2013

Fitness Proportionate Niching: Harnessing The Power Of Evolutionary Algorithms For Evolving Cooperative Populations And Dynamic Clustering

Abrham Tibebu Workineh
North Carolina Agricultural and Technical State University

Fitness Proportionate Niching: Harnessing the Power of Evolutionary Algorithms for Evolving Cooperative Populations and Dynamic Clustering

Abrham Tibebu Workineh

North Carolina A&T State University

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Department: Electrical and Computer Engineering

Major: Electrical Engineering

Major Professor: Dr. Abdollah Homaifar

Greensboro, North Carolina

2013

School of Graduate Studies
North Carolina Agricultural and Technical State University
This is to certify that the Doctoral Dissertation of

Abrham Tibebu Workineh

has met the dissertation requirements of
North Carolina Agricultural and Technical State University

Greensboro, North Carolina
2013

Approved by:

Dr. Abdollah Homaifar Major Professor	Dr. Gary Lebby Committee Member
Dr. Albert Esterline Committee Member	Dr. Jung Kim Committee Member
Dr. Robert Li Committee Member	Dr. John Kelly Department Chair

Dr. Sanjiv Sarin
Dean, The Graduate School

Biographical Sketch

Abrham Tibebu Workineh was born in 1982 in Amarasayint, South Wollo, Ethiopia. He received his bachelor's degree in Computer Engineering from Addis Ababa University (Ethiopia) in 2006 with Great Distinction and his master's degree in Electrical Engineering from North Carolina A & T State University in 2010, with Highest Honor.

Dedication

This dissertation is dedicated to my late grandmother, Kelem Atrsaw, to whom I am deeply indebted to her unceasing love and care and attribute to her most of my educational success and personal values. I wish you lived long enough to see this but your love will remain in my heart forever. I love you and rest in peace.

Acknowledgements

First and foremost, my heartfelt gratitude goes to Almighty God for He gave me the wisdom, passion and stamina throughout my work.

I am very thankful to my advisor, Dr. Abdollah Homaifar, for his invaluable technical and professional guidance in the entirety of this work. He has been a constant source of insight and guidance on most of the ideas in this work. His longtime curiosity on whether a default hierarchy is realizable in a competing environment channeled my interest in Evolutionary Algorithms into a productive research. I also express my sincere acknowledgement to all my committee members (Dr. Albert Esterline, Dr. Gary Lebby, Dr. Jung Kim and Dr. Robert Li) and graduate school outside observer (Dr. Scott Harrison) for their valuable critics and suggestions on the research approach and editing of this dissertation.

I am so grateful to my families for their great help, love and encouragement. My roommates and friends in Greensboro: you made my life simple, enjoyable and memorable; thanks for the togetherness, comfort and support throughout. I am particularly indebted to Ms. Sara Asfaw for her immeasurable help, motivation and comfort in many ways. I cannot thank her enough. And last, but not least, I would like to extend my appreciation to friends in the lab, whole ECE staff members and several other people around campus without whose support this work would have remained a dream.

Finally, completing this research would have been unthinkable without the generous support of the National Science Foundation (NSF) under Cooperative Agreement No. DBI-0939454. I am so much indebted to the US government for funding this research throughout my PhD program. Any opinions, findings, and conclusions or recommendations expressed in this material are, however, those of mine and do not necessarily reflect the views of NSF.

Table of Contents

List of Figures

List of Tables

Abstract

Evolutionary algorithms work on the notion of "best fit will survive" criteria. This makes evolving a cooperative and diverse population in a competing environment via evolutionary algorithms a challenging task. Analogies to species interactions in natural ecological systems have been used to develop methods for maintaining diversity in a population. One such area that mimics species interactions in natural systems is the use of niching. Niching methods extend the application of EAs to areas that seeks to embrace multiple solutions to a given problem.

The conventional fitness sharing technique has limitations when the multimodal fitness landscape has unequal peaks. Higher peaks are strong population attractors. And this technique suffers from the 'curse of population size' in attempting to discover all optimum points. The use of high population size makes the technique computationally complex, especially when there is a big jump in fitness values of the peaks. This work introduces a novel bio-inspired niching technique, termed Fitness Proportionate Niching (FPN), based on the analogy of finite resource model where individuals share the resource of a niche in proportion to their actual fitness. FPN makes the search algorithm unbiased to the variation in fitness values of the peaks and hence mitigates the drawbacks of conventional fitness sharing. FPN extends the global search ability of Genetic Algorithms (GAs) for evolving hierarchical cooperation in genetics-based machine learning and dynamic clustering. To this end, this work introduces FPN based resource sharing which leads to the formation of a viable default hierarchy in classifiers for the first time. It results in the co-evolution of default and exception rules, which lead to a robust and concise model description. The work also explores the feasibility and success of FPN for dynamic clustering. Unlike most other clustering techniques, FPN based clustering does not require any a priori information on the distribution of the data.

CHAPTER 1

Introduction

All living things have some degree of inherent intelligence and strive to adapt to situations through ongoing learning. We learn heuristics throughout our life and use them to solve various problems in day-to-day life. But machines have no intuition and hence fail to understand commonsense knowledge. One of the most striking features of nature is the existence of living organisms in a wide range of ecosystems adapted for surviving in a continuously changing environment. The unguided natural evolution of living things in response to environmental variations has attracted the attention of many researchers pursuing the development of computer systems with analogous features. Learning from nature has been the key aspect of Evolutionary Algorithms (EAs). Adaptation of this natural intelligence to machines has contributed to the advancement of technology. The essence of EAs is to use ideas from natural evolution in order to find a global optimum solution to a certain problem. EAs are population-based robust metaheuristic optimization algorithms that use biology-inspired mechanisms like mutation, crossover, natural selection, and survival of the fittest in order to refine a set of solution candidates iteratively. Evolution in living things is unguided and its goal is usually unpredictable (Back, 1996; Back and Schwefel, 1993). EAs introduce a change in semantics of natural evolution being goal driven. Compared to other optimization techniques, EAs have an advantage of being a “black-box” (i.e. making only few assumptions about the underlying objective function) approach to modeling a problem. The objective function does not require a deep knowledge of the structure of the problem space. This enables EAs to perform consistently well in a variety of problem categories (Weise, 2008, Floudas and Pardalos, 1996). In addition, unlike many other optimization techniques which suffer from the problem of

premature convergence to a local optima, EAs have a mechanism to avoid getting trapped in a local optima through discovery of new candidate solutions in the search space outside of the locally optimal regions by using a mutation operator (Floudas and Pardalos, 1996; Pardalos and Romeijn, 2002). The mutation operator is a means that enables EAs to get out of local optima.

1.1 Motivation

This work has been part of a large project funded by the National Science Foundation BEACON Center, which aims on harnessing the power of biological evolution to engineer better solutions to real problems. With the increasing complexity of real-world optimization problems, demand for robust, fast, and accurate optimizers in a wide variety of multi-dimensional, multiobjective and multimodal optimization problems is on the rise among researchers in various fields. In this era of huge amounts of digital data, efficient and robust computational methods are of utmost importance for knowledge discovery and information retrieval, which involves techniques for clustering, classification and analysis of dynamic data.

In most real scenarios, the working environment is dynamic and known only partially. In addition, the solution space to some problems is usually huge and an exhaustive search for finding the best of the many possible candidates might not be feasible cost and time-wise. This makes designing an accurate mathematical model for optimal solutions a very daunting task, especially when the learning system needs to model an environment with huge number of states (Lanzi and Riolo, 2000, Riolo, 1987). Building a model that adapts to a dynamic environment through ongoing learning is a major contemporary challenge. Evolutionary algorithms are population based search heuristics that address this. Interspecies interactions through recombination and mutation allow the population to rapidly identify regions of the fitness landscape with high fitness and hence locate a satisfactory solution to a given problem without

being trapped in local optima (Lee et al., 1999). Like any other meta-heuristics search technique, EAs cannot guarantee the global optimum solution. Instead, EAs provide an engineering solution to a given problem. By maintaining useful diversity in a population, EAs avoid early convergence to have sufficient exploration of the search space and locate multiple optima at the same time (Horn and Goldberg, 1996; Shir and Back, 2006).

1.2 Problem Description

This research has primarily focused on addressing key challenges in two related areas of machine learning: multimodal function optimization and classification. The conventional fitness sharing technique enables a Genetic Algorithm (GA) to discover multiple optima simultaneously by maintaining a useful diversity in a population. It is based on an ecological analogy of a finite resource model, where individuals in a given niche share the resource of that niche. Peaks of the multimodal fitness landscape represent the resource of a particular niche. In a multimodal fitness landscape, the traditional fitness sharing scheme tends to distribute the population along the various peaks in proportion to the fitness of each niche. Higher peaks in the fitness landscape are strong population attractors and hence a significant proportion of the population rushes to converge to those points. On the other hand, lower peaks remain as weaker attractors until the resource of higher peaks gets depleted and other individuals in the population will no longer have an incentive to migrate to those niche locations. The problem with this kind of sharing is that as higher peaks in the fitness landscape attract the majority of the population, the search technique is unable to discover other peaks of lower fitness value. In other words, if there is a large gap between the peaks of the multimodal fitness landscape, the EA typically requires a large population size in order to discover all the optimum points effectively. This makes the success of the search technique highly dependent on population size. Intuitively, the larger the

population size, the slower is the search process as the EA requires more computation time for convergence. Multimodal functions can also be mapped to represent the density of data for clustering: dense areas map to higher peaks and sparse areas to the location of lower peaks (see Figure 1.1). In such a scenario, higher peaks may not necessarily be more relevant than lower peaks. In Figure 1.1 for instance, clusters C_1 and C_5 are equally important from the point of view of categorizing the data in to appropriate number of clusters. Hence, discovering both cluster centers is equally important for the search technique. Is it possible to consider all the optima equally attractive from the EA search point of view so that the difference in the fitness of the highest and lowest peaks does not affect the performance of the search technique? This, if possible, would avoid the population size dependency and hence making the search process faster.

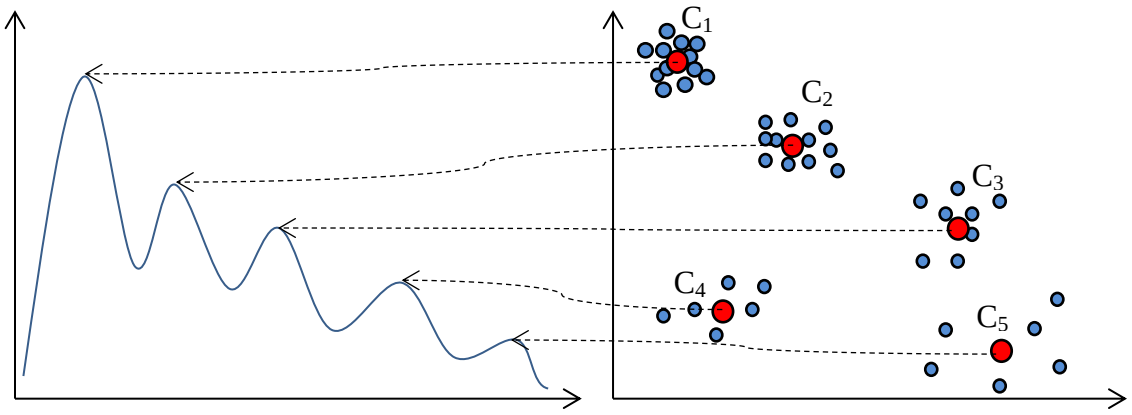

Figure 1.1. Mapping multimodal fitness landscape to clustering.

