random walk. The neighborhood size for the random walk selection is implicit and depends on the length of the walk and the topology.

Selection of individuals for reproduction at each grid point can be done in many different ways. One approach is to select both parents according to a selection distribution. An interesting variation is to select the central individual in each neighborhood as one parent and select the other parent according to the selection distribution. The central individual can be included or excluded from the selection pool while selecting the second parent. If the central individual is excluded, then both parents will be different. Another modification is to select one parent based on the probability distribution and then select the other parent at random. All these different methods induce different selection pressures (biases). Also having a local hill-climber within the algorithm makes it a highly exploitative search and tilts the balance towards exploitation rather than maintaining the balance. There have been a few studies that have experimented with different reproduction selection and deletion strategies.

Gorges-Schleuter (1989) in her implementation of ASPARAGOS always selected the individual at the grid location as one parent. The other parent was selected from the neighborhood based on the selection distribution. The selection algorithm used was the fitness proportional selection. The first parent was excluded from the second selection process and hence both parents selected were always different. She also studied different replacement (selection for deletion) strategies using a ladder topology having small population sizes (Gorges-Schleuter 1990). The main conclusion was that when using local fitness proportional selection, the strategy of replacing the individual at a grid only if the offspring had a better fitness worked best. This strategy acted as an implicit local elitist strategy for each neighborhood. In centralized GAs, elitist strategies are used to keep good solutions longer till better ones are found.

In Spiessens and Manderick (1991) the selection for reproduction was quite different from the canonical GAs. For each grid location, the first parent was selected based on the selection distribution and the other parent was selected at random. They used fitness proportional and tournament selections. In the case of tournament selection the tournament size was set to cover the entire neighborhood. The neighborhood sizes used were 5, 9 and 13. They conclude that a large neighborhood size relative to the population size appears to be able to give better performance if fitness proportional selection is used as the local mating strategy. But when local tournament selection is used, a smaller neighborhood size should be used since the tournament size is the entire neighborhood (Spiessens and Manderick 1991). It is unclear how large or small a neighborhood should be. Further, how should the neighborhood size vary with respect to changes in population size? How does selecting two parents instead of only one parent (as their study did) affect the performance? Also the study recommended using local tournament selection rather than fitness proportional selection, based on the results from only one test problem.

Studies that compared centralized GAs with global pools and decentralized GAs with local pools, all agree that decentralized local selection improves performance while the global selection tend to converge prematurely. Also diversity is maintained longer and thus convergence is delayed in decentralized GAs compared to centralized GAs. Collins and Jefferson (1991) showed in their study that the local random walk strategy was able to find both the optimal solutions and the larger population sizes were able to maintain them for a long time. In their study, at each grid location, both the parents were chosen from the neighborhood according to a selection distribution. They observed that random walks with lengths within the range 5-20 in the two-dimensional grids were able to generate solutions quite fast. Their use of very large populations sizes may partly be the reason why the random walk lengths were also large. They compared the decentralized local random walk strategy against centralized GAs which used fitness proportional and linear ranking selection strategies. It would be interesting to explore whether local fitness proportional and local ranking selection schemes would have performance comparable to the local random walk selection scheme.

Interestingly in McInerney's work (1992) the random walk strategies could not maintain two separate solutions for a long time unlike the results of Collins and Jefferson (1991). But the implementations of the local random walk in both these studies were different and so were the test problems. Hillis (1991) implemented a geographically distributed population GA on the Connection Machine CM-200 using a two-dimensional toroidal grid. The GA was used to solve the problem of general minimal sorting networks. He also included the competing coevolving species analogy to further speed up the progress of the algorithm towards the final solution. Truncation selection was used to select the best half, in terms of fitness values, of the population for reproduction. Then each individual at the grid performed a random walk to select the mate for recombination to produce an offspring.

Most of these studies have not considered the effects of crossover and mutation nor do they mention the settings of these parameters. These operators also affect the exploration/exploitation balance. The assumption appears to be that to obtain good performance from large populations one can impose a topology and use local information and that the performance is not sensitive to other parameter settings. The only study that looked at the effects of different types of crossover operators such as the 1-point, 2-point and uniform was performed by Spiessens and Manderick (1991). Their conclusion was that uniform crossover improved performance in small neighborhood sizes.

As we have seen, the studies used different strategies for reproduction as well as deletion selection, but the conclusions cannot be generalized. As mentioned earlier these different methods of selection induce selection pressure which can vary not only with respect to the topology used, but also with respect to the population and neighborhood sizes used. All studies were very concerned about the increase in communication costs with increasing neighborhood sizes and hence small neighborhood sizes were used. These studies provide a collection of interesting results but no global consensus. These empirical studies does not help one decide what local selection algorithm to use or what neighborhood size or shape to use. What we require is a characterization of the selection pressure of the different selection strategies with respect to the neighborhood sizes and topology.

2.4 Open Issues

Even though there have been many studies using the spatial topology and neighborhood concept, we still do not have a good insight into the behavior of these types of models. In these models, the neighborhoods are overlapping and there is an implicit migration of information between the subpopulations. In the centralized GAs, maintaining a balance between exploration and exploitation is important for an effective search. Similarly in decentralized spatially distributed GAs it is necessary to maintain the exploration and exploitation balance. This balance is affected (influenced) by different factors such as population size, topology, neighborhood size, local selection strategy, and the genetic operators.

The diffusion of genetic material through the topology is important. Too quick a diffusion can cause the search to converge to a non-optimal solution. On the other hand, if the diffusion is too slow, one will have to wait a long time before any solutions are found. So it is clear the effectiveness of the evolutionary algorithm is dependent not only on the local selection strategy one uses, but also on the neighborhood size and the topology. There are several open issues.

Issue 1: For a given class of problems is there an optimal topology?

How quickly the search progresses depends on how fast the genetic material can spread through the topology. Introduction of a topology is a semantic change and one would like to know how this affects the behavior of the algorithm. There may not be one optimal topology; the topology may be dependent on a number of factors like the type of problem to be solved, and the population size. The decision to use a particular topology can also be based on the hardware architecture of the parallel machine used.

Issue 2: For any class of problems, given a topology, is there an optimal neighborhood size? How does varying the neighborhood size influence performance?

Since the concept of population topology is introduced, the natural thing to do is to select mates from nearby individuals. It is not clear how one should define these neighborhoods. Since the neighborhoods overlap, each individual can belong to many subpopulations. If the neighborhood size expands, the number of subpopulations to which an individual can belong increases. This translates into faster migration of genetic material between subpopulations and there is a possibility that a super individual can takeover the population fairly quickly. Thus we could see the same problem of premature convergence as in the sequential GA if the right neighborhood size for a given topology and population size is not used. We would like to better understand how the neighborhood size is related to changes in the population size and topology.

Issue 3: Which selection strategies should be used for reproduction and deletion selection? How do these strategies affect GA search behavior?

The dispersion of the genetic material is dependent not only on the size of the neighborhood but also on the local selection strategy. The exploration versus exploitation balance is a function of the type of selection strategy, the neighborhood size and the topology used. In centralized GAs we understand how the selection pressure of the different selection schemes are characterized by their takeover time. But we do not have such an understanding with respect to the selection schemes used in decentralized GAs.

Selection pressure is also affected by the type of deletion (replacement) strategy.

Selection for reproduction decides who gets to reproduce while the selection for deletion is needed to put the offspring back into the population. An offspring replacing the individual at the grid only if it is better than the incumbent, causes an implicit elitist strategy and this could affect the search process.

Issue 4: How do genetic operators affect the exploration and exploitation balance? What rates should be assigned to genetic operators?

Most studies have not paid much attention to the rates of crossover or mutation and seem to convey the idea that these may not be relevant. But these parameter settings can affect the exploration/exploitation balance and may change the behavior of the decentralized GA.

Issue 5: How would one analyze decentralized spatially distributed GAs?

The behavior of a decentralized algorithm is a combination of the behavior of the different subpopulations within that topology. It is hard to specify what the emergent behavior will be in a decentralized GA which uses an explicit topology and overlapping populations. The theoretical foundations of the sequential GAs is given by the Schema Theorem (Holland 1975). The open issue here is how to extend the Schema Theorem to analyze these decentralized spatially distributed GAs.

Issue 6: What are the problem classes that are solvable by decentralized spatially distributed GAs?

Interest in these types of GAs have increased because of the belief that they may be better problem solvers. The open issue here is whether they are in fact better across all problem classes or only for specific classes of problems? We need to understand the strengths and weaknesses of these decentralized and spatially distributed GAs from a problem solving point of view.

In the next chapter we will present an overview of the issues which will be addressed in this dissertation.

Chapter 3

RESEARCH OVERVIEW

In this chapter we will outline the approach taken to understand and characterize the behavior of the decentralized and spatially distributed GAs. Several decisions have to be made to design a decentralized and spatially distributed GAs. The topology for the population has to be decided first. Then one has to decide the size and shape of the neighborhood and the local selection algorithm to be used. These design decisions were mainly based on the results of the empirical studies done in the past and these results can give conflicting choices. We do not have a very good understanding of the implications of choosing a particular neighborhood pattern or local selection algorithm, or the effects of a topology on the behavior of the GA. To get conclusive results, we need a controlled and systematic study with changes being introduced incrementally. We will use both empirical and analytical methods to achieve our goals.

3.1 Decentralization of the Selection Algorithms

Recall that among the three phases in an evolutionary algorithm, the selection process is not easy to decentralize because of the global statistics requirement. As a first step we try to understand the implications of decentralizing the selection phase on the behavior of the GA. Understanding the effects of this single change while keeping the semantics the same is important before we proceed to introduce semantic changes like introducing different topologies and neighborhoods. Keeping the semantics constant helps in making predictions about the expected behavior of the decentralized GAs based on the theoretical results of sequential centralized GAs.

The next change is to introduce a spatial topology on the population and define local neighborhoods. Selection now works on local neighborhoods. A selection process based on local neighborhoods rather than a global pool can alter the selection pressure induced by the selection algorithms. Changes in the induced selection pressures can result in different evolutionary search trajectories compared to the centralized GAs. The notion of topology, local neighborhoods, and local selection are all intertwined and have to be studied together. This poses a number of interesting design choices which are not particularly well understood.

3.2 Introduction of Topology

The population topology to be used is usually constrained by the underlying architecture on which the GA is going to be implemented, the problem to be solved, and the population size. The only study comparing the performances of the different population topologies showed that the two-dimensional torus performed better than the other topologies used (Schwehm 1994). The torus topology was better (in terms of the number of generations required to find the optimum) than both higher and lower dimension topologies used in the study. This is a limited study in the sense that the selection and replacement strategies, and the neighborhood size were fixed and common to all the topologies. His results are based on only one benchmark, a graph partitioning problem. Also from the results of the other studies done comparing the one-dimensional and two-dimensional topologies, the two-dimensional torus had better performance in terms of the best solution found. The number of combinations possible for the different topologies, neighborhood sizes, and selection strategies is enormous. Hence trying to answer the "optimal topology" issue can be difficult, if not impossible.

We do not attempt to answer the question of whether there is an optimal topology. Instead we make a design decision by selecting a topology which in our case is a square two-dimensional torus unless otherwise mentioned. First we characterize the behavior of the decentralized and spatially distributed GA with respect to this particular topology. Once this is done, we will extend and generalize these results to other topologies (non-square) and other dimensions (one, and greater than two).

3.3 Effects of Local Selection

The behavior of the decentralized and spatially distributed GAs is dependent on the effects of both the local selection algorithm used and the neighborhood pattern used. Changing either one of these or both can alter the behavior of these GAs. Hence the issues with respect to local selection methods and local neighborhoods will have to be studied together.

First, we will characterize the selection pressure induced by the different local selection algorithms in the square two-dimensional torus empirically. We will do this only for a few local neighborhoods sizes to limit the number of experiments. The induced selection pressure in the case of the decentralized and spatially distributed GAs will be compared to the induced selection pressure in the case of the centralized sequential GAs. This will give us some insights about the induced selection pressure in local neighborhood models.

3.4 Effects of Neighborhood Patterns

Next, we will examine the effects of the various neighborhood patterns with respect to each selection algorithm in the square two-dimensional torus. Selection is now happening in each of these local neighborhoods and these local interactions give rise to a collective or *emergent* global selection pressure. The formal analysis of this emergent global selection pressure will be done by extending the growth curve analysis used in sequential GAs to the local neighborhood models. By applying the growth curve analysis to the decentralized and spatially distributed GAs, we will provide a clearer understanding of the effects that the neighborhood patterns and selection algorithms have on the emergent global selection pressure. This will help in design decisions, as we will then have a better understanding of the behavioral implications of choosing a neighborhood pattern and selection algorithm.

3.5 Empirical Validation

This insight into the implications of choosing a particular selection algorithm and neighborhood pattern will help us make predictions of the expected behavior of a real GA. These predictions will then be validated by empirical experiments using function optimization problems.

3.6 Additional Properties

The decentralized and spatially distributed GAs will be applied to problems from the domain of multimodal functions and time-varying environment to understand the additional properties of these models. Thus instead of trying to answer the larger question of which problem classes can be efficiently solved by these decentralized and spatially distributed models, we restrict our attention to two classes of problems.

While we will not attempt to answer the broad question of what rates should be assigned to the genetic operators, the knowledge obtained from studying the selection algorithms should help us in making some observations about the crossover rates. Optimal parameter values in the case of the sequential GAs depend on the particular problem being solved (Hart and Belew 1991). Similarly, in the case of the spatially distributed GAs, the optimal parameter values might be problem dependent.

Chapter 4

ANALYSIS OF DECENTRALIZED SELECTION ALGORITHMS

The motivation for decentralizing the selection phase is to take advantage of parallelism. Performing the selection process concurrently can potentially yield a shorter execution time. In this chapter, the basic implications of the various methods of decentralizing the selection mechanism in an evolutionary algorithm will be studied. First, we will decentralize the selection phase without any changes in the semantics. If the semantics are kept constant, the behavior of the evolutionary algorithm will be the same even if implemented on different computing architectures. Second, we will investigate the effects of semantic change on performance behavior of the evolutionary algorithm when the selection process is decentralized.

Recall that for a search to be effective, the exploration of the search space has to be balanced by the exploitation of the information gathered. The selection process affects the exploitation part. How the individuals are selected for reproduction as well as for survival can alter this balance. The selection phase is a critical factor in the evolutionary algorithm as it can be used to bias the search to the more promising areas in the solution space. This bias is usually referred to as "*selection pressure*." The selection pressure is affected by how parents are selected to reproduce, how many offspring are produced and who survives or dies. Minor changes in the selection pressure can affect the search behavior.

Selection algorithms are usually characterized by the selection pressure they induce. Strong selection pressure means that high fitness value individuals are preferred for reproducing and surviving. On the other hand, if selection pressure is weak, then the search can flounder and not progress towards any good solutions. Thus the selection process leads to the higher or lower exploitation of the information gleaned from the search space depending on the level of selection pressure. Hence selection should be broad-based so that the individuals selected can be used to create offspring via genetic operators, which leads to more exploration of the search space. We will first review the selection algorithms used in GAs.

4.1 Selection Algorithms

The selection algorithm involves two elements, namely, a selection pool and a probability of selection for the individuals in that pool. The selection pool can be either a global or a local pool. A global pool consists of the individuals in the entire population while local pools are formed by some suitably defined criterion for similar or close-by individuals. Thus local pools are subsets of the population and these subsets may or may not overlap. The different selection schemes used to define the probability distributions are fitness proportional, linear ranking, tournament, truncation, (μ, λ), and $(\mu + \lambda)$ selections. Of these, the fitness proportional is the most commonly used selection scheme. We now provide a brief summary of the different selection algorithms.

Fitness Proportional Selection

In fitness proportional selection the probability distribution is defined in terms of the fitness of an individual relative to the fitness of the other individuals in the population. Fitness proportional selection probability is defined as:

$$p_s(i,t) = \frac{f_i}{\sum_{j=1}^{N} f_j}$$

where $p_s(i,t)$ is probability of selecting individual i at time t, f_i is the fitness of individual i, and N is the population size.

Fitness proportional selection can be implemented in many different forms. The roulette wheel selection (De Jong 1975) implementation is similar to the gambler's revolving wheel, except here the sections are proportional to the fitness value of the individuals. Other forms include the variations of stochastic selection like remainder stochastic selection, and stochastic universal selection (Baker 1987).

The major drawback of this selection strategy is that as the number of generations increases, the selection pressure decreases due to reduction in the fitness differentials. Scaling of the fitness values has been suggested as a method to maintain the selection pressure even when the population starts becoming homogeneous. Since the probability is defined in terms of the fitness of an individual proportional to the fitness of all other individuals in the population, global fitness statistics need to be maintained.

Rank Selection

In the rank selection scheme, first the entire population is sorted and ranked according to fitness values. Then an assignment function is defined which will assign each individual their selection probability. The selection probability for each individual is assigned on the basis of its rank in the population and not on its fitness value. The advantage of using rank selection is that, unlike fitness proportional selection, no scaling of fitness values is required (Whitley 1989). The selection pressure can be adjusted by changing the assignment function. When a linear assignment function is used, the term linear rank selection is used. Non-linear assignment functions can also be used but such implementations are very rare. Non-linear assignment functions induce higher selection pressure than linear functions. Baker (1989) assigned the selection probability using the following linear function:

$$p_s(i,t) = \frac{1}{N}\left(min + (max - min)\frac{rank_i(t) - 1}{N - 1}\right)$$

where max is the number of offspring allotted to the best individual in the population, min is number of offspring allotted to the worst individual in the population, and $rank_i$ is the rank of the individual i at time t. The constraints are that the number of offspring allotted to each individual be greater or equal to 0 and the sum of the offspring allotted be equal to N. The limits of the parameter max is $1 \leq max \leq 2$ and min is set to $2 - max$. For example, setting max equal to 2 implies that the best individual gets 2 offspring while the worst individual gets none ($min = 0$). Reducing the value assigned to max below 2 will reduce the induced selection pressure. By setting $max = 1.1$, the selection pressure induced by linear rank and fitness proportional selections will be similar (Baker 1989).

The disadvantage of rank selection is that the population has to be sorted every time before selection to determine the rank. As the population increases, so does the computation time. The advantage is that rank selection can maintain constant selection pressure through the generations, even when the population starts to become homogeneous.

Tournament Selection

A tournament selection involves randomly choosing k individuals from the selection pool, where k is the tournament size. The individual with the highest fitness is declared the winner and selected to be a parent. Every individual in the selection pool has a uniform probability of being selected. Like rank selection, tournament selection induces constant selection pressure independent of the fitness landscape of the applications.

The major advantage in using the tournament selection is that, unlike rank selection, there is no need to sort the population, thus saving computational time. Since binary tournament selection has identical expectation as linear rank, and requires less computational time, it is an ideal substitute to use instead of linear rank. And, unlike the fitness proportional selection, tournament selection can maintain constant selection pressure and hence there is no need to introduce scaling methods.

It is important to note that both linear rank and tournament selection are not affected by the absolute quantitative fitness values but only by the relative fitness values. Thus, unlike fitness proportional selection algorithm, these selection algorithms are insensitive to the transformations of the fitness values, which is why they produce constant selection pressure irrespective of the underlying fitness landscape.

Selection methods from Evolutionary Strategies

The two main forms of selection used in Evolutionary Strategies are (μ, λ), and $(\mu + \lambda)$, where μ is the number of parents or population size and λ is the number of offspring. In these selection methods, every individual in the selection pool has an uniform probability of being selected for reproduction. Unlike the selection algorithms mentioned above, there is no pre-selection of individuals based on fitness. Instead every individual in the population gets to participate in the reproduction process. Once the process of selection is completed, the survivors for the next generation are selected on the basis of fitness. This selection is performed either from the new offspring produced or from the combined set of parents and the new offspring. This is termed as post-selection. In fact the main difference between evolutionary strategies and GAs is this process of selection.

The number of offspring (λ) created is usually much greater than the parent (μ) population but resulting in a constant population size. In (μ, λ) selection, the survivors are selected from the offspring population. The offspring are ranked according to fitness values and the best μ are selected to replace the parent population. In $(\mu + \lambda)$ selection, both parents and offspring compete for survival. The combined set of parents and offspring are ranked based on fitness values and the best μ are selected to form the new population.

Truncation Selection

In truncation selection, a threshold t (where $0 < t \leq 1.0$) is usually defined. The individuals are first ranked according to their fitness values. The selection pool consists of $t\%$ of the best individuals in the population. The remaining individuals are deleted and not included in the selection process. All the remaining members in the selection pool have uniform probability of being selected for reproduction. The disadvantage of this method is that sorting is required before selection.

Selection for deletion strategies

So far we have discussed strategies for selecting individuals for reproduction. Selection of individuals for survival into the next generation is generally ignored because in most traditional GAs the entire parent population is replaced every generation by the offspring population. Recall that such a system is called the generational or non-overlapping GA. In overlapping GAs the survivors are selected based on fitness from the combined set of parents and offspring. Information is required in a central place to perform this survivor selection. Such GA systems are not easily and efficiently decentralized.

We want to have as many dimensions fixed as possible (*i.e.*, held similar to the sequential GA) so that if there is any change in the behavior of the evolutionary algorithm its cause can be traced. We use the generational model for deletion selection in all the experiments discussed in this chapter. The parents are always replaced by the offspring in all experiments unless otherwise stated. We are interested in decentralizing the selection phase so that it can be implemented on parallel machines efficiently. By keeping the semantics the same we can study the behavior of the system, and isolate the effects if any due to decentralization of the selection process.

4.2 Decentralized Selection using Global Pools

Since the selection phase can effect the search performance, any change in its semantics can alter the behavior of the evolutionary algorithm. But we should be able to get more power and speed in computing by performing the selection process concurrently. To do so the centralized selection process has to be decentralized. By maintaining the semantics constant for the two approaches, the expected behavior of the decentralized GA can be predicted using the theoretical knowledge of centralized GAs.

Recall that in most sequential GAs, the selection pool is the entire population (global pool). In all the selection algorithms with the exception of tournament selection, a central controller is required to perform selection. Thus, in order to decentralize these selection schemes and also keep semantics identical to the sequential GA involves maintaining a global controller. This can be done by identifying one processor as a "master control processor" which will act as the global controller, performing the task of assigning selection probabilities and selecting individuals for reproduction. All the other processors can be considered to be slave processors, who perform the job assigned to them by the master controller. This mapping is similar to the synchronous master-slave parallel model described by Grefenstette (1981). The task of selecting individuals for reproduction from the global pool is done only by the master processor thus maintaining constant semantics.

Such a master-slave implementation shows an increase in the decentralized execution time compared to the centralized execution time. However, the performance of the two systems, in terms of the optimum reached, is similar. Most of the execution time in the decentralized implementation is spent in communication between the master controller and the slave processors. The reliability is low in this form of master-slave implementation, as the system can collapse if the master processor goes down.

Hence the next step is to identify ways to increase reliability and also reduce communication overhead. Both these can be achieved by diminishing the master controller's role and distributing the control to the slave processors. This means that the selection method has to be decentralized. Each processor will now be responsible for selecting parents for reproduction. From Section 4.1, it is clear that tournament selection is an obvious candidate to use in decentralized GAs, since there is no need to maintain any global statistics.

In a decentralized tournament selection, we can have as many parallel processes as the number of individuals. Each process can run a tournament of size k to select a parent from the global pool. Thus two sets of tournament of size k is required to select the two parents to produce an offspring. The offspring replaces the individual in each process. This decentralized tournament selection meets our requirements as it can be implemented without any change in semantics.

We know that the binary tournament selection ($k = 2$) is equivalent in expectation to the standard linear ranking scheme in which the best individual gets two offspring and the worst gets none. Since the semantics have not changed the theoretical results are still valid. Hence the performance of the decentralized binary tournament selection should be similar to the performance of linear rank selection. However, this decentralized binary tournament selection does still have high communication costs, as each processor is selecting from the global pool of processors. We chose to ignore the communication costs for now and study the behavioral implications, if any, of such an implementation.

We modified GENESIS (Grefenstette 1984) to execute the decentralized form of tournament selection. This modified version is used to perform the experiments. The test suite consists of ten function optimization problems, both maximization and minimization, which were chosen from the evolutionary computation literature. (See Appendix A for a description of these functions.) As mentioned earlier, a generational GA is used with a population size 100. A two-point crossover operator and a bit-flipping mutation operator is used. The crossover probability and the mutation rate are set at 0.6 and 0.001 respectively, which are the settings generally used. No form of any elitism is used.

The usual performance metric for comparison of two GA systems is the "*best-so-far*" curves. Figures 4.1 and 4.2 are representative examples of the results obtained using linear rank and decentralized binary tournament selections. These plots show the best found so far as a function of the number of fitness evaluations executed. Each curve represents the average of 100 independent runs.

From Figures 4.1 and 4.2 it is clear that the decentralized binary tournament selection appears to perform worse than the equivalent linear ranking scheme. Recall

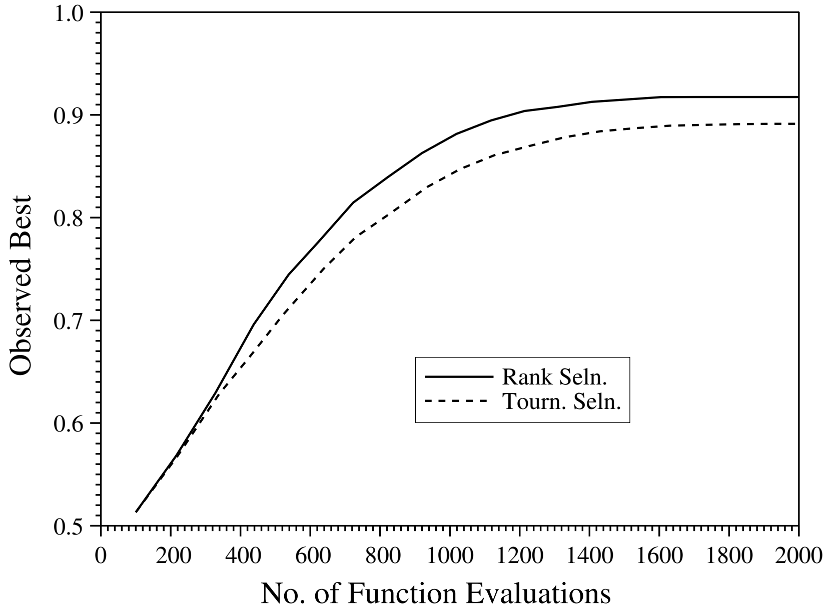

Figure 4.1: Best-so-far Performance for function HC10 using linear rank and decentralized binary tournament selection schemes and population size 100.

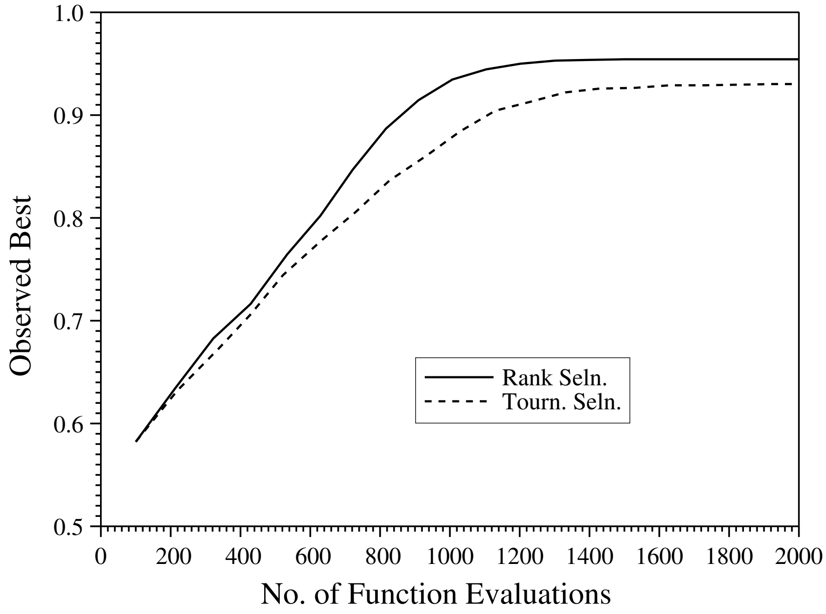

Figure 4.2: Best-so-far Performance for function Peak6 using linear rank and decentralized binary tournament selection schemes and population size 100.

that the performances of linear rank and binary tournament selections should be similar because both induce identical global selection pressure. But here the difference in performance between the binary tournament and linear rank selection is statistically significant. We ran a two-sample t-test and the difference is statistically significant at the 95% confidence level. Even though binary tournament has been shown to be equivalent to linear rank selection theoretically, the difference in performance in real GAs with small populations is significant.

To explain these results better, we analyzed these selection algorithms in a different manner. Here we study the selection pressure induced by the selection algorithm on the entire population at a certain point in time, which in our case is the first generation. It is possible to calculate the expected number of offspring for each individual in the population. Since linear rank and binary tournament selection are identical in expectation, we are interested in determining the variance in the number of offspring from the expected mean for these selection algorithms. Hence, for each selection method, we calculated the expected number of offspring for each individual in the initial population. The same initial population is used for both selection algorithms and all the 100 runs.

We collected statistics on the actual number of offspring produced by each individual parent in generation 1 of a generational GA. For each selection method, the actual number of offspring produced by each individual in the population is kept track of. This procedure of collecting statistics on actual number of offspring produced is repeated 100 times with different random number seeds. The data from the 100 runs on actual number of offspring is used to estimate the variance due to sampling error. The individuals are sorted from best to worst by fitness values for plotting purposes only. Then we plot for each individual in the sorted population its expected number of offspring and the variance from this expectation.

The results for decentralized binary tournament and linear rank selections are given in Figures 4.3 and 4.4 respectively. The black dots indicates the expected number of offspring for each individual and the variance is represented by the error bars. Remember that irrespective of the test function used, the selection pressure induced by both binary tournament and linear rank selections are the same. Hence the graphs for all the functions are essentially the same.

From Figures 4.3 and 4.4 one can see that both linear rank and binary tournament selections have the same expected selection pressure. But the most important point demonstrated in the figures is the fact that decentralized binary tournament selection exhibits considerably higher variance compared to linear ranking. For comparative purposes we also analyzed the effects of fitness proportional selection and the results for two test functions are shown in Figures 4.5 and 4.6. Note that the selection pressure induced by fitness proportional selection is very much dependent on the fitness landscape of the test function. From these figures we can see that the variance in fitness proportional selection is low just as in the case of linear rank selection. This is because linear ranking as well as proportional selection are generally implemented as an "*expected value model*" to minimize the sampling variance (De Jong 1975).

The expected value model was introduced by De Jong (1975) to improve performance in a GA by reducing the stochastic errors. The idea is to ensure that the actual number of offspring an individual receives is as close as possible to that individual's expected number of offspring. De Jong (1975) observed a significant improvement in performance when the sampling variance was reduced by keeping the difference between expected and actual number of offspring to a minimum.

The tournament selection can also be implemented as an expected value model by guaranteeing that all individuals participate in exactly the same number of tournaments (Goldberg, Korb, and Deb 1989). For example, in a binary tournament selection, each individual participates in exactly two tournaments. Hence, this expected value model of binary tournament will have reduced variance and improved performance. However, this expected value model is difficult to implement in an efficient way in a distributed system.

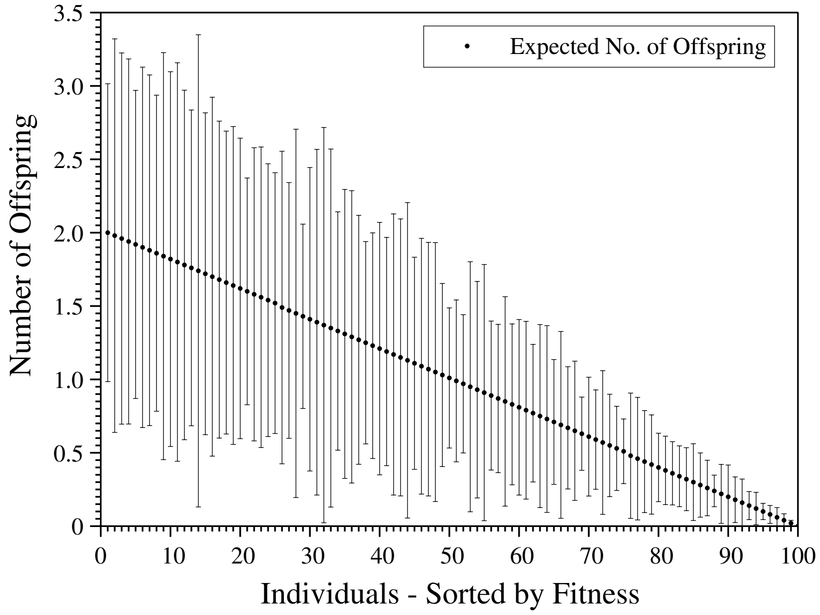

Figure 4.3: Variance in the expected number of offspring using binary tournament selection, and population size 100 for function HC10.

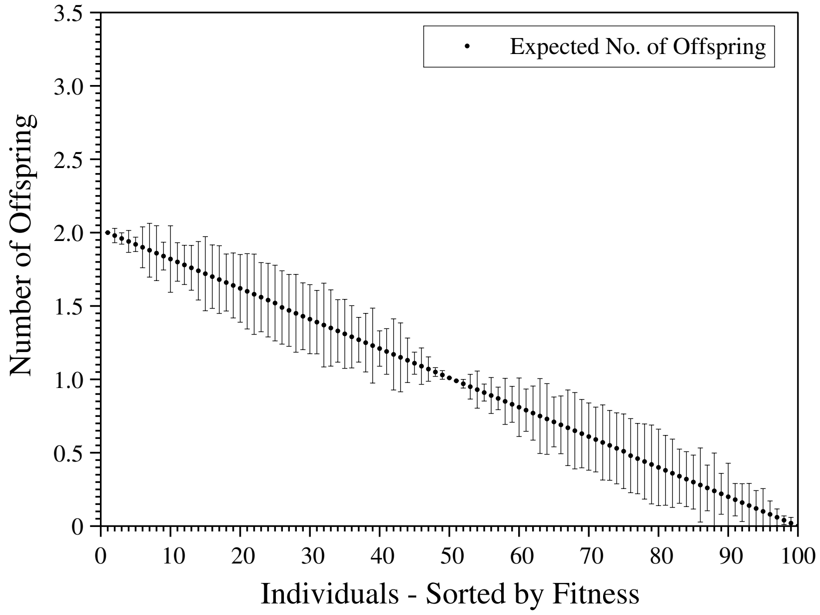

Figure 4.4: Variance in the expected number of offspring using linear rank selection, and population size 100 for function HC10.

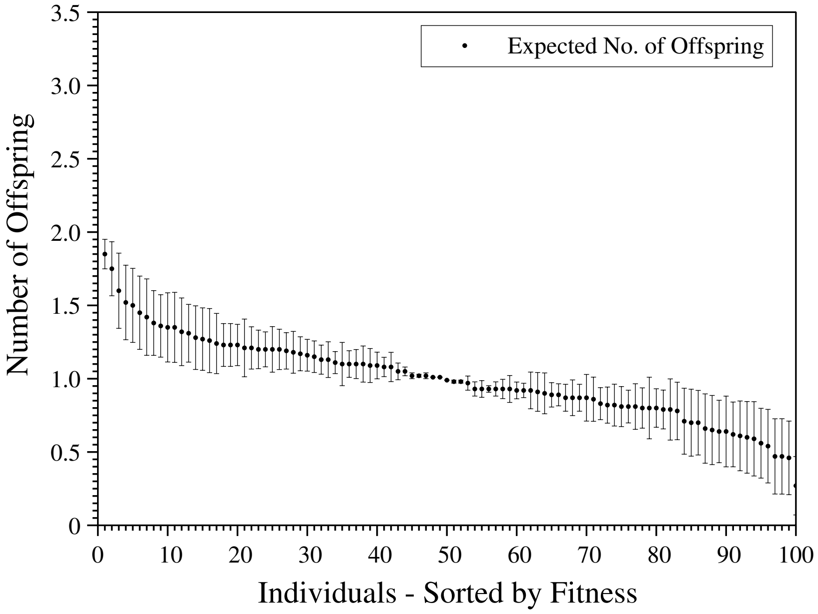

Figure 4.5: Variance in the expected number of offspring using proportional selection, population size 100 for function HC10.