The second focus area of this research is classification, particularly classification using Michigan style LCSs. LCSs are rule-based learning models guided by a reinforcement signal. Rules are classifiers that compete for a resource to maximize their strength. The goal of the learning system is to build a set of rules that work in coordination to accurately model a given environment. This requires a mechanism to evolve and sustain a diverse, cooperative population

of rules that together represent a concept or model a set of behaviors that solve a given problem. The LCS in its very nature has a unique power of discovering cooperating rules through a GA by the guidance of reinforcement learning. However, it would tend to lose diversity due to a strong selection pressure of the GA. Hence, to maintain a cooperative diversity while applying a selection operator to the population of rules, it must incorporate a mechanism to counterbalance the effect of selection pressure. An LCS that is to work in an environment with huge numbers of states can be modeled in either of two ways. One possible way is using a population of rules that never make mistakes. This makes the learning system computationally complex and unacceptable to model realistic problems due to the curse of large population size. Besides, an environment exhibiting perpetual novelty combined with a limited sampling of it adds another order of complexity. The other alternative is to build a hierarchical model where the task of the learning system is to categorize the states into groups that can be treated in a similar way.

The question this research has aimed to address is: is it possible to evolve a hierarchical set of cooperating rules in which general and specific rules coexist in a competing environment while the specific rules provide protection to the general rules without starving them? If this is realizable, it helps to build a concise concept description. A crude analogy from real life would be a scenario where general practitioners and specialists work in cooperation. A general physician can diagnose a variety of health problems that do not need a specialty. There is no need for a specialist when the general physician is good enough. But, a general physician needs to be covered by a specialist for special cases. So, a well optimized system is one that embraces the co-existence of both in the system. Similarly in LCS, is it possible to protect the default (general classifier) from making mistakes by the exception classifiers without starving it when its action is right? The starvation-protection dilemma has been the bottleneck of the research in this

community for over three decades. Our research provides a solution to the challenges mentioned above.

1.3 Contribution

This work makes two significant contributions to the field of evolutionary algorithms. The first contribution is the development of a novel niching technique, termed Fitness Proportionate Niching (FPN), for multimodal function optimization. FPN is a resource sharing strategy where individuals in a given environmental niche share the resource of that niche in proportion to their actual fitness. The niching technique developed is based on a finite resource model. It is analogous to an ecological scenario where a different amount of resource supports the same number of different species (for instance, a ten meters cube of water may support three lions whereas a six meters cube of water may suffice to support the same number of dogs for the same duration). This ecological scenario is mapped into the developed algorithm to model resource sharing in a rugged fitness landscape. The distribution of the resource is modeled by the fitness landscape and its amount refers to the fitness value. The species (dogs and lions in this example) refer to subpopulations at the location of each niche. The number of niches corresponds to the number of peaks (optimum points) of the fitness landscape. The variation in amount of a resource (e.g. $10m^3$ and $6m^3$) corresponds to peaks of different fitness value in the fitness landscape.

The second major contribution of this work is the adaption of FPN for evolving hierarchical cooperation in classifiers. Classification is a supervised learning where the learning system, once sufficiently trained, seeks to categorize previously unseen instances in to correct classes or labels. An LCS accomplishes this task by evolving a population of classifiers using a reinforcement signal. Classifiers are rewarded every time an input is correctly classified. All

classes (labels) are considered equally important (i.e., a classifier which correctly classifies an input from one class and another which correctly classifies a different input from another class are equally rewarded by the trainer). FPN is used as a resource sharing mechanism to reinforce classifiers that match to a given input. This effectively turns the classification problem into an optimization problem of a multimodal function with equal peaks. Depending on the complexity of the working environment, adequate modeling of the environment might require a huge number of rules that collectively give a better model of the environment. Building a hierarchical set of rules, where accurate and more specific rules respond to a subset of the situations covered by more general but less accurate default rules is vital to achieving a compact rule set size, especially when dealing with an environment that has huge numbers of states. This requires the co-existence of exception and default rules in the system so that the exception rules can protect the default rule from making mistakes without starving it. To the best of our knowledge, the techniques proposed so far have failed to provide protection without a subsequent starvation of the default. This work has filled the research gap in evolving hierarchical cooperation in a diverse population of classifiers. A new formulation of steady state analysis (expressions governing the group strength variation and subpopulation dynamics) is also part of our contribution to the field.

Several sections of this dissertation are published in journals and peer reviewed conferences. The most relevant publications are listed below (see the appendix for detail).

Workineh, A. & Homaifar, A. (2012). Evolving hierarchical cooperation in classifiers via fitness proportionate niching. *Journal of Complex Systems* (in review).

Workineh, A. & Homaifar, A. (2011). Robust bidding in learning classifier systems using loan and bid history. *Journal of Complex Systems*, Vol. 19, Issue 3, pp. 287-303.

Workineh, A. & Homaifar, A.(2012). *Fitness proportionate reward sharing: a viable default hierarchy formation strategy in LCS*. The 2012 International Conference on Genetic and Evolutionary Methods, Las Vegas, July 16-19.

Workineh, A. & Homaifar, A. (2012). *Fitness proportionate niching: maintaining diversity in a rugged fitness landscape*. The 2012 International Conference on Genetic and Evolutionary Methods, Las Vegas, July 16-19.

Workineh, A., & Homaifar, A. (2012). *Fitness proportionate niching: A different perspective on co-evolution of diverse population*. ALife13, Michigan State University, July 19-22 (Extended Abstract).

Workineh, A., & Homaifar, A.(2012). *A new bidding strategy in LCS using a decentralized loaning and bid history*. IEEE Aerospace Conference, 1-8, Big Sky, MT., March 03-12.

1.4 Dissertation Scope

The term EA is very broad and refers to a collection of global optimization heuristics that are inspired by natural evolution in living organisms. The focus of this work, however, is limited to the two types of EAs: Genetic Algorithms (GAs) and Learning Classifiers Systems (LCSs). In particular, it focuses on maintaining diversity in a competing population and its applications in evolving hierarchical cooperation in classifiers and clustering. We developed a novel niching technique that alleviates the limitations of traditional fitness sharing. Its validity is tested and compared with existing sharing techniques using benchmark problems used in the literatures. This research is not directed at any specific type of application. Instead, its practical significance for solving real problems in broad areas of application (classification, clustering and multimodal function optimization) is demonstrated. For classification, the niching scheme is used as a resource allocation technique in LCSs to evolve a hierarchical set of cooperating rules. It is

successfully applied in single label classification tasks. Though the simulations are carried out for Boolean function learning using an LCS, the algorithm can be extended to any evolutionary algorithm that seeks to evolve a hierarchical set of cooperating rules for a concise concept description. Its role in dynamic clustering is also explored using both synthetic and real data. The intent here, however, is not to compare its performance with or claim an improvement over a specific clustering algorithm. Most clustering algorithms rely on a priori knowledge (for instance number of clusters, the distribution of the data etc) on the data. We have demonstrated that the developed niching technique can be applied in clustering when such a priori information is not available in advance. This dissertation has also shown the applicability of the technique for multimodal function optimization.

1.5 Dissertation Outline

The remainder of the dissertation is organized as follows. Chapter 2 provides an overview of related research work with an emphasis on multimodal function optimization, niching GAs and LCSs. Research on various niching methods and the challenge of niche radius estimation are discussed in this chapter. A review of the state of the art on the challenges in the formation of default hierarchies in LCSs is also given in this chapter. The main contributions of this work are presented in chapters 3 and 4. Chapter 3 discusses the novel FPN technique. The mathematical formulation, performance measure and its ecological analogy are detailed. A complexity analysis by comparing it with the traditional fitness sharing technique is also provided. The methodology for evolving hierarchical cooperation in classifiers is discussed in chapter 4. A new formulation of the classifier format is presented and an FPN based reinforcement scheme is used to allocate a reward among competing classifiers in the advocate set. The learning cycle and steady state analyses for the proposed learner are also given in this chapter. In Chapter 5, we extended the

application of FPN for dynamic clustering. A technique based on the principle of ecological niche expansion is applied for niche radius estimation. Then, a mapping between the number of peaks in a multimodal function and the number of clusters in the data is made and the fitness function is formulated. Simulation is done using both synthetic and real data. Chapter 6 discusses the simulation results. This chapter has two parts. The first part presents the simulation results for FPN. Comparison with existing niching techniques using well-known multimodal test functions presented in the literatures is given. The second part discusses the results for the evolution of hierarchical cooperation in classifiers. Comparison on whether other techniques are able to attain a default hierarchy; and are able to sustain it under the selection pressure of the GA is also investigated using control runs for the same simulation setup. Chapter 7 concludes the dissertation by highlighting the contributions of the work and pinpointing directions for future research.

CHAPTER 2

Literature Review

With the increasing complexity of real-world optimization problems, demand for robust, fast, and accurate optimizers in a wide variety of multi-dimensional, multi-objective and multimodal optimization problems is on the rise among researchers in various fields. In this era of huge amounts of digital data, efficient and robust computational methods are of utmost importance for knowledge discovery and information retrieval. Evolutionary algorithms (EAs) are a form of stochastic search that utilize selection and inheritance to discover near-optimal solutions to arbitrary problems. Evolution in artificial systems follows the same basic principles as those of natural populations. Each individual possesses a coded solution to a given problem; the genotype. The genotype is decoded into a phenotype, which is a description of an individual's response to a given problem.

In this chapter, a detailed review of the current state of the art on niching methods for EAs is provided. We will review related research work and highlight some of the bottlenecks in this area that previous research has not addressed. The chapter is organized into four sections. The first section will explore the research in multimodal function optimization. The use of niching for maintaining diversity in a population for discovering multiple peaks in a multi-modal fitness landscape will be detailed in the second section. A brief summary of the various niching techniques is given in this section. The third section summarizes previous research work on estimating the niche radius. The idea of a finite resource model where individuals with in a given niche radius share the resource of that niche is analogous to implicit sharing in classifier systems. The fourth section discusses the evolution of hierarchical cooperation in classifiers. Applications of the niching technique for clustering and the related issues are discussed in the last section.

2.1 Global Optimization Issues

Unlike gradient based search techniques, which can be trapped in local optima, EAs are population-based global optimizers with a mechanism (using a mutation operator) to escape local optima. However, there is no guarantee of discovering the global optima every time. The chance of discovering the global optima depends on several factors, including the initial population setup, the type of fitness landscape (multimodality, ruggedness, deceptiveness etc) and the degree of exploitation and exploration. In population based search algorithms (Genetic algorithms (GAs), tabu search, genetic programming (GP), and particle swarm optimization (PSO)), premature convergence has been a common issue. It arises due to loss of diversity in a population which is related to maintaining a good balance between exploitation and exploration. Losing diversity means approaching a state where all the solution candidates look similar. Exploration is discovery of new solutions from the search space. These solutions introduce diversity to the solution list enabling the search algorithm to get out of trap in local minima and hence, allowing the algorithm to discover better solutions. The mutation operator in GAs helps to discover new solutions by providing a mechanism to get out of local optima. On the other hand, exploitation improves currently known solutions by combining individuals. The cross over operator speeds up convergence by refining previously discovered solutions in the population. A balance between exploration and exploitation is very essential in population based search algorithms. Too much exploration (i.e. high mutation rate) destroys good solutions and would turn the search technique to a blind search. This delays convergence to optimum points in the search space. Also, too much exploitation (i.e. high crossover rate) leads to premature convergence and ignoring possibly better solutions located at distant areas of the problem space which have not been explored. Higher exploitation rate means higher convergence rate and

accepting the risk of not finding the optimal solution (as the search may get stuck at a local optimum).