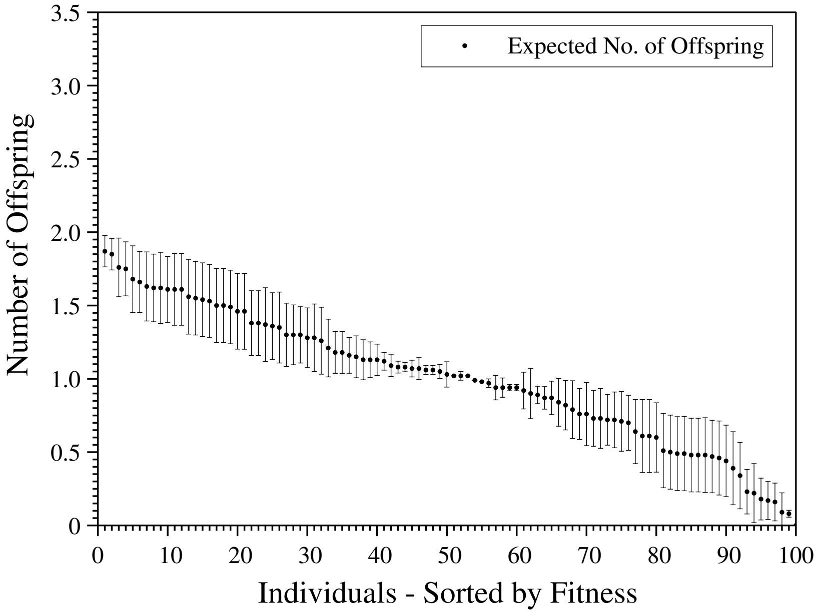

Figure 4.6: Variance in the expected number of offspring using proportional selection, population size 100 for function F1.

4.2.1 Effects of Variance

In centralized sequential GAs with finite populations, higher sampling variance increases genetic drift (allele loss). This is especially true when smaller population sizes are used where the higher variance makes it easier for alleles to disappear. This higher allele loss in turn can have negative effects on the search performance like converging to a suboptimal solution (De Jong 1975). Thus the larger variance of the decentralized form of binary tournament selection provides the most plausible explanation as to why it is always outperformed by a linear ranking scheme which has lower variance.

One way to offset the allele loss in the centralized case is to use larger population sizes. Reducing allele loss improves the performance of the evolutionary algorithm. Hence by increasing the population size we should expect to see a reduction in the "anomalous" results (difference in the performances of linear rank and decentralized binary tournament selections). To test this hypothesis, we performed experiments with several population sizes ranging from 20 to 1024. We observed that differences in the performance of linear rank and decentralized tournament selection decreases as the population size increases. The results are shown in Figures 4.7 and 4.8. Figure 4.7 shows that as the population size increases, the observed best so far performance curve improves when the decentralized binary tournament is used. We observed that the anomalous results disappear when a large population size is used. This result is shown in Figure 4.8 for a population size of 1024. We can see that the observed best performance curve of binary tournament and linear rank selection are almost identical (*i.e.*, the performance difference is not statistically significant).

Thus the effects of the higher variance are not likely to have too much of a negative impact on search performance when decentralized tournament selection is used with large population sizes. However, performing a decentralized tournament selection on large population sizes does not lend itself to an efficient implementation on parallel and distributed systems. An efficient implementation can be achieved by using local selection pools instead of global pools.

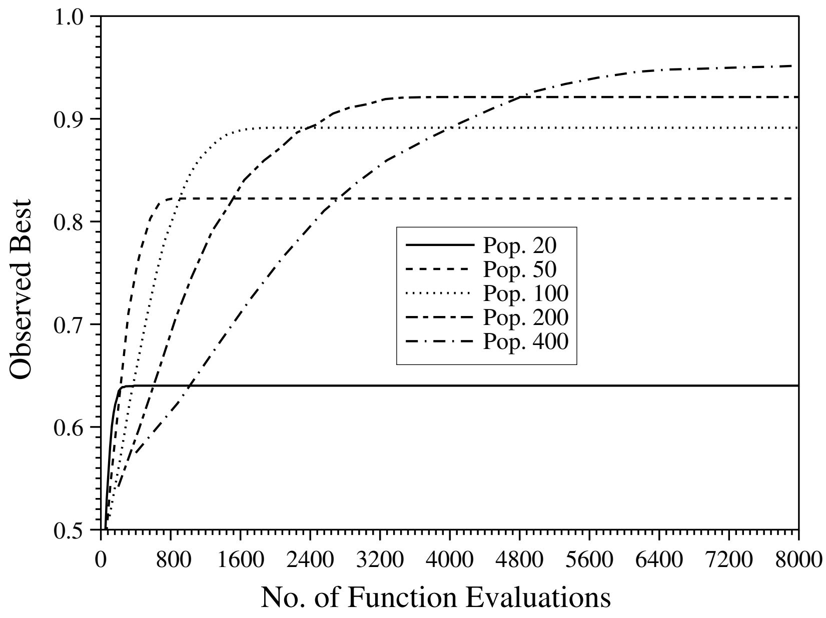

Figure 4.7: Best-so-far Performance using binary tournament selection with different population sizes for function HC10.

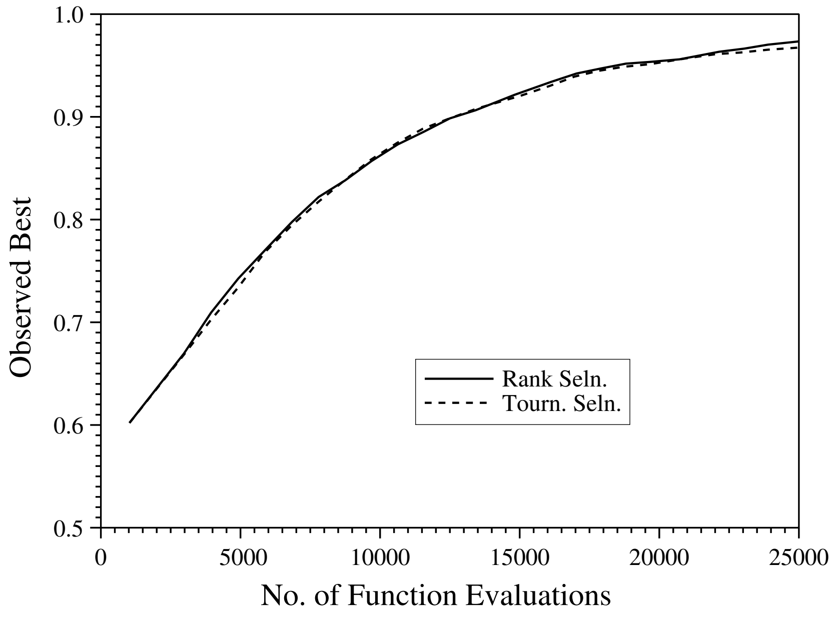

Figure 4.8: Best-so-far Performance using binary tournament and linear rank selection for function HC10 (population size = 1024).

It is generally agreed that tournament selection is easy to parallelize (Goldberg and Deb 1991; Fogel 1994). But as we have shown, when small population sizes are used, the decentralized form of tournament selection with global pools have higher variance. One suggestion is to implement a binary tournament selection in a decentralized manner by ensuring that each individual competes only with its immediate right and left neighbor. This way each individual will compete in two tournaments only. Such an implementation will be an expected value model as each individual now participates in only two tournaments and will lead to lower sampling variance. However, such an approach implies that a topology is imposed on the population, because selection is done from a local pool consisting of the immediate left or right neighbor and not from a global pool. Moreover, the semantics of such a system will not be the same as the sequential GA and can lead to performance differences.

4.3 Decentralized Selection with Local Pools

In the previous section, we concluded that an efficient implementation can be achieved by performing selection from a local pool rather than from a global pool. Local selection pools can be formed by either by defining a suitable measure for similar or nearby individuals. Some of the methods used for defining similar individuals are distance metrics in the genotype and/or phenotype space, or adding a tag template to each individual in the representation space. These cases requires comparison of each individual with the other individuals in the population to determine who belongs to the different local selection pools. Thus it is difficult to implement these measures efficiently in a decentralized form.

Local selection pools can also be created by imposing a spatial topology on the population in which individuals live on a lattice. Neighborhoods for each cell are defined in terms of nearby cells. The individuals in these nearby cells form the local selection pool for each cell. This decentralized and spatially distributed population model mimics natural populations much better than a centralized sequential model.

Biological populations are geographically and spatially located and there is more interaction between nearby individuals. Selection done from a local pool rather than a global pool changes the semantics of the algorithm. Since the semantics have changed, this decentralized local neighborhood model can be considered a new algorithm, for which we need new theoretical results.

Different types of topologies have been proposed and used in the empirical studies (see Section 2.2.2). We decreased the number of degrees of freedom by deciding to use a square two-dimensional torus because this particular topology has been shown to have better performance than lower or higher order dimensions (see Section 3.2). In all the experiments in this section we used a square ($N \times N$) two-dimensional torus. A torus is used so that there are no edge effects. Each individual is assigned to a cell in the two-dimensional grid. Neighborhoods surrounding a particular grid cell are usually defined in terms of the number of steps taken from that cell. Remember that every grid cell has a neighborhood which overlaps with the neighborhoods of nearby cells. This overlap provides an implicit mechanism for the dispersal (migration) of the genetic material throughout the population topology. A GA is concurrently executed in each grid cell with code to select parents from its neighborhood, produce offspring, and replace the current individual assigned to that cell.

Recall that two of the open issues are: "How does the behavior of the evolutionary algorithm change when selection is done from local pools in a decentralized spatially distributed model?" and "What size should the local pools (neighborhoods) be with respect to the population topology?". In the following sections we investigate the effect on the induced selection pressure when selection algorithms are used in local neighborhoods or pools instead of global pools.

4.3.1 Neighborhood terminology

Before discussing the choice of selection algorithms we introduce the various neighborhood patterns that will be used in the experiments. Numerous studies have considered different types of neighborhoods. The reason these different types of neighborhoods were used in the various studies is mainly due to the interconnection network of the parallel machine on which the decentralized GA was implemented. For the same neighborhood size the pattern or shape of these neighborhoods can be different. So we assign a unique label to each neighborhood shape and size combination. Since one of our goals is to understand the effect of the different neighborhood shapes and sizes on the performance of the evolutionary algorithm, we try to include as many different patterns for neighborhoods as possible. Also this gives us a label with which we can refer to the neighborhoods.

We use a two-dimensional torus in all our experiments. Figures 4.9, 4.10, and 4.11 depict only a planar view of the torus. An individual is assigned to each grid location in the two-dimensional lattice and this is indicated by a circle in Figures 4.9, 4.10, and 4.11. Selection is performed in each grid location from the local pool formed by including the individual at that grid location and its neighbors. Another thing to remember is that all neighborhoods overlap, though this overlap is not shown in the figures below. Overlap implies that an individual can be a member in several different local pools. We define three different classes of neighborhoods. As mentioned earlier, the neighborhoods are defined in terms of the number of steps taken from each grid point.

For the first case, the shape label **L** (**L**inear) is assigned to neighborhoods defined as the grid points reachable in n steps taken in a fixed axial direction (north, south, east, or west) from a grid location. Once a direction is chosen, the next step (neighbor) can be in that direction only. This is illustrated in the Figure 4.9. For the central individual (indicated by a square), the neighbors that can be reached in steps 1, 2, and 3, by that individual are highlighted by dark filled circles. The local selection pool for an individual is formed by including itself (central individual) and the highlighted individuals. The number after the shape label indicates the number of individuals in a local selection pool.

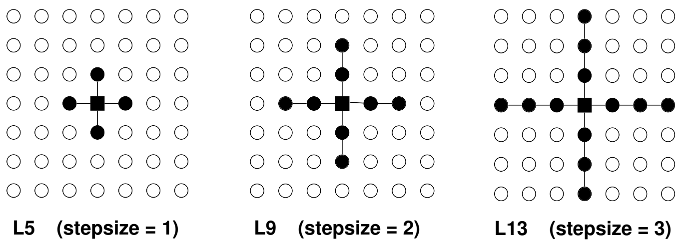

Figure 4.9: Linear Neighborhoods.

In the second case the shape label **D** (**D**iamond) is used for the neighborhood that contains all the grid points reachable in n steps taken in fixed axial direction (Figure 4.10). Once a direction is chosen, in the next step the movement can in one of the four axial directions. Note the difference between this shape and the Linear shape. The neighborhoods under this shape are larger than the ones in case of Linear for the same step size (where $n =$ step size). Only for step size 1, the linear (L5) and diamond (D5) neighborhoods are the same. As the step size increases, the neighborhood sizes in **D** category are larger than in the **L** category.

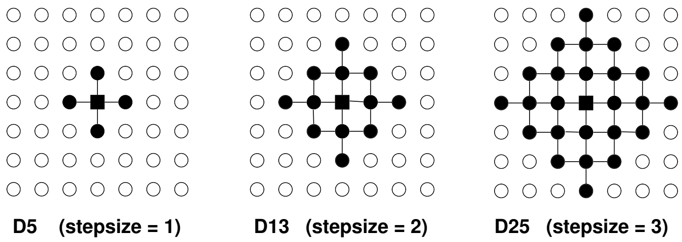

Figure 4.10: Diamond Shaped Neighborhoods.

Finally, the shape label **C** (**C**ompact) contains the points reachable when diagonal movement is allowed in addition to the fixed axial direction movement (Figure 4.11). The neighborhoods in the **L** category are contained within the corresponding **C** and **D** type neighborhoods. The neighborhoods in the **D** category are contained within the corresponding **C** type neighborhoods.

Notice that for the same neighborhood size 9 or 13, the pattern of the neighbor-

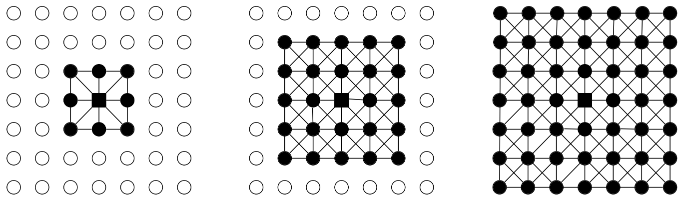

Figure 4.11: Compact Neighborhoods.

hood can be quite different. We have shown only two possible patterns (for example, L9 and C9, L13 and D13), many more for the same size are possible. Also note that, within any shape we can create additional neighborhood sizes by dropping off a few grid locations. We will be using these labels henceforth to refer to these neighborhoods.

4.3.2 Local Selection Algorithms

All the selection for reproduction algorithms described in Section 4.1 can also be used in the case of local pools. However, in this section we examine only those selection schemes which were used in the case of global pools (Section 4.2). Those strategies for reproduction selection are fitness proportional, linear ranking, and binary tournament selections. Random walk selection strategy is not included in this section, even though it has been used as a selection method with local pools (Hillis 1991; Collins and Jefferson 1991; McInerney 1992). Random walk selection strategy does not have a corresponding centralized sequential version against which it can be compared. We will analyze random walk selection in the next chapter.

Each individual in the grid is now performing selection from the local pool formed from the nearby neighbors. Thus the total number of such local selection pools is the same as the population size. Each individual at the grid is following some given rules to select parents and create an offspring. All the individuals follow the same update rule which can be either fitness proportional, or linear ranking or binary tournament selection. These interacting individuals give rise to an emergent global behavior which is hard to predict.

The selection probabilities for the individuals in the local pool are defined using fitness proportional, linear ranking or tournament selection algorithms. In the case of fitness proportional algorithm, the fitness average is calculated for each such local pool by the central grid individual. The local fitness proportional selection probability can now be defined as:

$$p_s(i,t) = \frac{f_i}{\sum_{j=1}^{n} f_j}$$

where $p_s(i,t)$ is probability of selecting individual i at time t, f_i is the fitness of individual i, and n is the neighborhood (local pool) size.

In case of the rank proportional selection, the individuals in the local pool are first ranked and then assigned local selection probability using the following function:

$$p_s(i,t) = \frac{min + (max - min)\frac{rank_i - 1}{n-1}}{n}$$

where max is the number of offspring allotted to the best individual in the local neighborhood, min is the number of offspring allotted to the worst individual in the local neighborhood, and $rank_i$ is the rank of the individual i at time t in the local neighborhood pool. In all the experiments reported in this chapter, max was set to 2 and min was equal to 0.

The selection for deletion for this set of experiments using local pools is the same as the previous one. Since a generational GA system is used all the individuals in the current generation are always replaced by the offspring. Thus in every generation, each individual at the grid is replaced by the offspring created.

In the local pool case, each individual participates in the selection process of different neighborhoods since these are overlapping neighborhoods. Hence an individual can have a different selection probability in each of these neighborhoods. Thus an individual may have above average fitness in one local selection pool but may have below average fitness in another neighborhood pool. The combined effect of the selection process in these various local neighborhoods can give rise to an *emergent* selection pressure and can be different from the selection pressure induced in a system using global pools. We define emergent selection pressure to be the collective selection pressure arising from the selection pressure occurring in each of these overlapping local neighborhoods.

The main purpose here is to understand the behavioral implications of choosing a particular selection algorithm in an overlapping neighborhood model. (The effects of neighborhood sizes and shapes will be investigated in detail in the next chapter.) Since the focus is on understanding selection algorithm effects, we used only three small neighborhoods of increasing sizes. The neighborhood sizes were fixed at 5, 9, and 13. The labels of the four neighborhood patterns used in the experiments are L5, C9, L9, and D13. The geometries of these neighborhoods are illustrated in Section 4.3.1.

Each such neighborhood defines a local selection pool. The dynamics of parallel GAs operating in overlapping neighborhoods is extremely complex and difficult to analyze formally. We first provide some insights into the choice of local selection algorithms by analyzing them empirically. The different local selection algorithms will be characterized based on the emergent selection pressures induced by them. This emergent selection pressure will be compared to the global selection pressure induced by selection algorithms when global pools are used.

For each grid location two parents are selected to produce an offspring from the local selection pool. Selection of the two parents at each grid location is based on the update rule which in this case is the selection algorithm used. In our experiments all the grid locations use the same update rule (although one can imagine a system in which the grid locations use different selection algorithms for selecting parents). Choosing the two parents via the selection algorithm can be implemented in various ways. One method is to choose both parents from the neighborhood based on the local selection probability distribution. Another method is to choose one parent from the pool based on the probability distribution and select the other at random from the neighborhood. Yet another method is to pick the individual at the central grid to be one parent and select the other parent from the neighborhood based on the probability distribution. In the last two cases mentioned, the individual already chosen, may or may not be part of the selection pool. All these different methods for choosing parents have been implemented by the various studies (see Section 2.3.2). How and which parents are chosen can affect the selection pressure and can lead to different emergent selection pressures.

In our experiments in Section 4.2, two parents were selected based on the probability distribution to produce the offspring. To be consistent, in our experiments in this section two parents are selected at each grid location using the local selection probabilities. The offspring produced replaces the current individual at the grid location. Again a generational model is used as the current population is always replaced by the new offspring. We used the same test suite as in the earlier experiments. Since there are no standard settings for crossover and mutation for the local pool case, we used the same setting as in Section 4.2. The crossover probability is set at 0.60 and the mutation rate is set at 0.001. Again no form of elitism was used. The genetic operators (two-point crossover and bit-flipping mutation) are the same as before. The population size is 1024 set up as a 32×32 torus.

We first compare the local selection algorithms, on the basis of the "best so far" curves. Figures 4.12, 4.13, 4.14, and 4.15 illustrate the performance curves obtained using a 32×32 grid and neighborhood types L5, C9, L9, and D13 respectively. For all the four neighborhoods, we see that the performance of fitness proportional selection is significantly below the others. This observation holds for all the ten test problems used. Linear ranking selection appears to produce the best results in a neighborhood size of 5. As the pool (neighborhood) size increases, binary tournament selection produces search behavior equivalent to linear ranking. Just as increasing the population size reduces the variance for the binary tournament selection in the global pool case, increasing the neighborhood size seems to have the same effect in the local pool case. This can plausibly explain the better performance observed in the case of binary tournament selection as the neighborhood size increases. Another thing to note is that there is some performance improvement when neighborhood size increases from 5 to 9, but there does not seem to be a significant improvement in performance when the neighborhood size increases from 9 to 13. The differences in the performance of these neighborhoods will be explored in detail in the next chapter.

In Section 4.2, we examined the variance from the expected number of offspring in the global pool case. We do a similar analysis for the local pool to understand the characteristics of the emergent selection pressure induced on the entire population by a local selection algorithm. Each individual can now participate in different neighborhoods and thus can have an expected mean in each of these local pools. Thus for the local pool case, the expected mean is the average of the expected number of offspring an individual receives in each of these local pools. As in the case of the decentralized selection using global pool, for each selection method, we monitored the actual number of offspring produced by each member of the population. In order to facilitate comparison with the global pool case, the same population size of 100 was used here too. The topology is a 10×10 torus and the neighborhood size is 5. The same initial population was used for each selection method and each run. This process of collecting statistics was repeated 100 times with different random number seeds in order to estimate the variance due to sampling errors. The results are plotted as if there were a single global pool with members sorted by fitness from best to worst. Figures 4.16, 4.17, and 4.18 illustrate the results for binary tournament, linear ranking and fitness proportional selections respectively.

We can make a few observations by comparing Figures 4.16, 4.17, and 4.18 with Figures 4.3, 4.4 and 4.5 involving a global pool. First, under local ranking and binary tournament selection, members with similar fitness can have quite different expected

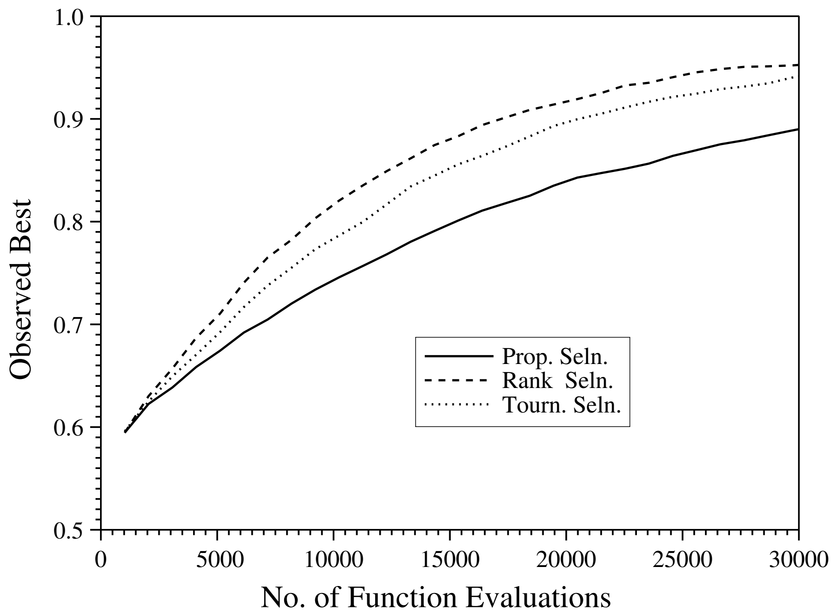

Figure 4.12: Best-so-far Performance for the three selection schemes using a 32×32 toroidal grid and a neighborhood size of 5 (L5) for function HC10.

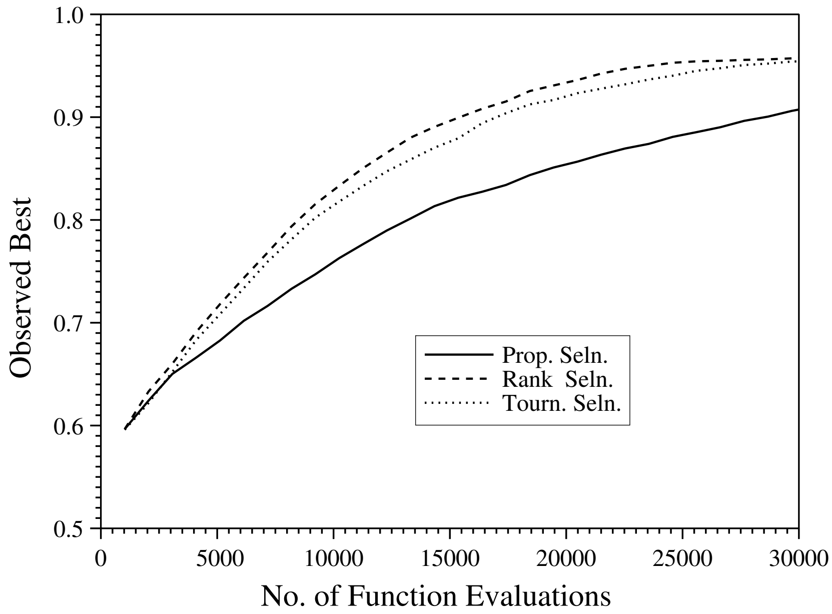

Figure 4.13: Best-so-far Performance for the three selection schemes using a 32×32 toroidal grid and a neighborhood size of 9 (C9) for function HC10.

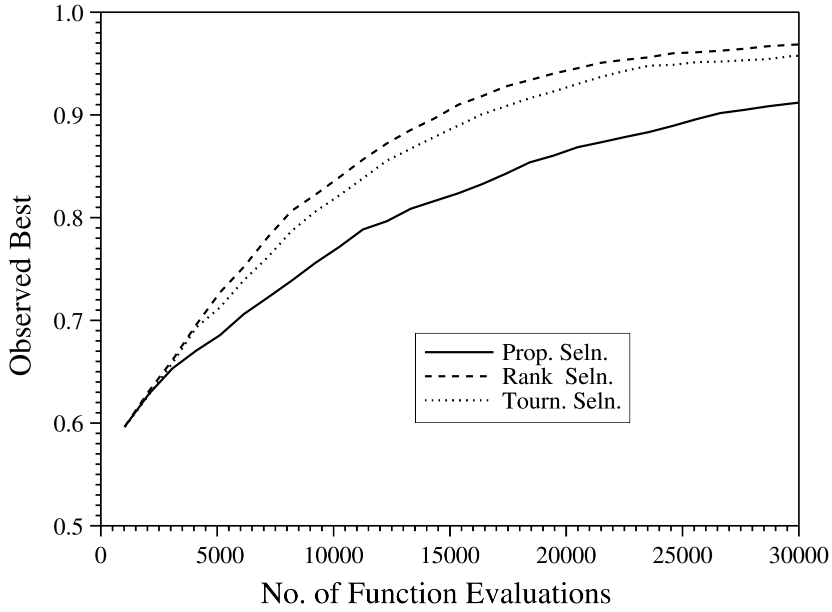

Figure 4.14: Best-so-far Performance for the three selection schemes using a 32×32 toroidal grid and a neighborhood size of 9 (L9) for function HC10.

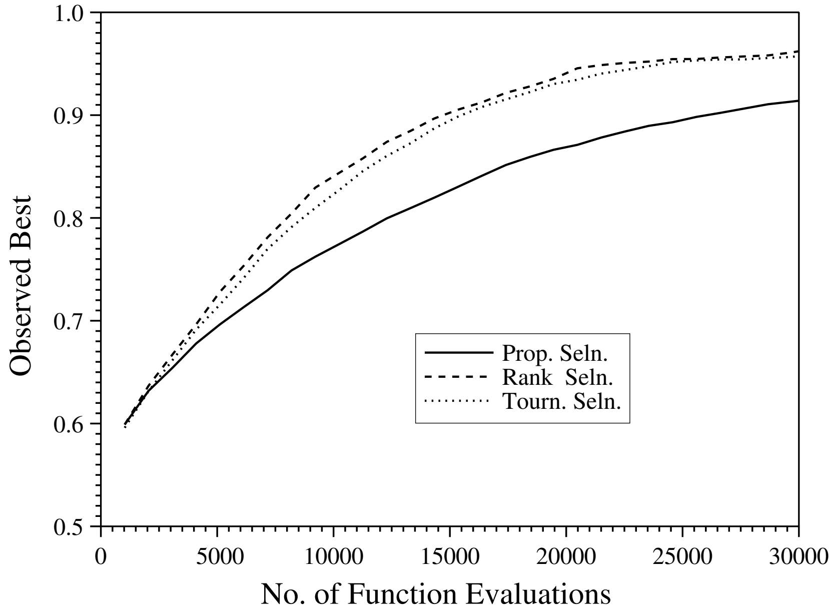

Figure 4.15: Best-so-far Performance for the three selection schemes using a 32×32 toroidal grid and a neighborhood size of 13 (D13) for function HC10.

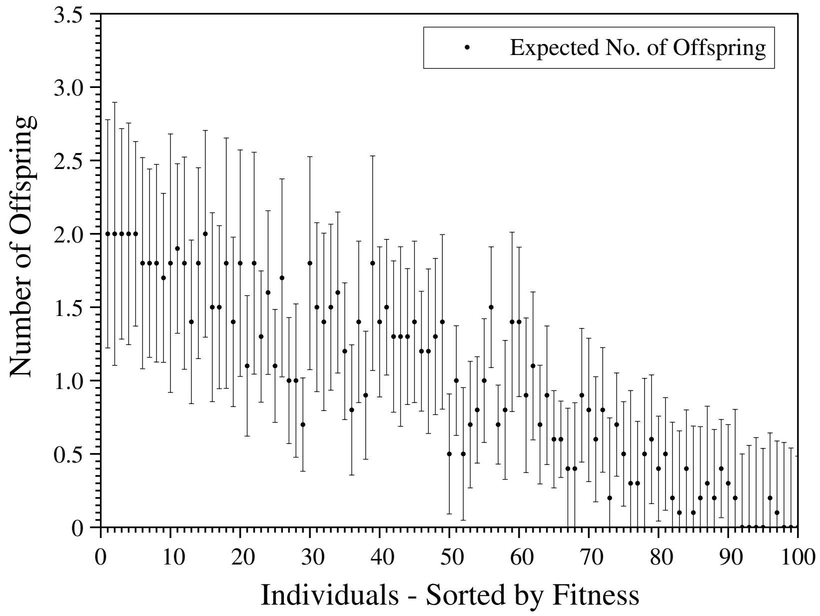

Figure 4.16: Variance in the expected number of offspring using local binary tournament selection, a 10×10 toroidal grid and a neighborhood size of 5 for function HC10.

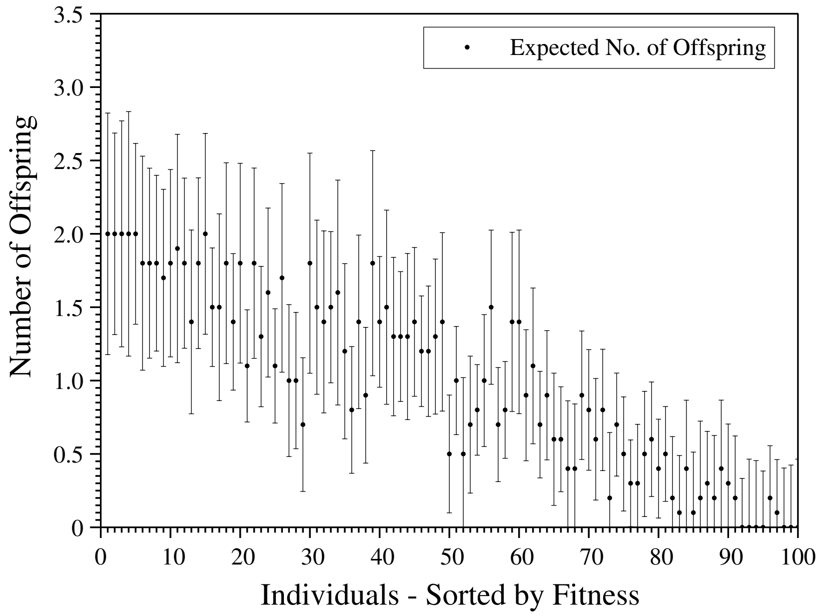

Figure 4.17: Variance in the expected number of offspring using local linear ranking selection, a 10×10 toroidal grid and a neighborhood size of 5 for function HC10.

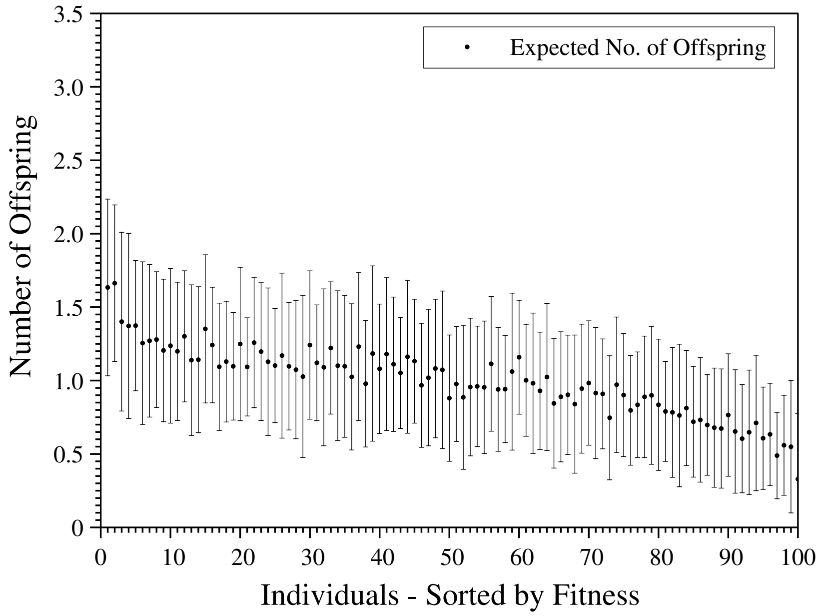

Figure 4.18: Variance in the expected number of offspring using local proportional selection, a 10×10 toroidal grid and a neighborhood size of 5 for function HC10.

number of offspring. Second, in the global pool case the expected number of offspring appears as a straight line, while in the case of the local pool the same appears to be quite jagged. The expected mean is jagged due to the effects of individuals having to compete only in small local neighborhoods. Third, among the three selection algorithms in the local pool case, fitness proportional selection induces a much more uniform but weaker selection pressure than either local linear ranking or local binary tournament selection. The most plausible explanation for this phenomena is that ranking and binary tournament selection produce a constant selection pressure that is independent of the actual fitness values, while proportional selection is quite sensitive to them.

A different way to visualize this induced selection pressure is to use the method of least squares to fit a curve to the observed actual number of offspring. For this we use a population size of 1024 laid out as a 32×32 torus. As before the crossover and mutation rate are 0.60 and 0.001 respectively. The neighborhood size used is 5 (L5) for the local pool selection. Using the same procedure as before we collected statistics on the actual number of offspring each individual receives in the first generation. This was done for both the global and local pool cases. The number of offspring plotted for each individual (represented by a dot) is the average of 100 runs and is sorted by fitness from best to worst. Then the least square method is used to fit a curve based on the actual number of offspring data. In the case of the sequential (global pool) case, the fitted curve is identical to the calculated expected number of offspring line. In the local pool case we can call this fitted curve the "expected number of offspring" and compare this to the expected number of offspring in the global pool case. This comparison is done for each selection algorithm.

Figures 4.19 and 4.20 illustrate the actual number of offspring and the emergent selection pressure induced on the entire population for linear ranking selection with local and global pools respectively. The emergent and the global selection pressure induced by binary tournament selection with local and global pools are shown in Figures 4.21 and 4.22 respectively. Figures 4.23 and 4.24 present the selection pressure induced by fitness proportional selection for the local and global pools respectively.

From the figures (Figures 4.19 to 4.24) it is quite clear that the slope of the fitted line in the case of the local pool case is milder than the global pool case. The other striking observation is that there is lower variance (dispersion) in the case of global pools while there is more dispersion in the local pool system. The milder slope implies that the induced selection pressure in the local neighborhoods is quantitatively weaker than but qualitatively similar to the global pool case. Also, for the local neighborhood model, one can also see that the emergent selection pressure induced by fitness proportional selection is weaker than the binary tournament selection, which in turn is slightly weaker than linear ranking selection.

In some implementations of the spatially distributed GA, the central grid individual of a neighborhood is always selected as the first parent. Then a mate is chosen

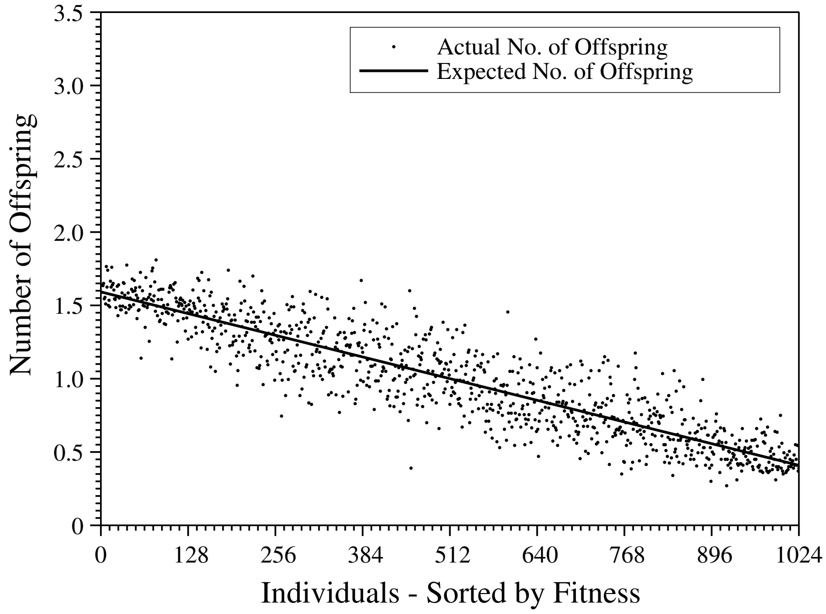

Figure 4.19: The emergent selection pressure induced by linear ranking selection using a 32×32 toroidal grid and neighborhood size 5 for function HC10.

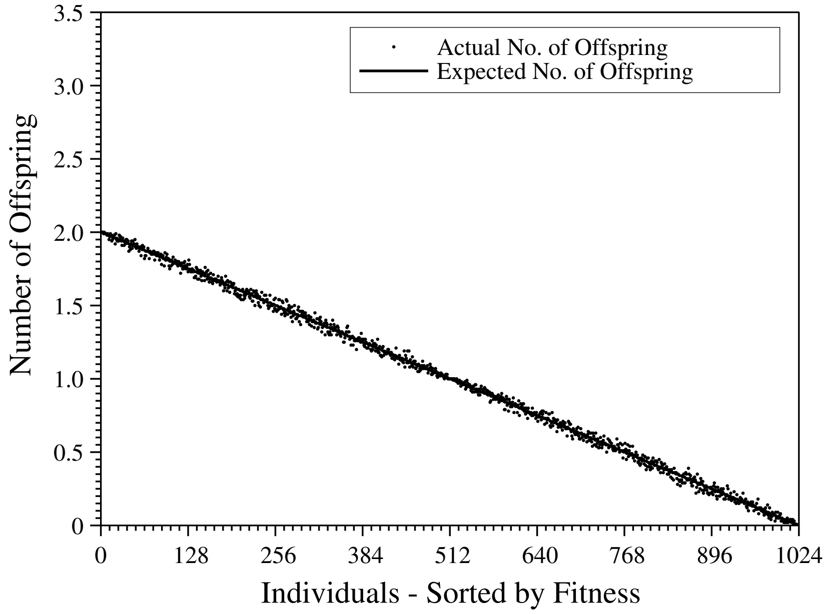

Figure 4.20: The global selection pressure induced by linear ranking selection using population size 1024 for function HC10.

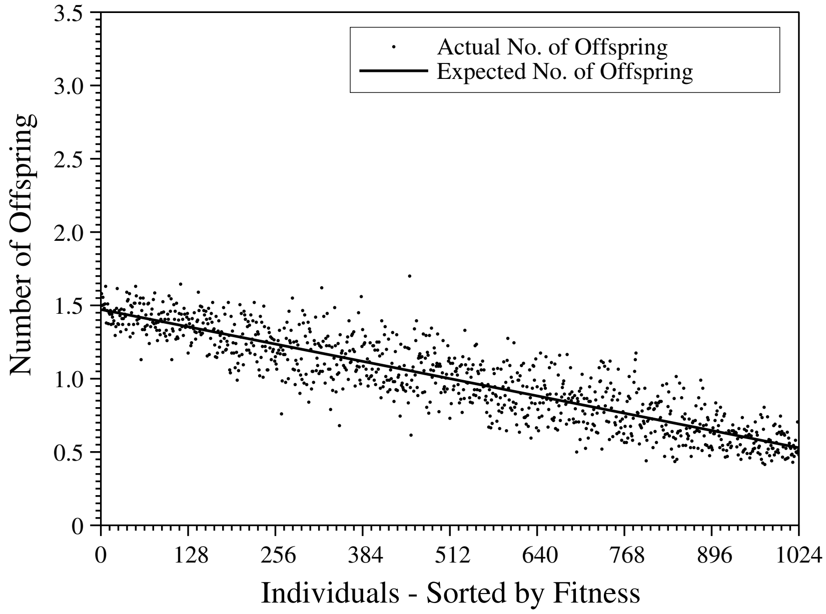

Figure 4.21: The emergent selection pressure induced by binary tournament selection using a 32×32 toroidal grid and neighborhood size 5 for function HC10.

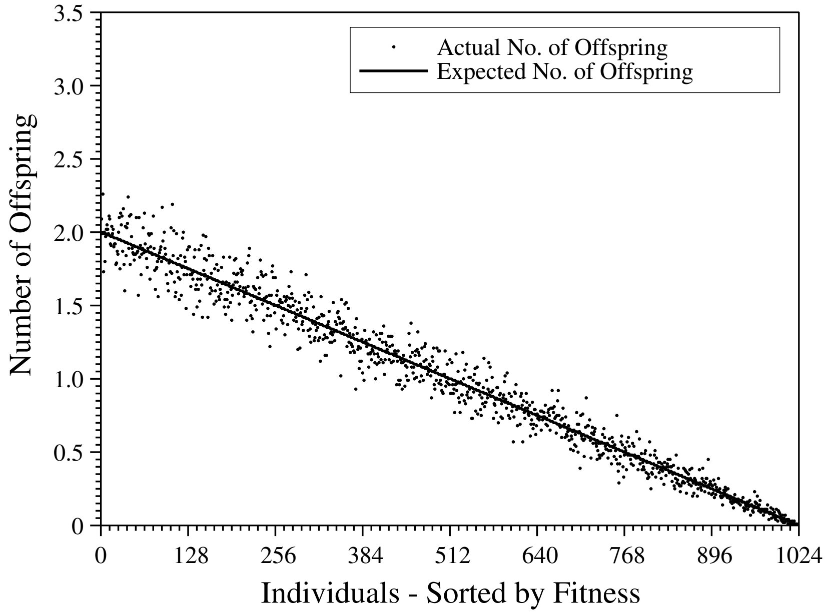

Figure 4.22: The global selection pressure induced by binary tournament selection using population size 1024 for function HC10.

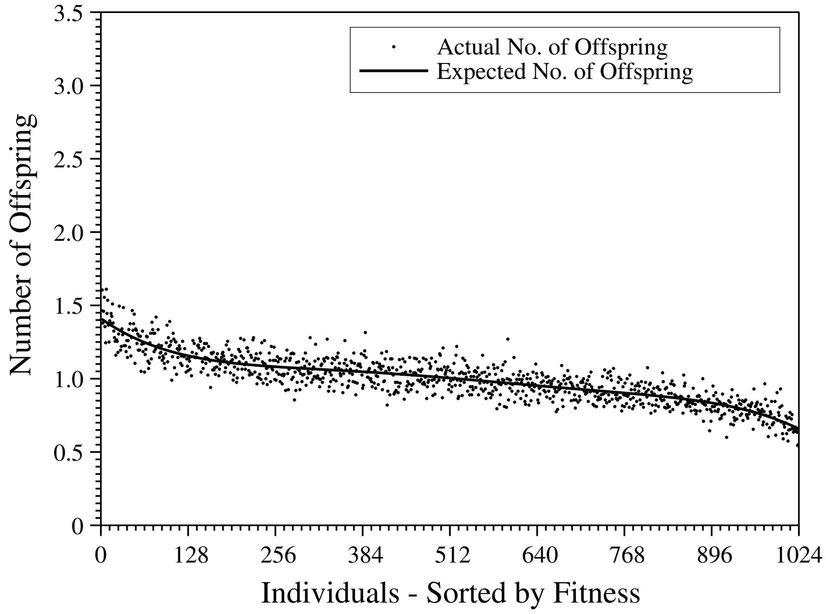

Figure 4.23: The emergent selection pressure induced by fitness proportional selection using a 32×32 toroidal grid and neighborhood size 5 for function HC10.

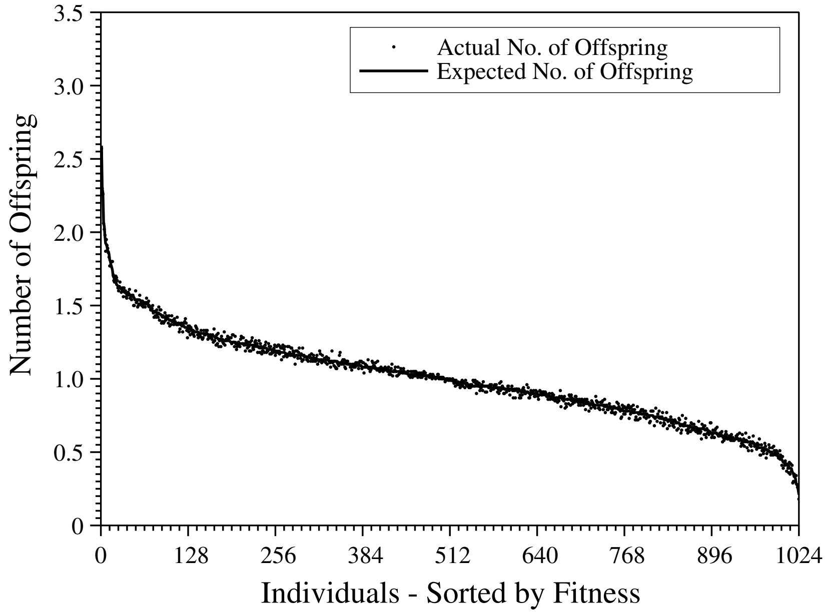

Figure 4.24: The global selection pressure induced by fitness proportional selection using population size 1024 for function HC10.

from the neighborhood based on fitness. We call this the one parent case. This is another difference between the sequential GA and the spatially distributed model. In sequential GAs both parents are chosen based on fitness. Each individual in the population is not automatically ensured to participate in the reproduction process. Selecting the individual at each grid point as one parent and then choosing another parent from the neighborhood pool will have the effect of reducing the emergent selection pressure induced by the selection algorithm. Figures 4.25, 4.26, and 4.27 illustrate the selection pressure induced by the local selection algorithms when the central grid individual is always selected to be one parent and the other parent is selected based on fitness from the neighborhood. From the figures it is clear that the induced selection pressure is quantitatively weaker than the case when both parents are chosen based on fitness.

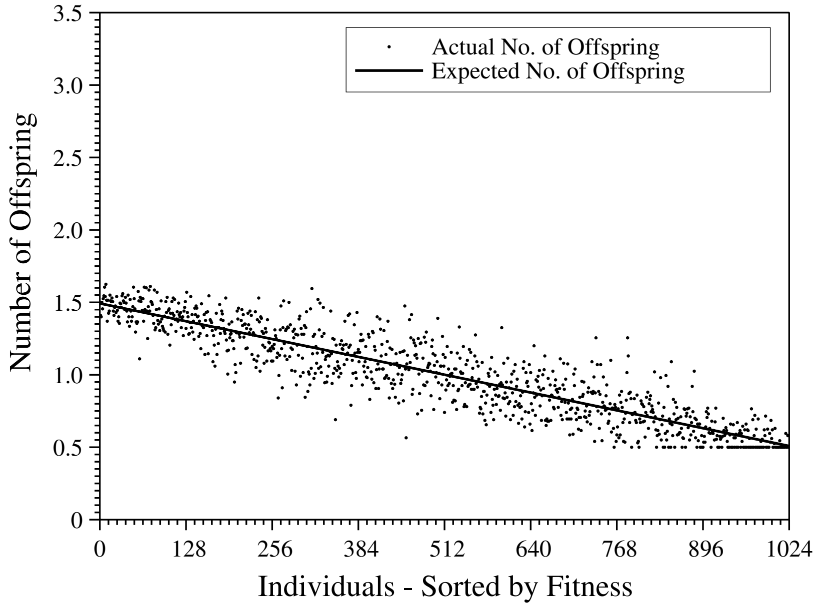

Figure 4.25: The emergent selection pressure induced by linear ranking selection using a 32×32 toroidal grid, neighborhood size 5 for the one parent case.

It should be clear now that we have more degrees of freedom in a spatially dis-

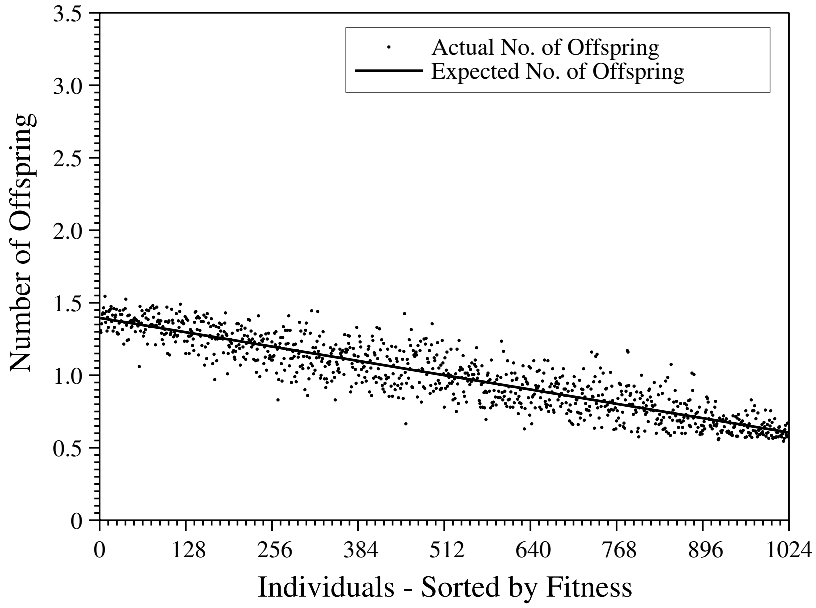

Figure 4.26: The emergent selection pressure induced by binary tournament selection using a 32×32 toroidal grid, neighborhood size 5 for the one parent case.

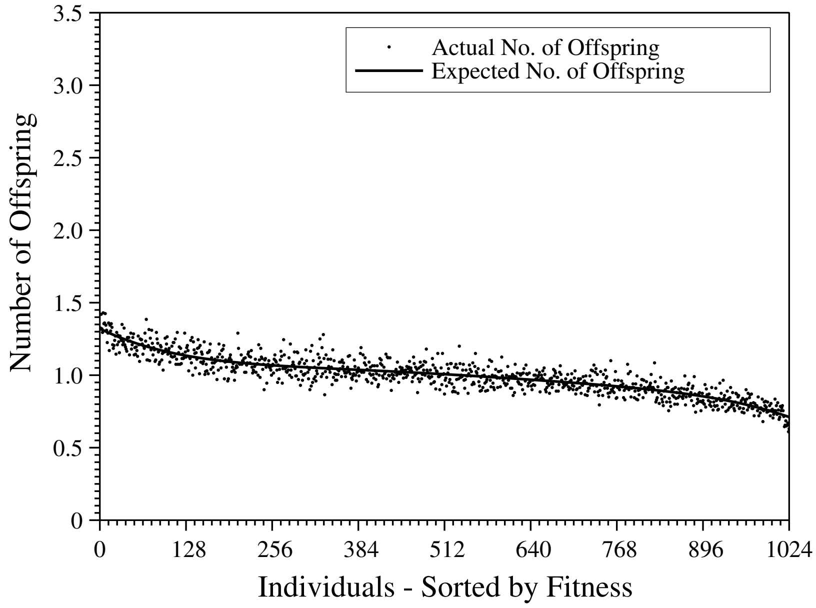

Figure 4.27: The emergent selection pressure induced by fitness proportional selection using a 32×32 toroidal grid, neighborhood size 5 for the one parent case.

tributed GA to control the selection pressure. Picking the individual at the grid cell to be one parent and then selecting the other parent based on probability distribution will weaken the induced selection pressure compared to the case where both parents are chosen probabilistically. Similarly picking one parent at random and the other probabilistically will have weaker emergent selection pressure than when both parents are selected probabilistically.

In addition to insights into local selection methods, these results also provide some initial hints about the role of neighborhood size in spatially distributed models (see Figures 4.12–4.15). We noted that there is a performance improvement in the case of all the three selection schemes when the neighborhood size is increased from 5 to 9. The improvement is negligible when the neighborhood size is increased further from 9 to 13. Is there a way to characterize the effects of neighborhood size? This question will be dealt with in greater detail in the next chapter.

4.4 Summary

We started by exploring how the selection phase could be decentralized when using a global selection pool and yet keep the semantics of the algorithm the same. Of all the selection algorithms examined, tournament selection could be easily decentralized without any change in semantics.

The results of using a decentralized selection algorithm emphasize the importance of an analysis of the variance of the selection schemes. Without it one can fall into the trap of assuming that selection algorithms that have equivalent expected selection pressure produce similar search behavior. Higher variance is strongly correlated with poor search performance especially when small population sizes are involved. Hence to improve performance one must reduce selection variance.

Selection variance can be reduced either by using better sampling techniques and/or by increasing population size. Increasing the population size can lead to an increase in the communication overhead especially if implemented on a parallel machine. Hence, this implies that a local selection pool will have to be used instead of a global selection pool. Local selection pools are defined by introducing a topology on the population. However, this violates our requirement that the semantics should not be changed. Since the semantics are changed, we have new algorithms whose behavior are not well understood.

We studied the selection pressure induced by local selection algorithms in a spatially distributed model. The three standard selection algorithms examined are fitness proportional, linear ranking, and binary tournament. The selection pressure induced by these selection algorithms on the entire population is termed the emergent selection pressure.

We observed that the emergent selection pressure induced by the three selection algorithms is qualitatively similar to its sequential counterpart but quantitatively weaker. By changing the parent selection implementation to always choosing the central grid individual to be a parent and selecting its mate from the neighborhood based on fitness, the emergent selection pressure is further weakened quantitatively compared to the sequential GA.

We saw that in a fairly small neighborhood of size 5, the emergent global selection pressure induced by binary tournament selection was quantitatively weaker than linear ranking selection. But as the neighborhood size increased the selection pressure induced by linear ranking and binary tournament were similar. Of the three selection algorithms, the selection pressure induced by fitness proportional selection was lower than both linear ranking and binary tournament selections for the three neighborhoods studied and was dependent on the fitness landscape of the problem. We raised the question, "What is the effect of increasing the neighborhood size and is there some way to quantify it?". We will examine this issue in detail in the next chapter.

Chapter 5

ANALYSIS OF NEIGHBORHOOD EFFECTS

In the preceding chapter, we studied the emergent global selection pressure induced on the entire population by standard local selection algorithms. Using small neighborhood sizes and shapes we presented results which indicated that the emergent selection pressure of a particular local selection algorithm was qualitatively similar to its sequential counterpart, but quantitatively weaker.

In this chapter we extend the results obtained in the previous chapter by examining more closely the effects of local neighborhood size and shape. We generalize the techniques used to analyze selection methods for sequential GAs and apply them to local neighborhood models. In particular, we extend the growth curve analysis to local neighborhood models to understand the effects that the local selection algorithm, the neighborhood size and shape have on the emergent global selection pressure.

5.1 Growth Curve Analysis

A standard technique for studying and comparing selection algorithms in sequential GAs is to characterize the selection pressure they induce on the individuals in a population. A simplified GA model is used in which only selection is active. There are no genetic operators like mutation and crossover, hence no new individuals are created in the population. There are two events that can happen in a system with only selection and no genetic operators. One is the selection of an individual for reproduction (birth), and the other is the selection of an individual to be replaced (death) of an individual. In most sequential GAs, all the individuals in the population are replaced each generation, that is, everyone is selected to die. Hence this deletion selection is never explicitly considered.

Selection, reproduction, and deletion drive the system to a state of equilibrium in which one of the members of the initial population takes over the entire population. Of particular interest in this simplified system is the growth rate of the best fitness class in the initial population. The growth curves of the best fitness class help visualize the differences in the selection pressure induced by the selection algorithms. The term "*takeover time*" was used by Goldberg and Deb (1991) to indicate the growth rate of the best fitness class in the population in a sequential GA.

In sequential GAs, this growth rate indicates how much selection pressure is being applied by a selection algorithm. Faster growth rates indicate that the selection pressure is high while slower growth rates imply weaker selection pressure. The selection pressure induced by the different selection algorithms, in terms of takeover time, is well understood in the centralized sequential GAs. The takeover time is usually given in terms of the number of generations needed for the best fitness class to fill the population. Table 5.1 gives the takeover times for the different selection algorithms described in Section 4.1 in Chapter 4.

Table 5.1: Takeover times for different selection algorithms in sequential GAs

Selection Algorithm	Takeover Time
Fitness Proportional	$O(N \ln N)$
Linear Ranking $1 < max < 2$ $max = 2$	 $O(\frac{2}{max-1} \ln(N-1))$ $O(log_2 N + log_2(\ln N))$
Tournament k=tournament size	 $O(\frac{1}{\ln k}(\log N + \log(\ln N)))$
(μ, λ)	$O(\ln \lambda / \ln(\lambda/\mu))$

The ideal growth analysis is generally performed using an infinite population model with a finite number of unique phenotypes,which implies that there are only a limited number of fitness values. In an infinite population model, each individual will get its expected number of offspring. Thus the individuals in the population can have fractional offspring. It is important to note that we are dealing with the proportion of each fitness class in the population. For example, assume we have a search space with only 10 unique phenotypes. Also assume that the initial population of the infinite model contains one instance of each of the 10 phenotypes. Hence each fitness class will start with a proportion of 0.1 in the population. The proportions of these fitness classes in the population are updated every generation based on the selection algorithm used. What is of interest here is determining how quickly the proportion of the best phenotype (fitness class) grows from 0.1 to 1.0, for each selection algorithm. The growth curves of the best fitness class for the standard selection algorithms, such as fitness proportional, linear ranking, and tournament selection, are logistic in nature but vary in their growth rates (Ankenbrandt 1991; Goldberg and Deb 1991).

The two questions that arise are: "What does it mean to be an infinite population model in a spatially distributed GA?", and "How does one extend this growth curve model to the spatially distributed neighborhood systems?". We first explain what it means to be an infinite population model in a spatially distributed system. Recall that there are only a finite number of cells in a spatially distributed GA. To understand this extension let us imagine the $N \times N$ torus to be a landscape with N^2 cells. The infinite population is then divided equally into these N^2 cells. What we have now is one infinite population in each cell and the total number of infinite populations is N^2. To mimic an infinite population model, each cell in the torus has a proportion associated with the best and worst fitness class. (For the sake of simplicity, we assume that there are only two unique phenotypes – the best and the worst.) Also recall that in these spatially distributed GAs, each cell has a defined neighborhood and selection is done only from this local neighborhood. Unlike the sequential centralized GAs, in these decentralized neighborhood systems, the proportions associated with the fitness classes have to be updated in each cell independently based on the information from the local neighborhood. Since these neighborhoods overlap, the propagation of the information is done implicitly. The emergent growth curve in a spatially distributed GA is a combination of the proportions from each of these N^2 cells.

In Section 4.3.2 in Chapter 4 we examined the emergent selection pressure induced by the different local selection algorithms. There we examined the selection pressure induced at a particular point in time; we chose the first generation as the time of this analysis. On the contrary, the growth curve analysis is cumulative over time. In the decentralized and spatially distributed GAs, selection is occurring in each of the local neighborhoods and these local interactions give rise to an emergent growth curve. Hence this tool which we are going to analyze in this chapter is quite different from the one we used in Chapter 4. This extension of growth curve analysis will generate some new analytical tools.

For the emergent growth curve analysis, a simplified model is used in which only reproduction is present and genetic operators are absent. We assume that the starting proportion of the best fitness class is $1/N^2$, since we have N^2 cells. This implies that in the initial population, only one cell in the torus contains an instance of the best fitness class while the rest of the cells $(N^2 - 1)$ are instances of the worst fitness class. Our main interest in is finding out how long does it take to increase the proportion of the best fitness class from $1/N^2$ to 1.0. As before the primary concern is how reproduction selection changes the proportion of the individuals in the population. In each cell (grid point) information from its specified neighborhood is collected and the proportions associated with the two fitness classes are updated. This update is done in parallel in all the cells. These proportions are changed using the local selection algorithm and the neighborhood information. Even though this is an idealized model (only selection is used), we can generate some insights as to how to choose a neighborhood size for a given grid size.

This is similar to an $N \times N$ torus in which only one cell is infected and all the other cells are not infected. How quickly the infection spreads through the torus is dependent on the neighborhood pattern. This gives the lower bound or the worst case of the dispersion rate. But each cell in the neighborhood is not infected completely as the local selection algorithm also plays a role. The local selection algorithm determines the level of infection which can slow down the dispersion. Thus the emergent takeover time will depend on both the local selection algorithm and the neighborhood pattern used. The additional element to be taken into consideration for these spatially distributed populations are the effects of the neighborhood patterns.

The emergent growth curve model is first analyzed for the standard selection algorithms. By standard we imply the following selection algorithms - fitness proportional, linear ranking, and tournament selections. Next the analysis is applied to random walk selection strategy which is a commonly used selection algorithm in spatially distributed populations.

5.2 Local Selection Algorithms

In this section the three local selection algorithms studied are the fitness proportional, linear ranking, and tournament selections. The topology used in all the experiments in this chapter is a two-dimensional $N \times N$ toroidal grid. The neighborhood patterns and labels used are as described in Section 4.3.1. We assume that there are only two unique fitness classes. The phenotype labeled best has a higher fitness value than the worst phenotype. Just as in the case of sequential GAs, we are interested in the growth rate of the best fitness class in the population. For any given selection algorithm, we want to track how quickly the best genotype disperses through the entire spatial topology for different neighborhood patterns. This will give us an idea as to how the growth rate of the best individual in the population is affected by the local selection algorithm and the different neighborhood sizes and shapes.

The proportions with respect to the two classes is updated every generation in all the cells. First, in each cell, information regarding the two classes from the local neighborhood is collected every generation. This is done simultaneously in each cell. Second, in every cell the selection probability for the individuals in the local selection pool is calculated. The selection probability depends on the selection algorithm used. Third, the proportion of the fitness classes is updated in each cell. This process is done for each of the three selection algorithms.

We know from sequential GAs that the growth rate of fitness proportional selection is dependent on the fitness ratio ($fitness_{best}/fitness_{worst}$) in the current population. Reducing the fitness ratio in the case of fitness proportional selection will lead to a decrease in induced selection pressure and vice-versa. On the other hand, linear ranking and tournament selections are not sensitive to the changes in the fitness value ratios. In real world applications the fitness ratio varies from problem to problem. Moreover, the fitness ratio can also vary over time for a given function. We experimented with different fitness ratios ranging from 1.1 to 3.0. We opted to use a fitness ratio of 2.0 as it seemed to be a reasonable assumption to make about the expected fitness ratio in real world problems.

In centralized sequential GAs, linear ranking and binary tournament selections are identical in expectation. This is true only when the expected number of offspring for the best individual and the worst individual is two and zero respectively, in linear ranking selection. The parameter max defined in the linear function to assign ranks, determines the expected number of offspring. For example, if parameter $max = 2.0$ then it means that the expected number of offspring for the best individual is 2. Reducing the value of the max parameter reduces the selection pressure induced. We wanted to verify if the equivalence in expectation holds in the spatially distributed GAs also. Hence, in the case of linear ranking selection the parameter max is set to 2.0 for all the experiments.

Figure 5.1 illustrates how the growth rates change when we move from a sequential centralized GA to a spatially distributed population with local neighborhood selection. In this particular case the growth curves exhibited by the best fitness class in a sequential GA using fitness proportional, linear ranking, and binary tournament selections are compared with the growth curves obtained from a spatially distributed GA using a two-dimensional 32×32 toroidal grid and a local neighborhood size of 5.

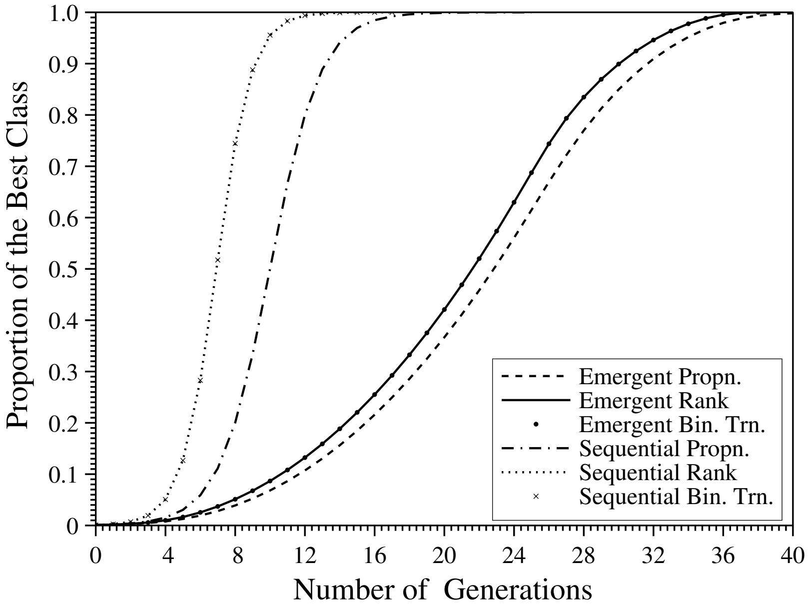

Figure 5.1: Sequential and emergent growth curves for fitness proportional, linear ranking, and binary tournament selection.

We can observe from Figure 5.1 that the spatially distributed GAs exhibit the familiar logistic growth curves but have slower growth rates than the sequential counterpart. Also note that in the case of sequential GAs, the selection pressure induced by fitness proportional selection (for the assumed fitness ratio) is weaker than that of linear ranking selection. From the figure we can see that this rank order of the selection pressure is preserved; using weaker local selection pressure (in this case, fitness proportional selection) induces a weaker emergent global selection pressure. We know from sequential GAs that linear ranking and binary tournament selections have the same expectations and hence have identical growth curves. Similarly in the spatially distributed GA, the growth curves of linear ranking and binary tournament selections are identical.

The intuitive explanation for this phenomenon of having slower growth curve is that it is the result of the combined effects of logistic growth occurring in each of the local neighborhoods and the propagation times necessary to spread the best individual globally throughout a spatially distributed population. The selection algorithm used defines the logistic growth that occurs in the local neighborhoods. In Chapter 4, we showed that, at a particular generation, the emergent selection pressure of a local selection algorithm (fitness proportional, linear ranking, and binary tournament selections) is qualitatively similar to but quantitatively weaker than that produced by fitness proportional or linear ranking or binary tournament in sequential GAs with global mating pools. (For details, see Section 4.3.2 in particular Figures 4.19 to 4.23 in Chapter 4.) As a result, much of our understanding of the selection algorithms in sequential GAs can also be applied to spatially distributed GAs. The second hypothesis involves the effects of propagation times on growth rates. Propagation times depends on the neighborhood sizes used. We now examine the effects of propagation times on the growth rates.

5.3 Local Neighborhoods

The local neighborhoods in the spatially distributed population overlap and this has the effect of implicit migration of information through the topology. The local neighborhood pattern affects how slowly or quickly the migration of information takes place.

5.3.1 Effect of Neighborhood Size

The propagation time can be controlled by the size of the local neighborhood since these neighborhoods overlap. Increasing the neighborhood size will create a larger overlap and hence information will propagate faster through the grid. Similarly reducing the neighborhood size will slow down the propagation. Thus one can see that propagation times are closely related to neighborhood sizes. Hence we can hypothesize that by increasing the local neighborhood sizes, while keeping the grid size fixed will result in higher growth rates of the best fitness class.

To test this hypothesis, the grid size is kept fixed and is a 32×32 toroidal grid. The neighborhood size is first increased from 5 to 9 and then from 9 to 13. The different neighborhood patterns associated with these three sizes are L5, C9, L9, D13, and L13. (Section 4.3.1 contains a description of these different neighborhood patterns. Neighborhood patterns L5, L9, and L13 are illustrated in Figure 4.9, C9 is shown in Figure 4.11, and D13 is shown in Figure 4.10.) The growth curves obtained for fitness proportional, linear ranking, and binary tournament selection algorithms for these different neighborhood sizes are illustrated in Figures 5.2, 5.3, and 5.4.

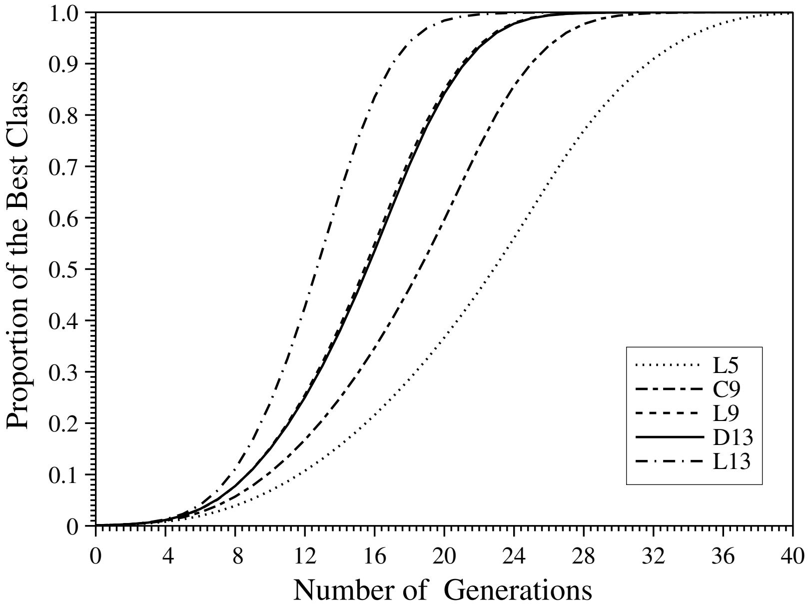

Figure 5.2: Emergent growth curves for fitness proportional selection using different neighborhood sizes.

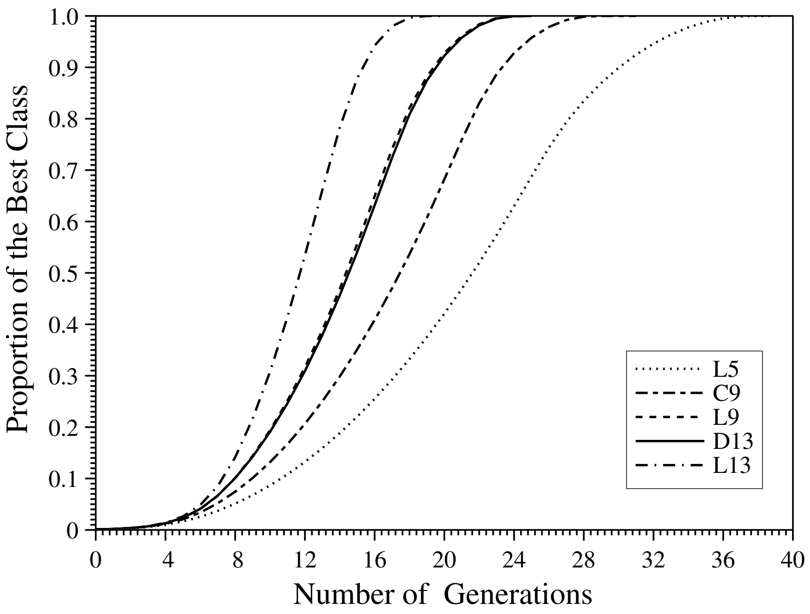

Figure 5.3: Emergent growth curves for linear ranking selection using different neighborhood sizes.

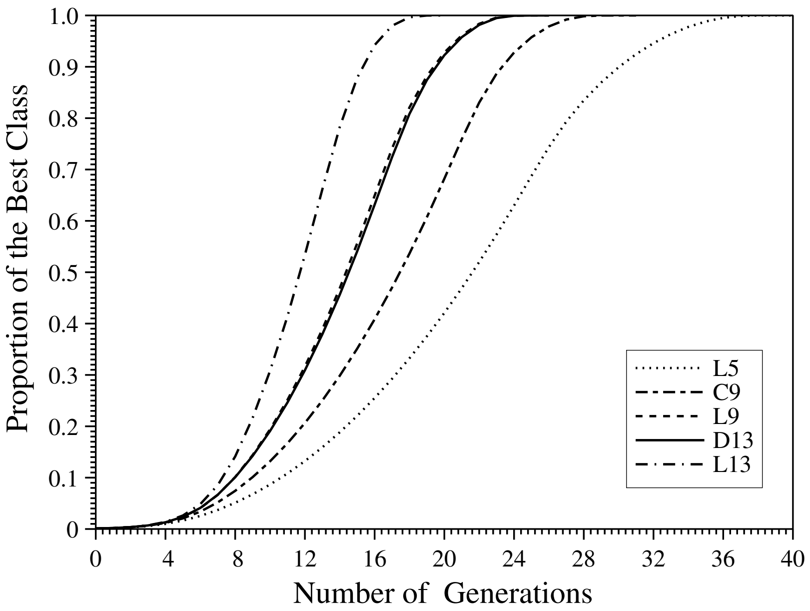

Figure 5.4: Emergent growth curves for binary tournament selection using different neighborhood sizes.