EAs are applicable to a wide range of problems with varied fitness landscapes. Simple EAs are effective in finding a single optimum of a unimodal fitness landscape (Jong, 1975). For instance in Figure 2.1, the function has one global optimum and the path to the optimum point is smooth. A more difficult problem arises when the function to be optimized is multimodal (see Figure 2.2) and has many local optima.

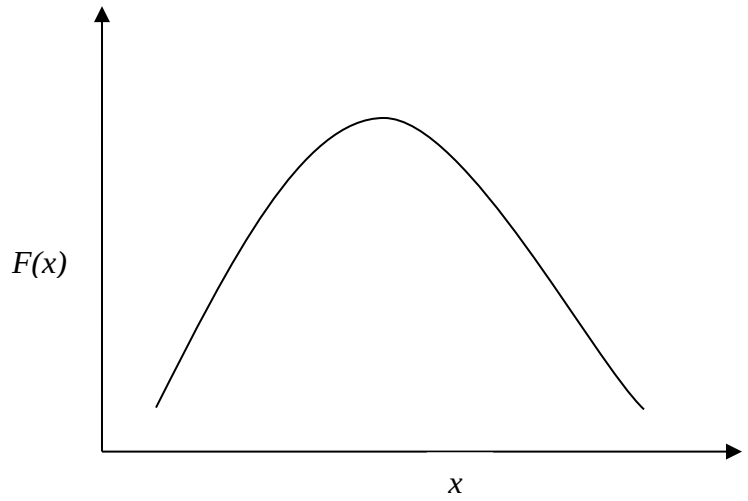

Figure 2.1. A unimodal objective function with a smooth path to the global optimum.

The optima in a multimodal fitness landscape may have different peaks unevenly distributed throughout the search space. EAs have an intrinsic drawback when dealing with such multimodal fitness landscapes to locate all optima simultaneously. This is due in part to the genetic drift (Asoh and Muhlenbein, 1994; Cioppa et al., 2007) that results from the selection pressure of a GA, operator disruption and selection noise and that drives the population to converge to the highest fitness. The drawback is also due to the evaluation mechanism, which computes the fitness of each individual in the population independent of the fitness of other individuals (Cioppa et al., 2004, 2007). So, to deal with multimodal fitness landscapes, EAs need

to have a mechanism to maintain a diverse population by counterbalancing the effect of genetic drift.

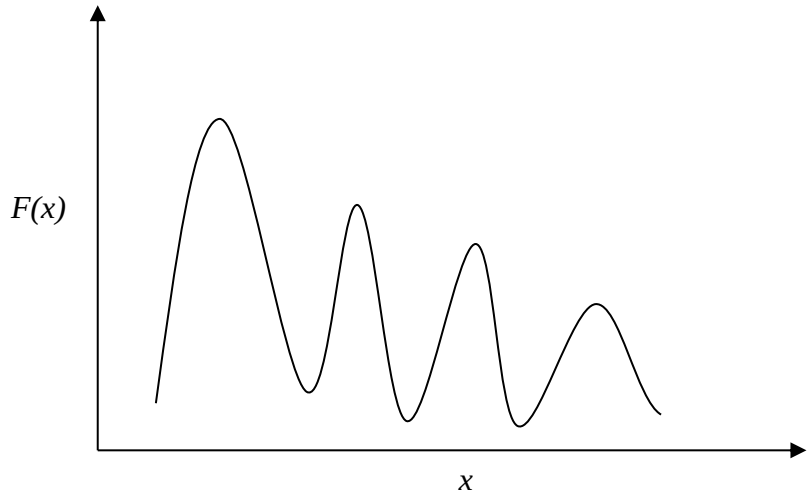

Figure 2.2. A multi-modal objective function with 4 unequal peaks.

There is no general solution to premature convergence but possible solutions have been suggested in the literatures. The most popular include, restarting the process with a randomly initialized state, increasing the exploration rate (higher mutation rate for instance) and extending the duration of evolution by steering the search away from the already sampled or frequently visited areas using techniques like niching and implicit fitness sharing. The principle behind most of the existing diversity maintaining mechanisms is based on an analogy with natural ecosystems, which encourages the formation of species or niches, each representing candidate solutions (Forrest et al., 1993; Goldberg and Richardson, 1987; R. Smith et al., 1993).

2.2 Niching Methods for Evolutionary Algorithms

The motivation behind niching was to promote diversity in a population. Traditional GAs evolve the whole population towards convergence, i.e. individuals in the population soon become nearly identical. In other words, they suffer from early convergence and in the case of multimodal functions; GAs can only find one of the solutions. Hence effective implementation of

niching is very crucial to the success of GAs in multimodal function and multi-objective optimization, machine learning and classification problems. Maintaining diversity in a population has two major advantages. First, it helps to avoid premature convergence and hence allowing sufficient exploration of the search space. The most important advantage of keeping diversity in a population is to discover multiple optima in multimodal function optimization. So far there are two major approaches to niching: preventing premature convergence by altering the selection operators in the GA and the multi-population strategy. Fitness sharing (Goldberg, 1989; Goldberg and Richardson, 1987), crowding (Jong, 1975), deterministic crowding (Mahfoud, 1995) and clearing (Petrowski, 1996) techniques fall under the first category. The second approach includes multi-population GAs, island population models and forking where the algorithms subdivide the population into subpopulations and optimize locally. The need for maintaining useful diversity in a population to reduce the effect of genetic drift in the standard GA has been emphasized by several researchers in previous work (Deb and Goldberg, 1989; Goldberg et al., 1992; Horn et al., 1994; Mahfoud, 1995; Sareni and Krahenbuhl, 1998). To date, various niching strategies have been proposed in the literatures and this section presents a brief survey of the state of the art in the three most notable niching techniques: crowding, fitness sharing and clearing.

2.1.1 Fitness sharing. Fitness sharing is the most well-known method for creating stable subpopulations of individuals around the multiple local or global optimum points in the search space (Goldberg et al., 1992; Goldberg and Richardson, 1987). The inspiration for adapting the sharing technique to traditional GAs emanates from natural ecosystems where individuals of the same species share a finite natural resource in an environment. The hierarchical organization of species in a competing world of limited resources is shaped by the location and distribution of

these resources. Traditional GAs assume an infinite resource model where there is no need for competition for resources and all individuals can comfortably coexist on the same peak and receive the same fitness that they would have if they were the only individual on that peak. Hence, in the case of multimodal functions of unequal peaks, all individuals tend to seek the highest peak and converge to that point. Also, in multimodal functions of equal peaks, the population will converge to one of the peak locations arbitrarily. The feasibility of resource sharing in evolutionary algorithms was first pointed out by Holland (J. Holland, 1975). But the first implementation of fitness sharing to model a resource contention within a simple GA was given by Goldberg and Richardson (Goldberg and Richardson, 1987). It is based on the idea that a point in a search space has limited resources which must be shared by all individuals that occupy similar search space (Davidor, 1991). As more and more individuals get attracted to the highest peak, the resource at that peak gets depleted and other lower peaks in the search space begin to attract individuals.

Sharing in an Evolutionary Algorithm (EA) is implemented by scaling the fitness of an individual based on the number of "similar" individuals present in the population (Cioppa et al., 2007; Goldberg and Richardson, 1987). It lowers each individual's fitness by an amount nearly equal to the number of similar individuals in the population. The raw fitness of the individual is reduced by the number of similar solutions in the population belonging to the same niche (Goldberg et al., 1992; Goldberg and Richardson, 1987). Scaling an individual's fitness is controlled by two operations, a similarity function, which measures the distance between two individuals in either the genotypic or phenotypic space, and a sharing function (Deb and Goldberg, 1989; Goldberg et al., 1992; Horn et al., 1994). The sharing function is shown in equation (2.1).

$$sh(d_{i,j}) = \begin{cases} 1-(\frac{d_{i,j}}{\sigma_{sh}})^{\alpha}, & \text{if } d_{i,j}<\sigma_{sh} \\ 0, & \text{otherwise} \end{cases} \quad (2.1)$$

Here $d_{i,j}$ is the distance between individuals *i* and *j*, σ_{sh} is the niche radius and the constant α is usually set to 1 for a triangular sharing function. And the niche count, n_i is calculated by summing a sharing function over all individuals of the population as given in equation (2.2), where *m* is the number of individuals occupying the same niche.

$$n_i = \sum_{j=1}^{m} sh(d_{i,j}) \quad (2.2)$$

Now, the shared fitness of an individual *i* is calculated using equation (2.3).

$$F_{sh,i} = \frac{F_i}{n_i} \quad (2.3)$$

Where, F_i is the raw fitness of the individual.

As can be seen from the equation above, the degree to which two individuals are considered to belong to the same niche is controlled by the sharing radius. And, the performance of the fitness sharing relies strongly on the proper choice of the niching radius. This is one of the limitations of the fitness sharing technique. In general choosing the optimum niche radius requires a priori knowledge of the distribution of the peaks in the objective function (Dick, 2010; Dick and Whigham, 2006).

2.1.2 Crowding. The standard crowding method was first introduced by De Jong to promote useful diversity in the population to prevent premature convergence of the GA (Jong, 1975). In this method, a fraction of the total population called the generation gap is allowed to reproduce at each generation. The crowding factor (CF) determines the number of individuals selected from the population for comparing the similarity of the new offspring. Similarity of individuals can be determined by means of a distance measure, either genotypic or phenotypic

distance between individuals. The new offspring then replaces the most similar individual taken from this randomly drawn subpopulation of size CF. Later, Mahfoud introduced a modified crowding technique termed “deterministic crowding” (Mahfoud, 1994a, 1995) to improve the standard crowding by introducing competition between children and parents of identical niches. In a deterministic crowding, the new offsprings replace the nearest (measured in phenotypic distance) parent provided it has a higher fitness. Deterministic Crowding is simple, fast and requires no additional parameters. Its weakness is that as species of higher fitness value always tend to win over species of lower fitness, it fails to provide sufficient restorative force to maintain diversity. Mengshoel and Goldberg introduced a probabilistic crowding to mitigate this shortcoming (Mengshoel and Goldberg, 1999). In this scheme, stronger individuals do not always win over weaker ones; rather they win proportionally to their fitness.

2.1.3 Clearing. The clearing type niching is essentially similar in principle to the explicit fitness sharing technique. But, instead of uniformly distributing the resource to the entire subpopulation in a given niche, it allocates the whole resource only to the best members of the subpopulation. It is based on a winner-takes-all strategy where it preserves the fitness of the best individuals of each niche and resets the fitness of the others with in the niche radius (Petrowski, 1996). This convergence to only one of the alternatives is undesirable in multimodal optimization of real problems, because we are interested in getting information about good points and better solutions.