The results are consistent with our expectation that as the neighborhood size increases the selection pressure intensifies. One can observe that the growth curves for neighborhood size 13 is faster than that of neighborhood size 5. However, there is an interesting "anomaly" in the Figures 5.2, 5.3, and 5.4. The L9 and D13 neighborhoods exhibit nearly identical growth curves even though D13 is almost 50% larger than L9. Another observation we can make from the figures is that there is quite a difference in the growth rates of the neighborhoods which have the same size but different shapes, viz., C9 and L9, D13 and L13. The growth rate of the L9 neighborhood is higher than that of the C9 neighborhood. Similarly the growth rate of L13 neighborhood is much higher than that of D13. The anomaly in the growth rates of L9 and D13 as well as the difference in growth rates for the same size neighborhoods is present in all the selection algorithms used. Hence this suggests that the neighborhood shape in addition to the size plays an important role in the propagation times.

5.3.2 Effect of Neighborhood Shape

We will examine why the L9 produces a growth rate nearly identical to D13 and significantly higher than C9. For easier reference, Figure 5.5 illustrates the pattern of these three neighborhoods. Looking at the geometries of these three neighborhoods, the intuitive notion is that what is really important here is the radius of a neighborhood. We need to have an appropriate measure to calculate the neighborhood radius in order to be able to analyze these growth curves.

We can define the radius of a neighborhood to be the radius of the smallest hyper sphere containing that neighborhood. We can use the simple city-block or Manhattan distance measure to define the radius of the neighborhood. With this measure all the three neighborhoods (C9, L9, and D13) have a radii of 2. If the radius is the same for the three neighborhoods then the growth rate of these neighborhoods should be the same. But the growth rates are different as shown in the previous section. This distance measure is not a good one for our purposes as it does not discriminate

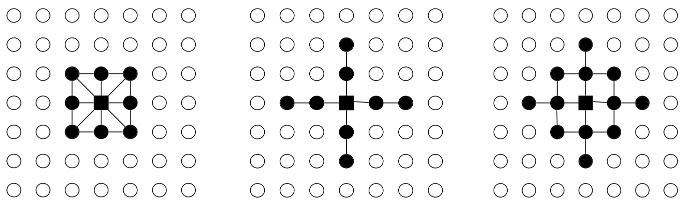

Figure 5.5: Neighborhood patterns.

among the three neighborhoods. The Euclidean distance measure is commonly used to calculate the radius. Using this distance measure the C9 neighborhood has a radius of $\sqrt{2}$ while L9 and D13 neighborhoods have radii of 2. This calculated radius correlates reasonably well with the observed differences in growth rates of L9 and C9 noted in the previous section. However, this definition of neighborhood radius does not distinguish between L9 and D13 which have identical radii, but have slightly different growth rates. Both these measures ignore the finer interior details of neighborhood shape. The internal layout of a neighborhood is important and has to be distinguished. We need a measure that will discriminate the internal shape. Unless the radius is well defined, we will not be able to analyze the local selection algorithms.

To account for the finer internal details, we adopt an alternative definition for neighborhood radius based on a standard distance measure used in spatial analysis (Unwin 1981):

$$\textbf{effective radius} = \sqrt{\frac{\sum(x_i - \bar{x})^2 + \sum(y_i - \bar{y})^2}{n}}$$

which measures the spatial dispersion of a point pattern and can be thought of as the radius of the circle centered on the mean center $(\bar{x}, \bar{y})$ of a neighborhood pattern of n points where

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}, \qquad \bar{y} = \frac{\sum_{i=1}^{n} y_i}{n}$$

Using this definition, the radii of C9, D13, and L9 are 1.16, 1.47, and 1.49 respectively. A closer look at the Figures 5.2, 5.3, and 5.4 indicate that this radius measure correlates well with the observed growth curves, with L9 actually exhibiting slightly higher growth rates than D13. Similarly the radius of the L13 neighborhood is 2.08 which is much larger than the radius of D13 neighborhood. Looking at Figures 5.2, 5.3, and 5.4 we can observe that the difference is growth rates of L13 and D13 neighborhoods is well reflected by their radii. This new measure for calculating the neighborhood radius takes into account the internal layout of the neighborhood in the radius calculation. Note that this distance measure can also be used to calculate the radius of a grid. For example, the radius of a 32×32 grid is 13.058.

5.3.3 Effect of Neighborhood Radius

Now that we can compute neighborhood radius such that the internal layout is also taken into consideration, we can suggest a simple, intuitive model in which selection pressure can be varied as a function of the radii of the local neighborhoods. By increasing the radii of the local neighborhoods, while keeping the grid radius constant, the growth rates can be driven arbitrarily close to the selection pressures of the corresponding sequential GAs. Figure 5.6 illustrates this for three different radii on the same 32×32 grid using fitness proportional selection. Figure 5.7 shows similar results for linear ranking selection. Binary tournament selection data is identical to linear rank selection and hence the plots are not shown, as the graph is the same as Figure 5.7. We can see from the figures that the takeover times (number of generations) gets smaller as the neighborhood radius is increased. Our hypothesis that selection pressure varies as a function of the radius of the local neighborhood can be verified from these figures.

Along the same lines we can now hypothesize that by decreasing the grid size (reduces the radius of the grid) while keeping the neighborhood radius fixed, we should observe similar effects. What we should see for a given neighborhood radius is that the growth rates of the smaller grids should be higher than the larger grid

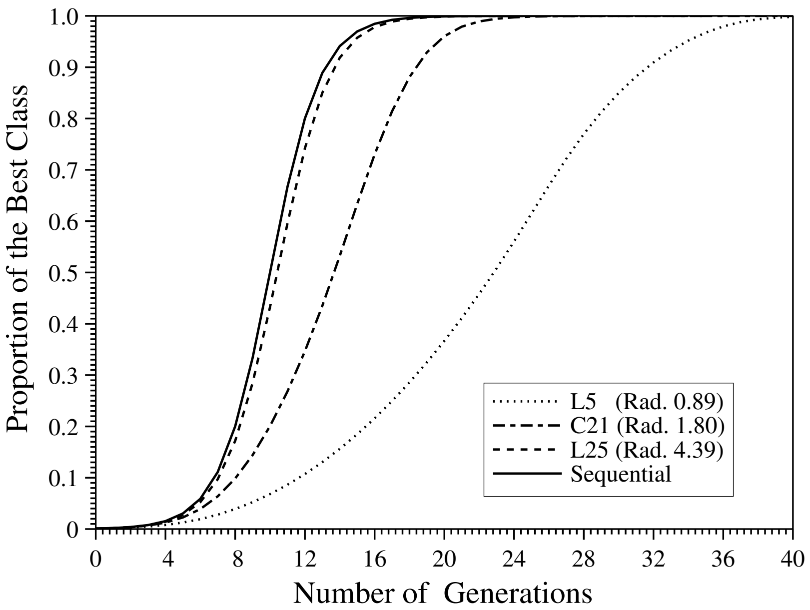

Figure 5.6: Emergent growth curves for fitness proportional selection using different radii.

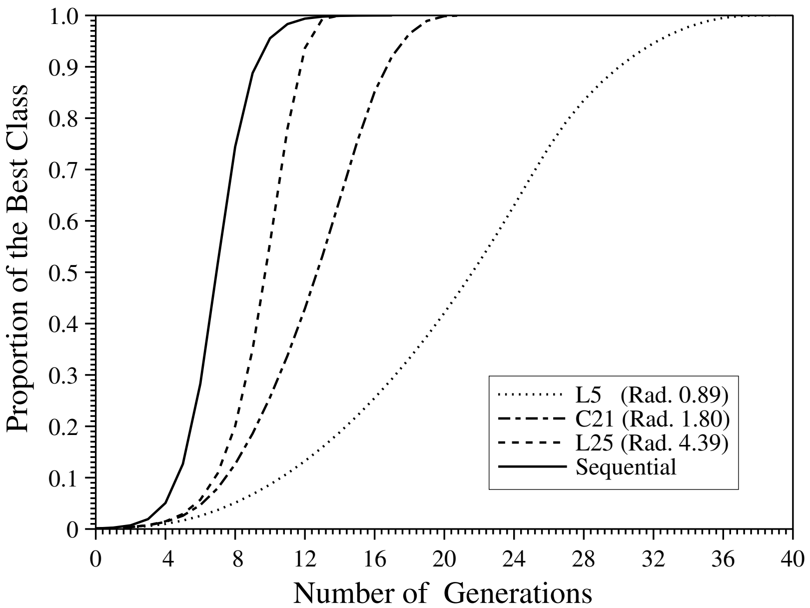

Figure 5.7: Emergent growth curves for linear ranking selection using different radii.

sizes. Figures 5.8 and 5.9 illustrate that this is precisely what happens when the neighborhood radius is kept fixed at 0.89 (neighborhood pattern L5). The grid sizes are reduced to a 16×16 and a 8×8 toroidal grids. We can observe from the figures that as the grid size decreases the selection pressure increases (faster growth curves) for the same neighborhood radius.

These results in turn suggest that the key factor in controlling selection pressure is the ratio of the neighborhood radius to the grid radius. What this implies is that if the radii ratio is kept constant, the growth curves for these different neighborhoods should be the same. Figures 5.10 and 5.11 illustrate that this is precisely the case. Here an L5 neighborhood in a 32×32 grid with a radii ratio of 0.0685 produces a growth curve nearly identical to a C21 neighborhood in a 64×64 grid with a radii ratio of 0.0692.

5.4 A Simple Quantitative Model for Selection Pressure

The observations of the previous section suggest how one might derive a simple quantitative model of selection pressure for these local neighborhood models. Like their sequential counterparts, all of the growth curves generated by the local neighborhood models are well approximated by a logistic equation of the form:

$$N = \frac{N^*}{1 + (\frac{N^*}{N_0} - 1)e^{-at}}$$

where N^* is the asymptotic value of N for very large t and N_0 is the initial value of N (Banks 1994). Since in our case we know that N^* is 1.0 we can rewrite the equation as:

$$P_{b,t} = \frac{1}{1 + (\frac{1}{P_{b,0}} - 1)e^{-at}}$$

where a is the growth coefficient, $P_{b,t}$ is the proportion of the best fitness class in the population at time t, and $P_{b,0}$ is the initial proportion of the best fitness class. The

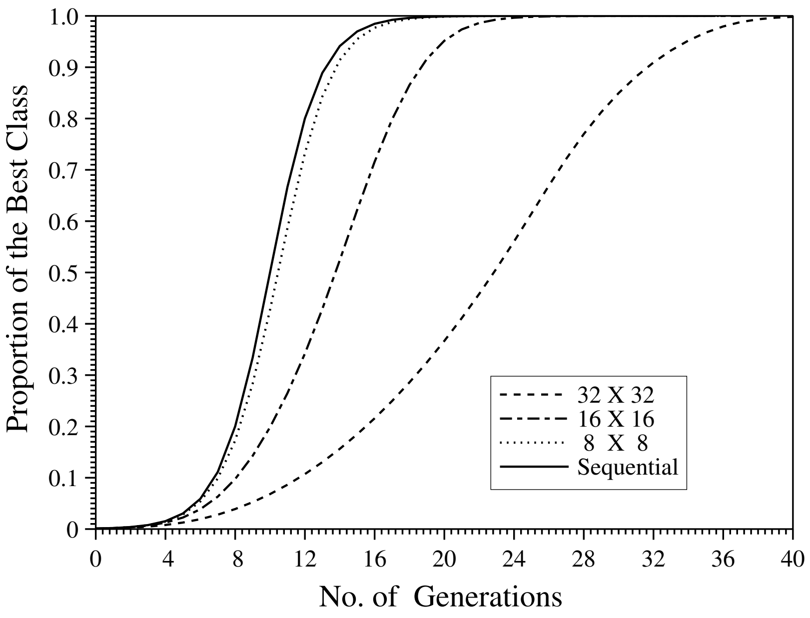

Figure 5.8: Sequential and emergent growth curves for fitness proportional selection using different grid sizes and radius 0.89.

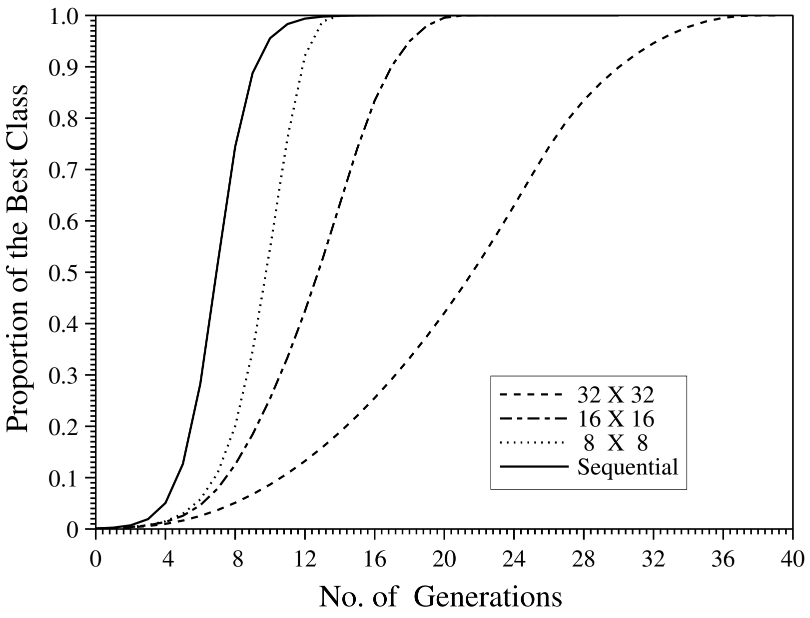

Figure 5.9: Sequential and emergent growth curves for linear ranking selection using different grid sizes and radius 0.89.

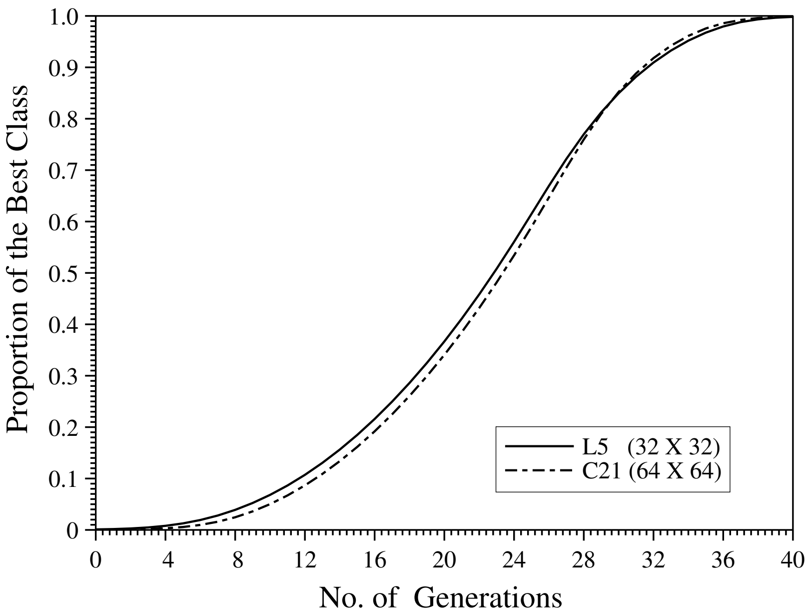

Figure 5.10: Emergent growth curves for fitness proportional selection using constant radii ratios.

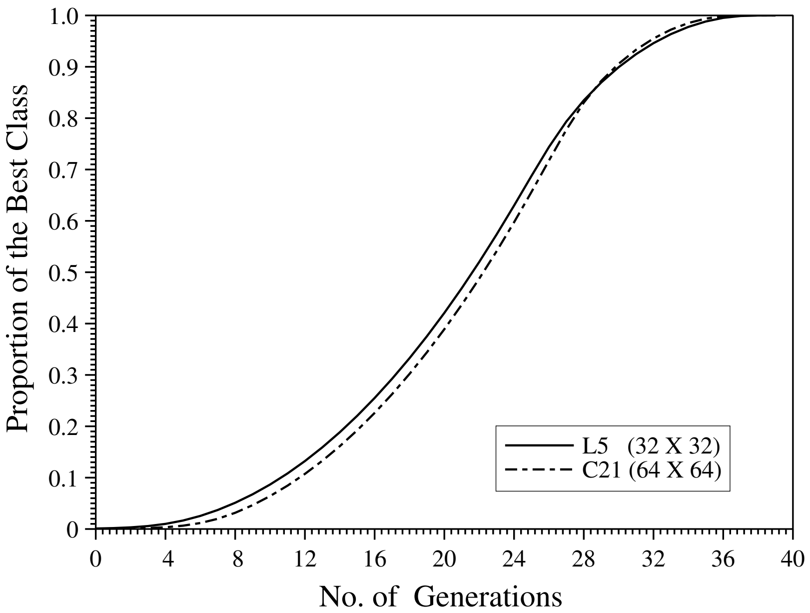

Figure 5.11: Emergent growth curves for linear ranking selection using constant radii ratios.

growth rate is controlled by a single parameter a, the coefficient in the exponent of the above equation.

In the case of centralized sequential GAs, we know that the growth of the best fitness class is a function of the selection algorithm. Hence, the growth coefficient is dependent only on the selection algorithm used. The value of a is the natural logarithm of the fitness ratio for proportional selection and 1 for linear ranking and binary tournament selections (Ankenbrandt 1991; Goldberg and Deb 1991). However, for spatially distributed populations we have shown that the growth curve is a function of both the local selection algorithm used and the neighborhood pattern used. Thus, the growth coefficient a will also be dependent on the ratio of the neighborhood radius to the grid radius. The radii ratio rr controls the rate of propagation of information through the grid.

We need to understand the relationship between the growth coefficient and the radii ratio. To do so, we generate the growth curves associated with various radii ratios in the interval $(0, 1)$. Different grid sizes are used to generate the radii ratios in the given interval. The $N \times N$ grid starts with $N = 8$ and is incremented by 4 up to $N = 32$ and then the size is doubled till $N = 512$. For each selection algorithm, the growth curve data associated with different neighborhoods is generated for these various grid sizes. Tables 5.2, 5.3, and 5.4 give neighborhood radius and radii ratios for two different grid sizes for the three neighborhoods shapes defined. The radii of the 28×28 and 64×64 grids are 11.424 and 26.125 respectively.

The value of the growth coefficient a for the various radii ratios can then be estimated by using a least squares fit of the best logistic curve to each generated growth curve. In order to use the least square fit, the logistic equation has to be transformed into a linear form. The logistic equation given above can be rewritten as follows:

$$\ln\left(\frac{1}{P_{b,t}} - 1\right) = \ln\left(\frac{1}{P_{b,0}} - 1\right) - at$$

This is now in the form of a linear equation $y = c + mx$, where $y = \ln(\frac{1}{P_{b,t}} - 1)$

Table 5.2: Radius and radii ratios for Compact neighborhoods.

Neighborhood Label	Step Length	Neighborhood Radius	Radii Ratio	
			28×28	64×64
C25	2	2.000	0.175	0.077
C81	4	3.652	0.320	0.140
C169	6	5.292	0.463	0.203
C289	8	6.928	0.605	0.265
C441	10	8.564	0.750	0.328

Table 5.3: Radius and radii ratios for Diamond shaped neighborhoods.

Neighborhood Label	Step Length	Neighborhood Radius	Radii Ratio	
			28×28	64×64
D13	2	1.468	0.129	0.056
D41	4	2.613	0.229	0.100
D85	6	3.764	0.330	0.144
D145	8	4.916	0.430	0.188
D221	10	6.090	0.531	0.232

Table 5.4: Radius and radii ratios for Linear neighborhoods.

Neighborhood Label	Step Length	Neighborhood Radius	Radii Ratio	
			28×28	64×64
L9	2	1.491	0.131	0.057
L17	4	2.657	0.232	0.102
L25	6	3.816	0.334	0.146
L33	8	4.973	0.435	0.190
L41	10	6.129	0.537	0.235

and $c = \ln(\frac{1}{P_{b,0}} - 1)$. The method of least squares can now be used to determine the values for c and m. The slope m gives us the growth coefficient a. Since we have the data for each generated growth curve with respect to each radii ratio, we can use that data to compute the "best" values for the slope and the intercept.

Once these values are obtained, we can then plot the value of the growth coefficient a as a function of increasing radii ratio rr. The two limits are: if the neighborhood radius is zero then the growth coefficient is zero, and if the radii ratio is 1.0, then the growth coefficient is equivalent to the sequential GA. There can be no propagation if there is no defined neighborhood and hence there is no growth. Similarly a radii ratio of 1.0 implies that the neighborhood radius is equal to the grid radius, and this is equivalent to selecting from a global pool (the sequential case).

Figure 5.12 illustrates the growth coefficient data for different radii ratios for the fitness proportional selection scheme. The solid line is the fitted line and shows the relationship between the growth coefficient and the radii ratio. Figure 5.13 portrays the data and the relationship between growth coefficient and radii ratio for linear ranking selection. Figure 5.13 also applies to binary tournament selection. We can conclude that there is a clear inverse exponential relationship between the two parameters, rr and a. This implies that increasing the radii ratio will lead to faster growth curves and result in higher induced selection pressure.

Recall that for the purpose of these experiments the fitness value of the best class is assumed to be 2.0 and that of the worst class is 1.0. We can see from the two figures, that for any given radii ratio, linear ranking induces higher growth rates than fitness proportional selection algorithm (for the assumed fitness values of 2.0 and 1.0). Reducing the fitness value of the best class from 2.0 to 1.5 will have the effect of lowering the induced selection pressure in the case of fitness proportional selection algorithm but will make no difference in the case of linear ranking scheme. From Figure 5.12 we can see that for fitness proportional scheme, selection pressure comparable to sequential GAs are achieved already with radii ratio (rr) values of

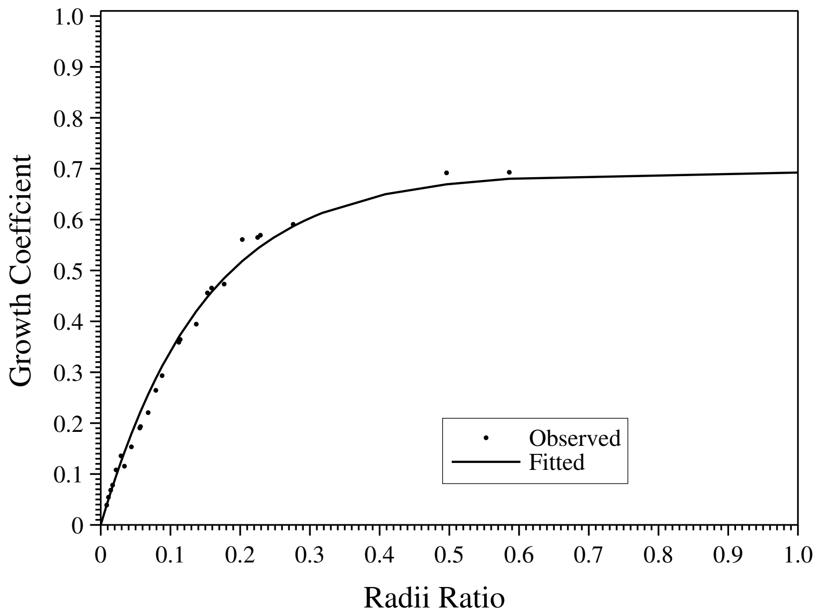

Figure 5.12: Emergent growth coefficients for fitness proportional selection plotted against radii ratios rr.

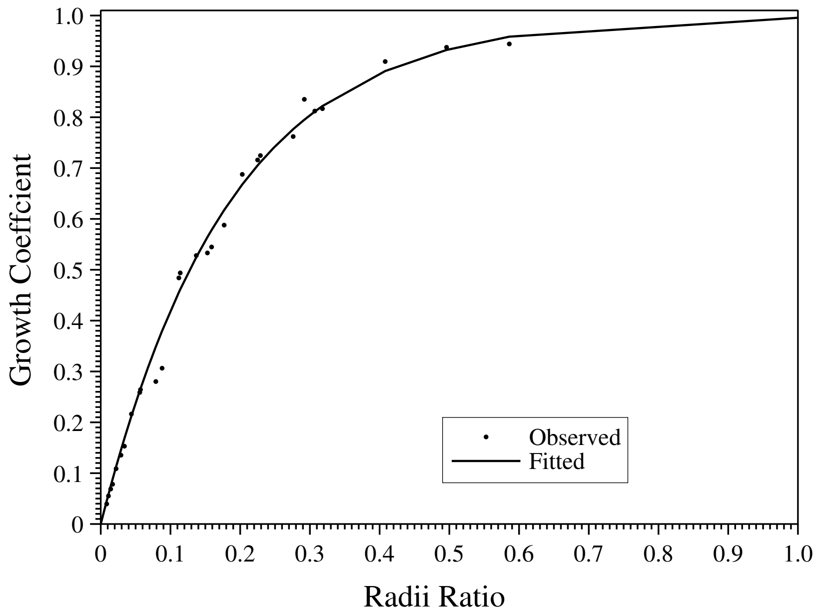

Figure 5.13: Emergent growth coefficients for linear rank selection plotted against radii ratios rr.

0.4. While in the case linear ranking and binary tournament selections, the growth coefficient is greater than 0.5 for radii ratios exceeding 0.15. When the radii ratio is equal to 1, the value of the growth coefficient a corresponds to the sequential case.

We will now briefly examine the emergent selection pressure induced by tournament selection for tournament sizes k greater than 2. From the characterization of selection pressure in the sequential GAs, we know that as the tournament size increases the selection pressure induced intensifies. Figure 5.14 illustrates this effect for a grid size 64×64 and a neighborhood size 13 (D13). The tournament size k is increased from 2 to 4. We can see that the growth curve is higher in the case of $k = 4$ compared to $k = 2$ when both the neighborhood size and grid size are kept constant. The emergent selection pressure induced by tournament selection becomes stronger when the tournament size k is increased.

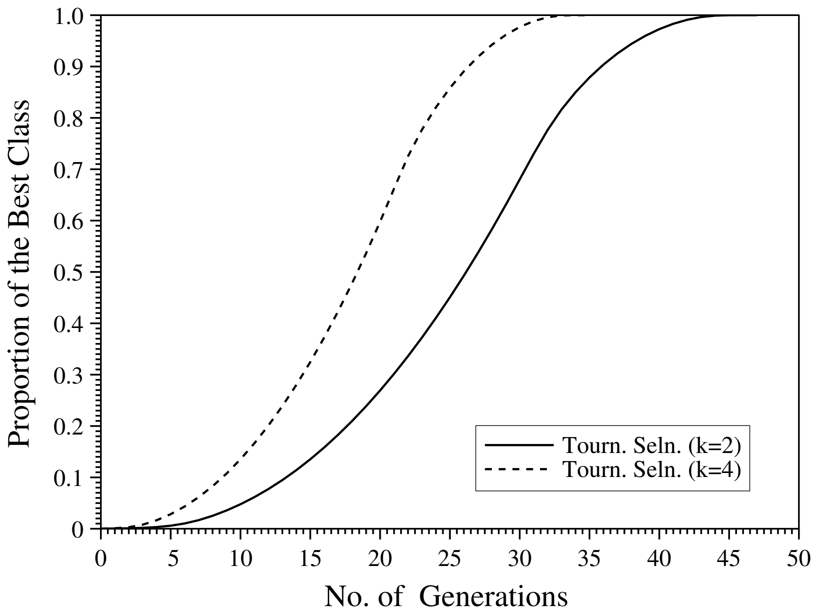

Figure 5.14: Emergent growth curves for tournament selection ($k = 2$ and $k = 4$) using a 64×64 toroidal grid and neighborhood size 13 (D13).

We know that as the tournament size increases, the selection pressure intensifies. Hence what one would expect to see is that for same radii ratios the growth coefficient values for tournament selection ($k > 2$), will be higher than both linear ranking and fitness proportional selections. Figure 5.15 depicts the relationship between the growth coefficient a and the radii ratio rr for tournament size 4. Here also we see an inverse exponential relationship as observed in the other selection algorithms.

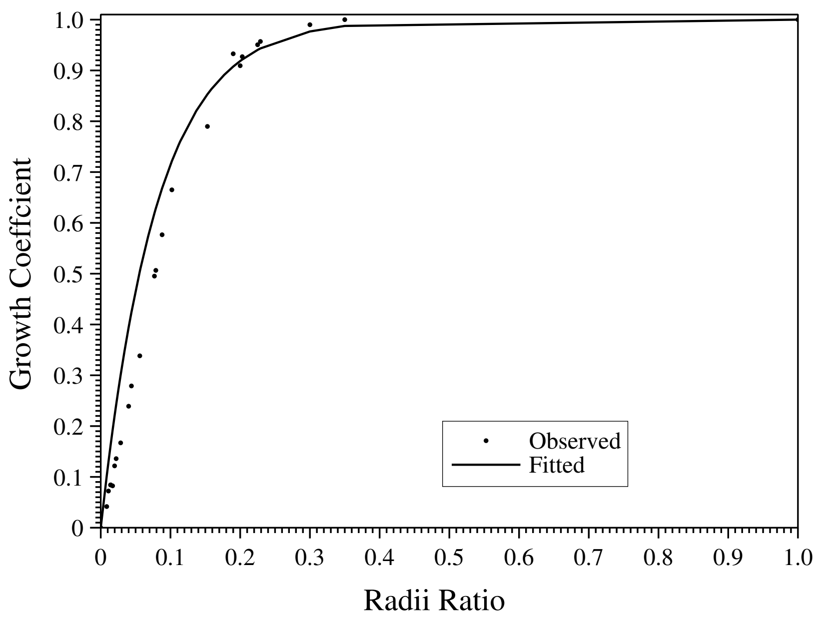

Figure 5.15: Emergent growth coefficients for tournament selection ($k = 4$) plotted against radii ratios rr.

5.4.1 A Predictive Tool

Figure 5.16 portrays a summary of the relationship between the growth coefficients and the radii ratios for the standard selection algorithms. We can see that the rank order of the standard selection algorithms is the same in both sequential and spatially distributed GAs with respect to the induced selection pressure.

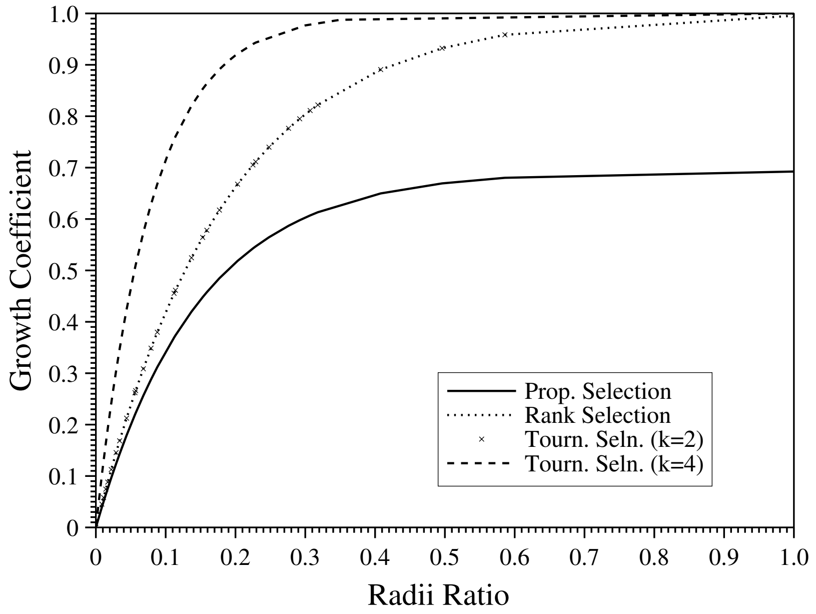

Figure 5.16: Emergent growth coefficients plotted against radii ratios for the standard selection algorithms.

We have now provided an analytical tool which can be used to make predictions about expected behavior of the spatially distributed GAs. Given a radii ratio we can make strong predictions about the expected selection pressure for the different selection algorithms. This tool can also be utilized to make decisions about what neighborhood size and shape to use for a given population topology. The tool can also be used to analyze new forms of selection methods introduced. Random walk selection method which is commonly used in spatially distributed GAs can be analyzed using this quantitative model.

5.5 Random Walk Selection

Random walk is an alternate method that is used to select parents in spatially distributed models (Hillis 1991; Collins 1992; McInerney 1992; Schwehm 1993). The main motivation for the random walk selection method comes from Sewall Wright's work on geographical populations (Wright 1969). Natural populations are geographically separated. Spatially laid-out GA populations resemble natural populations much better than the global population of a sequential GA. In his *isolation-by-distance* model, which uses a spatially structured population, Sewall Wright (1943) assumes a normal distribution between the offspring and the parent locations. A "random walk" of length w provides a normal distribution between the parent and offspring locations. (For derivation, see Appendix B.) Hence, the GA researchers have used selection via random walks to pick parents in a spatially distributed population.

In the standard selection methods discussed in the previous sections, all the individuals within a neighborhood are eligible to participate in the local selection process, while the individuals that are outside of the neighborhood boundary are ineligible to participate. The only bias that exists among the individuals in the local selection pool is based on the fitness value. This is rather a crude distance bias in the sense that the individuals within the neighborhood are "visible" and those outside are "invisible" for the point of view of interaction. Other forms of distance biases can also be defined.

A random walk method provides a distance bias in which the probability of interaction decreases as a function of the distance. That is, the individuals closer to the grid location have higher probability of being chosen while the individuals further away have lower probability. Hence we now have both a distance and a fitness bias in the selection process. Since selection via random walks is similar to tournament selection except for the additional distance bias, we should be able to analyze random walk selection based on the analytical model for the tournament selection introduced in the earlier.

Before analyzing random walk selection, we need to examine the neighborhood patterns that a random walk creates. A random walk of length w induces both a neighborhood size and shape. The exact shape and size of the neighborhoods generated by random walk selection depends on the *length* of the walk and the *directions* traversed. The neighborhood size is the set of grid points reachable in exactly w (the random walk length) randomly chosen steps. The neighborhood shape depends on the directions traversed. We use the same neighborhood terminology introduced in Section 4.3.1 in Chapter 4. If steps are allowed in only four directions (up, down, left or right), the neighborhoods created are diamond (D) shaped (refer to Figure 4.10.) If steps along the diagonal are also permitted, then compact square neighborhoods (C) are generated (see Figure 4.11.) Linear neighborhoods, defined in Section 4.3.1, are not usually generated when a random walk method is used.

The choices regarding the possible directions of movement (four or eight) of a random walk are generally motivated by the interconnection architecture of the underlying parallel machine. The important point to note here is that random walks of length w in eight directions result in significantly larger neighborhoods than those involving only four directions.

5.5.1 Endpoint Selection

Recall that one way to introduce *both* a distance and a fitness bias into the selection mechanism is to perform a tournament selection using random walks. Instead of selecting k members uniformly from a neighborhood, k random walks of length w are performed and the endpoints of the walks are used as the competitors of the tournament. Hence we call this the "endpoint" selection method. Since random walks are more likely to terminate at grid points close to the starting point, individuals in the neighborhood which are nearer the starting grid point are more likely to be selected than those that are further away.

The normal distribution describes the probability distribution induced by a random walk in a continuous space. The question that arises is how do we use that normal distribution to approximate probability in a discrete space? Since an actual random walk of length w creates a well-defined neighborhood, we can enumerate all the paths for the random walk and generate the probabilities in a discrete space.

Thus, it is possible to calculate the exact probabilities for all the endpoints in a random walk of length w by enumerating all the paths. These exact probabilities are used in the experiments described in this section.

We are now interested in the effects of the addition of this distance bias on the overall emergent selection pressure. We will first find out the effect the introduction of a distance bias has on the effective radius of the neighborhood. Using the radius formula given in Section 5.3.2, we can compute the new effective radius of the neighborhood induced by a random walk of length w. We use the probability of ending in a particular cell as weights to compute the effective neighborhood radius. The radius calculation is now weighted and is:

$$\textbf{effective radius} = \sqrt{\frac{\sum p_i(x_i - \bar{x})^2 + \sum p_i(y_i - \bar{y})^2}{\sum p_i}}$$

where p_i is the probability of ending the random walk in cell i.

Tables 5.5 and 5.6 below gives a sample of the effective (weighted) radius and radii ratios for random walks in eight and four directions for two different grid sizes. Comparing the neighborhood radius in Tables 5.2 and 5.5, we can see that the neighborhood radius has reduced in the case of the distance biased endpoint selection. Similar observation can be made for the diamond shaped neighborhoods (compare Tables 5.3 and 5.6). Thus the result of introducing a distance bias is that the effective radius of the neighborhood is reduced. As a consequence we can see that the effective radii ratios in the case of endpoint selection has decreased for the same neighborhood pattern.

This difference in radius between distance–biased and no–distance–bias neighborhoods is not significant when the random walk length is small. For example, the effective radius of C25 neighborhood (random walk of length 2 in eight directions) is 1.732 while the radius of C25 neighborhood with no–distance–bias is 2.00. But as the random walk length increases the difference in radius is large. For a random walk of length 10 (C441) the radius of the distance–biased neighborhood is 3.873. The radius of the same neighborhood (C441) with no–distance–bias is 8.564. In this

Table 5.5: Radius and radii ratios for Compact neighborhoods for endpoint selection method.

Neighborhood Label	Random Walk Length	Neighborhood Radius	Radii Ratio	
			28×28	64×64
C25	2	1.732	0.152	0.066
C81	4	2.450	0.215	0.094
C169	6	3.000	0.263	0.115
C289	8	3.464	0.303	0.133
C441	10	3.873	0.339	0.148

Table 5.6: Radius and radii ratios for Diamond shaped neighborhoods for endpoint selection method.

Neighborhood Label	Random Walk Length	Neighborhood Radius	Radii Ratio	
			28×28	64×64
D13	2	1.414	0.124	0.054
D41	4	2.000	0.175	0.077
D85	6	2.450	0.215	0.094
D145	8	2.828	0.248	0.108
D221	10	3.162	0.277	0.121

case introducing a distance bias decreases the effective radius of the neighborhood by more than half. One point to note from Tables 5.5 and 5.6 is that random walks in eight directions (compact neighborhoods) have larger radii than random walks in four directions (diamond shaped neighborhoods).

In an endpoint random walk of length w and tournament size 2, two competitors are selected for the tournament by performing two random walks of length w and choosing the endpoints of the walk. This selection method is essentially the same as the standard binary tournament selection. The only thing that has changed is the addition of a distance bias on the neighborhood. We can use our inverse exponential model for the standard binary tournament (see Figure 5.13) to predict growth curves for the endpoint selection method. First, we calculate the weighted radii ratio. Second, we use our inverse exponential mapping scheme to predict the corresponding growth coefficient. For example, a random walk of length 3 in eight directions defines a neighborhood of size 49 (C49) and has an effective radius of 2.12. If a grid size of 64×64 is used the weighted radii ratio is 0.081. Using the predicted growth coefficient, we compute the growth curve. This predicted growth curve is compared with the actual growth curve in Figure 5.17. One can see that the prediction in terms of the number of generations to takeover the population is fairly accurate. This suggests that we can use the inverse exponential model of the standard tournament selection for making predictions when an distance biased tournament selection is used.

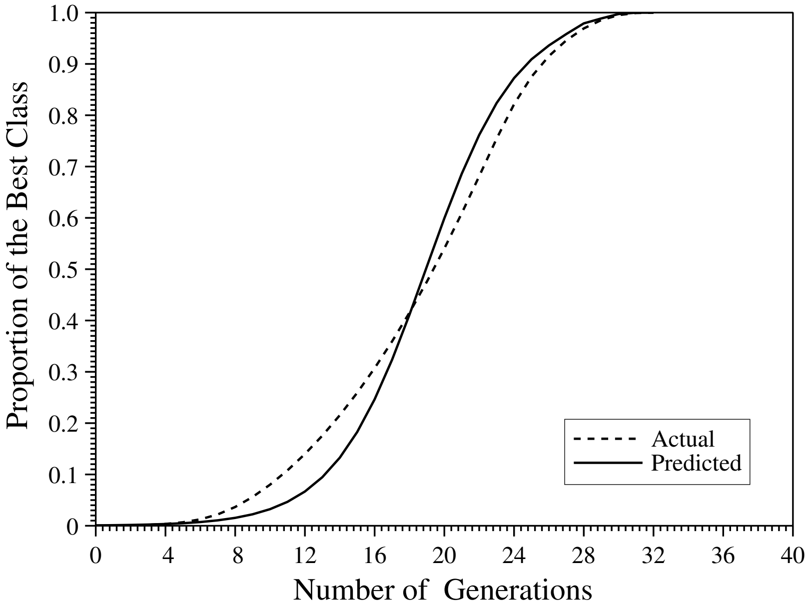

Figure 5.17: Predicted and actual growth curves for binary tournament selection using random walks of length 3 in 8 directions on a 64×64 toroidal grid.

We have analyzed, in this section, an exact probability model generated by enumerating all the paths for a random walk of length w. For small values of w obtaining the endpoint of a random walk is usually accomplished by simulating a random walk. For larger values of w it is generally more efficient to obtain the endpoints by randomly sampling from the underlying induced probability distribution.

An important caveat is that when an actual walk is simulated to select endpoints for the tournament in the case of the random walk in four directions, we will have cells which will never be an endpoint of the walk. That is the probability of ending in a certain cell is zero. We call these locations as the unreachable cells. Such a situation does not arise in a random walk in eight directions, i.e., there are no unreachable states. Hence an implementation of a random walk in four directions, which simulates a random walk to select endpoints, could potentially produce anomalous results, as there may be cells which will never get to participate in the selection process.

5.5.2 Path Elitism Selection

Since an endpoint tournament selection involving multiple random walks can be expensive to implement, a variation involving "path elitism" is sometimes used. In most actual implementations of random walk selection method, a single random walk of length w is performed and every individual encountered during the walk us selected to be a competitor in the tournament (Collins and Jefferson 1991; McInerney 1992). This implies that the tournament size is the same as the random walk length. The best individual found during the walk is selected as a parent. A variation in this implementation is to include the individual at the starting grid point in the competition. By doing this, local elitism is included implicitly. We call this method path elitism selection model since local elitism is included. This selection method is analyzed in this section.

We are interested in understanding how such a change will affect the selection pressure. We would expect the emergent selection pressure of path elitism of random walk length w to be similar to that produced by a tournament of size w using the endpoints of random walks to select the competitors. Again it is possible to enumerate all the possible paths and calculate the exact probabilities of visiting each cell in the neighborhood for a random walk length w. Using these probabilities as weights, the effective radius of the neighborhood can be calculated. A sample of the effective (weighted) radius of the neighborhood and the radii ratios for the path elitism random walk selection are given in Tables 5.7 and 5.8.

Table 5.7: Radius and radii ratios for Compact neighborhoods for path elitism selection method.

Neighborhood Label	Random Walk Length	Neighborhood Radius	Radii Ratio	
			28×28	64×64
C25	2	1.251	0.110	0.048
C81	4	1.788	0.157	0.068
C169	6	2.198	0.192	0.084
C289	8	2.541	0.222	0.097
C441	10	2.843	0.249	0.109

Table 5.8: Radius and radii ratios for Diamond shaped neighborhoods for path elitism selection method.

Neighborhood Label	Random Walk Length	Neighborhood Radius	Radii Ratio	
			28×28	64×64
D13	2	1.045	0.091	0.040
D41	4	1.489	0.130	0.057
D85	6	1.827	0.160	0.070
D145	8	2.110	0.185	0.081
D221	10	2.359	0.207	0.090

We can observe that the weighted radius of neighborhoods for path elitism selection is lower than the radius of neighborhoods in endpoint selection (Tables 5.2, 5.3, and 5.7, 5.8). This is because a single path generated by taking a random walk of length w is contiguous and more compact than the endpoints generated by taking w

random walks. This has the effect of reducing the radius of the neighborhoods even further. For example, the effective radius of the neighborhood created by the single path of a random walk of length 3 in eight directions (C49 neighborhood) is 1.54 compared to a radius of 2.12 for the same neighborhood for the endpoint selection.

Once again, we can use our analytical tools to predict the growth curves for the path elitism selection method. We use the weighted radius to compute the radii ratio and then derive the growth coefficient. The inverse exponential relationship for tournament selection method is used to predict the growth curves for the path elitism model. Figure 5.18 illustrates this by comparing the predicted and the actual growth curves for the path selection in a 64×64 grid.

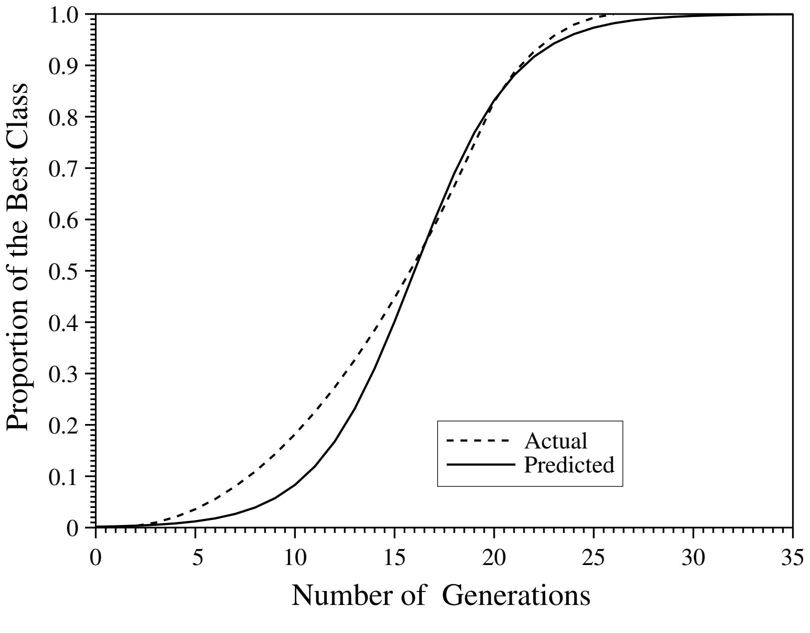

Figure 5.18: Predicted and actual growth curves for path elitism selection using a random walk of length 3 in 8 directions on a 64×64 torus.

We can see from Figure 5.18 that the prediction is not as accurate as before in terms of the number of generations needed to takeover the population. This is because we are using the growth coefficient model for non-elitist tournament selection to predict the behavior of an elitist selection method. Adding local elitism to the model can result in slightly higher selection pressure than if an non-elitist model is used.

The main difference between the path elitism model and the endpoint model is that in the endpoint selection model, the tournament size has to be specified and can be different from the random walk length. In the path elitism model, the random walk length w specifies *both* the length of the random walk *and* the tournament size ($k = w$). For a given tournament size, we can expect the selection pressure induced by the path elitism selection method to be lower than that of the endpoint selection method. This is because the effective neighborhood radius has reduced further in the case of path elitism compared to the endpoint selection. For example, if the random walk length is 10 and the walk is in eight directions then the neighborhood size is 441. The effective (weighted) neighborhood radius for the endpoint selection is 3.783 while for the path elitism selection it is 2.843. The selection pressure induced in the case of endpoint selection is dependent on the tournament size used.

Figure 5.19 compares the growth curves of endpoint and path elitism selection methods for a tournament size of 10 in a 256×256 toroidal grid. The figure also includes the growth curve for a tournament size 2 in the case of endpoint selection. We can see from the figure that the selection pressure induced by endpoint selection method is greater than the path elitism method when the tournament size is 10 as hypothesized. But reducing the tournament size to 2 leads to decline in the selection pressure. In terms of communication costs, performing ten random walks of length 10 is more expensive than performing a single random walk of length 10. So it is a trade-off between communication costs and selection pressure.

The emergent selection pressure induced in a path elitism model is a function of the neighborhood radius and the selection pressure in that neighborhood. The selection pressure in the neighborhood depends on the tournament size used which in turn depends on the length of the random walk. For example, if a random walk of length 5 is used then the tournament size is also 5 assuming that the individual at the

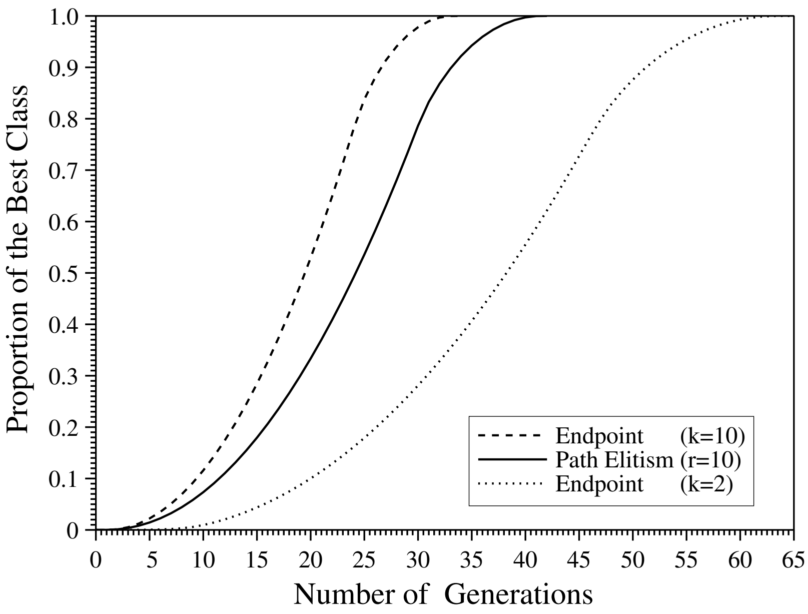

Figure 5.19: Growth curves for endpoint and path elitism selection methods using a random walk of length 10 in 8 directions on a 256×256 torus.

starting grid is not included in the tournament. We know that as the tournament size increases the induced selection pressure intensifies. Therefore in the case of the path elitism model, the induced selection intensifies as the random walk length increases since the tournament size also increases. If the individual at the starting grid is also included in the case of path elitism model, then the tournament size is $k + 1$. The selection pressure can also increase due to an elitist policy.

As before, it is possible to derive the relationship between growth coefficient and radii ratio for the path elitism random walk selection method. Using random walks of length 2, 3, and 4, we generated growth coefficients for different radii ratio for the path elitism selection method. The resulting relationship between the growth coefficients and the radii ratio is still an inverse exponential relationship. Note however the fit is not a good one like the other selection methods. We did not generate a similar curve for the endpoint random walk method because the analytical model of tournament selection can be used to predict accurately the growth curves for the endpoint selection. This, however, is not possible for path elitism selection.

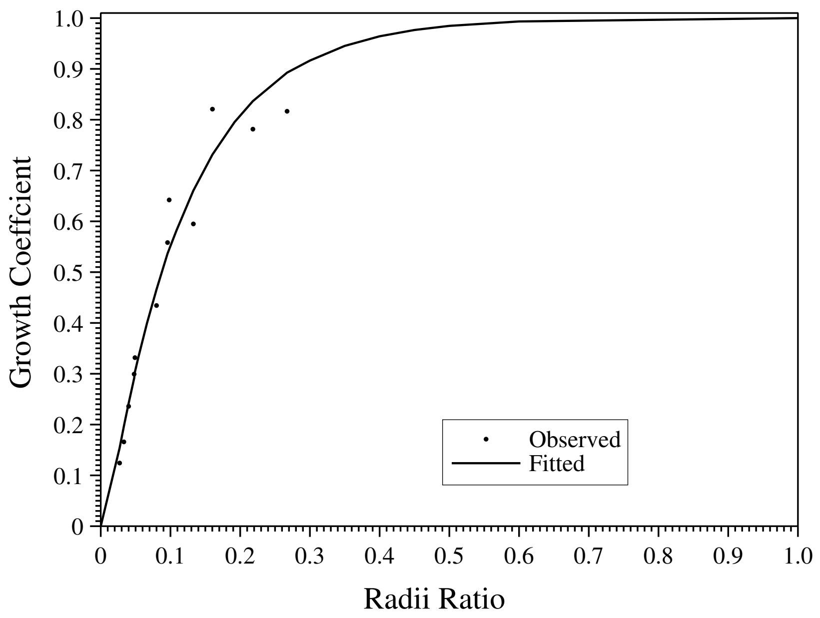

Figure 5.20: Emergent growth coefficients for random walk selection (path elitism) plotted against radii ratios rr.

5.6 Extension to Other Topologies

So far we have been discussing square two-dimensional topologies. We observed that the key factor in controlling the selection pressure in a spatially distributed GA is the ratio of the neighborhood radius to the grid radius. Selection pressure can be intensified by increasing the radii ratio. Similarly reducing the radii ratio can lower the selection pressure. One way to reduce the radii ratio is to increase the grid size while keeping the neighborhood radius constant. We do not have to necessarily use square two-dimensional grids. A rectangular two-dimensional grid can also be used. A particular population size can be laid out in many different ways in two-dimensional form. A rectangular grid will have a larger grid radius than a square grid for the same population size. This in turn decreases the radii ratio and reduces the selection pressure induced if the neighborhood radius is kept constant.

The question that arises is: "Is it possible to say something about the neighborhood shapes and sizes in dimensions greater than 2?". The largest radius that can be achieved is by using a one dimensional topology. We can expect to see the same inverse exponential relationship between the radii ratio and the growth coefficient. As the dimensions increase, the radius of the grid will start to reduce if the population size is kept constant. The radius measure introduced in Section 5.3.2 is a metric than can be easily extended to dimensions greater than two.

For example, using the same definition as before the radius measure for a three-dimensional topology will be :

$$\textbf{effective radius} = \sqrt{\frac{\sum(x_i - \bar{x})^2 + \sum(y_i - \bar{y})^2 + \sum(z_i - \bar{z})^2}{n}}$$

Another point to remember is that, as the dimensions get larger, there is a corresponding increase in the number of neighbors surrounding a grid point. Larger neighborhoods result in an increase the neighborhood radius. This will lead to an increase in the radii ratio since grid radius has reduced and neighborhood radius has increased for a given step length. So increasing the number of dimensions while keeping the population size constant can lead to an increase in the selection pressure induced. For example, a three dimensional grid of $8 \times 8 \times 16$ (population size 1024) has a radius of 5.64 while a two dimensional grid 32×32 (same population size as before) has a radius of 13.06. Even when laid out as a grid of $4 \times 8 \times 32$ it still only has a radius of 9.58.

The radii ratio defines the selection pressure induced. To maintain the same selection pressure as the dimensions increase, the grid radius and the neighborhood radius will have to be selected such that the radii ratio is the same as in a two-dimensional topology. In general, in higher dimensions, larger population sizes will have to be used to maintain the radii ratio.

5.7 Summary

We extended the growth curve analysis used in centralized sequential GAs to spatially distributed GAs to study the selection pressure induced by selection algorithms in local neighborhoods. The selection algorithms examined are fitness proportional, linear ranking, tournament and selection using random walks.

A distance measure from spatial analysis techniques was introduced to compute the radii of the neighborhoods and the population grid. This metric was useful in differentiating the different neighborhood shapes and sizes as it takes into account the internal layout of the neighborhood in the radius calculation. Thus neighborhoods with the same number of members, but not of the same shape, can have different radii.

We showed that the critical parameter is the ratio of the neighborhood radius to the radius of the population grid. The emergent selection pressure induced on the global population can be directly controlled by varying this radii ratio. We also presented a quantitative model of this relationship in which the coefficient of the growth rate of the best fitness class is shown to be an inverse exponential function of the radii ratio.

We have developed a quantitative model which characterizes the selection pressure induced by the different selection algorithms in terms of the radii ratio. Figure 5.21 presents a summary of the inverse exponential relationship between growth coefficient and the radii ratios for the different selection algorithms. Higher radii ratios indicate faster growth rates which in turn implies that the induced selection pressure will be stronger.

From Figure 5.21, we can observe that fitness proportional selection has the weakest selection pressure of the four local selection schemes considered for the assumed

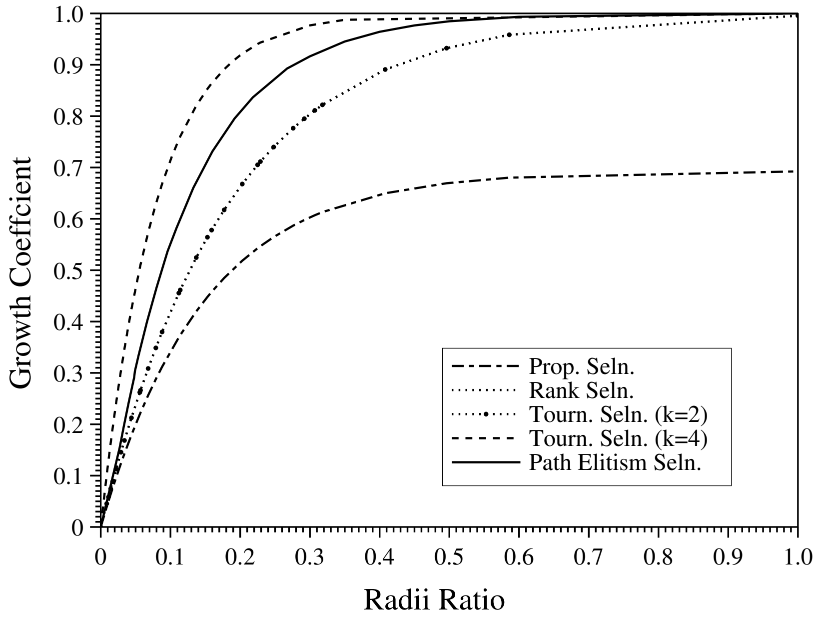

Figure 5.21: Emergent growth coefficients plotted against radii ratios for different selection algorithms.

fitness ratio. Linear rank and binary tournament selections induces slightly higher selection pressure than fitness proportional selection. Tournament sizes greater than two increase the selection pressure. In the case of random walk selection method, the selection pressure induced depends on the method of implementation. Selection via path elitism method induces higher selection pressure than the standard selection algorithms for two reasons. First, implicit local elitism causes higher selection pressure. Second, the tournament size increases as the random walk length increases. Increasing the tournament selection results in higher selection pressure. The selection pressure induced by random walk endpoint selection depends on the tournament size used. One has to consider the tradeoff between the additional communication overhead and perceived benefit of increasing the selection pressure.

This quantitative model can be used to make predictions about the expected behavior based on either growth coefficients or radii ratio. Even though it has been developed using an infinite model with only selection active, yet this analytical model can be used to make predictions about the expected behavior in a “real” spatially distributed GA with finite populations and genetic operators. These predictions will be validated in the next chapter using the function optimization domain. This analytical model can help make design decisions.

Chapter 6

EMPIRICAL VALIDATION

One of the recurring themes in Holland's work is the importance of a proper balance between exploration and exploitation when adaptively searching an unknown space for high performance solutions (Holland 1975). This balance is affected by many different factors like the population size, genetic operators used, selection mechanism, and the complexity of the search space of the problem being solved. The selection process affects the exploitation of the information while the genetic operators alter the exploration of the search space.

The emergent selection pressure induced by the selection algorithm in a decentralized and spatially distributed GA is dependent on both the local selection algorithm used and the neighborhood pattern. This is the major difference between the centralized sequential GAs and the decentralized spatially distributed GAs. In Chapter 4 we showed how the emergent selection pressure induced by the different local selection algorithms is quantitatively similar to its sequential counterparts. This implies that much of our knowledge about the selection algorithms in sequential GAs can be transferred to spatially distributed GAs.

In Chapter 5 we analyzed the selection algorithms by extending the growth curve techniques used in sequential GAs. We provided an analytical model (Figure 5.21) in which the selection pressure could be varied as a function of the ratio of the neighborhood radius to the grid radius. This analytical model provides an insightful analysis of the selection pressure resulting from the choices of selection method and neighborhood patterns. This model allows us to make quantitative predictions concerning the selection pressure produced by particular choices of selection method and neighborhood pattern. For example, by reading horizontally across the graph in Figure 5.21 we are able to determine which combinations of selection methods and neighborhood patterns produce equivalent selection pressure. Similarly, by reading vertically one can determine the quantitative change in the emergent global selection pressure effected by changing the local selection method for a given neighborhood pattern.

This ability, combined with the fact that the emergent selection pressures induced by the various local selection methods is qualitatively similar to the well-understood sequential versions, suggests we can also use these models to make predictions about the actual performance of real spatially distributed GAs on difficult function optimization problems. In this chapter we evaluate these tools to see whether or not they make useful predictions about the performance of spatially distributed GAs on difficult function optimization problems. The result is a better understanding of both the tools and how to design more effective spatially distributed GAs.

6.1 Experimental Methodology

First, we must choose a set of test functions to empirically validate our analytical model developed in the previous chapter. The hardest task is to pick an appropriate set of test functions on which the algorithm can be tested so that the results can be generalized. We know that GAs are robust with respect to various types of fitness landscapes from the vast number of studies performed. Design of a suitable suite of test problems is another extensive research field by itself (Davis 1991; Whitley, Mathias, Rana, and Dzubera 1995; De Jong, Potter, and Spears 1997).

Our test suite consists of problems taken from the domain of function optimization. We ensured that the test suite consisted of scalable functions, unimodal and multi-modal, continuous and discontinuous, noisy functions and Boolean functions.

Both maximization and minimization functions were included. The test functions were chosen from the evolutionary computation literature and include the following: Rosenbrock, Schwefel, Rastrigin, and Griewangk functions, and Boolean Satisfiability problems like the N-Peak and Hamiltonian Circuit problems from De Jong and Spears (1989). The test functions are explained in detail in Appendix A.

Next, we have to define the performance measures to be used to compare these different systems. Since our goal is to predict and analyze performance, the measures chosen should be able to validate the predictions made of the behavior of these systems. The *best-so-far* metric measures the effectiveness of the search process as a function of time (measure in generations). As the name indicates, the best solution (in terms of fitness value) found is tracked every generation. (The population may or may not contain the best value at the end of that run.)

We have to understand the effects of genetic drift in finite populations since it can have an adverse impact on the diversity of the population. Hence the two other performance metrics we chose are *allele loss* and *inbreeding coefficient*.

The allele loss measures the number of gene positions that are completely fixed to either a 0 or a 1. This allele loss metric provides information about the diversity in the population. Diversity is not only required to explore new areas in the search space to find all optimum solutions but also to delay convergence to a single optimum solution. The diversity in the population will determine whether there is any possibility for improvement of the search process. The lower the allele loss, the higher are the chances that the optimal solution will be found.

Inbreeding coefficient measures the similarity of the two individuals selected to produce the offspring. If the individuals selected to reproduce are quite similar, then the inbreeding coefficient is closer to 1.0. On the other hand if the individuals selected are not similar, then the inbreeding coefficient is much lower. Selecting similar individuals for reproduction can lead to the loss of diversity in the population rather quickly.

The inbreeding coefficient is a metric adapted from population genetics. In natural populations the genotypes are diploid. The inbreeding coefficient is the probability that for an individual at a locus the two alleles of a gene are identical by descent (Smith 1989). The main effect of inbreeding in natural populations is to reduce the proportion of heterozygous genotypes and increase the proportion of homozygous genotypes. The effect of inbreeding is measured in terms of the reduction in the heterozygosity that occurs in the population. The inbreeding coefficient F is defined as:

$$F = \frac{H_0 - H}{H_0}$$

where H_0 and H are the initial and current proportions of heterozygous genotypes respectively. The initial proportion of heterozygous genotypes is given by $H_0 = 2p(1-p)$ where p is the frequency of the 0 allele (Smith 1989). The frequency is measured for each location in the genotype. In the initial population, the heterozygosity frequency (H_0) is around 0.5 for each location. As the generations increase and the diversity reduces, heterozygosity reduces, i.e., the individuals become similar to each other and H will be closer to 0.

The inbreeding coefficient measures how the proportion of heterozygous individuals deviate from what is expected when the population is in Hardy-Weinberg equilibrium. The Hardy-Weinberg principle states that in a randomly mating population, the genotype frequencies of the alleles AA, Aa, and aa will be p^2, $2pq$, and q^2 respectively (Smith 1989). The assumption is that there are only two alleles a and A. The frequency of allele A is given by p while q is the frequency of allele a and $p + q = 1.0$.

In the case of GAs the two alleles are 0 and 1. We give an example of how the expected heterozygosity is calculated. Suppose, at location 5 the number of 0 alleles in a population of 1000 individuals is 435. Then the frequency of 0 alleles at locus 5 is 0.435 and the expected proportion of heterozygosity is $2(0.435)(1.0 - 0.435)$ which is equal to 0.49. Inbreeding implies that related individuals are mated. Hence, in GAs the inbreeding measure is used at the time of reproduction during each generation.

For each locus, the number of heterozygous pairs (01 and 10) are monitored. H is the actual proportion of heterozygous pairs during mating at each locus. The inbreeding coefficient F is the average over all the locations. Larger the value of F, higher the similarity of the two individuals that are being mated.

We use the analytical model to predict the expected performance behavior in real spatially distributed GAs with finite population and genetic operators based on either the growth coefficient or the radii ratio. These predictions are validated using the test suite of problems from the function optimization domain. First, the performance predictions based on the growth coefficient will be validated. Since the induced selection pressure for all selection algorithms is the same it essentially means that the expected performance of these systems should be similar. However the radii ratio for the local selection algorithms can be quite different. Next, the performance predictions for a selected radii ratio will be validated. In this case the growth coefficients of the local selection algorithms will be different but the neighborhood pattern can be kept the same for most selection algorithms. Note that there are two degrees of freedom when trying to adjust the selection pressure in a spatially distributed GA. In a centralized sequential GA, the user can change the type of selection algorithm to adjust the selection pressure. While in a spatially distributed GA the user, in addition to changing the local selection algorithm, can also alter the radii ratio to control the selection pressure. We explore these ideas in the rest of this chapter.

Recall that the analytical tool was developed using a simplified model (using an infinite population with only selection and no genetic operators) in which the fitness values of the different classes are exact and constant. Recall that for the growth curve analysis the fitness ratio ($fitness_{best}/fitness_{worst}$) in the current population was assumed to be 2.0. Only fitness proportional selection algorithm is sensitive to the actual fitness ratios while the other three selection algorithms are not. In real GAs, every problem landscape has a different fitness ratio and moreover this fitness ratio can change over time. Just as in the sequential GA, due to this simplifying assumption, the behavior predictions of fitness proportional selection algorithm may not be as accurate as the other selection algorithms.

6.1.1 Experiment Parameters

In the experiments discussed in this section, the following parameter settings were used unless explicitly mentioned otherwise. At each grid location two parents are selected using the selection algorithm. We use a two-point crossover operator and a crossover rate of 1.0. We use a bit-flipping mutation operator and the mutation rate is set at 0.001. For each selection algorithm, the experiments are averaged over 100 runs. The selection algorithms used in this study are fitness proportional, linear ranking, binary tournament and random walk (path elitism). Henceforth "random walk" selection will imply path elitism random walk selection.

6.2 Predictions based on Growth Coefficients

Recall that the major decisions once a topology is chosen are the choice of the neighborhood size and the local selection algorithm to be used. These choices were mainly based on the results of the empirical studies (discussed in Section 2.3) done in the past. However now we have developed an analytical tool that can be used to make effective design decisions. If the growth coefficient is the same for all the selection algorithms, it implies that the selection pressure induced by these selection algorithms is equivalent. Hence we can predict that the four selection algorithms will have similar performance if the growth coefficient is the same for these selection algorithms. We can expect the best-so-far performance, the allele loss measure and the inbreeding coefficient statistics to be similar among the four selection algorithms.

To illustrate these ideas, we chose a growth coefficient of 0.30. Using this growth coefficient, the corresponding radii ratios for the selection algorithms is used to decide the neighborhood pattern and population topology. The neighborhood size and shape can be quite different for the selection methods as the radii ratios are not the same for the chosen growth coefficient. For fitness proportional selection the neighborhood pattern is D (refer Chapter 4) with a size of 41 in a 64×64 grid. Linear ranking and binary tournament both have a neighborhood size of 13 (D pattern) in a 64×64 grid. For random walk selection the neighborhood pattern is C and the neighborhood size is 25. Note how varied the neighborhood size and shapes are for the different selection algorithms.

Figures 6.1 and 6.2 present the best-so-far measure for Rastrigin and Hamiltonian Circuit functions (see Appendix A) for the four selection algorithms. In the case of Rastrigin function we are minimizing the function and hence lower the curve, better the performance. While in the case of Hamiltonian Circuit function we are maximizing and hence higher the curve, better the performance. As predicted, the performance of the selection methods with the exception of fitness proportional selection are similar. Recall that the assumption of the analytical model in the case of fitness proportional regarding the fitness ratio in the current population is fairly strong. The sensitivity of fitness proportional selection algorithm to the actual fitness values, unlike the other selection algorithms, accounts for the lower level of performance in some of the test functions.

In sequential GAs scaling of fitness values is dynamically done to maintain the selective pressure when solving problems in function optimization domain (De Jong 1992). A slight modification was made to the implementation of fitness proportional selection in the decentralized spatially distributed model. The fitness values were scaled in every local neighborhood before calculating the selection probabilities. The behavior of a spatially distributed GA using scaled fitness proportional selection should be closer to the analytical model than an unscaled version. The results of the scaled fitness proportional selection for the same growth coefficient and neighborhood as used earlier experiments is compared with the other selection algorithms in Figures 6.3 and 6.4. As predicted the behavior of the scaled fitness proportional selection is now similar to the other selection algorithms.

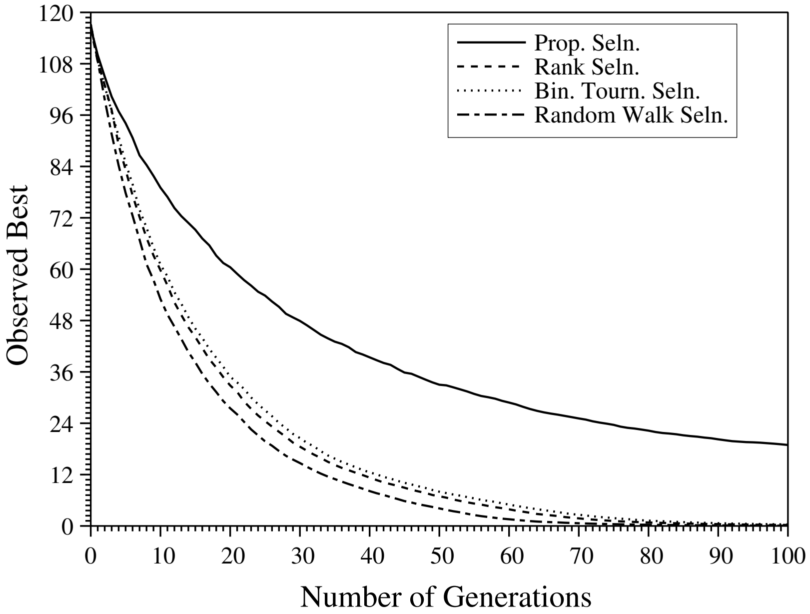

Figure 6.1: Best-so-far Performance for Rastrigin function for a growth coefficient of 0.30.

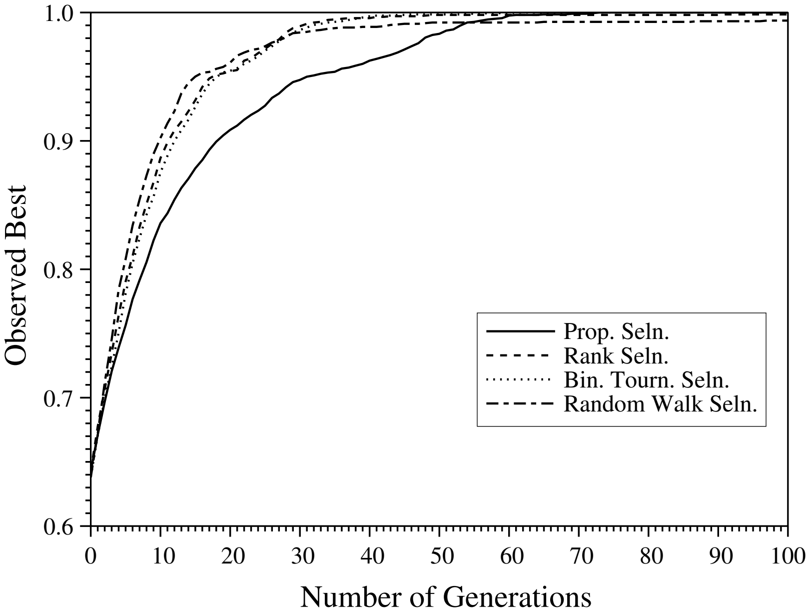

Figure 6.2: Best-so-far Performance for HC10 function for a growth coefficient of 0.30.