2.3 The Niche Radius Problem

In a multimodal fitness landscape, the goal of a niching technique is to evolve the population in to stable subpopulations. The number of niches should map to the number of optimum points of the fitness landscape. The fitness sharing method has been the most popular

niching techniques proposed in the literatures. However, the success of the algorithm entirely depends on the appropriate choice of the niche radius. Using too small niche radius results in discovering too many fictitious niches while a too large radius cannot discriminates between neighboring niches which turns the niching technique to that of a simple GA. If a priori information about the fitness landscape (such as the number of peaks and the distance between them) is known, then estimating the niche radius would not be a problem. But in most of the real world problems, a priori information about the fitness landscape is not available. This puts a serious limitation on the practicability of the radius dependent sharing techniques (fitness sharing and clearing schemes). This limitation has inspired previous research to go in two main directions in the past. One area of research has primarily focused on discovering alternative ways of niching techniques. Spatially structured GAs (Dick, 2010; Dick and Whigham, 2008), clustering based niching techniques (Ando et al., 2005; Streichert et al., 2004; Zhang et al., 2006), crowding GA (Jong, 1975), deterministic crowding (Mahfoud, 1995), a species conserving genetic algorithm (SCGA) (Li et al., 2002) and many others fall under this line of research. These methods do not consider any sharing scheme. In deterministic crowding, diversity is introduced to the population through a guided replacement (i.e. parents are replaced by offsprings only when the offsprings have a better fitness) (Mahfoud, 1994a, 1994b). SCGA maintains the fittest individual for each species until a fitter individual for that species is discovered in a later iteration (Li et al., 2002).

The second research direction has entirely focused on how to estimate the niche radius dynamically. Deb and Goldberg, proposed a technique for estimating the niche radius given the heights of the peaks and that their distances are known a priori (Deb and Goldberg, 1989). This approach is very limited as in most real applications there is very little prior knowledge about the

fitness landscape. In implicit fitness sharing (R. Smith et al., 1993), similar individuals in a population compete for limited and explicit resources with no limitation on the distance between peaks. Though there is no need of estimating the niche radius, the algorithm's performance depends on appropriate choice of other parameters such as the size of the sample of individuals that compete, the number of competition cycles and the definition of a matching procedure.

Miller and Shaw developed a dynamic niche sharing technique that is able to efficiently identify and search multiple niches in a multimodal domain (Miller and Shaw, 1996). The algorithm attempted to identify the multiple peaks dynamically to classify all individuals as either belonging to one of these dynamic niches (if individuals are within the niche radius of a dynamic peak) or else belonging to the "non-peak" category. The authors claimed a better performance as compared to the standard sharing and deterministic crowding techniques. The algorithm still shares the drawback of the standard sharing method as it made assumptions on the number of niche peaks and the distance between them, which in most real scenarios are not known a priori.

In (Shir and Back, 2006), the authors introduced the concept of adaptive individual niche radius to address the niche radius problem . The idea brings a new representation where the niche radius is encoded as part of the chromosome structure of each individual and adapts along with other parameters during the course of evolution (Shir and Back, 2005; Shir et al., 2007).

A dynamic niche identification technique based on the characterization of the dynamic behavior of the evolutionary algorithm in terms of the mean and standard deviation of the number of niches discovered during the evolution was proposed in (Cioppa et al., 2004, 2007). By varying population size and the niche radius iteratively and observing the pattern of the variation of the mean and standard deviation at each generation, they gave an estimate on the

optimal values of the population size and the niche radius without any a priori information on the fitness landscape. The dynamic niche identification technique proposed does not make any assumption on the priori knowledge of the fitness landscape but it is still based on knowing the number of peaks. The applicability of this algorithm is limited especially when the search space is too big and the fitness landscape is too rugged, as iteratively varying the niche radius would involve too much computation. In addition, the algorithm is very subjective and it is difficult to accurately estimate the optimum value of the niche radius by simply looking in to the variation of the mean and the standard deviation.

The authors in (Chang et al., 2010) introduced a dynamic identification of the niches based on the idea of niche migration to automatically evolve the optimal number of niches. The algorithm requires only an initial guess of the niche radius, possibly set to the minimum value that could at least result in the prevalence of two niches. The idea of niche migration is based on the analogy of population dynamics that when a given city is crowded people in that city start migrating to a nearby city (Chang et al., 2010; Chang et al., 2011). The technique was successfully applied to data clustering with no assumption on the number of clusters and the distribution of the data (Chang et al., 2011).

The Adaptive Isolation Model (AIM) introduced in (Streichert et al., 2004) used a clustering algorithm to identify different regions of attractors and then the subpopulation that makes up the clusters are isolated and optimized independently. It was claimed that the algorithm is both efficient (since crossovers will only include parents from the same attractors reducing the number of offspring sampled outside of the attractors) and comprehensive (the chance of discovering a suboptima in weaker attractors increases by isolating strong attractors). This algorithm is based on the assumption that for a multi-modal fitness landscape the GA distributes

the population among the different attractors. This however requires some form of diversity preserving mechanism (e.g. crowding or some other form of inducing diversity).

There is quite a limited research on dynamically estimating the niche radius. The earlier work was by Deb and Goldberg (Deb and Goldberg, 1989). The authors made a comparison between crowding and fitness sharing methods and an estimate on the niche radius was also given. The estimation on the niche radius was based on the assumption that number of peaks in the fitness landscape is known a priori. Dick also proposed a local clearing technique for automatic adaptation of the niche radius for spatially structured population (Dick, 2010; Dick and Whigham, 2006, 2008). In our implementation, we used a modified version of the dynamic niche identification techniques introduced by the authors in (Chang et al., 2010) to estimate the niche radius.

2.4 Evolving Hierarchical Cooperation in Classifiers

2.4.1 LCS overview. A Learning Classifier System (LCS) is a machine learning paradigm where an agent learns to perform a certain task by interacting with a partially known environment via guidance of a reward signal that indicates the quality of its action (J. Holland, 1980; R. Smith and Goldberg, 1990). Classifiers are rules in the form of "if condition then action" format. In an LCS, the solution domain initially contains a large population of candidate classifiers. The learning process begins with this random population and needs to evolve it to optimal solutions through training. The goal of the classifier in the learning process is to accumulate as much reward as possible. The reinforcement learning guides the search for solution by rewarding classifiers that propose a correct action.

Holland's formulation of LCS involves a bucket brigade algorithm that takes care of the credit assignment task in LCSs (J. Holland, 1985; J. Holland and Holyoak, 1988). Each classifier

is assigned a strength that is adjusted by the bucket brigade algorithm in such a way that it reflects the classifier's overall usefulness to the system (Booker et al., 1990; J. Holland and Reitman, 1977). In the bucket brigade, classifiers that match to the current input bid for posting their message in the message list (J. Holland, 1985). The message list allows classifiers to communicate directly with each other through paying and charging of a reward (J. Holland, 1992; J. H. Holland, 1995). The learning objective in LCSs can be very general, for instance to survive. Unlike most reinforcement learning algorithms, LCSs do not make any particular assumption on the environment and on the agent's goal which is generally defined in terms of adaptation to the environment (Kovacs, 2004).

There are two major types of LCS: Michigan and Pittsburgh style LCSs. The classification is based on their advocator's affiliation (University of Michigan and University of Pittsburgh respectively). In the Pittsburgh formulation of LCS, individuals in a population are complete solutions to the problem (Bacardit 2004; S. Smith, 1980). An individual is a rule set and the length of a rule is fixed while the number of rules in one rule set varies. Individuals in the population compete among themselves to correctly classify the training samples. The working principle of Pittsburgh LCSs is essentially similar to GA. The fitness of an individual in the population is measured by the classification accuracy of the rule set (Bacardit et al., 2007). But in Michigan style LCS, individuals are rules and the solution to the problem is the whole population. An individual rule covers part of the solution and coordination among rules and a mechanism to evaluate the performance of rules in the form of reward or punishment is essential. The use of the term LCS in this work adheres to Michigan style LCS. The standard LCS consists of three major components: the message and rule system, the apportionment of credit and the discovery component (Booker et al., 1990; J. Holland, 1980, 1986).

The apportionment of credit subsystem in an LCS addresses the issue of credit assignment which serves as a measure of the classifier's performance. It is based on an economic analogy where a classifier garners credit in the form of strength (a kind of capital). It involves a bid competition among classifiers that match to the current environmental input. Accordingly, matched classifiers bid a certain proportion of their strength and rule conflicts are resolved based on a probability distribution over the bids (R. Smith and Goldberg, 1990). A winner classifier has to pay out all its debt through the clearing house hence risking a certain percentage of its strength with the possibility of getting a reward. Also, to promote the exploration of the classifier space, a random noise is added to the deterministic bid (Homaifar et al., 1988). In this system, classifiers communicate with each other through the message list in addition to directly interacting with the environment. Many of the variants of classier systems are alternative schemes for apportioning or accumulating credit.

An LCS uses a GA as a search engine to discover new rules from a population of candidate rules based on the experience of existing rules. GAs are a class of computerized search procedures that are based on the mechanics of natural genetics (Goldberg, 1989). However, the GA used in an LCS is different from a standalone GA. Consider for instance the problem of "function optimization". In a standalone GA, the intention is to find parameter settings which correspond to the extremum of the fitness function. There is no notion of generalizing across states and no need for a measure of the accuracy with which this is done. Also, there is no concept of environmental state (i.e. the GA structure lacks the condition part of the classifier). It only manipulates a set of parameters corresponding to the action part of a classifier. So a standalone GA is a function optimizer that seeks for points of maximum functional value in the search space, whereas the GA in an LCS serves as a function approximator (Kovacs, 2004).

Since its first conceptual inception by John Holland in 1976 (J. Holland, 1976), the LCS idea has stimulated a number of investigations of its merit for some real-world applications such as gas pipeline control (the first application) (Goldberg, 1983), Boolean function learning (Wilson, 1985), sequence prediction (Robertson and Riolo, 1988), letter recognition (Frey and Slate, 1991) and job-shop scheduling (Hilliard et al., 1987). The basic structure of an LCS is depicted in Figure 2.3.

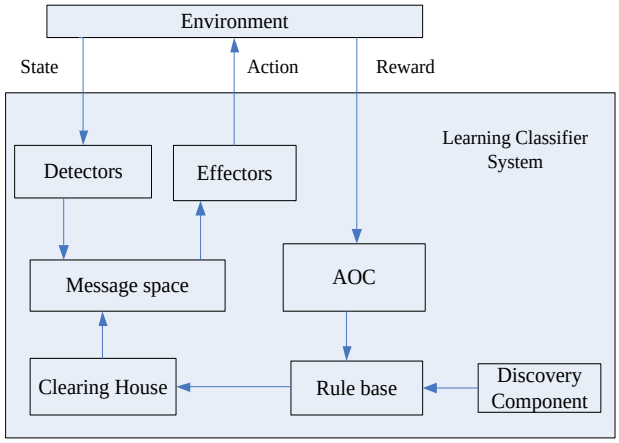

Figure 2.3. The structure of a learning classifier system.

The LCS detects its environment through its detectors and takes appropriate action with the help of its effectors. During the learning phase, a feedback signal is provided by the environment in the form of a reward or punishment to guide the learning system. In application mode, no external feedback is provided and the LCS applies its knowledge to predict the environment in the form of an action when triggered by a given input.

To date, several modifications have been made to the traditional LCS. Wilson introduced a strength based learning classifier system known as the zeroth-level classifier system (ZCS) in 1994 (Wilson, 1994). A year later, he introduced accuracy based classifier system (XCS) which brought a major change in an LCS's rule fitness calculation (Wilson, 1995). The fitness is made to represent the accuracy of the prediction instead of the prediction itself. Hybrid techniques,

such as fuzzy-xcs are also proposed for single step reinforcement problems (Casillas et al., 2007). XCS maintains in the population both classifiers that consistently receive high reward and those that consistently receive low reward. As a result, it usually requires a higher population size as compared to strength-based fitness, which keeps only classifiers receiving high reward in the system (Butz et al., 2004).

In strength based LCSs, the past performance of a classifier is measured by the amount of its current strength. Strength is used as a means of resolving conflicts and as a fitness for the genetic algorithm. In our previous work, we introduced a modified bidding strategy in LCSs by allowing classifiers to get a loan from a loaning agent termed a 'bank' during auctions (Workineh and Homaifar, 2011). The loaning approach followed was centralized in the sense that there is only one central bank issuing the loan. A bid history variable that gives classifiers a clue about the potential of competent classifiers was also introduced. We also implemented a more compact, less complex and more realistic distributed loaning strategy that allows a loan exchange among classifiers to play a dual role (loaners and borrowers) among classifiers in the systems (Workineh and Homaifar, 2012c). The generalization capability of an LCS by means of using hash symbols in its condition string gives it the potential to develop a compact representation of the concepts learned.

2.4.2. Default hierarchy. Smith and Goldberg (R. Smith and Goldberg, 1990) introduced a modified bidding strategy to allow the formation of default hierarchies. The bid amount is made proportional to the specificity of the classifier's condition to allow more specific classifiers to fire instead of more general inaccurate classifiers. It was also indicated that incorporating a specificity factor in to the bid computation does not enable the LCS to distinguish between default and exception rules at steady state. To alleviate this shortcoming taxation was introduced.

Tax in an LCS has a dual purpose: to eliminate classifiers that serve no purpose but whose strengths are not low enough to qualify them for deletion by the GA and to provide a separation between default and specific classifiers at steady state. Besides, the authors suggested a necessity auction to improve the LCS's simulation of auction dynamics and to induce bid separation that responds to the entire system's performance (R. Smith and Goldberg, 1990; R. Smith and Goldberg, 1992).

2.4.2.1 The starvation–protection dilemma. The starvation-protection dilemma has been a bottleneck to research in attaining a working default hierarchy. The intention here is to protect the default rule from firing when it is wrong without starving it. This requires a bidding strategy that favors the exception when both match a given input. In (J. Holland et al., 2000; Riolo, 1987; R. Smith and Goldberg, 1990), a bid amount proportional to the specificity is proposed. In this kind of bidding strategy, exception classifiers bid a higher amount as compared to more general classifiers in the system. The shortcoming of this type of protection (in a standard LCS) is the consequent starvation of the default classifier when it is right. Protection is always associated with an immediate starvation of the default.

In the standard formulation of LCS, starvation can happen in two ways. One is when a more specific classifier having the same action as the default out bids the default and hence prevents it from entering the active set and possibly getting a payoff (J. Holland, 1980; R. Smith and Goldberg, 1990). The active set is a subset of the match set that contains the highest bidding classifiers. The specific classifier tends to flourish in the population as a result of getting a payoff from its environment. The other scenario is when an exception classifier with a wrong action bids with the default classifier of the correct action. The exception classifier may prevent the default from entering the active set again. In effect, the default has to wait until the exception

classifiers with wrong action die out of the system. But, as the default always belongs to the match set, it incurs an overhead tax at every computation time. Both scenarios prevent the default from firing and possibly receiving a boost in its strength in the form of a reward from the environment. Thus, starvation augmented with continuous taxation continuously weakens the default classifier until it is eventually eliminated by the GA. In such circumstances, a hierarchy, even if emerges some time along the learning process, is unlikely to survive the selection pressure of the GA.

2.4.2.2 Extending decision making to the match set. The objective is to allow the default classifier to take over the decision making every time it is right. To avoid the starvation problem, Wilson, in a detailed experiment using Boole (Wilson, 1989), proposed key modifications to the standard LCS formulation. One remedy to avoid starvation is extending system decision to the match set instead of limiting them to the active set. In Holland's formulation of LCS, only classifiers in the active set make decision on the system and any reward by the external environment goes to the active set. Wilson's experiments proved that extending the decision making and resource sharing scheme to a rather bigger set of classifiers (the match set) improves the system's performance. The default is always part of the match set and an external reward is distributed among all classifiers in the match set proposing the same action as the winner classifier. This effectively avoids starvation of the default due to outbidding of other specific classifiers in the system to qualify for the active set. As far as its action agrees with the winner's action, it always gets a fraction of the reward coming from the external environment. But, under the same bidding and paying policy as proposed in (R. Smith and Goldberg, 1990; R. Smith and Goldberg, 1992) and (Riolo, 1987), omitting the active set may lead to rampant overgeneralization. To overcome this problem, Wilson suggested a different bidding and paying

policy in which specificity is retained in the bid calculation but eliminated when calculating the classifier's payout (J. Holland et al., 2000; Wilson, 1989). The bid amount is scaled by the specificity while the actual pay out of the classifier depends entirely on its strength and the bid coefficient.

2.4.2.3 The need for niching. In Michigan type LCSs, the GA searches through the space of all possible rules to find and maintain a diverse, cooperative subpopulation. On one hand, an LCS requires the strong selection pressure of the GA to discover new rules. On the other hand, without some form of niching, the LCS cannot maintain a diverse set of cooperative rules in the population due to the selection pressure of GA. Niching provides a restorative force to balance the selection pressure that causes early convergence by maintaining useful diversity in the population (Horn and Goldberg, 1996). Maintaining a diverse set of cooperative rules is particularly important for the formation of default hierarchies and for temporal rule chains.

Many researchers have pointed out the importance of implicit (reward sharing) and explicit (fitness sharing) niching for the evolution of cooperative classifiers. In previous work, Booker (Booker, 1982, 1989) implemented niching using an indirect form of sharing and introduced mating restrictions to limit mixing between niches. Wilson applied implicit niching in LCSs by uniformly distributing a reward among classifiers that agreed with the system decision for learning a Boolean concept. Smith and Valenzuela-Rendon later proposed explicit niching in LCSs by applying fitness sharing using a hamming distance as a metric to find the niche radius on the space of rules (R. Smith and Valenzuela-Rendón, 1989). Mahfoud also applied explicit niching for GA-based classification of Boolean concepts which is analogous to a stimulus response LCSs (Mahfoud, 1995). The model of a stimulus-response LCS is similar to the natural immune system. The analogy between implicit niching in the immune system and resource

sharing in an LCS is also analyzed by the authors in (R. Smith et al., 1993). Horn and Goldberg also stressed the importance of niching and gave a model to approximate the niche existence and extinction times (Horn et al., 1994). Assuming a two niche scenario and all classifiers have the same specificity, the authors demonstrated that the dynamics of an interacting and coevolving population can be analyzed, predicted and controlled. The environment provides a reinforcement signal to the learning system when it responds correctly. In a stimulus-response LCS, the system takes action on its environment directly and receives an immediate reward or punishment as a consequence of this action (Booker, 1982; Booker et al., 1990).

There are two major notions on how to distribute the reward among classifiers in the active set. Holland first suggested that all classifiers in the active set should receive a constant reward R and pay out their bid (J. Holland, 1980; J. Holland and Reitman, 1977). This notion of sharing however does not lead to the formation of a default hierarchy as it does not distinguish between correct and wrong classifiers. The limitation of this kind of reward sharing on the formation of default hierarchy can be explained as follows. Consider for instance a scenario where a default classifier of action zero exists in the system. Assume also that its strength is high enough to outbid other specific classifiers in the match set and join the active set. This classifier may or may not agree with the winning classifier's decision but is going to receive a reward from the environment either way. This kind of indiscriminate rewarding leads to the emergence of sneaky classifiers that survive on the back of other reward generating classifiers and take over the system.

To overcome this shortcoming, Wilson suggested that reward should be shared only among classifiers that agree with the system's decision. Wilson's sharing scheme assumes a fixed amount of external reward R to be divided evenly among all classifiers in the advocate list

(Wilson, 1989). The advocate list consists of classifiers in the match set that propose the same action as the winner classifier.

The theory of equal reward sharing ignored the whole notion of competition in an LCS. The intention of a classifier is to build up its strength by garnering as much reward as possible from its environment. A good analogy here would be a competitive market economy where whoever strived hard should be rewarded and accumulate wealth. Wealth in classifiers is measured in terms of strength. Hence instead of using equal resource sharing, the rewarding scheme should somehow be biased towards stronger classifiers. Strength is a measure of the quality of a classifier and the rewarding scheme has to adjust the strength to reflect the classifier's overall usefulness to the system. We applied a fitness proportionate resource sharing scheme where classifiers proposing the same action as the winner will get credit in proportion to their strength. The proposed rewarding scheme resulted in a remarkable improvement in the performance of the system and produced a viable and robust default hierarchy.

CHAPTER 3

Fitness Proportionate Niching (FPN)

Traditional Genetic Algorithms (GAs) fail to maintain useful diversity in the population as a result of a genetic drift due to selection pressure, selection noise and operator disruption. Genetic drift leads to early convergence making simple GAs suitable only for optimizing unimodal functions. However, most real world optimization problems often deal with multi-modal functions and hence require a technique to discover the location of multiple optima in the search space. The conventional fitness sharing scheme based on the niche count has a limitation when there is a high gap between the peaks of the multimodal function. It requires a high population size in order to discover all the peaks simultaneously. The use of high population size makes it computationally complex, especially when there is a big jump in fitness values of the peaks.

FPN provides a solution to this limitation by uniformly distributing the population along the various peaks. Like the traditional fitness sharing based on niche counts, this technique is also based on the notion of limited resources where individuals in a given niche share the resource of that niche. But, here individuals share the resource in proportion to their actual fitness. Unlike the conventional sharing scheme, the difference in the fitness values of the highest and lowest peaks does not affect the performance of the proposed niching scheme. This chapter presents a comparison of the two techniques using both mathematical analysis and simulations on well-known multi-modal test functions. A technique for estimating the niche radius, complexity analysis and ecological analogy of the proposed niching technique are also provided here. The last section presents a chi-square like deviation comparison for some of the test bench functions in the literatures.

3.1 Test Functions

For testing the performance of the algorithm, multimodal functions of different difficulty level (see Figures 3-1 to 3-5) are used from literatures during simulations (Cioppa et al., 2004, 2007; Deb and Goldberg, 1989; Sareni and Krahenbuhl, 1998). The mathematical expressions for these functions are given in equations (3.1) to (3.5).

$$F_a(x)=\sin^6(5\pi x) \tag{3.1}$$

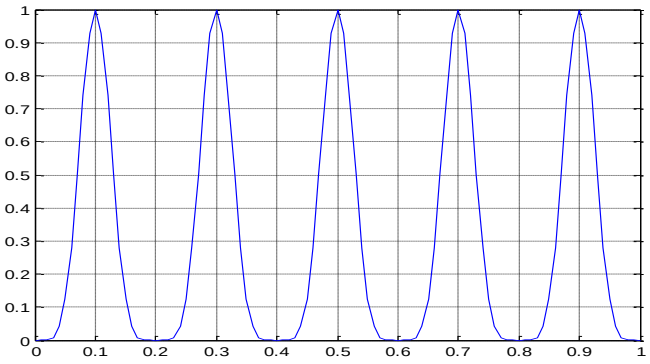

Figure 3.1. $F_a(x)$-a multimodal function with 5 equal peaks.