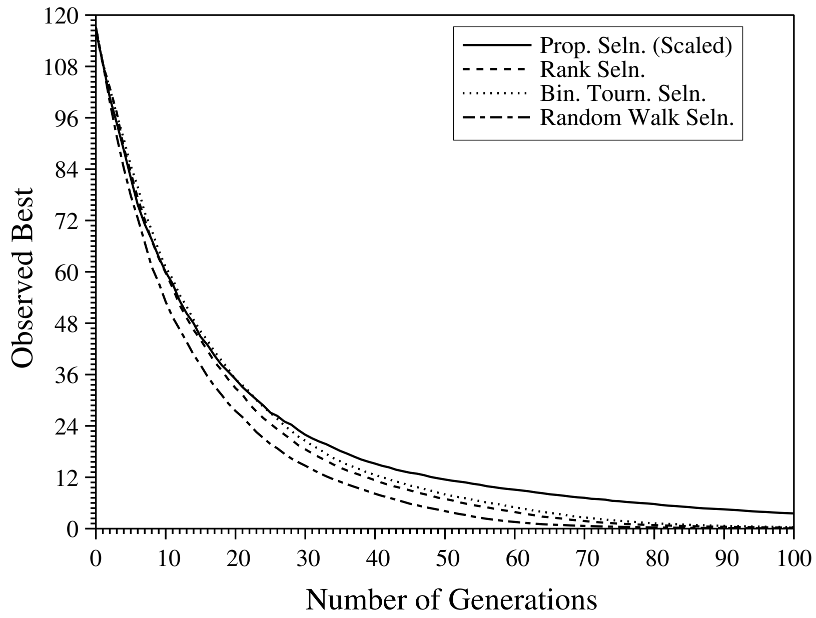

Figure 6.3: Best-so-far Performance for Rastrigin function using scaled fitness proportional selection.

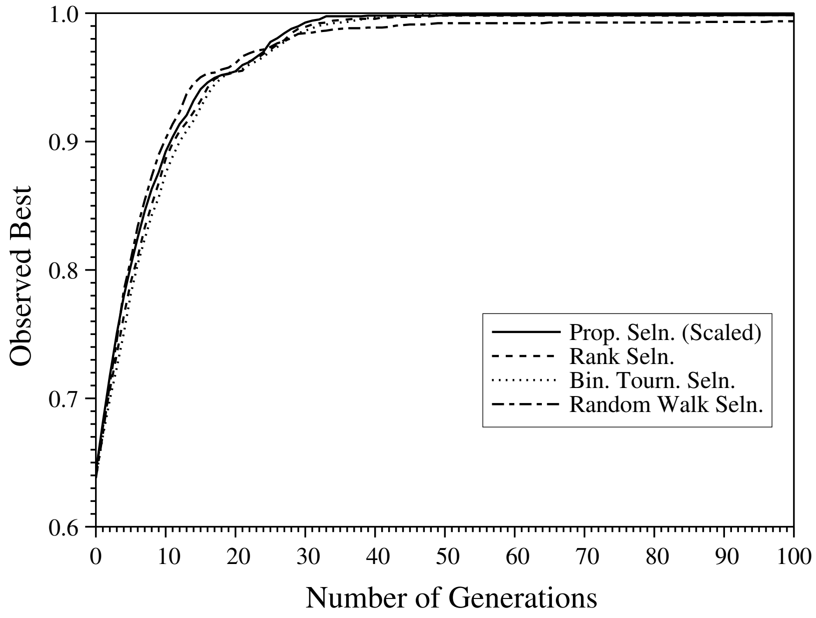

Figure 6.4: Best-so-far Performance for HC10 function with scaled fitness proportional selection.

The allele loss statistics are not shown as there was no allele loss for the four selection algorithms in all functions. The inbreeding coefficient for the two functions is shown in Figures 6.5 and 6.6. The prediction was that one should observe similar inbreeding coefficients as the selection pressure is the same in all the selection algorithms. As expected, only linear ranking and binary tournament have similar inbreeding coefficients and are quite different from both random walk and fitness proportional selection.

Recall that the implementation of random walk selection has an implicit form of elitism. Adding elitism makes the search more exploitative and hence leads to higher inbreeding coefficients than a selection algorithm with no elitism. This is clearly reflected in Figures 6.5 and 6.6. In the case of the fitness proportional selection algorithm we know that the analytical model's assumption regarding the fitness ratio is strong. The actual fitness ratios are much lower and this in turn induces a weaker selection pressure making the search more explorative. This leads to low inbreeding coefficient values which can be seen from Figures 6.5.

Introducing local scaling of fitness values, in the case of fitness proportional selection, produced performance behavior closer to the predicted behavior. Scaling has the effect of increasing the induced selection pressure and can affect the inbreeding measure. The inbreeding coefficients for the scaled fitness proportional selection is compared with the other selection algorithms in Figures 6.7 and 6.8. As expected, the inbreeding coefficients are higher compared to the unscaled version of fitness proportional selection.

6.3 Predictions based on Radii Ratios

If the choice of the neighborhood pattern is limited by the hardware architecture then the choice of selection method can be made using the analytical model. By taking a vertical slice of the analytical model, we get the induced selection pressure for a given radii ratio. The desired selection pressure can help in choosing the selection

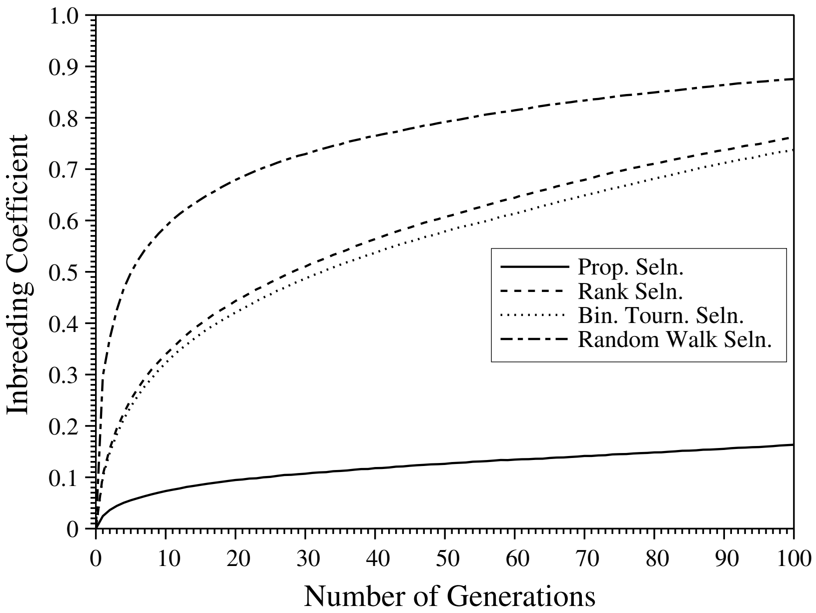

Figure 6.5: Inbreeding Coefficient statistics for Rastrigin function for a growth coefficient of 0.30.

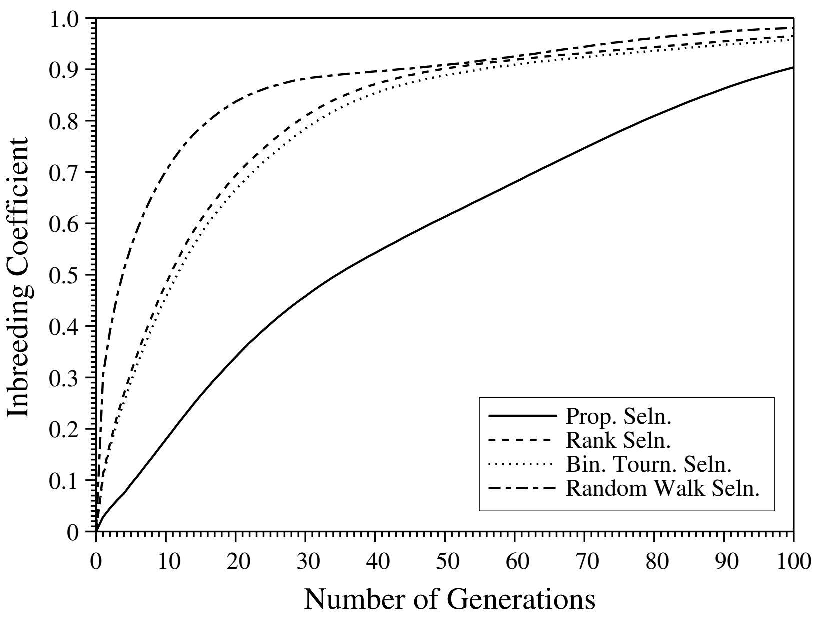

Figure 6.6: Inbreeding Coefficient statistics for HC10 function for a growth coefficient of 0.30.

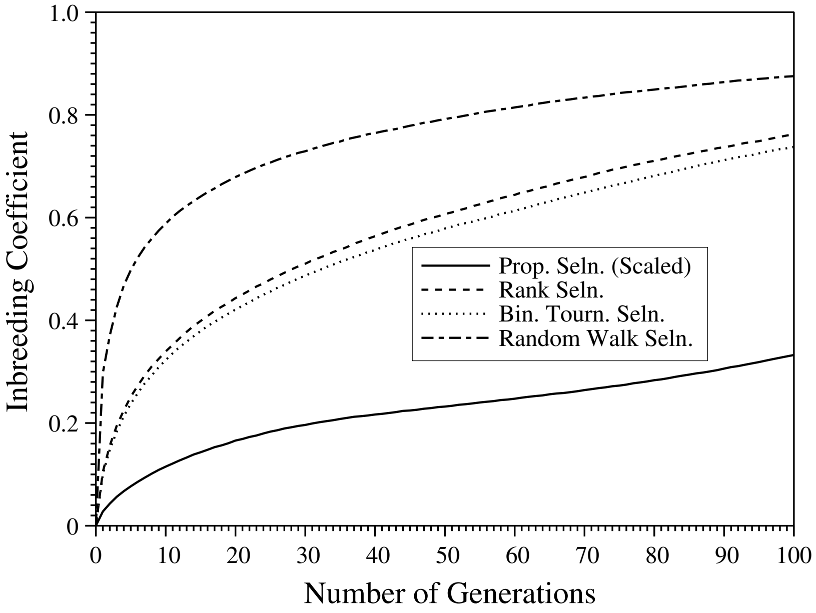

Figure 6.7: Inbreeding Coefficient statistics for Rastrigin function with scaled fitness proportional selection.

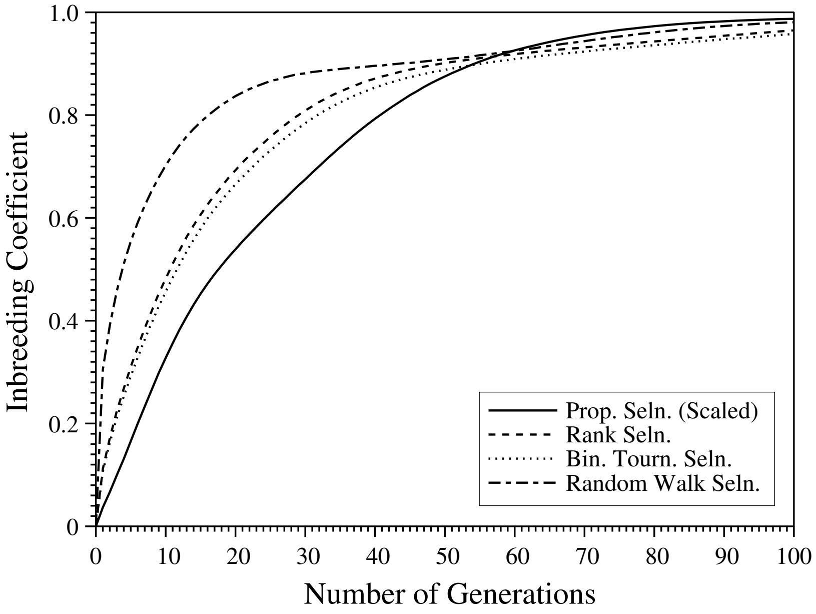

Figure 6.8: Inbreeding Coefficient statistics for HC10 function with scaled fitness proportional selection.

algorithm to be used. In the previous section we validated the predictions about the performance of spatially distributed GAs based on the growth coefficients. Now we make such predictions about the expected performance based on the radii ratios. At a given radii ratio the growth coefficients can be radically different for the four selection algorithms. The predicted behavior will depend on the growth coefficient values for the different selections corresponding to the given radii ratio. A high growth coefficient value implies that the GA will be exploitative while a low growth coefficient leads to a more explorative behavior.

For this validation exercise, a radii ratio of 0.15 is used. Having a fixed radii ratio means that the neighborhood radius can be kept similar for most of the selection algorithms. The topology is a 32×32 torus and the neighborhood pattern is the compact shape of size 25 for fitness proportional, linear ranking and binary tournament selections. A random walk of length 5 in eight directions (compact neighborhood shape) in the 32×32 grid gives a radii ratio of 0.15 for the random walk selection method.

The growth coefficient for each selection algorithm for the given radii ratio is calculated using the analytical model (see Figure 5.21 in Chapter 5). Even at this small radii ratio the difference in the growth coefficients of fitness proportional, linear ranking, binary tournament, and random walk selections is fairly large. Random walk selection has the highest growth coefficient.

We use these growth coefficients to generate the predicted growth curves for the different selection algorithms which are shown in Figure 6.9. From the characterization of the selection pressure and the predicted growth curves, we can hypothesize that the highly exploitative random walk selection (higher selection pressure than the rest) will generate good solutions quickly. Linear ranking and binary tournament should exhibit similar performances as they are equivalent in expectation. Fitness proportional selection has the weakest selection pressure for the assumed fitness ratio. Hence we can expect it to be more explorative than the other selection algorithms.

We know from sequential GAs that when the selection pressure is high, above

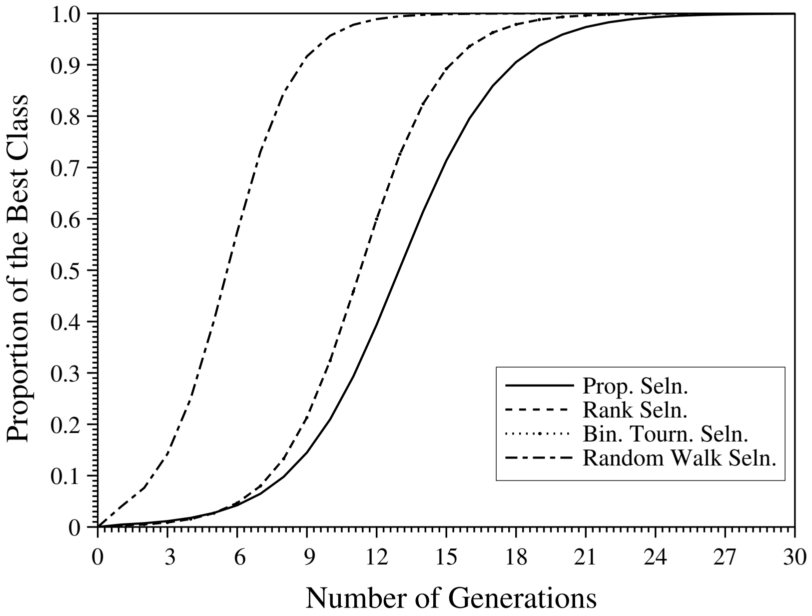

Figure 6.9: Predicted Growth Curves for radii ratio 0.15.

average individuals are selected for reproduction at the expense of the below average individuals. We also know that higher selection pressure implies that the diversity in the population is lost much faster and can expect the allele loss to be greater. Hence we can predict higher allele loss in the case of random walk selection, while linear ranking and binary tournament will have similar allele loss with fitness proportional having the least allele loss. Similarly, when the selection pressure is high we can expect to see larger values for the inbreeding coefficient as more similar individuals will be selected for reproduction. We can predict that the random walk selection will have larger inbreeding coefficient values, linear ranking and binary tournament will have similar values and fitness proportional selection will have smaller inbreeding coefficient values.

Figures 6.10 and 6.11 show a representative summary of the best-so-far performance for two of the test functions. As predicted, random walk selection exhibits a higher level of performance than the other selection algorithms in all the functions.

Linear ranking and binary tournament selections are similar in performance while the performance of fitness proportional is significantly lower. In all the test functions random walk selection method leads to a solution fairly quickly, with no further improvement in performance. Unlike linear ranking, binary tournament, and fitness proportional selection methods, which exhibit better performance albeit slowly, random walk selection does get “stuck” in a suboptimal solution.

As predicted, higher allele loss is consistently observed in the case of random walk selection which had the highest selection pressure among the selection algorithms. Figures 6.12 and 6.13 are a representation of the allele loss observed for the Rastrigin and Hamiltonian Circuit problems. In all the test functions, fitness proportional selection had the lowest allele loss. In the case of random walk selection, the allele loss is around 50% of the string length at the end of 100 generations for all test functions. This indicates that there is still some diversity in the population.

Higher selection pressure is also reflected by a larger inbreeding coefficient (See Figures 6.14 and 6.15). Random walk selection has much higher inbreeding coefficient compared to the other selection algorithms. The induced selection pressure in the case of random walk is much stronger than the other selection algorithms for the given radii ratio. Recall that this implementation of random walk selection has implicit local elitism. Hence the two individuals selected for mating are more similar to one another in random walk selection than in the other selection methods.

6.3.1 Adjusting Selection Pressure

So far we have used the analytical model to make predictions based on either the growth coefficient or the radii ratio. Another way to use the analytical model is to assist in adjusting the selection pressure. The advantage of using a spatially distributed GA is that the selection pressure can be adjusted either by changing the local selection algorithm used or by changing the radii ratio used.

Recall that the the complexity of fitness landscape of the problem being optimized

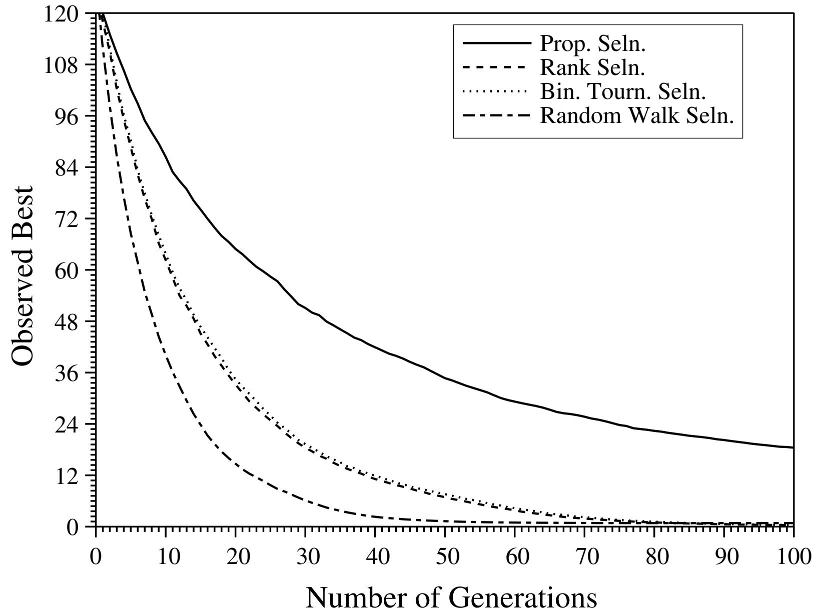

Figure 6.10: Best-so-far Performance for Rastrigin function for radii ratio 0.15.

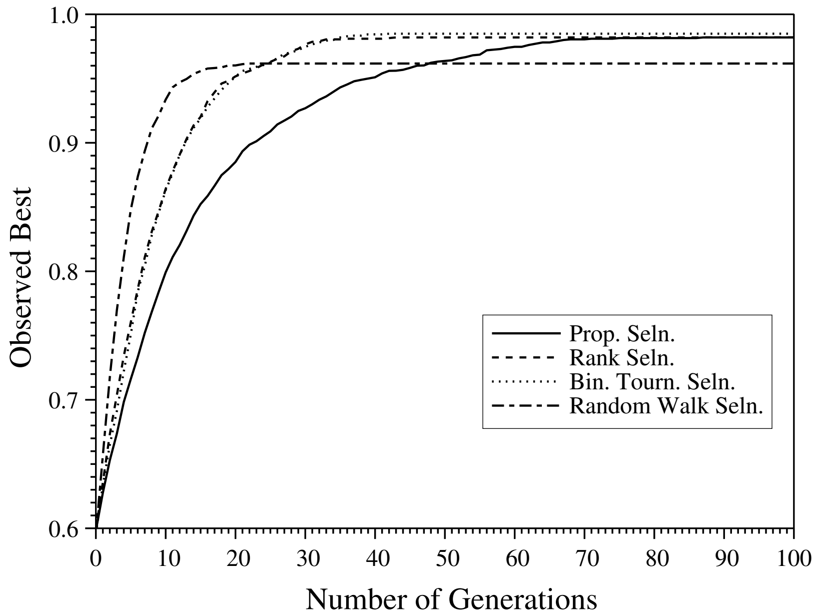

Figure 6.11: Best-so-far Performance for HC10 function for radii ratio 0.15.

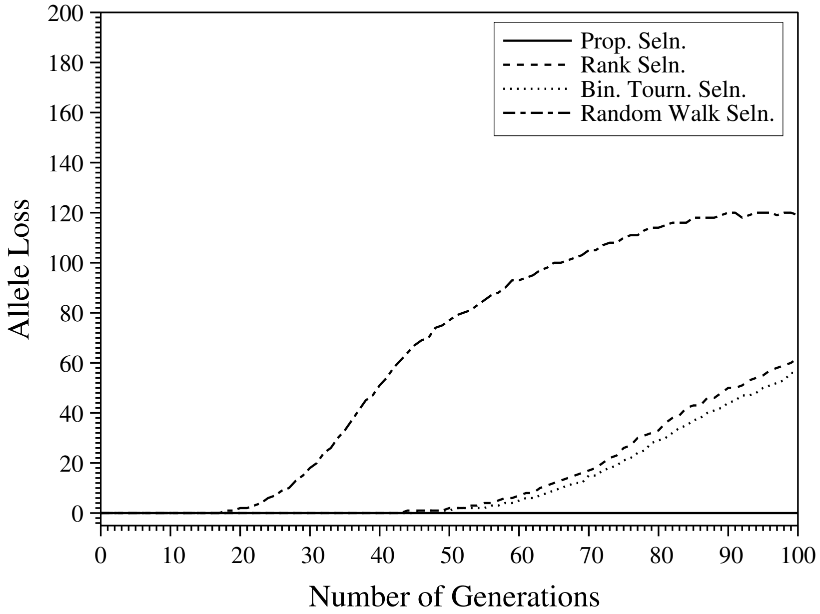

Figure 6.12: Allele loss statistics for Rastrigin function for radii ratio 0.15.

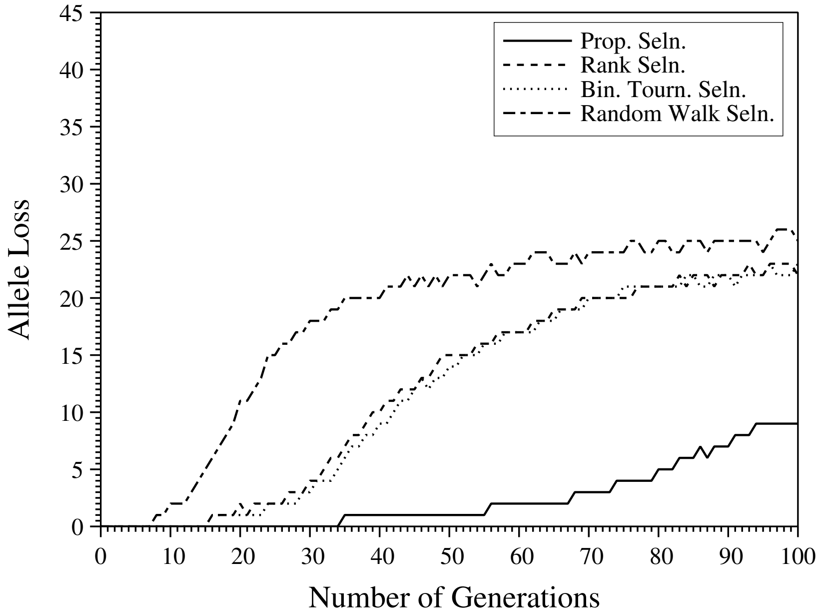

Figure 6.13: Allele loss statistics for HC10 function for radii ratio 0.15.

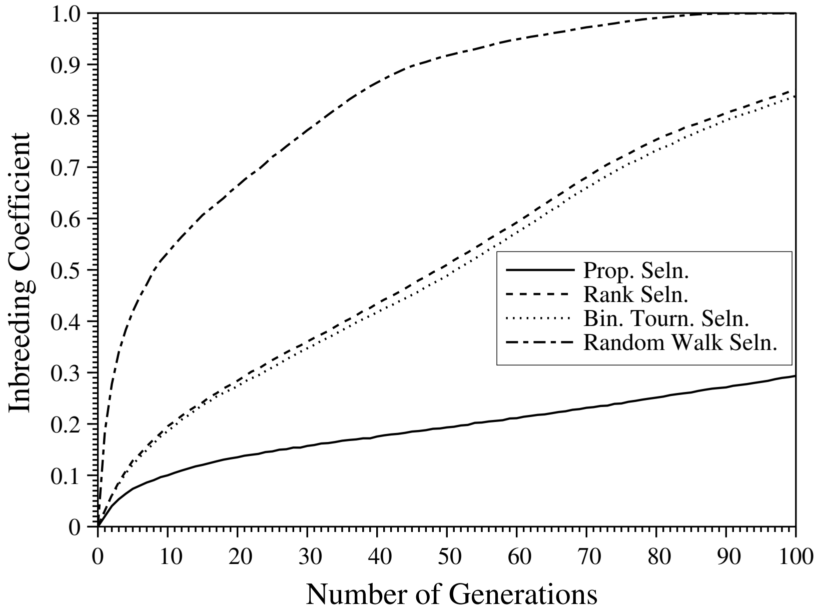

Figure 6.14: Inbreeding coefficient for Rastrigin function for radii ratio 0.15.

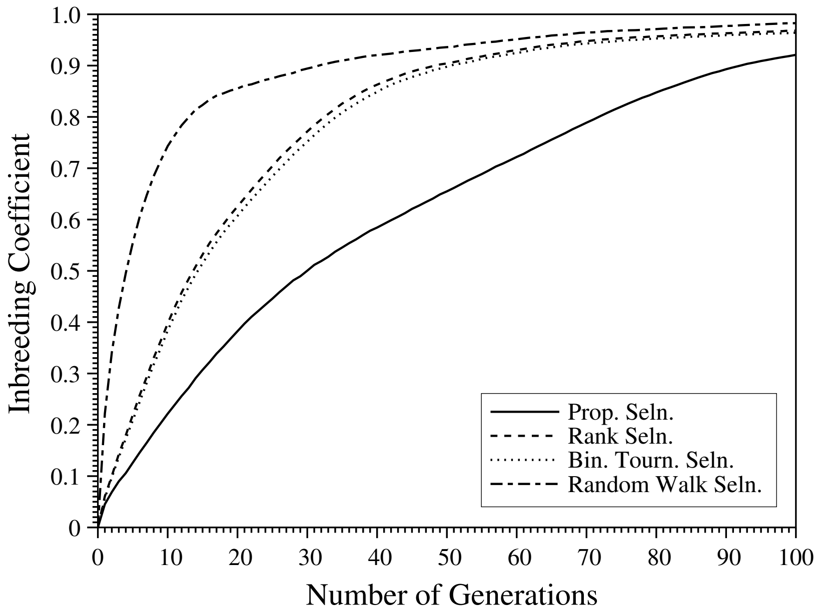

Figure 6.15: Inbreeding coefficient for HC10 function for radii ratio 0.15.

can affect the tension between exploration and exploitation. In the previous section we observed that the performance of random walk selection method in all the test functions lead to a solution close to the optimum fairly quickly but later showed no improvement. This trend was most noticeable in the boolean satisfiability problems. Since the induced selection pressure in the case of random walk selection is strong, the bias is towards more exploitation. The exploitation level can be lowered by decreasing the selection pressure. An improvement in performance will be seen if the balance is better than before.

The growth coefficient for the selected radii ratio in the case of random walk selection is fairly high (close to 0.70) compared to the other selection algorithms. One way to decrease the selection pressure is to use a smaller radii ratio, thus reducing the growth coefficient. This was done by decreasing the neighborhood radius while keeping the grid radius constant. Recall that for random walk selection method, the neighborhood size is controlled by the length of the random walk. By reducing the random walk length the neighborhood radius reduced, and the radii ratio as well as the growth coefficient for random walk selection decreased. The growth coefficient is now closer to linear ranking and binary tournament selections.

Reducing the selection pressure in the case of random walk selection leads to a performance improvement in all the test functions. Figure 6.16 shows the best performance for the two radii ratios using random walk selection method for HC10 function. A slight improvement in performance can be seen. A decrease in the number of alleles lost and inbreeding coefficient values was observed when the selection pressure was lowered.

The analytical model can also be used to adjust the selection pressure if the observed performance is not satisfactory with a particular selection algorithm. To do so, the radii ratio was reduced which lead to a lower growth coefficient. In the case of random walk selection, a slight improvement in performance was noticed. Another point to note is that reducing the radius in the case of random walk selection, the

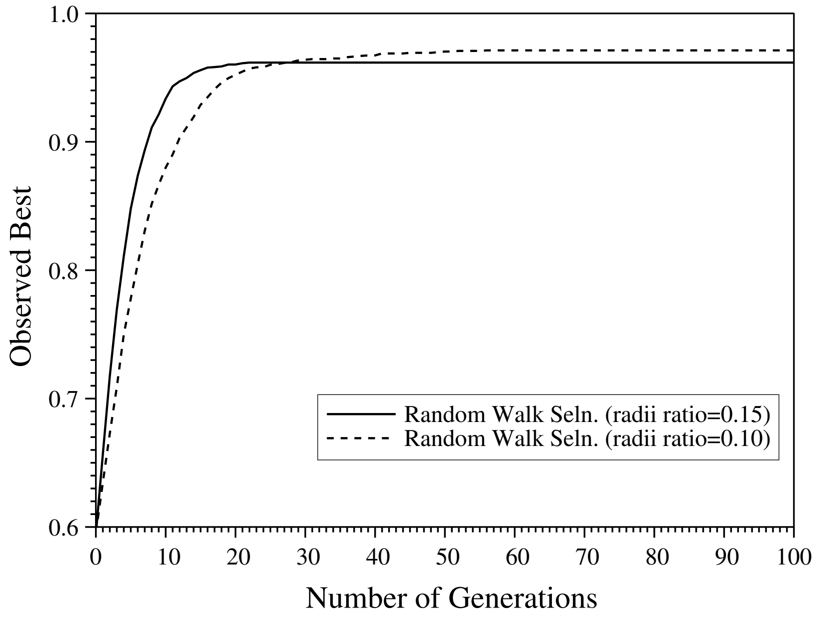

Figure 6.16: Best-so-far Performance using random walk selection for HC10 function.

length of the walk is decreased. Thus, the communication costs can be lowered without sacrificing the performance.

6.4 Parent Selection

There are different ways of implementing parent selection which affect the exploration of the search space. So far, in all the experiments, two individuals were selected based on fitness from the neighborhood as parents. We call this the two-parent case. But other forms of implementations are also possible. One way is to always chose the individual at the central grid as the first parent and then find a mate from the neighborhood based on fitness (Gorges-Schleuter 1989; Mühlenbein 1989). Another form of implementation is to first select a parent based on fitness from the neighborhood and replace the central individual of the neighborhood by this chosen parent. Then a mate for that parent is chosen at random from the neighborhood (Spiessens and Manderick 1991). A variation to this implementation is to select the mate based on fitness and not at random. This is termed "double selection" (White and Pettey 1997). However, these two variations increase the communication cost as well as the computation cost. The fitness information from the neighborhood has to be collected to select a parent based on fitness. If the second parent is selected at random, it involves an additional communication step. If the second parent is selected based on fitness, then the process of collecting fitness information and computing selection probabilities has to be repeated.

The analytical model can be used for predictions for these forms of implementations too. In Chapter 4 it was shown that the induced selection pressure when one parent is the individual at the grid and the other is selected based on fitness is quantitatively weaker than when both parents are chosen based on fitness (see Figures 4.25 to 4.27). This is called the one-parent case.

The prediction is that for the one-parent case, the behavior will be quantitatively weaker than the two-parent case. Using the same radii ratio of 0.15, we can predict, based on the analytical model, that the highly exploitative random walk selection will generate solutions quickly while fitness proportional selection will be the slowest. Linear ranking and binary tournament selection will have similar behavior. What we can expect to see is that the performance of all the selection algorithms will be lower than the two-parent case.

Using the same radii ratio as before, experiments are run on the test suite for the one-parent case. We found that for the four selection methods used the best-so-far performance is quantitatively weaker compared to the two-parent case, for the same number of generations. This is illustrated in Figure 6.17 for the Rastrigin function. As predicted, random walk selection generates better solutions and linear ranking and binary tournament selection have similar performance behavior, while fitness proportional selection is the weakest. As expected, both allele loss and the inbreeding coefficient are also lower for all functions compared to the two parent case.

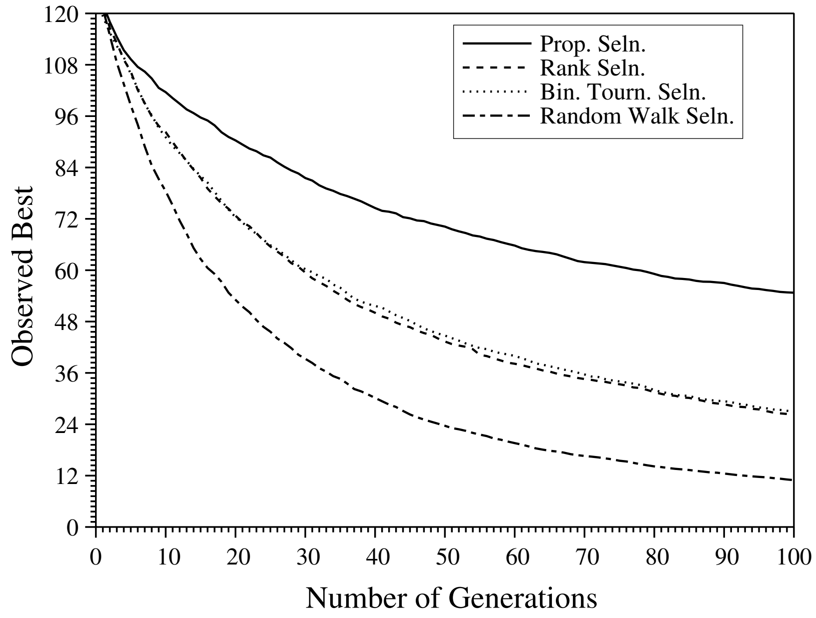

Figure 6.17: Best-so-far Performance for Rastrigin function for the one parent case.

6.5 Local Elitism

In the analytical model, elitism was not included except for the path elitism random walk selection method. Adding elitism increases the bias towards exploitation and how the behavior of the system changes depends on whether the balance was proper or not. Using elitism will definitely increase the allele loss as well as the inbreeding coefficient.

In a spatially distributed model, at each grid location two parents are selected based on fitness and recombined to form an offspring. The offspring replaces the current individual at the grid location irrespective of the fitness of the offspring. The growth curves were studied using an infinite population model. In a finite population model, especially in small neighborhoods, the best individual may not necessarily take over the population in all the runs, due to genetic drift. We can increase the neighborhood size to combat the genetic drift and approximate the infinite model. But increasing the neighborhood size increases the communication costs. An alternative is to use some form of elitism to get a good approximation of the infinite model. Adding an elitist policy means that a good solution is always preserved till a better one is found.

Recall that the implementation of the random walk selection has implicit local elitism. This is because the central individual always participates in the selection process and if that individual is the best in the neighborhood then it always gets selected. The other selection algorithms do not have an elitist policy which can be easily added to make it more exploitative.

Adding an elitist policy might be a good alternative to fitness scaling in the case of fitness proportional selection algorithm. Scaling involves additional computation while for elitism only a comparison of fitness values is required. Elitism would help to strengthen the local differentials without the additional computation. A slight change in the replacement strategy is made. Instead of always replacing the individual at each grid by the offspring, we replace the individual only when the offspring is better.

Figure 6.18 shows the performance improvement in the fitness proportional selection for Rastrigin function when local elitism is added. The introduction of an explicit local elitism in the case of fitness proportional selection, does improve the performance significantly in some functions but hurts the performance in other functions. For the class of Boolean satisfiability functions, we found that adding local elitism to fitness proportional selection method, hurt the performance instead of improving it. In fact local scaling was a better choice than local elitism for boolean satisfiability problems as the performance improved. One point to note is that local elitism is not an effective solution for all test problems.

As expected there is a much higher allele loss as well as higher inbreeding coefficient. When local elitism is not used, there is no allele loss until generation 100. There is plenty of diversity still in the population and the experiments can be run much longer than 100 generations. On the other hand adding local elitism significantly increases the allele loss. This suggests, that instead of having a deterministic replacement policy, some other softer forms of local elitism ought to be explored.

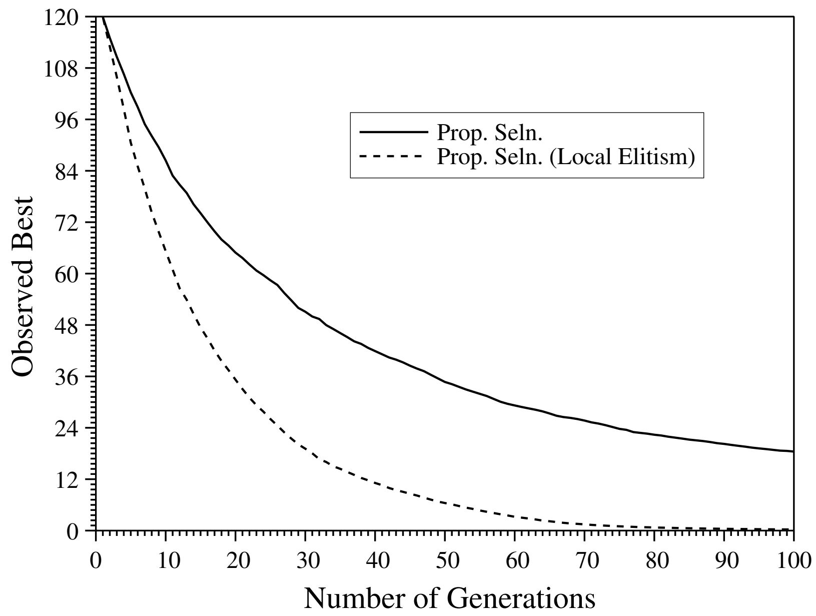

Figure 6.18: Best-so-far Performance using fitness proportional selection with and without local elitism for Rastrigin function.

Linear ranking and binary tournament selections have similar performance for all the test functions when a radii ratio of 0.153 and neighborhood size 25. But when neighborhood radius is reduced while keeping the grid size constant (smaller neighborhood sizes are used) binary tournament does not perform as well as linear ranking selection. The question that arises is: "How does one improve the search performance without increasing communication overhead?". It was shown in De Jong and Sarma (1995) that the performance of binary tournament selection in small neighborhood sizes can be improved by using a local elitist policy at the time of replacement. This again has the effect of strengthening the local selection differentials. Binary tournament with explicit local elitism and linear ranking selection have similar performance in the small neighborhood sizes. Thus one way to improve search performance without increasing communication overhead is to use a binary tournament with local elitism when the neighborhood size is fairly small.

6.6 Genetic Operators

The type of genetic operators used can offset the exploitation of the information with exploration of the search space. In our analytical model we made no assumptions about the specific genetic operators which are used. The question that arises is what happens when the mutation rate or crossover rate is switched to something other than what we have used so far. The performance curve is either raised or lower depending on the equilibrium between exploration and exploitation. The effects of using a different type of crossover or mutation operator is not addressed.

We found that using a crossover rate less than 1.0 leads to lower performance behavior in all functions. The higher mutation rate is generally used to increase the diversity in the population and to combat the problem of premature convergence in sequential GAs. In the the spatially distributed GA experiments the mutation rate was kept at 0.001 similar to the rate used in sequential GAs. We ran a few experiments using mutation rates of 0.005 and 0.01. Of the four selection algorithms, only random walk selection method shows any performance improvement when mutation rate is increased to 0.005 and 0.01. This is because random walk selection has implicit elitist policy which helps to preserve good solutions longer. In the other selection methods there was a slight decrease in best-so-far performance. Increasing the mutation rate to greater than 0.005 requires an elitist policy in the selection algorithms other than random walk to generate a good performance.

6.7 Summary

We have demonstrated that the growth coefficient or the radii ratio can be used to predict the expected behavior for the different local selection algorithms. We used the analytical tools which characterized the selection pressure of commonly used selection algorithms to make behavior predictions in “real” spatially distributed GAs. This was empirically evaluated using complex function optimization problems.

First we validated predictions based on the growth coefficient. The radii ratios corresponding to the growth coefficient were considerably different. Similar emergent selection pressure leads to comparable behavior as predicted by the model. We saw that the performance improved considerably when local scaling was used with fitness proportional selection method. The behavior of the system using fitness proportional selection with scaling is much closer to the predictions made using the analytical model. Next we validated predictions based on the radii ratio. We showed how the analytical model can be used to adjust the selection pressure by changing the radii ratio.

We found that adding an elitist policy to selection strategies other than random walk, improved the search performance in some functions but not in others. Fitness proportional selection with explicit elitism showed improved performance in some functions. Linear ranking and binary tournament selection algorithms had very similar performance behavior except in small neighborhood sizes. In small neighborhoods, using a local elitist policy with binary tournament selection improved the performance and made it similar to linear ranking selection. The advantage of using binary tournament selection is that it does not have the additional computation required by linear ranking selection. Moreover communication costs can be reduced by using smaller neighborhood sizes and local elitism in the case of binary tournament selection. In fact, the model and the results indicate that there is no reason to even consider using linear ranking selection with spatially distributed GAs.

We demonstrated that the analytical model can be used to make accurate qualitative and quantitative predictions in a real spatially distributed GA. This means that design decisions regarding the selection methods and the neighborhood size and topology, can be made based on the analytical model. A direct consequence of the validation of the predictions based on the analytical model is the observation that the results of the various empirical studies in spatially distributed GAs can be explained using our analytical model.

As is the case with centralized GAs, there is no optimal level for selection pressure; rather it depends on the fitness landscape of the problem being solved. In centralized sequential GAs the usual method to adjust selection pressure is to change the selection algorithm. The advantage of using a spatially distributed GA is that the selection pressure can be adjusted by changing either the local selection algorithm or the radii ratio. We showed how the selection pressure in the spatially distributed GAs can be controlled by adjusting the neighborhood radius. An extension would be to dynamically adapt the selection pressure during the search process by changing the neighborhood radius.

In general, since the emergent global selection pressure induced in the case of spatially distributed GAs is weaker than the global selection pressure in centralized GAs, diversity is maintained longer. The ability to sustain diversity can be useful for classes of problems where the goal is to find multiple solutions or to track a time-varying environment. It is clear that the balance between exploration of the search space and exploitation of the information gleaned is very much dependent on the underlying problem being solved as evidenced by the experimental results. Since the spatially distributed GA has a finer way of controlling selection pressure this equilibrium can be adjusted to apply to these classes of problems. The effectiveness of the spatially distributed GAs in such classes of problems will be explored in the next chapter.

Chapter 7

ADDITIONAL PROPERTIES OF SPATIALLY DISTRIBUTED MODELS

The centralized sequential GA loses diversity fairly quickly and converges to a solution which may not necessarily be the optimal solution. This is a drawback when the centralized sequential GA is applied to classes of problems like multimodal function optimization and nonstationary environments. In both these classes of problems, maintaining a high level of diversity is important in order to sustain exploration. In the case of multimodal functions, the focus is on the ability to identify as many of the optimum solutions as required by the user. In nonstationary environments, the GA has to have the ability to adapt and track the changing optimum. Many real world problems fall into the category of multimodal functions and nonstationary environments. To apply the centralized sequential GAs to these problems, the usual approach is to either reduce the selection pressure induced by changing the selection algorithm or make some other suitable modifications to the GA.

Unlike the centralized sequential GAs, in decentralized and spatially distributed GAs the selection pressure can be controlled by either changing the local selection algorithm or altering the radii ratio. Thus the spatially distributed GAs seem to have the potential to be applied to these classes of problems as there are multiple controls to reduce selection pressure. In this chapter we explore the suitability of the decentralized and spatially distributed population model for the above two classes of problems, namely, multimodal function optimization and nonstationary environments. This will provide some insights into the additional properties of these decentralized and spatially distributed GAs.

7.1 Multimodal Function Optimization

Multimodal function optimization usually implies that there are many optima (local and/or global) in the fitness landscape. The goals of the optimization may differ depending on the problem. We define only two goals for the purposes of our study, though there are many other possibilities. In the first case, the goal is to locate the single global solution in the presence of many local optima (solutions). The problems in the test suite used in Chapter 6 fall into this category. In the second case, the goal is locate the competing alternate solutions. These competing alternate solutions may all be global optima or some global and some local optima. The additional requirement in the second case may be the need to maintain all these alternate solutions (whether local or global) within the same population. Each of these optima or alternate solutions is considered to be a niche that can support one or more species.

We know that the drawback of the centralized sequential GA is that it loses diversity quickly and has to be modified to be applied to problems with competing alternate solutions. Diversity is not only required to explore new areas in the search space to find all solutions but also to delay convergence to a single solution. Niching and speciation is a focus of much research in the sequential GA community and considerable work has been done in this area (De Jong 1975; Goldberg and Richardson 1987; Deb and Goldberg 1989; Mahfoud 1992; Beasley, Bull, and Martin 1993; Spears 1994).

We will briefly discuss some of the changes made to the centralized sequential GAs to solve multimodal functions (problems with competing alternate solutions). The crowding scheme was introduced by De Jong (1975) as an alternative to increasing the mutation rate and as a method for overcoming the problem of premature convergence in multimodal functions when using small populations. Later studies showed that the crowding scheme could maintain subpopulations only on a few global peaks of the multimodal problem (Deb and Goldberg 1989). Goldberg and Richardson (1987) introduced the sharing function as a method of maintaining diversity while solving multimodal functions. The population is divided into subpopulations using a distance metric which can be defined in either the genotype or the decoded parameter space. The fitness of an individual is then reduced based on the number of individuals in its subpopulation. This process is called fitness sharing. Additionally, mating was also restricted to occur only within the subpopulations consisting of similar individuals (Deb and Goldberg 1989). Variations of the crowding scheme and preselection (every individual gets to reproduce irrespective of fitness) strategies were studied by Mahfoud (1992). He showed that the scheme of preselection combined with phenotypic sharing could maintain all the peaks even when the peaks are of differing heights. Adding tag bits or labels to the bit string representation is another method used for creating subpopulations and locating the competing alternate solutions in multimodal problems (Spears 1994).

All these methods involve changing the centralized sequential GA to control the induced selection pressure. We have already seen that in a spatially distributed GA the selection pressure can be controlled by adjusting the radii ratio. Moreover since the genetic data is spatially distributed in the local neighborhood model, different parts of the search space may be explored simultaneously. Hence the hypothesis is that if the appropriate radii ratio is chosen, the spatially distributed model will be able to locate the competing alternate solutions in a multimodal function.

Collins and Jefferson (1991) used a spatially distributed GA with very large population sizes to solve a multimodal function with two global optima. The local selection algorithm used was the random walk (path elitism) selection. They showed that the two global optima could be located and maintained over a long period of time for random walk of length 1. The radius of the grid was quite large while the radius of the neighborhood generated by a random walk of length 1 was small and hence the radii ratio was low ($\ll 0.1$). From our analytical model we know that the selection pressure induced by such a low radii ratio is rather weak. This suggests that if very small radii ratio and correspondingly low growth rates are used, the chances of locating the global optima in a multimodal function are fairly high.

Davidor's ECO GA (1991) uses a two-dimensional spatially distributed GA to successfully create niches and form species. Unlike our model wherein all the grid locations perform reproduction and creation of a new offspring concurrently, Davidor's ECO model allowed only one grid location to reproduce and create new offspring each generation. Also the induced selection pressure was weaker in the ECO model than a sequential GA. This allowed more exploration, thereby locating the competing alternate solutions.

The objective in this section is to verify whether using a spatially distributed GA would help in locating and maintaining the competing alternate solutions in a multimodal function. To test this hypothesis we used the following one dimensional real-valued sinusoidal functions from Deb and Goldberg (1989).

MF1: $f(x) = \sin^6(5\pi x)$

MF2: $f(x) = \exp^{-2\ln 2(\frac{x-0.1}{0.8})^2} \sin^6(5\pi x)$

In both the functions the parameter x ranges between 0 and 1. The function MF1 has five global optima in the domain of equal height and the global peaks are equally spaced. In function MF2 the five global optima are equally spaced but have differing heights. In both cases there are 5 competing alternate solutions and the goal is to find all the five solutions. The hypothesis is that choosing the appropriate radii ratio will help the spatially distributed GA to locate all the five peaks in these two test functions. However, the selection pressure forces the GA to choose individuals with better fitness. Hence in function MF2, the lower level peaks will be taken over by the higher level peaks unless some explicit effort is made to maintain the lower level peaks.

In centralized GAs usually small population sizes are used, with fitness proportional selection and the standard settings for crossover (0.60) and mutation (0.001). Fitness proportional selection has the weakest induced selection pressure in a centralized GA. Hence the options available for reducing the selection pressure is limited to modifying the algorithm. But in a spatially distributed GA the selection pressure can be decreased or increased by changing the radii ratio.

If the induced selection pressures in the decentralized spatially distributed GA and the centralized sequential GA are the same then we can expect the two GA systems to have similar performance behavior. We configure the two GA systems same way. Both systems use the same population size, selection algorithm, crossover and mutation rates. The population size used is 225 which is fairly close to the population sizes used in centralized GAs. We use fitness proportional selection with the standard settings for the genetic operators. For the decentralized spatially distributed GA the radii ratio chosen is 0.33 which had a growth coefficient quite close to that of the centralized sequential GA for fitness proportional selection. The population size of 225 is laid out as a 15×15 toroidal grid and a neighborhood size of 25 (D25) gives the desired radii ratio of 0.33. What we can expect to see is that the GA will converge to a single peak instead of locating the alternate solutions.

The performance measure generally used is the number of subpopulations (peaks) maintained at the end of a certain number of generations, which is usually set at 200. Since the test functions are one dimensional functions, it is easy to plot the number of peaks represented in the population at the end of generation 200. We ran the spatially distributed GA for 200 generations and the population at the end of generation 200 is plotted to find the number of peaks represented.

Figures 7.1 and 7.2 portray the number of peaks maintained at the end of generation 200 for function MF1 and MF2 respectively when the radii ratio is 0.33. As expected at the end of generation 200 only one peak is maintained in both the functions. But we know that in the spatially distributed GA the selection pressure can be reduced by changing the radii ratio. The radii ratio was lowered from 0.33 to 0.15 for the same toroidal grid by reducing the size of the neighborhood.

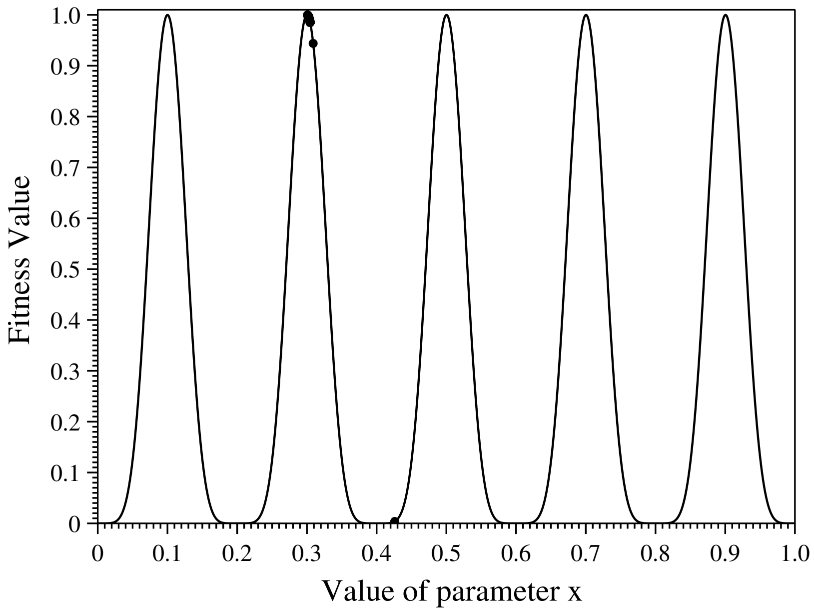

Figure 7.1: Number of peaks represented at the end of generation 200 for function MF1.

Figures 7.3 and 7.4 show the results at the end of 200 generations for function MF1 and MF2 when a radii ratio of 0.15 is used. One can see that for function MF1, 3 of the 5 peaks are still represented in the population at the end of 200 generations. Thus by altering the radii ratio, the selection pressure decreased and this improved the performance of the system as it was able to sustain three peaks instead of only one. But function MF2 has five peaks of differing heights and selective pressures should force the GA to converge to the higher peaks. As expected, there is no change in the number of peaks sustained for function MF2.

These experiments are first approximations of the applicability of the spatially distributed GAs to multimodal function optimization problems. As predicted, changing the radii ratio shows an improvement in performance even in such small population

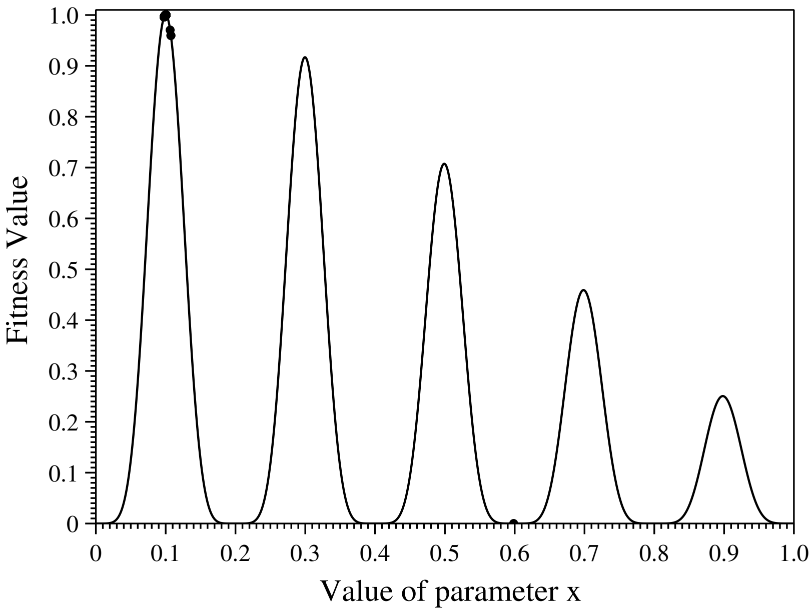

Figure 7.2: Number of peaks represented at the end of generation 200 for function MF2.

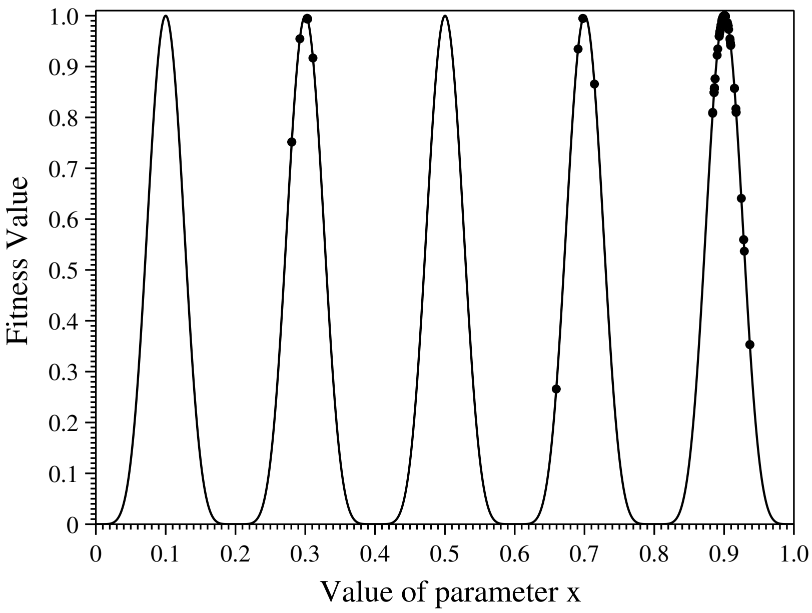

Figure 7.3: Number of peaks represented at the end of generation 200 for function MF1.

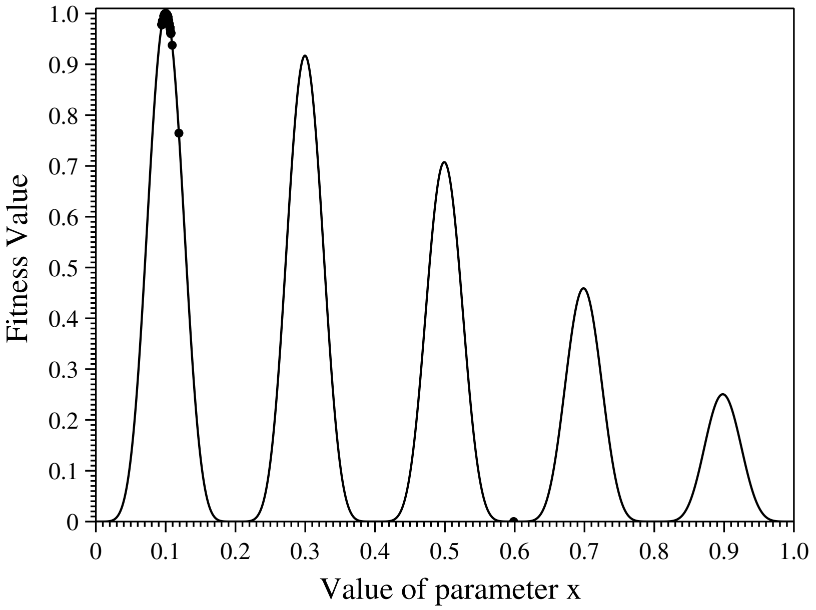

Figure 7.4: Number of peaks represented at the end of generation 200 for function MF2.

size for the case where the competing alternate solutions are all the same height.

7.2 Nonstationary Environments

An environment is defined to be nonstationary if the response from the environment for the inputs varies with time. The response can either vary every time step by a certain quantity or change every δ time steps. First, we will try to define a few types of nonstationary environments. In the first case, the environment changes periodically. This implies that the fitness landscape is constant over a fixed time period (δ) after which it changes. This change can oscillate between two different states or can have a larger number of different states. In the second case, the fitness landscape varies continuously with time either in small or large increments. In both cases, either the entire landscape changes or only certain parts of the landscape change. In the last type, multiple learning systems interact and the actions of one learning system change the observed landscape (environment) of another learning system. There is no fixed time period and the change can occur often or infrequently. Cobb (1990) discusses two different types of nonstationary environments.

The GA is adaptively searching an unknown space of potential solutions to find high performance solutions. When the environment varies with time it is essential that the GA be able to adapt to these changes by adjusting its search pattern. The major drawback of using sequential GAs for nonstationary problems is that it converges quickly to a solution and loses diversity. Hence when the environment state varies, the sequential GA is unable to track the change unless some modifications are made to it. In the next section we review the different types of modifications usually made to the sequential GA to "track" the changing environment.

7.2.1 Sequential GAs and Nonstationary Environments

Very few studies have been performed using centralized, sequential GAs to adaptively track a changing environment. The two main classes of modifications that have been attempted are (1) expanding the memory of the sequential GA via the representation scheme, and (2) using some form of the mutation operator to introduce diversity in the population. The first approach is to modify the representation to carry along redundant information and this increases the memory of the sequential GA. The rationale for carrying redundant information is that if the environment suddenly changes, this redundant data will help the GA to quickly adapt to the new environment. The second approach uses the mutation operator to introduce diversity into the population whenever the fitness landscape changes. Both these approaches have their advantages and disadvantages. Increasing the population size does not help the sequential GA track the moving optimum as it still needs some modification to adapt to the changing environment.

Pettit and Swigger (1983) applied the sequential GA to track a target solution which changed at random every generation. The rate of fluctuation was either slow, fast or medium. They found that the sequential GA was unable to successfully track the changing environment even when the rate of fluctuation was slow. Goldberg and Smith (1987) used a diploid (pairs of chromosomes) representation instead of a haploid (single chromosome) representation to effectively track a nonstationary 0–1 blind knapsack problem that periodically changed between two different states. A new operator called the “dominance” operator was introduced to handle the new representation. The diploid representation helped to carry redundant information along till the need arose to use them. They concluded that a GA with a diploid representation was able to track the changing environment better than a GA with a haploid representation. Another method to expand the memory of the sequential GA was to use a multi-layered (a tree structured) chromosome which helped to successfully track the optimum in a nonstationary 0–1 blind knapsack problem (Dasgupta and McGregor 1992).

The other studies have all concentrated on using the mutation operator to introduce diversity into the population whenever the environment changes. The *triggered hypermutation* operator was used in the sequential GA to introduce diversity into a population where the environment is changing continuously (Cobb 1990). This hypermutation operator was triggered when the time-averaged best performance declined. Essentially a higher mutation rate was used to change the population composition so that the new optimum can be tracked. Grefenstette (1992) used a method called the *random immigrants* to increase the diversity in the population to successfully track the moving optimum in a periodically changing environment. Each generation a certain percentage of the population is replaced by new randomly created individuals which kept the exploration level high.

Cobb and Grefenstette (1993) compared these two mechanisms (triggered hypermutation and random immigrants) and the sequential GA (with higher mutation rate) on different forms of nonstationary environments. They concluded that triggered hypermutation will be able to perform well in slowly changing nonstationary environments while the random immigrant approach is better suited for sudden massive changes in the environment. Yet another modification made to the sequential GA is the use of a shift operator which is initiated when the time averaged best performance decreased (Vavak, Fogarty, and Jukes 1996; Vavak, Jukes, and Fogarty 1997). The shift operator enabled a local search within a certain radius of each individual. This is similar to mutating the individual but here each individual had its own mutation level. They showed that this mechanism was able to track the optimum in a slowly changing environment.

Both the approaches of either increasing memory or using mutation operator to increase diversity have some drawbacks. The use of the diploid representation in the sequential GA was tested on a two-state problem and it is not clear how this approach will scale up when applied to problems with larger number of states. The other approach of using different mutation mechanisms perform well only in certain classes of nonstationary environment. In general, in real world problems one is not sure when the changes are occurring, how fast or slow these changes are and how drastic these changes are. Hence it is hard to decide beforehand which mutation scheme would be appropriate for the given nonstationary problem. All these modification methods essentially alter the exploration and exploitation balance to adaptively search the changing environment. It is essential to achieve the appropriate balance in these nonstationary environments to successfully track the moving optimum.

In the previous chapter, the spatially distributed GA was applied to stationary function optimization problems. We showed how the selection pressure can be adjusted by changing the radii ratio to achieve a better level of performance. It would be desirable to have a GA which can be used for both stationary and nonstationary environments without many modifications. In the local neighborhood GA the genetic information is spatially distributed. This can act as extended memory for the GA as the various neighborhoods can potentially search different areas in the search space. The spatial distribution of the genetic information can prove to be especially useful in nonstationary environments. The information in the different neighborhoods can be beneficial during times of change and can aid the GA to track the changing optimum. We will explore this hypothesis in the following sections.

7.2.2 Experimental Methodology

We are interested in understanding the behavior of the spatially distributed GAs in a time varying environment. To understand this better, for our experiments, we used the test problem called "Landscape A" from Cobb and Grefenstette (1993). It is a two-dimensional test problem and the fitness landscape consists of 14 sinusoidal hills with varying heights. The height of the tallest hill is 60. The range of each dimension is +32.768 to -32.768. The length of the string is 32 bits.

The two types of nonstationarity we consider in this study are: (1) the translation of the maximum hill while keeping the rest of the landscape fixed, and (2) the translation of the entire landscape (all hills) either in a fast or slow mode. There is no distortion of the hills, which implies that the widths and heights are constant.

Normally in the sequential GAs small population sizes and the standard setting for crossover are used. The mutation rate used is usually higher than the standard setting of 0.001. As mentioned earlier, increasing the population size is not a solution for the sequential GA to track the optimum in a dynamic environment as maintaining diversity is still a problem.

All the experiments in this section are averaged over 100 runs. The experiment was stopped when generation 200 was reached. The best individual found was kept track of every generation. Two point crossover and bit flipping mutation is used in all experiments. The standard setting of 0.60 for crossover rate is used. Cobb and Grefenstette (1993) in their study concluded that the standard sequential GA with high mutation rate (0.10) was able to track the moving optimum. But the goal here is to get some insights into the ability of the decentralized spatially distributed GAs to track the moving optimum in a nonstationary environment. We opted to use a mutation rate of 0.01 instead of the standard setting of 0.001. All parameter settings are held similar to the centralized sequential GA to study the effects of decentralization and spatial distribution. The toroidal grid used is 15×15. We used fitness proportional selection algorithm as the selection pressure induced by it is weaker than the other selection algorithms.

7.2.3 Translation of the Maximum Hill

The first type of nonstationary environment used for testing the behavior of the spatially distributed GA is the periodically changing landscape. To simulate the periodically changing environment, the location of the maximum hill was moved randomly to a different spot every 20 generations. The rest of the hills remained stationary. The reason for choosing 20 generations is because it was considered to be ample time for a sequential GA to converge to the new optimum (Grefenstette 1992).

First we used the decentralized spatially distributed GA with a radii ratio of 0.33 which had similar selection pressure as the centralized sequential GA. Here we can expect the two systems to have similar performance behavior. Then we reduced the selection pressure in the case of the decentralized spatially distributed GA by reducing the radii ratio to 0.15.

Figure 7.5 shows the best performance for the fitness proportional selection algorithm when a radii ratio 0.33 is used. The induced selection pressure with a radii ratio of 0.33 in a spatially distributed GA is similar to that of the centralized sequential GA. Even though the induced selection pressures are similar, we can see that the spatially distributed GA is able to track the moving optimum better than the sequential GA. The spatial distribution of genetic information does seem to help in tracking the moving optimum. A centralized sequential GA with similar settings is unable to track the moving optimum even when a higher mutation rate of 0.01 is used.

In the case of multimodal function optimization we saw that reducing the selection

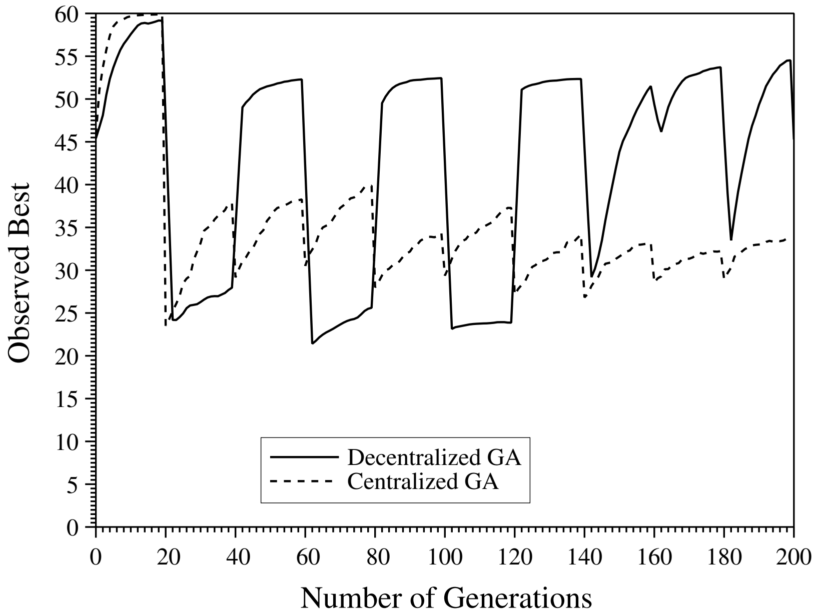

Figure 7.5: Performance curves for decentralized (radii ratio=0.33) and centralized GAs for max location moved every 20 generations.

pressure helped improve performance. We reduced the radii ratio to 0.15 from 0.33 to study the effects on the tracking ability. Figure 7.6 presents the results for fitness proportional selection for the decentralized and spatially distributed GA using a radii ratio of 0.15 and the centralized sequential GA. Both the GAs used a mutation rate of 0.01.

From Figure 7.6 we can make a few observations. We know that the induced selection pressure is fairly strong in the case of the centralized GA compared to the decentralized GA. In the beginning the strong selection pressure helps the centralized GA locate the maximum peak faster than the decentralized GA. But this is also the drawback in the case of centralized GAs as diversity is lost more quickly. Hence when the location of the maximum peak is moved, the centralized GA is unable to find it while the decentralized GA performs better by quickly tracking it. Reducing the selection pressure by reducing the radii ratio does appear to help improve the tracking

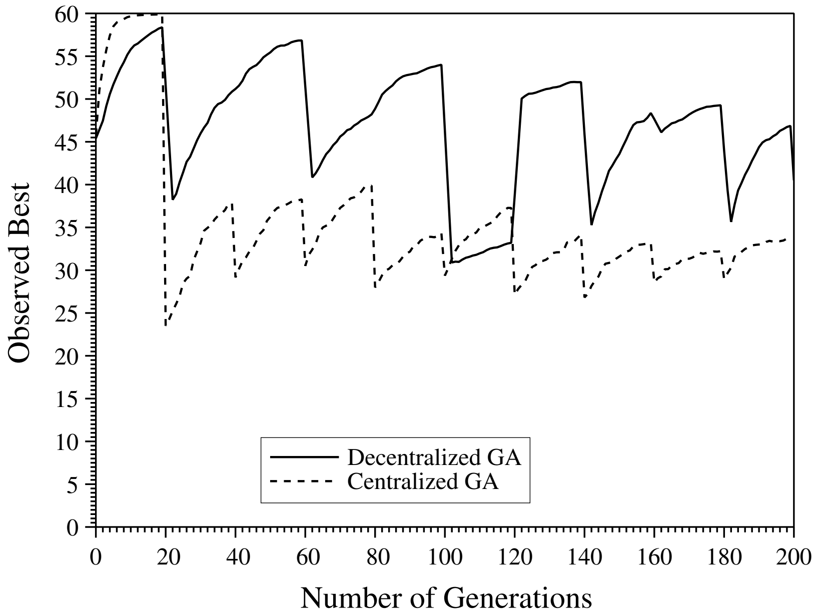

Figure 7.6: Performance curves for decentralized (radii ratio=0.15) and centralized GAs for max location moved every 20 generations.

ability in the case of the decentralized spatially distributed GA.

In the spatially distributed GAs the individual at the grid location is always replaced by the offspring created, irrespective of the fitness value of the offspring. One of the things that seemed to affect the performance of the decentralized spatially distributed GA is this replacement rule. Potentially good solutions were lost before they could be explored further. A deterministic form of elitism (replace individual only if offspring is better) is not a suitable alternative as the diversity is lost fairly rapidly and this hinders the tracking ability. Some other softer forms of elitism need to be examined. Also the behavior of the other selection algorithms like tournament and random walk selection should be studied.

7.2.4 Translation of the Entire Landscape

In this section, the ability of the spatially distributed GA to track a landscape that shifts in entirety is determined. The entire landscape can shift with time. This was simulated by moving all the hills, either in slow mode or in fast mode, every 5 generations. In the slow mode, the location of the hills were incremented by ± 2.0. In the fast mode the location of the hills were incremented by ± 5.0. Each dimension could increase or decrease and their movements are independent. This setup is similar to the one used by Cobb and Grefenstette (1993). The landscape is varied slowly (increments of ± 2.0) every 5 generations for the first 50 generations then switched to larger increments (± 5.0) for the next 50 generations. This translation is combined with the random movement of the location of the max hill at fixed intervals. Hence for the last 100 generations the location of the maximum hill is varied every 20 generations.

Figure 7.7 displays the results of this experiment using a decentralized GA with a radii ratio of 0.33 and a centralized GA. Figure 7.8 compares the performance of the two systems (decentralized GA with a radii ratio of 0.15 and a centralized GA). From these two figures we can see that the higher selection pressure in the case of centralized GA helps in the beginning where it is able to find the optimum faster than the decentralized GA. But as the landscape varies with time, the spatially distributed GA is able to track the optimum when the entire landscape is shifting.