Most of the functions ($F_a(x)$ to $F_d(x)$) are one dimensional and defined on the interval [0, 1]. The first three functions have 5 peaks located at approximate values of 0.1, 0.3, 0.5, 0.7 and 0.9. $F_a(x)$ has equal peaks whereas $F_b(x)$ and $F_c(x)$ have unequal peaks (equations (3.1) to (3.3).

$$F_b(x)=e^{-2\log 2\left(\frac{x-0.1}{0.8}\right)^2}\sin^6(5\pi x) \tag{3.2}$$

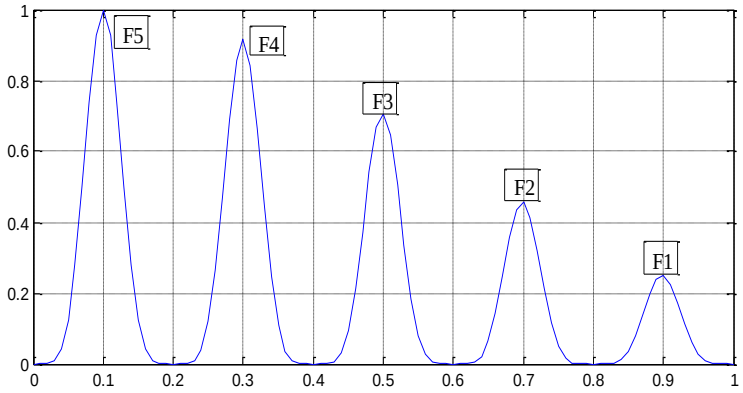

Figure 3.2. $F_b(x)$-a multimodal function with 5 unequal peaks.

In $F_c(x)$, there is a big gap in the fitness values of the highest and smallest peaks. The location of the highest peak tends to attract a significant proportion of the population. $F_d(x)$ has 10 peaks unevenly distributed over the search space. The two-dimensional shekel foxhole function ($F_e(x)$) has 25 peaks over the interval [-40,40].

$$F_c(x)=\begin{cases} 10\, e^{-2\log 2\left(\frac{x-0.1}{0.8}\right)^2} \sin^6(5\pi x), & \text{if } 0\le x\le 0.2 \\ e^{-2\log 2\left(\frac{x-0.1}{0.8}\right)^2} \sin^6(5\pi x), & \text{if } 0.2<x\le 1 \end{cases} \tag{3.3}$$

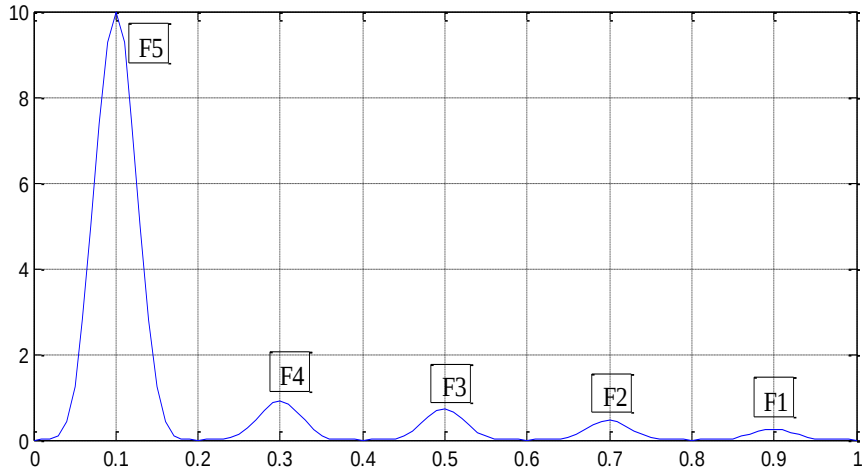

Figure 3.3. $F_c(x)$-a function with 5 unequal peaks with high fitness variation.

$$F_d(x)=\begin{cases} \sin^6(15\pi x), & \text{if } ((x\in[0.07,\ 0.27])\cup(x\in[0.4,\ 0.6])\cup(x\in[0.73,\ 0.94])) \\ 0\ , & \text{Otherwise} \end{cases} \tag{3.4}$$

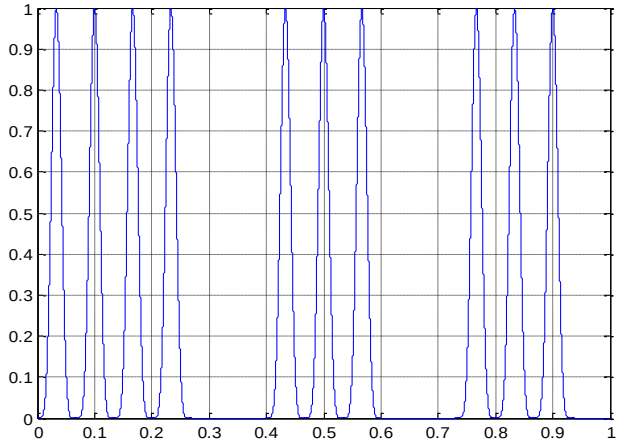

Figure 3.4. $F_d(x)$-a multimodal function with unevenly distributed peaks.

$$F_e(x,y)=500-\frac{1}{0.002+\sum_{i=0}^{24}\frac{1}{1+i+(x-a(i))^6+(y-b(i))^6}} \tag{3.5}$$

Where

$$a(i)=16*\big((i \bmod 5)-2\big) \text{ and}$$

$$b(i)=16*\left(\left\lfloor\frac{i}{5}\right\rfloor-2\right)$$

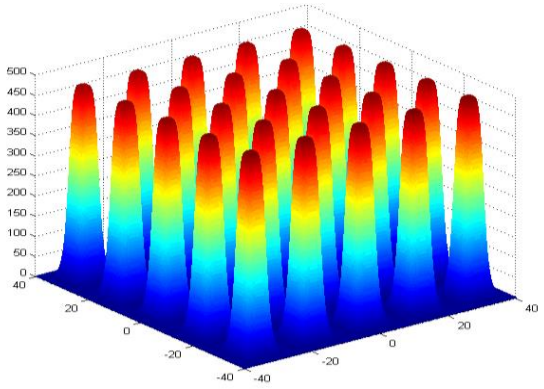

Figure 3.5. $F_e(x)$-Shekel's foxhole function.

3.2 Mathematical Formulation

In FPN, sharing is in proportion to actual fitness and hence the niche count of an individual *i* is given by equation (2.1) and its shared fitness is given by equation (3.7).

$$sh(d_{i,j})=\begin{cases} f_j \text{ , if } d_{i,j} < \sigma_{sh} \\ 0, \text{ otherwise} \end{cases} \tag{3.6}$$

$$f_{sh,\,i}=\frac{f_i}{\sum_{j=1}^{M} sh(d_{i,j})} \tag{3.7}$$

Where, M is the number of individuals in a given niche, f_i is the raw fitness and $f_{sh,i}$ is the shared fitness and $d_{i,j}$ is the phenotypic distance between individuals *i* and *j*.

The feasibility of the proposed niching scheme can be verified both mathematically and using simulation. It can be demonstrated analytically that FPN is indeed insensitive to the difference in fitness of the peaks. Like every other population based stochastic search algorithm,

GAs also require a certain reasonable population size for sufficient exploration of the search space. It can be shown that unlike the traditional fitness sharing, the FPN will tend to form a stable subpopulation around all the niches with no additional restriction on the size of the population for multi-modal functions of unequal peaks.

To get an insight on the performance improvement of FPN over TFS, let's consider a multimodal fitness landscape with M unequal peaks with F_1 and F_M being the minimum and maximum fitness values of the peaks respectively. Let the subpopulation size at each of the niches be denoted by n_1 to n_M respectively and N be the population size.

Using the traditional fitness sharing scheme, the shared fitness of an individual *i* in the k^{th} niche at steady state is given by equation (3.8) (Goldberg and Richardson, 1987).

$$f_i' = \frac{F_k}{n_k} \tag{3.8}$$

Assuming that after sufficient iteration almost all the population distributes around the M peaks, we get equation (3.9).

$$n_1 + n_2 + \ldots + n_M = N \tag{3.9}$$

To discover all the peaks, it is required that the shared fitness values at each niche should be approximately equal (i.e. $f'_1 = f'_2 = \ldots = f'_M$).

Substituting and rearranging terms, the number of individuals at the k^{th} niche is governed by equation (3.10)

$$n_k = \frac{F_k}{\sum_{i=1}^{M} F_i} N \tag{3.10}$$

But using FPN, the shared fitness of an individual in the k^{th} niche is given by equation (3.11).

$$f_k' = \frac{f_k}{\sum_{i=1}^{n_k} f_i} \tag{3.11}$$

Where, n_k is the subpopulation size at the k^{th} niche (location of a peak). After sufficient generations, individuals in each niche will have approximately equal fitness (i.e. $f_i=f_j$, for two individuals *i* and *j* in the same niche). Hence a simplified form of equation (3.11) is shown in equation (3.12).

$$f_k^{'}=\frac{1}{n_k} \tag{3.12}$$

From equation (3.12), for the shared fitness values to be equal, the population has to be evenly distributed among all the peaks, irrespective of the difference in the fitness value at the peaks (i.e. $n_1=n_2=....=n_M=N/M$). So, for a multimodal function having M peaks, the expected number of individuals at the k^{th} peak using TFS scheme is given by equation (3.13). But FPN distributes the population around the optimum points uniformly as shown in equation (3.14).

$$n_k=\frac{F_k}{\sum_{i=1}^{M} F_i} *N \tag{3.13}$$

$$n_k=\frac{N}{M} \tag{3.14}$$

It can be observed that, for a multimodal function with equal peaks (e.g. $F_a(x)$), equation (3.13) would degenerate to equation (3.14). Hence, for multimodal functions with equal peaks, FPN is essentially the same as the traditional sharing scheme. The underlying principle in FPN considers all the peaks as equally important for the GA and hence the proportion of the population at the different peaks does not really matter. This makes sense because from the perspective of the GA, what is important is whether the niching scheme is able to form a stable sub population around all the multiple optimum points. Once all the peaks are discovered, a preference between the different peaks can be made by arranging the final population based on the fitness values.

3.3 Complexity Analysis

Performance speed in GA depends on the population size. The higher the population size, the more computation time it takes for the GA to converge to the optimum points. Let P_{min} be the minimum size of a niche (i.e, the minimum number of individuals a niche needs to have) and P_{FPN} and P_{TFS} be the population size for the FPN and TFS algorithms respectively.

As discussed in the previous section, FPN distributes the population uniformly at each of the peaks (see equation (3.14)). Hence, the lower bound of the population size required by the FPN algorithm to discover all the peaks is given by:

$$P_{FPN} \geq M * P_{min} \tag{3.15}$$

And for the TFS, we know that the subpopulation size at each of the niches is proportional to the fitness values of the niches. Hence, the population size at the minimum peak (P_1 in this case) is given by:

$$P_1 = \frac{F_1}{\sum_{i=1}^{M} F_i} * P_{TFS} \tag{3.16}$$

Which means the lower bound population size for TFS is given as follows:

$$P_{TFS} \geq P_1 * \frac{\sum_{i=1}^{M} F_i}{F_1} \tag{3.17}$$

The niche with the lowest fitness value needs to have at least P_{min} individuals to be considered as a valid niche. Hence, equation (3.17) becomes

$$P_{TFS} \geq P_{min} * \frac{\sum_{i=1}^{M} F_i}{F_1} \tag{3.18}$$

This indicates that the traditional sharing scheme based on the niche count has a threshold requirement on the minimum population size to discover all the peaks when the objective function has unequal peaks. As the gap of the peak values increases, the required minimum population size also increases drastically.

Comparing equations (3.15) and (3.18), we get equation (3.19).

$$\frac{P_{TFS}}{P_{FPN}} = \frac{\sum_{i=1}^{M} F_i}{M*F_1} \tag{3.19}$$

From equation (3.19) it can easily be observed that FPN and TFS require the same population size for a multimodal function of equal peaks. However, for a rugged fitness landscape with large variation in fitness values of the peaks, the ratio in equation (3.19) becomes very large. Consider for instance a two peak scenario where the highest peak is C times the smallest one (i.e. $F2=C*F_1$, where C is a constant greater than 1). Then, the ratio in equation (3.19) is simplified as shown in equation (3.20).

$$\frac{P_{TFS}}{P_{FPN}} = \frac{F_1+C*F_1}{2*F_1} = \frac{1+C}{2} \tag{3.20}$$

For instance, if F_2 is 3 times higher than F_1 equation (3.20) then indicates that FPN runs twice faster than TFS.

For the test functions $F_b(x)$ and $F_c(x)$ given above, M=5, F_1=0.251, F_2=0.45, F_3=0.7, F_4=0.91 and the value of F_5 is 1 for $F_b(x)$ and 10 for $F_c(x)$. $F_c(x)$ has a large fitness gap between its highest and lowest peaks. If a niche size of at least two individuals is required at the lowest peak (i.e. n_1>=2) and substituting in the values, the expected subpopulation size at F_1 for $F_b(x)$ is given by equation (3.21).

$$\begin{aligned} n_1 &= \frac{F_1}{F_1+F_2+F_3+F_4+F_5} * N \\ &\cong \frac{0.251}{1+0.91+0.7+0.45+0.251} * N \\ &= 0.0758*N \end{aligned} \tag{3.21}$$

This implies that for the niching technique to locate all the peaks of $F_b(x)$, a population size of at least 27 is required. In practice, the desired population size has to be much larger than this ideal mathematical threshold. The optimum population size to discover all the peaks is

largely dependent on the ratio of the fitness at the peaks. The higher the fitness ratio between the peaks, the larger is the size of the population required to discover all the peaks. This is more evident in $F_c(x)$. $F_c(x)$ has a much higher fitness gap between its highest and lowest peaks as compared to $F_b(x)$.

Using the same expression given above and substituting in the numerical values $F_c(x)$, the expected number of individuals at F_1 will be:

$$\begin{aligned} n_1 &= \frac{F_1}{F_1+F_2+F_3+F_4+F_5} * N \\ &\cong \frac{0.251}{10+0.91+0.7+0.45+0.251} \\ &= 0.02 * N \end{aligned} \tag{3.22}$$

Quantitatively speaking, a population size of at least 100 is required to have at least two individuals at the lowest peak (F_1). FPN overcomes this requirement on the minimum size of the population by uniformly distributing the total population among the various peaks, irrespective of the difference in the fitness value. From equation (3.15), it only requires a population size of at least twice the number of peaks to have at least two individuals at each of the niches (i.e. one tenth of the population size required by the traditional sharing scheme in this particular example).

3.4 Performance Criteria

To verify the performance of FPN and compare it with the existing approach, three main criteria are used in literatures. The first criterion is the percentage of number of peaks discovered by the niching algorithm as a function of search cycle (generation). This, in effect is equivalent to comparing the ratio of the sum of the fitness of the local optima identified by the niching technique divided by the sum of the fitness of the actual optima in the search space. The second criterion is the distribution of the population around the optimum points. This shows whether the

niching technique is able to evolve a stable and diverse subpopulation. Another way of evaluating the performance of a niching technique is to measure the deviation of the actual population distribution from the expected population distribution. This is commonly called the chi-square like distribution and is discussed in the next section.

The goal of any niching technique in a multimodal optimization is to discover all the unique niches corresponding to the optimum points that prevail in the fitness landscape. The standard error measures the deviation of the number of niches discovered at each generation from the expected number of niches. The simulation is repeated for a total of R runs each time storing the number of niches discovered in a matrix of R rows and G columns. Then, the expected number of niches and the standard error are calculated using equations (3.23) and (3.24).

$$\langle V(g)\rangle = \frac{1}{R}\sum_{r=1}^{R} W_{r,g}, \qquad \forall g \epsilon (1,\ldots G) \tag{3.23}$$

$$E(g) = \sqrt{\frac{1}{R} * \left(\frac{\sum_{r=1}^{R} (W_{r,g} - \langle v(g)\rangle)^2}{R-1}\right.} \tag{3.24}$$

Where V(g) and E(g) are the mean and standard error at generation *g*, $W_{r,g}$ is the number of niches discovered at the r^{th} experiment and g^{th} generation, G is the number of generations, R is the number of times the simulation is repeated.

3.5 Ecological Analogy

The sharing technique introduced here has its inspiration from species interaction in biological ecosystems. Ecological niches maintain diverse species where similar individuals share the resource of that particular ecological niche. The traditional sharing technique is based on the notion of spatial distribution of species following the location of a given resource. There

is no diversity in the type of the individuals in the species. The traditional GA assumes the resource of a niche is infinite and hence a niche with the highest fitness value can accommodate all of the population. The whole population then rushes to converge to the global optimum losing diversity and hence premature convergence. The sharing concept is based on a finite resource model (i.e. the resource of a given niche is finite). Higher peaks in the fitness landscape attract more individuals and as more and more individuals come to that location, the resource of that particular niche gets depleted and niches of smaller fitness value tend to attract other individuals in the population.

FPN is also based on a finite resource model approach. The underlying principle in FPN is analogous to an ecological system that embraces diverse species sharing a non-uniform environmental resource (for instance, in an ecosystem antelope, hyenas and rhinoceros can occupy spatially different locations consuming different amount of the same resource, water for example). Diversity is not only in spatial distribution but also in the type of individuals. Under such an analogy different amount of the same resource can support equal number of different species (for instance, a 10 meters cube of water may support 3 lions whereas 6 meters cube of water may suffice to support the same number of dogs for the same duration). The non-uniformity in the distribution of a resource is accounted by the multi-modal fitness landscape with unequal peaks. Unlike, the traditional sharing scheme where most of the population converges at a niche of more resources, FPN distributes equal number of different individuals along the various peaks. This approach makes sense because from the perspective of the GA, what is important is whether the niching scheme is able to form a stable subpopulation around all the multiple optimum points. In other words, FPN considers all the peaks as equally important and hence the subpopulation size is independent of the fitness of the peaks.

Figure 3.6 shows an ecological analogy where a different amount of a resource (F1 to F4) can support equal number of (N) individuals of different species.

3.6 Dynamic Niche Identification with Niche Expansion (DNINE)

As pointed out earlier in this dissertation, the performance of both the clearing and fitness sharing types of niching is highly dependent on the use of proper values of the niche radius, population size and the number of peaks of the multi-modal function. That requires a priori knowledge of the search space and fitness landscape which makes their application to optimizing most real world problems nearly impossible.

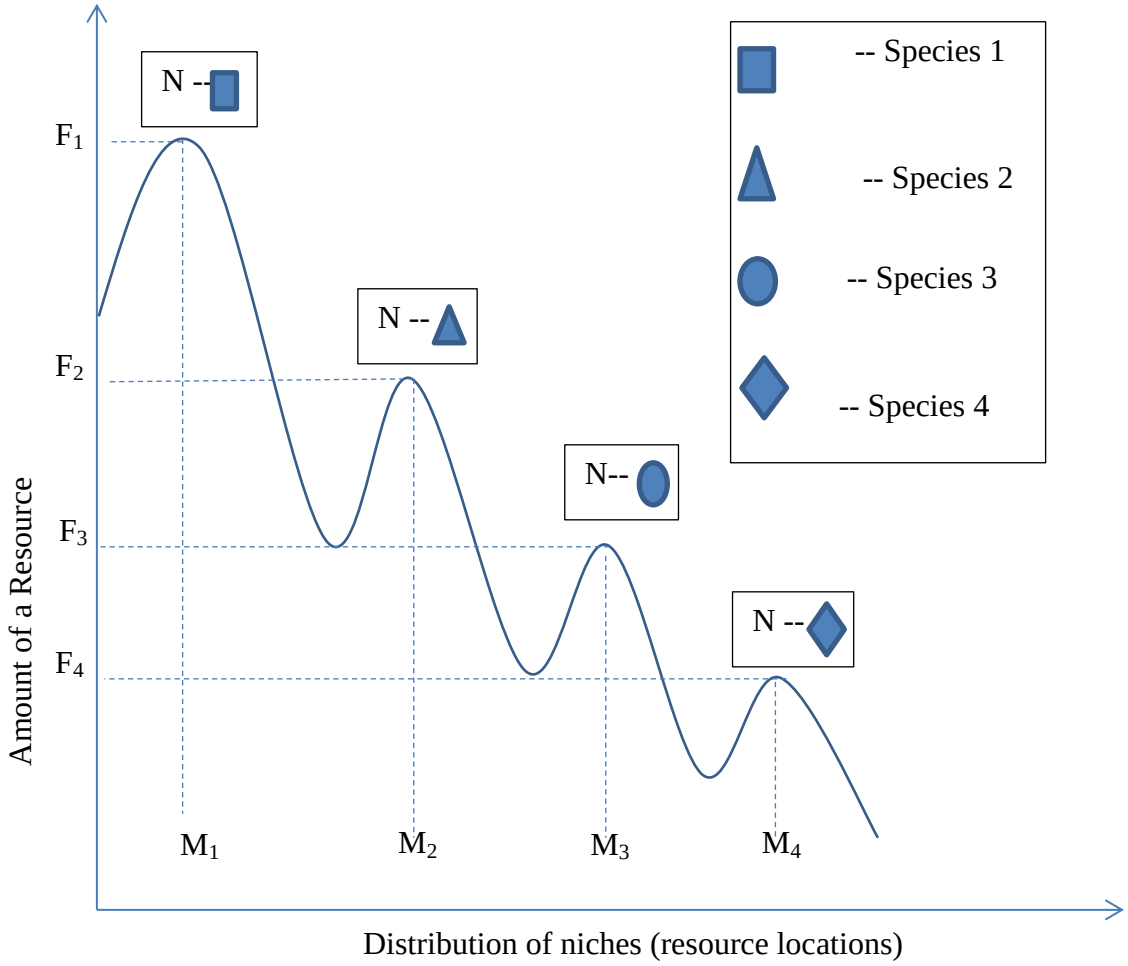

Figure 3.6. Ecological analogy for a multimodal fitness landscape.

Without an accurate estimate of the niche radius and knowledge of the number of peaks and the gap between them, the performance of the TFS and clearing techniques can be very dismal. Particularly, a correct value of the niche radius is very important for the sharing technique so as to maintain a stable and diverse subpopulation at the various peaks. For a higher niche radius, the sharing technique cannot discriminate between neighboring niches, while a very small radius value will lead to the creation of false niches and reduce the overall performance of the technique. Our niching technique, though it is insensitive to the difference in the fitness values of the various peaks, still relies on the accurate estimation of the niche radius. In our work, we applied a modified version of the dynamic niche identification technique proposed in (Chang et al., 2010). The algorithm is based on the idea of population dynamics in a given city. When a city is crowded, scarcity of resource and living cost motivates part of the population to migrate to nearby cities or leads to the emergence of new cities. Similarly, when a niche is overcrowded part of its population migrates to nearby niches. There is no niche migration in our implementation. Instead, the idea of niche expansion is applied by merging two niches when there is no valley between them. The algorithm starts with a small initial niche radius and subdivides the population based on this radius. Normally, the use of a small niche radius results in the formation of fictitious niches. Niche refining is done using the niche expansion principle (merging any communicating niches) to identify actual niches that prevail in the population. To clarify the working principle of the niche expansion technique, we provided the definition of related terminologies below. The algorithmic implementation of the DNINE algorithm is given in Appendix A.