Interestingly in the later stages of the search when the location of the maximum hill is being translated the decentralized GA is able to track the moving optimum better than the centralized GA. We have reduced the neighborhood size to the smallest possible for the grid used and the only other way to reduce the radii ratio is to increase the grid radius. It is possible to change the square grid to a rectangular grid to achieve a larger grid radius and thus reduce the radii ratio. We believe that our current theoretical results will hold rectangular grids also. This has to be tested in future studies.

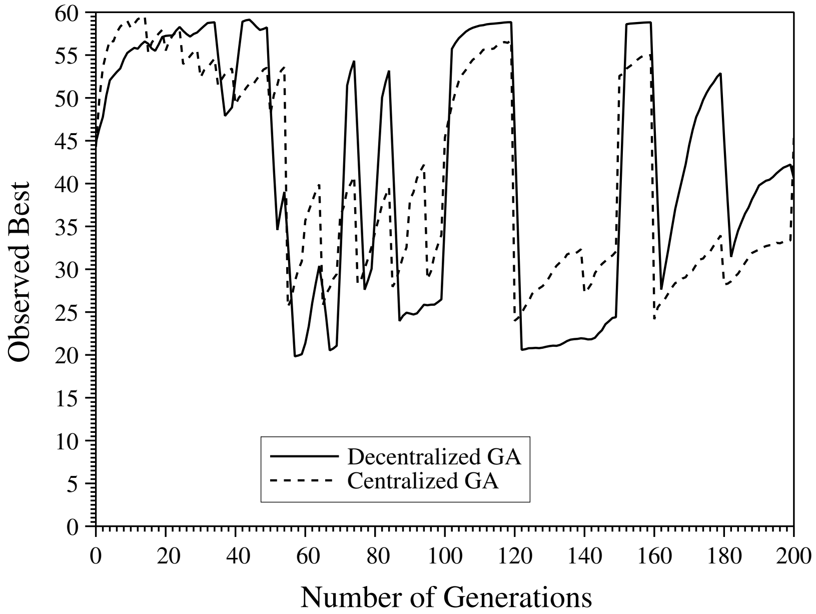

Figure 7.7: Performance curves for decentralized GA (radii ratio=0.33) and centralized GA for translation of entire landscape.

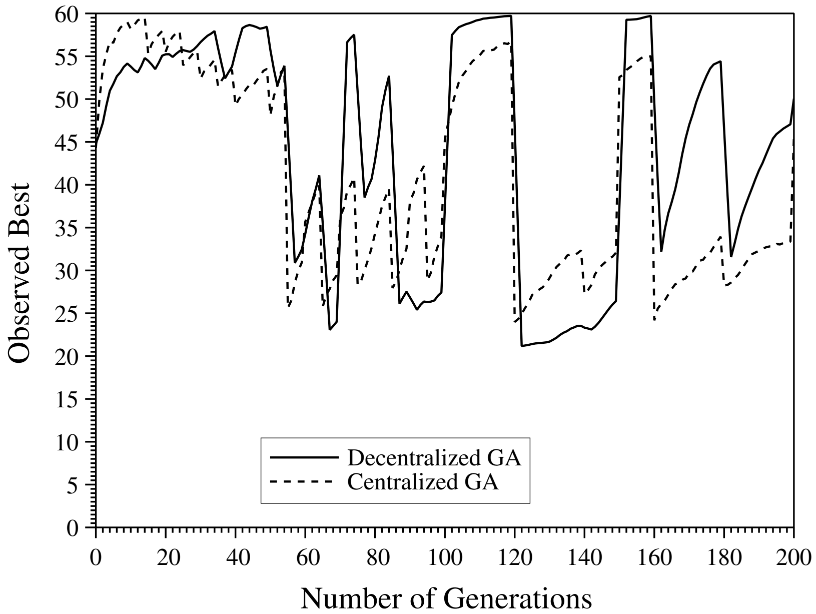

Figure 7.8: Performance curves for decentralized GA (radii ratio=0.15) and centralized GA for translation of entire landscape.

The spatially distributed does appear to have the potential to be applied to problems in the nonstationary environment without much changes. We can observe that no random replacement of the population or hypermutation is required to track the moving optimum in the spatially distributed GA. By reducing the radii ratio an improvement in the tracking ability is seen. These results suggest that changing the radii ratio dynamically will aid in tracking the moving optimum better.

7.3 Summary

We first explored the ability of the spatially distributed GA in solving multimodal functions. We found that if the competing alternate solutions are of the same height (fitness value) then the spatially distributed GA is able to locate and maintain some of these peaks. The spatial distribution and a low radii ratio does help to locate the peaks in a multimodal function without any fitness sharing or mating restriction. But when the peaks differ in height, as expected, the search converges to the highest peak. Some form of the crowding scheme or mating restriction will be necessary to sustain the lower peaks in a fitness landscape with differing heights.

We then explored the ability of the spatially distributed GAs to track a moving optimum using different scenarios. These are very preliminary results, but are enough proofs of principle to show that the spatially distributed GAs will be able to track a moving optimum in a nonstationary environment.

One of the things that hurt the performance in this model, was the loss of potentially good solutions before they could be explored further. Recall that each individual at the grid location is always replaced by the offspring, irrespective of the fitness value of the offspring. Using a deterministic form of elitism (replace only when better) is not a suitable alternative as the diversity is lost fairly quickly and the spatially distributed GA is unable to track the moving optimum. Some other softer forms of elitism should be examined. Also the behavior of the other selection algorithms like tournament and random walk selection should be studied.

It is apparent that the spatially distributed GA, without any major modifications, can be used to solve problems in nonstationary environments and to locate competing alternative solutions in multimodal functions. This is an area that has not been explored prior to this study and certainly merits a more rigorous treatment.

Chapter 8

SUMMARY AND CONCLUSIONS

A number of interesting design issues arise when one wants to implement and use a decentralized and spatially distributed GA. The implications of these design decisions have not been well understood. The objective of this research has been to understand and characterize the behavior of the decentralized and spatially distributed genetic algorithms. This will help in making a more informed choice when designing and implementing a spatially distributed GA. Some of the open issues identified were: (1) is there an optimal topology for a given class of problems, (2) is there an optimal neighborhood size given a topology and how does changes in the neighborhood size influence the performance, and (3) what selection strategy should be used and how do the various selection strategies affect the search behavior.

Most of the commonly used sequential GAs have fairly strong explicit and implicit forms of centralized control. For these centralized GAs there are both theoretical and experimental results. Decentralization of these GAs involves changing the semantics of the algorithm which alters the behavior. Decentralizing the sequential GA helps to implement them on parallel and distributed systems easily. In chapter 4, we explored the possibility of decentralizing the selection strategy used in the GA while keeping the semantics unaltered. Of the different selection strategies, only tournament selection could be easily decentralized without any change in the semantics. But using a decentralized tournament selection leads to a higher variance which is strongly correlated with poor search performance especially in small population sizes.

Selection variance can be reduced either by using better sampling techniques and/or by increasing the population size. Using large population sizes with global selection pool can lead to an increase in communication costs when implemented on a parallel machine. The alternative is to define local selection pools. This was done by introducing a topology on the population which, however, changed the semantics of the algorithm.

But changing the semantics alters the behavior of the algorithm which needs to be studied. So, we examined the selection pressure induced by the standard local selection algorithms (fitness proportional, linear ranking and binary tournament) in a local neighborhood model. The selection pressure induced by the local selection algorithms on the entire population is called the "emergent" global selection pressure. We demonstrated that the emergent global selection pressure induced by the three selection algorithms was qualitatively similar to its sequential counterpart but quantitatively weaker.

Next, in chapter 5, we formally analyzed the selection methods used in a spatially distributed GA by extending the growth curve techniques to characterize the selection pressure induced by selection algorithms in sequential GAs. In the spatially distributed GAs, the growth curve is a combination of the effects of logistic growth occurring in each local neighborhood and the propagation times required to disperse the best individual globally through the grid. The selection algorithm defines the logistic growth that occurs in the local neighborhood while the propagation time depends on the neighborhood size. Since the local neighborhoods in the spatially distributed GA overlap, this provides an implicit mechanism for the migration of information through the topology. Thus, how slowly, or how quickly the information propagates depends on the neighborhood pattern.

We analyzed the propagation times when fitness proportional, linear ranking and binary tournament selection algorithms are used in the local neighborhood model. We had to define an appropriate measure based on spatial analysis techniques, to calculate the effective radius of the different neighborhood patterns. The measure had to take into account the internal layout of the neighborhoods. The radius measure was able to discriminate between neighborhoods that had the same size but had different layouts. The critical parameter was the ratio of the neighborhood radius to the radius of the grid (radii ratio). By varying this radii ratio, the emergent selection pressure induced on the global population can be directly controlled. A quantitative model of this relationship was presented in which the coefficient of the growth rate of the best individual in the population was shown to be an inverse exponential function of this radii ratio. Higher radii ratios lead to faster growth rates which implies that the induced selection pressures are stronger. Hence, the radii ratio could be adjusted to control the selection pressure. The analytical model was then used to investigate an alternate selection method, called the random walk selection, used for selecting parents in a spatially distributed GA. We showed that the random walk method is similar to the tournament selection method but has a distance bias introduced on the members in the neighborhood.

In chapter 6 we made predictions using the analytical tool developed in the chapter 5. These predictions were validated using test functions from the function optimization domain. We demonstrated that the analytical model can be used to make reasonably good qualitative and quantitative predictions in a "real" spatially distributed GA. One of the advantages of the decentralized and spatially distributed GA is that the selection pressure can be controlled by both the choice of selection method and the neighborhood radius. This advantage gives the user a finer degree of control over selection pressure than is possible with centralized sequential GAs. The analytical model provides the information necessary to make appropriate design decisions regarding the selection method, the neighborhood size and the topology. The analytical model can also explain the results of the various empirical studies in spatially distributed GAs. We showed how the model can be used to adjust the selection pressure by changing the neighborhood radius.

In chapter 7, using the insights gained about the behavior of the spatially distributed GA, it was applied to problems from time-varying and multimodal function optimization domains. We showed that the selection pressure can be adjusted by changing the radii ratio to improve performance in multimodal function optimization. We then demonstrated that the spatially distributed GAs without any major modifications can be used to track the moving optimum in nonstationary environments.

8.1 Contributions

We showed that the *emergent* global selection pressure induced by the selection algorithms in a decentralized and spatially distributed GA is qualitatively similar to the corresponding centralized sequential GA, but quantitatively weaker. The emergent global selection pressure is the selection pressure induced by the local selection algorithms on the entire population. The direct consequence of this result is that much of our knowledge about the selection algorithms in the centralized sequential GAs can be transferred to the spatially distributed GAs.

We introduced a distance measure from the spatial analysis techniques to compute the neighborhood radius, as the existing measures could not take into account the internal layout of the neighborhood. This new measure discriminates between neighborhoods which have the same overall size but different shapes. A major accomplishment of this research is the characterization of the selection pressure in terms of the radii ratio (ratio of the neighborhood radius to the grid radius) and the development of a quantitative model. The selection algorithms analyzed are fitness proportional, linear ranking, binary tournament and random walk selections. We demonstrated how the selection pressure in a spatially distributed GA can be adjusted either by changing the type of selection algorithm used or by changing the radii ratio.

Our analytical model can be utilized by an user to make effective design decisions. How changes in neighborhood size influence performance behavior, what selection strategy to use, and how these selection strategies affect search behavior are answered by our analytical model. If the performance is not satisfactory for a given problem, the user can use the analytical model to either increase or decrease the selection pressure to improve performance. The radii ratio or the selection method can be changed to achieve better performance. This was demonstrated in chapter 6.

The analytical and empirical results provide clearer insights into the behavior of the spatially distributed GAs. As is the case with sequential GAs, there is no optimal level for selection pressure; rather it depends on the fitness landscape of the problem being solved. Also, there is no optimal neighborhood size or shape; rather the neighborhood topology controls the selection pressure which can be adjusted to improve performance.

Our analytical model was used to select the topology and neighborhood radius for locating the competing alternate solutions in a multimodal function landscape. The spatially distributed model was tested in a time-varying landscape. We gained additional insights into the behavior of these spatially distributed model in these two classes of problems.

8.2 Limitations

The focus of the analysis has been on the effects of selection while keeping the genetic operators fixed. The effects of using a different type of crossover operator, like uniform crossover, multi-point crossover, were not explored. Changing the type of crossover operator to something other than two-point crossover has the effect of altering the balance between the exploration and exploitation and can affect the performance behavior.

The behavior of the spatially distributed GAs in multimodal functions was studied using two simple test functions with a small number of competing solutions. When the competing peaks differed in height the search converged to the highest peak. How to sustain these lower peaks in the fitness landscape was not studied.

The ability of the spatially distributed GAs to track a moving optimum was a preliminary study. Only one test function and two different types of nonstationarity was considered. A more extensive study is required to generalize the results.

In this dissertation, only square two-dimensional topologies were considered and the analytical model is based on these types of topologies. We believe that the theoretical results can be extended to non-square topologies and also to dimensions greater than two.

8.3 Future Research Directions

The advantage of using a spatially distributed GA is that the balance between exploration and exploitation can be easily altered by using the analytical model to select a different radii ratio or selection algorithm. An extension would be to dynamically adapt the selection pressure during the search process by changing the neighborhood radius to maintain this balance. This would also help in solving problems in the time varying environment. Future studies should examine the effects of dynamically changing the selection pressure.

The spatially distributed GA was not very successful in maintaining the different peaks in a multimodal landscape. In centralized sequential GAs subpopulations are created by explicitly defining a distance metric in the genotype or phenotype space or adding a tag template to each individual in the representation space. Unlike the techniques used in centralized sequential GAs to form and maintain niches and species, in spatially distributed GAs the subpopulations are formed implicitly depending on the location of the individuals. Using some method like preselection or crowding can help in maintaining some of the peaks. The formation of niches and species need to examined in more detail.

In a spatially distributed model, the individual at the grid is always replaced by the offspring. This caused the loss of potentially good solutions before they could be explored further. Using a deterministic form of elitism (replace only when better) was not a suitable alternative as the diversity was lost quickly and the spatially distributed GA was no longer able to track the moving optimum. Some other probabilistic forms of local elitist policy need to be explored when spatially distributed GAs are applied to problems in time-varying environments. The spatially distributed GA needs to be applied to real world nonstationary problems to further test and validate its applicability.

The operators used can offset the higher selection pressure induced by a selection algorithm. We used only the standard operators and their settings. Different types of crossover operators, like uniform crossover operator, which are considered to be more exploratory than two point crossover, should be tested with the spatially distributed GA.

Recent studies have developed new test generators for studying the behavior of evolutionary algorithms (De Jong, Potter, and Spears 1997). The spatially distributed GA should be tested using these problem generators to gain further insights into its behavior. Also the predictions of the analytical model should be tested with non-square grids to test its applicability in those topologies.

BIBLIOGRAPHY

BIBLIOGRAPHY

Ankenbrandt, C. A. (1991). An extension to the theory of convergence and a proof of the time complexity of genetic algorithms. In *Foundations of Genetic Algorithm*, San Mateo, CA, pp. 53–68. Morgan Kaufmann.

Bäck, T. and H.-P. Schwefel (1993). An overview of evolutionary algorithms for parameter optimization. *Evolutionary Computation 1*(1), 1–23.

Baker, J. E. (1987). Reducing bias and inefficiency in the selection algorithm. In *Proceedings of the Second International Conference on Genetic Algorithms*, Cambridge, MA, pp. 14–21. Lawrence Erlbaum Associates.

Baker, J. E. (1989). *An Analysis of the Effects of Selection in Genetic Algorithms.* Ph. D. thesis, Vanderbilt University, Nashville, TN.

Baluja, S. (1993). Structure and performance of fine-grained parallelism in genetic search. In *Proceedings of the Fifth International Conference on Genetic Algorithms*, Urbana–Champaign, IL, pp. 155–162. Morgan Kaufmann.

Banks, R. B. (1994). *Growth and Diffusion Phenomena.* Springer-Verlag.

Beasley, D., D. R. Bull, and R. Martin (1993). A sequential niche techniquie for multimodal function optimization. *Evolutionary Computation 1*(2), 101–126.

Cobb, H. G. (1990). An investigation into the use of hypermutation as an adaptive operator in genetic algorithms having continuous, time-dependent nonstationary environments. Technical Report Memorandum Report 6760, Naval Research Laboratory.

Cobb, H. G. and J. J. Grefenstette (1993). Genetic algorithms for tracking changing environments. In *Proceedings of the Fifth International Conference on Genetic Algorithms*, Urbana–Champaign, IL, pp. 523–530. Morgan Kaufmann.

Cohoon, J. P., W. N. Martin, and D. S. Richards (1991). A multi–population genetic algorithms for solving the K-partition problem on hypercubes. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, San Diego, CA, pp. 244–248. Morgan Kaufmann.

Collins, R. J. (1992). *Studies in Artificial Evolution.* Ph. D. thesis, University of California, Los Angeles.

Collins, R. J. and D. R. Jefferson (1991). Selection in massively parallel genetic algorithms. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, San Diego, CA, pp. 249–256. Morgan Kaufmann.

Dasgupta, D. and D. R. McGregor (1992). Nonstationary function optimization using the structured genetic algorithm. In R. Männer and B. Manderick (Eds.), *Proceedings Parallel Problem Solving from Nature - PPSN II*, Brussels, Belgium, pp. 145–154. Elsevier Science Publishers.

Davidor, Y. (1991). A naturally occuring niche & species phenomenon: The model and first results. In R. K. Belew and L. B. Booker (Eds.), *Proceedings of the Fourth International Conference on Genetic Algorithms*, San Diego, CA, pp. 257–263. Morgan Kaufmann.

Davis, L. (1991). Bit climbing, representational bias, and test suite desgin. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, San Diego, CA, pp. 18–23. Morgan Kaufmann.

Deb, K. and D. E. Goldberg (1989). An investigation of niche and species formation in genetic function optimization. In J. D. Schaffer (Ed.), *Proceedings of the Third International Conference on Genetic Algorithms*, Fairfax, VA, pp. 42–50. Morgan Kaufmann.

De Jong, K., M. A. Potter, and W. M. Spears (1997). Using problem generators to explore the effects of epistasis. In *Proceedings of the Seventh International Conference on Genetic Algorithms*, East Lansing, MI, pp. 338–345. Morgan Kaufmann.

De Jong, K. and J. Sarma (1995). On decentralizing selection algorithms. In *Proceedings of the Sixth International Conference on Genetic Algorithms*, Pittsburgh, PA, pp. 17–23. Morgan Kaufmann.

De Jong, K. A. (1975). *An Analysis of Behavior of a Class of Genetic Adaptive Systems*. Ph. D. thesis, University of Michigan, Ann Arbor.

De Jong, K. A. (1985). Genetic algorithms: A 10 year perspective. In *Proceedings of an International Conference on Genetic Algorithms and their Applications*, Pittsburgh, PA, pp. 169–177. Lawrence Erlbaum Associates.

De Jong, K. A. (1988). Learning with genetic algorithms: An overview. *Machine Learning 3*(3), 121–138.

De Jong, K. A. (1992). Are genetic algorithms function optimizers? In R. Männer and B. Manderick (Eds.), *Proceedings Parallel Problem Solving from Nature - PPSN II*, Brussels, Belgium, pp. 3–13. Elsevier Science Publishers.

De Jong, K. A. and J. Sarma (1993). Generation gaps revisited. In L. D. Whitley (Ed.), *Foundations of Genetic Algorithms - 2*, pp. 19–28. San Mateo, CA: Morgan Kaufmann.

De Jong, K. A. and W. M. Spears (1989). Using genetic algorithms to solve NP-Complete problems. In *Proceedings of the Third International Conference on Genetic Algorithms*, Fairfax, VA, pp. 124–132. Morgan Kaufmann.

De Jong, K. A. and W. M. Spears (1993). On the state of evolutionary computation. In *Proceedings of the Fifth International Conference on Genetic Algorithms*, Urbana–Champaign, IL, pp. 618–623. Morgan Kaufmann.

Feller, W. (1968). *An Introduction to Probability Theory and Its Applications - Vol. I*. New York: John Wiley & Sons, Inc.

Fogel, D. B. (1994). Evolutionary programming: an introduction and some current directions. *Statistics and Computing 4*, 113–129.

Fogel, L. J., A. J. Owens, and M. J. Walsh (1966). *Artificial Intelligence through Simulated Evolution*. New York: John Wiley & Sons.

Goldberg, D. E. and K. Deb (1991). A comparative analysis of selection schemes used in genetic algorithms. In *Foundations of Genetic Algorithms*, San Mateo, CA, pp. 69–93. Morgan Kaufmann.

Goldberg, D. E., B. Korb, and K. Deb (1989). Messy genetic algorithms: Motivation, analysis, and first results. *Complex Systems 3*, 493–530.

Goldberg, D. E. and J. Richardson (1987). Genetic algorithms with sharing for multimodal function optimization. In *Proceedings of the Second International Conference on Genetic Algorithms*, Cambridge, MA, pp. 41–49. Morgan Kaufmann.

Goldberg, D. E. and R. E. Smith (1987). Nonstationary function optimization using genetic algorithms with dominance and diploidy. In *Proceedings of the Second International Conference on Genetic Algorithms*, Cambridge, MA, pp. 59–68. Morgan Kaufmann.

Gorges-Schleuter, M. (1989). ASPARGOS an asynchronous parallel genetic optimization strategy. In *Proceedings of the Third International Conference on Genetic Algorithms*, Fairfax, VA, pp. 422–427. Morgan Kaufman.

Gorges-Schleuter, M. (1990). *Genetic Algorithms and Population Structures - A Massively Parallel Algorithm*. Ph. D. thesis, University of Dortmund, Germany.

Grefenstette, J. J. (1981). Parallel adaptive algorithms for function optimization. Technical Report T.R. no. CS-81-19, Vanderbilt University.

Grefenstette, J. J. (1984). GENESIS: A system for using genetic search procedures. In *Proceedings of the 1984 Conference on Intelligent Systems and Machines*, pp. 161–165.

Grefenstette, J. J. (1992). Genetic algorithms for changing environments. In R. Männer and B. Manderick (Eds.), *Proceedings Parallel Problem Solving from Nature - PPSN II*, Brussels, Belgium, pp. 137–144. Elsevier Science Publishers.

Grosso, P. B. (1985). *Computer Simulations of Genetic Adaptation: Parallel Subcomponent Interaction in a Multilocus Model.* Ph. D. thesis, University of Michigan, Ann Arbor, MI.

Hart, W. E. and R. K. Belew (1991). Optimizing an arbitrary function is hard for the genetic algorithm. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, San Diego, CA, pp. 190–195. Morgan Kaufmann.

Hillis, D. W. (1991). Co-evolving parasites improve simulated evolution as an optimization procedure. In C. G. Langton, C. Taylor, J. D. Farmer, and S. Rasmussen (Eds.), *Proceedings of Artificial Life II*, pp. 313–323. Addison-Wesley.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems.* Ann Arbor: University of Michigan Press.

Mahfoud, S. W. (1992). Crowding and preselection revisited. In R. Männer and B. Manderick (Eds.), *Proceedings Parallel Problem Solving from Nature - PPSN II*, Brussels, Belgium, pp. 27–36. Elsevier Science Publishers.

McInerney, J. (1992). *Biologically Influenced Algorithms and Parallelism in Non–linear Optimization.* Ph. D. thesis, University of California, San Diego.

Mühlenbein, H. (1989). Parallel genetic algorithms, population genetics and combinatorial optimization. In *Proceedings of the Third International Conference on Genetic Algorithms*, Fairfax, VA, pp. 416–421. Morgan Kaufmann.

Mühlenbein, H. (1991). Evolution in time and space – the parallel genetic algorithm. In *Foundations of Genetic Algorithm*, San Mateo, CA, pp. 316–337. Morgan Kaufmann.

Pearson, K. (1905). The problem of the random walk. *Nature 72*, 294.

Pettey, C. C. and M. R. Leuze (1989). A theoretical investigation of a parallel genetic algorithm. In *Proceedings of the Third International Conference on Genetic Algorithms*, Fairfax, VA, pp. 398–405. Morgan Kaufmann.

Pettit, E. and K. M. Swigger (1983). An analysis of genetic–based pattern tracking and cognitive–based component tracking models of adaptation. In *Proceedings of the National Conference on Artificial Intelligence (AAAI-83)*, pp. 327–332. Morgan Kaufmann.

Rechenberg, I. (1973). *Evolutionsstrategie: Optimierung Technischer Systeme nach Prizipien der Biologischen Evolution.* Stuttgart: Frommann-Holzboog.

Schwefel, H.-P. (1981). *Numerical Optimization of Computer Models.* New York: John Wiley & Sons. English translation of Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie (1977).

Schwehm, M. (1992). Implementation of genetic algorithms on various interconnection networks. In M. Valero, E. Oñate, M. Jane, J.L.Larriba, and B. Suárez

(Eds.), *Parallel Computing and Transputer Applications*, pp. 195–203. Amsterdam: IOS Press.

Schwehm, M. (1993). A massively parallel genetic algorithm on the maspar MP-1. In *Proceedings of the International Conference Artificial Neural Nets and Genetic Algorithms*, Innsbruck, Austria, pp. 502–507. Springer-Verlag.

Schwehm, M. (1994). Massively parallel genetic algorithms. In *Proceedings of the International Conference Massively Parallel Processing*, Delft, The Netherlands, pp. 505–512. Elsevier Science Publ.

Smith, J. M. (1989). *Evolutionary Genetics*. Oxford University Press.

Smith, R. E., S. Forrest, and A. S. Perelson (1993). Searching for diverse, cooperative populations with genetic algorithms. *Evolutionary Computation 1*, 127–149.

Spears, W. M. (1994). Simple subpopulation schemes. In A. V. Sebald and D. B. Fogel (Eds.), *Proceedings of the Third Conference on Evolutionary Programming*, pp. 297–307. World Scientific Publ.

Spears, W. M., K. A. De Jong, T. Bäck, D. B. Fogel, and H. de Garis (1993). An overview of evolutionary computation. In T. be filled (Ed.), *Proceedings of the European Conference on Machine Learning*, pp. 442–459. To be filled.

Spiessens, P. and B. Manderick (1991). A massively parallel genetic algorithm: Implementation and first analysis. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, San Diego, CA, pp. 279–287. Morgan Kaufmann.

Suh, J. Y. and D. V. Gucht (1987). Distributed genetic algorithms. Technical Report T.R. no. 225, Indiana University.

Syswerda, G. (1989). Uniform crossover in genetic algorithms. In J. D. Schaffer (Ed.), *Proceedings of the Third International Conference on GeneticAlgorithms*, Fairfax, VA, pp. 2–9. Morgan Kaufmann.

Syswerda, G. (1991). A study of reproduction in generational and steady-state genetic algorithms. In G. J. E. Rawlins (Ed.), *Foundations of Genetic Algorithms - 1*, pp. 94–101. San Mateo, CA: Morgan Kaufmann.

Tanese, R. (1989). Distributed genetic algorithms. In J. D. Schaffer (Ed.), *Proceedings of the Third International Conference on GeneticAlgorithms*, Fairfax, VA, pp. 434–439. Morgan Kaufmann.

Unwin, D. (1981). *Introductory Spatial Analysis*. Methuen and Co. Ltd.

Vavak, F., T. C. Fogarty, and K. Jukes (1996). A genetic algorithm with variable range of local search for tracking changing environments. In *Proceedings Parallel Problem Solving from Nature - PPSN IV*, Berlin, Germany, pp. 376–385. Springer-Verlag.

Vavak, F., K. Jukes, and T. C. Fogarty (1997). Adaptive combustion balancing in multiple burner boiler using a genetic algorithm with variable range of local search. In *Proceedings of the Seventh International Conference on Genetic Algorithms*, East Lansing, MI, pp. 719–726. Morgan Kaufmann.

White, P. M. and C. C. Pettey (1997). Double selection vs. single selection in diffusion model gas. In *Proceedings of the Seventh International Conference on Genetic Algorithms*, East Lansing, MI, pp. 174–180. Morgan Kaufmann.

Whitley, D. (1989). The GENITOR algorithm and selection pressure: Why rank-based allocation of reproductive trials is best. In *Proceedings of the Third International Conference on Genetic Algorithms*, Fairfax, VA, pp. 116–123. Morgan Kaufmann.

Whitley, D., K. Mathias, S. Rana, and J. Dzubera (1995). Building better test functions. In *Proceedings of the Sixth International Conference on Genetic Algorithms*, Pittsburgh, PA, pp. 239–246. Morgan Kaufmann.

Whitley, D. and T. Starkweather (1990). GENITOR II: a distributed genetic algorithm. *Journal of Experimental and Theoretical Artificial Intelligence 2*, 189–214.

Wright, S. (1932). The role of mutation, inbreeding, crossbreeding and selection in evolution. In *Proceedings of Sixth International Congress of Genetics*, Ithaca, NY, pp. 356–366.

Wright, S. (1943). Isolation by distance. *Genetics 28*, 114–138.

Wright, S. (1969). *Vol. II The theory of Gene Frequencies.* The University of Chicago Press.

APPENDICES

Appendix A

DESCRIPTION OF THE TEST FUNCTIONS USED

The hardest task is to define an appropriate set of test functions on which the algorithm can be tested so that results can be generalized. We know that GAs are robust with respect to various types of fitness landscape from the vast number of studies performed. In this dissertation we are more interested in characterizing the behavior of the algorithm with respect to the topology, different neighborhood sizes, and selection algorithms used. Design of a suitable test problems is another extensive research field by itself (Davis 1991; Whitley, Mathias, Rana, and Dzubera 1995). We ensured that the test suite consisted of scalable function, unimodal and multi-modal, continuous and discontinuous, noisy functions and Boolean functions. The functions are described below.

The Rastrigin function :

$$f(\vec{x}) = An + \sum_{i=1}^{n} x_i^2 - A\cos(2\pi x_i) \tag{A.1}$$

where $n = 20$, $A = 3$, and $-5.12 \leq x_i \leq 5.12$.

The global minimum of zero is at the point $\vec{x} = (0, 0, \ldots)$. The primary characteristic of this function is the existence of many suboptimal points whose values increase as the distance from the global optimum point increases. The variables in this equation is separable and can be optimized independently.

The Schwefel function :

$$f(\vec{x}) = An + \sum_{i=1}^{n} x_i \sin x \tag{A.2}$$

where $n = 10$, $A = 418.9829$, and $500 \leq x_i \leq 500$.

The global minimum of zero is at $\vec{x} = (0, 0, \ldots)$. There are many local optima which are far apart in the space. The second best minimum is far away from the global minimum, which can trap an optimization algorithm thus produce a suboptimal solution. The variables in this equation are separable.

The Griewangk function :

$$f(\vec{x}) = \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \cos(\frac{x_i}{\sqrt{i}}) + 1 \tag{A.3}$$

where $n = 10$, and $-600 \leq x_i \leq 600$.

The global minimum is zero is at $\vec{x} = (0, 0, \ldots)$. In 10 dimensions, the function has 4 local optima. There is an interdependency between the variables because of the product term.

The Rosenbrock function :

$$f(\vec{x}) = 100(x_1^2 - x_2)^2 + (1 - x_1)^2 \tag{A.4}$$

where $-2.048 \leq x_i \leq 2.048$.

The global minimum of zero is at the point (1,1). In this function too there is a strong interdependency between the variables. When the number of dimensions is greater than 2, it is called the generalized Rosenbrock function.

Boolean Satisfiability Functions:

The Hamiltonian Circuit (HC) and the N-Peak (NP) problems are a class of the boolean satisfiability functions (De Jong and Spears 1989). The goal in a boolean satisfiability problem is find the correct assignment to the variables in the boolean expression so that boolean expression is true. Some of the problems that falls in this category are the traveling salesperson, job shop scheduling, bin packing, graph partitioning, and quadratic assignment problem.

The N-Peak problem has only one global optimum but has (N-1) local optima. The complexity of the problem can be increased by using raising the value of N. The 5-Peak (NP5) and 6-Peak (NP6) are the two problems used. The string length is 30. The Hamiltonian Circuit problem consists of finding a path through a directed graph such that all the nodes are touched exactly once. This is a NP-complete problem. The number after the HC denotes the number of nodes in the graph and in our case we used 10 (HC10) and 12 (HC12) nodes. For each value of N, the directed graph contains approximately $\frac{N^2}{2}$ edges and there is only one solution. But there are many suboptimal (local) solutions. The optimal solution in both cases has a fitness value of 1.0 while solutions that partially satisfied the boolean expression have fitness values less than 1.0.

Appendix B

RANDOM WALK

The term *random walk* was introduced by Karl Pearson (1905), where the problem was to find the probability that a person starting from a certain location will be within a distance of d and $d + \delta d$ from his starting point. Random walk examples are generally posed as the aimless wandering of a drunkard. The statisticians try to answer questions like where a drunkard will be after a certain number of steps, or the probability the drunkard will reach home. The normal or Gaussian distribution is generally used to characterize the movement of the drunkard (Feller 1968). An one dimensional random walk of length w induces a normal distribution with mean 0 and a standard deviation $\sqrt{w}$. A two-dimensional random walk of length w follows a normal distribution with mean 0 and a standard deviation $\sqrt{2w}$.

We will first show the derivation for a one-dimensional random walk and then extend to higher dimensions. Assume that a person is starting a walk along an axis (for example the x-axis). The person starts the walk at position $x = 0$ at time t. The following assumptions are made:

1. The person can move either to the left or to the right every t seconds.
2. The distance covered in each time step be δ.
3. The two variables t and δ are constants and are equal to 1.
4. The probability of going to the right is p and the probability of going to the left is q and $p + q = 1$.

5. The walk is not biased towards any side, i.e., $p = q = 0.5$. This kind of walk is said to be a symmetric walk.

Let the starting position be $S_0 = 0$. If n is the number of steps taken, then after the n^{th} step, the walker will be at position S_n. Thus

$S_n = X_1 + X_2 + \cdots + X_n$,

where $X_1, X_2, \ldots$ are independent random variables assuming the values $+1$ and -1 with probabilities p and q respectively.

Since the position of the walker after the n^{th} step differs from the previous step by $\pm\delta$, i.e.,

$S_n = S_{n-1} \pm 1$, where $\delta = 1$.

We would like to know the mean, variance and the shape of the distribution of the random walk. The probability that the walker will move exactly k times to the right in n trials is given by the binomial distribution (Feller 1968)

$b(k; n, p) = \binom{n}{k} p^k q^{n-k}$

Hence the location the walker after n steps, S_n, will be equal to the number of steps taken to the right minus the number of steps taken to the left times the step length δ (which is assumed to be equal to 1).

$S_n = [k - (n - k)] = 2k - n$

Since k follows a binomial distribution, S has to be a binomial distribution too. The expected mean of S_n is :

$E(S_n) = 2E(k) - n$

We know that $E(k)$, the expectation of k, is np (Feller 1968). Substituting that in the above equation, we get

$E(S_n) = (2p - 1)n$

Since the random walk is assumed to be symmetric, i.e., $p = q = 1/2$, the expected mean of the random walk is zero. The variance of the random walk of length n is $4npq$ (Feller 1968). Substituting the values of p and q, we find that the variance is equal to the length of the random walk n. Thus a random walk of length n induces a normal distribution with mean 0, and a standard deviation $\sqrt{n}$.

In a two-dimensional random walk, the walker can move in any of the four directions (along the $x-$ or $y-$ axis). Thus each point will have four neighbors. The random walk is assumed to be symmetric and hence the probability of moving in any direction is the same. The two-dimensional random walk can be analyzed along the same lines as above by assuming that the walks along an axis is independent and symmetric (Feller 1968). Thus it can be shown that a two-dimensional random walk of length n follows a normal distribution with mean 0 and a standard deviation $\sqrt{2n}$.

CURRICULUM VITAE

Jayshree Sarma was born in India and is now a US citizen. She received a Bachelors of Commerce degree and a Masters of Commerce degree from the University of Bombay, India. She received a Master of Science degree in Information Systems and a Doctor of Philosophy degree from George Mason University. Her research interests are in the areas of evolutionary computation, neural networks, machine learning and parallel computation.

Jayshree Sarma has presented papers in these areas in several conferences including the International Conference on Genetic Algorithms, Foundations of Genetic Algorithms, and the International Joint Conference on Neural Networks. She has coauthored a chapter on Generation Gap methods in "The Handbook of Evolutionary Computation". She was a co-organizer of the workshop on Parallel Genetic Algorithms at the Sixth International Conference on Genetic Algorithms.

Before coming to the USA, she worked as a Systems Engineer with Computer Maintenance Corporation, Bombay, India. Prior to that she worked as a Finance Executive with the Tata group of companies in India.