Definition 1: Niche Master. Each niche (subpopulation) is represented by its master. The niche master of a given niche is the individual with the highest fitness in that particular niche.

Definition 2: Distance between Niches. The distance between two niches is defined as the distance between their niche masters. Suppose M_i and M_j be the two niche masters of N_i and N_j, then the distance between these two niches is calculated using equation (3.25).

$$D_n(N_i,N_j)=d(M_i,M_j)=\|M_i-M_j\|^2 \tag{3.25}$$

And the line that intersects the two niche masters can be expressed using equation (3.26).

$$X=M_i+k(M_j-M_i), \text{ where } k\in[0,1] \tag{3.26}$$

Definition 3: Communication between Niches. Two niches are said to be communicating with each other when there is no valley between them. To check whether there is a valley between the two niche masters, a series of points ($x_1, x_2, \ldots x_q$) are generated between the two niche masters. If there exists a point x_m that satisfies the inequality in equation (3.27), then there exists a valley between the two niches and hence there is no communication between them.

$$f(x_m)<\min(f_i,f_j) \tag{3.27}$$

Where f_i and f_j are the fitness values of niche masters *i* and *j* respectively.

Absence of communication between the two niches indicates that the two niches are actual independent niches and they should be maintained in the population. However, if there is a communication between niches, it means that both niches are on the same side of a valley and possibly they can be merged. If no point has lower fitness than either of the end points, then it indicates that no valley lies between the two niches. The core idea behind the niche expansion is that niches with no valley in between can be merged and a new niche master representing the bigger niche is selected.

In Figure 3.7, M_1 and M_2 communicate where as M_1 and M_4 do not, as there is a valley between them. Accordingly, the niche expansion algorithm merges M_1 and M_2 in subsequent iterations while M_4 stays as an independent niche.

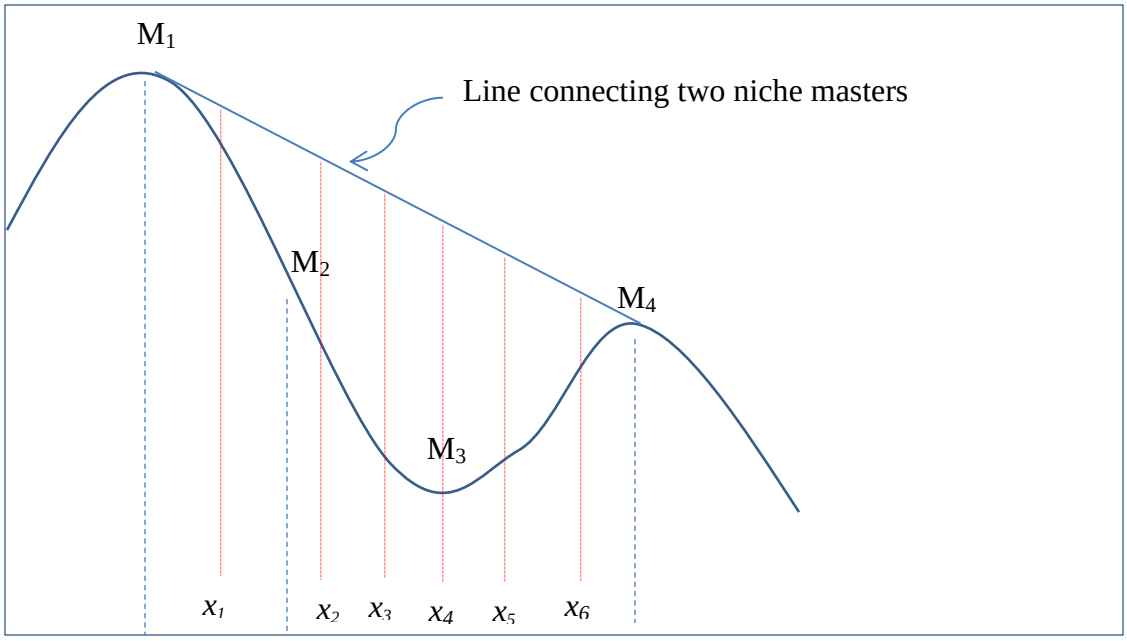

Figure 3.7. Niche distribution for a multimodal fitness landscape.

A sequence of points on the line joining the two niches are generated (see Figure 3.7, x_1 to x_6, for instance) to determine whether the two niches communicate or not. Then, the fitness of those points is compared with the fitness of the niche masters. As can be seen from Figure 3.7 above, there exists a point (x_4) satisfying the inequality in equation (3.27). This indicates that there is a valley in between the two niches, M_1 and M_4 in this case.

Figure 3.8 shows the impact of noise on the performance of the niche expansion algorithm. M_2 has a lower fitness as compared to M_1 and M_3 indicating that there is a valley between M_1 and M_3. This however can be due to the impact of noise and M_1 and M_3 should not be considered as two independent actual niches. To overcome the effect of noise, a noise tolerance factor is introduced to modify the inequality given in equation (3.27) as shown in equation (3.28).

$$f(x_m) < \gamma * \min(f_i, f_j) \tag{3.28}$$

Where γ is a random number between 0.8 and 1.

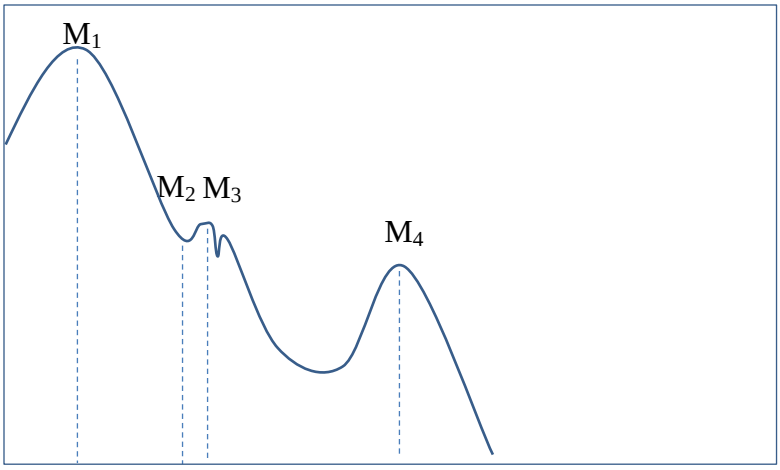

Figure 3.8. Effect of noise on niche migration.

3.7. Chi-square like Distribution

To test the statistical soundness of the algorithm, a chi-square like deviation is computed for two of the benchmark functions (F_a (x) & F_c(x)). The chi-square like deviation for the *q* niche peaks plus the nonpeak niche is given by equation (3.29).

$$csd = \sqrt{\sum_{i=1}^{q+1}\left(\frac{X_i-\mu_i}{\sigma_i^2}\right)^2} \tag{3.29}$$

Where,

$$\mu_i = N^* \frac{f_i}{\sum_{k=1}^{q} f_k}, \quad \sigma_i = \mu_i\left(1-\frac{\mu_i}{N}\right)$$
$$\sigma_{q+1} = \sum_{i=1}^{q} \sigma_i^2, \qquad \mu_{q+1} = 0 \tag{3.30}$$

The chi-square like deviation is a measure of the deviation of the actual population distribution from the ideal population distribution at each of the peaks. The smaller the chi-square value, the lesser is the deviation from the ideal distribution and hence the better is the algorithm. As can be seen from Figure 3.9, FPN and TFS have nearly the same distribution for a

multi-modal function of equal peaks ($F_a(x)$). This is expected because, for a multi-modal fitness landscape with equal peaks, TFS also tends to distribute the population evenly across the various peaks at steady state.

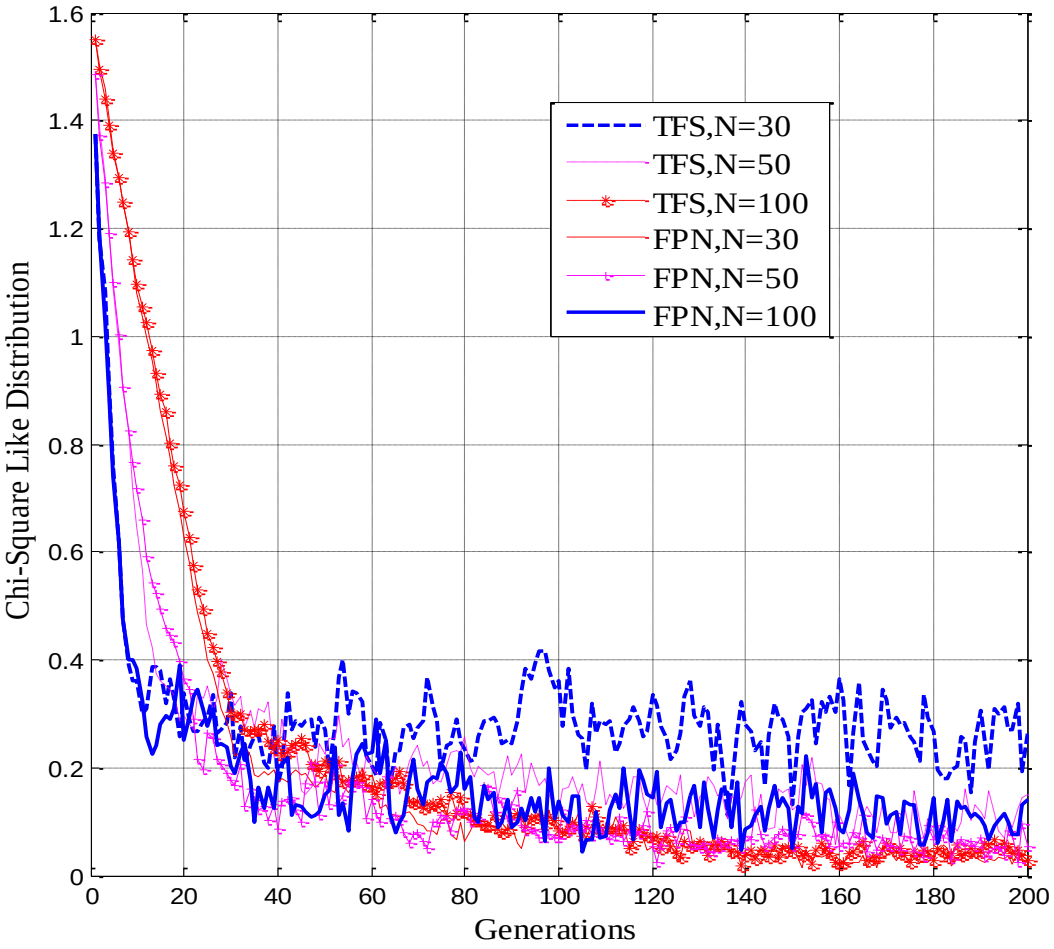

Figure 3.9. Chi-square like deviation for Fa(x).

However, as the gap in fitness values of the peaks increases, FPN starts to outperform TFS. This is clearly evident from Figure 3.10, where there is a big difference in the chi-square like values. The FPN algorithm is able to distribute the population along the various peaks with a small deviation from the ideal mathematical distribution irrespective of the gap in fitness values at the peaks. The results are of course in harmony with the distribution obtained from simulation results in Chapter 6 (see Table 6.1 and Table 6.2).

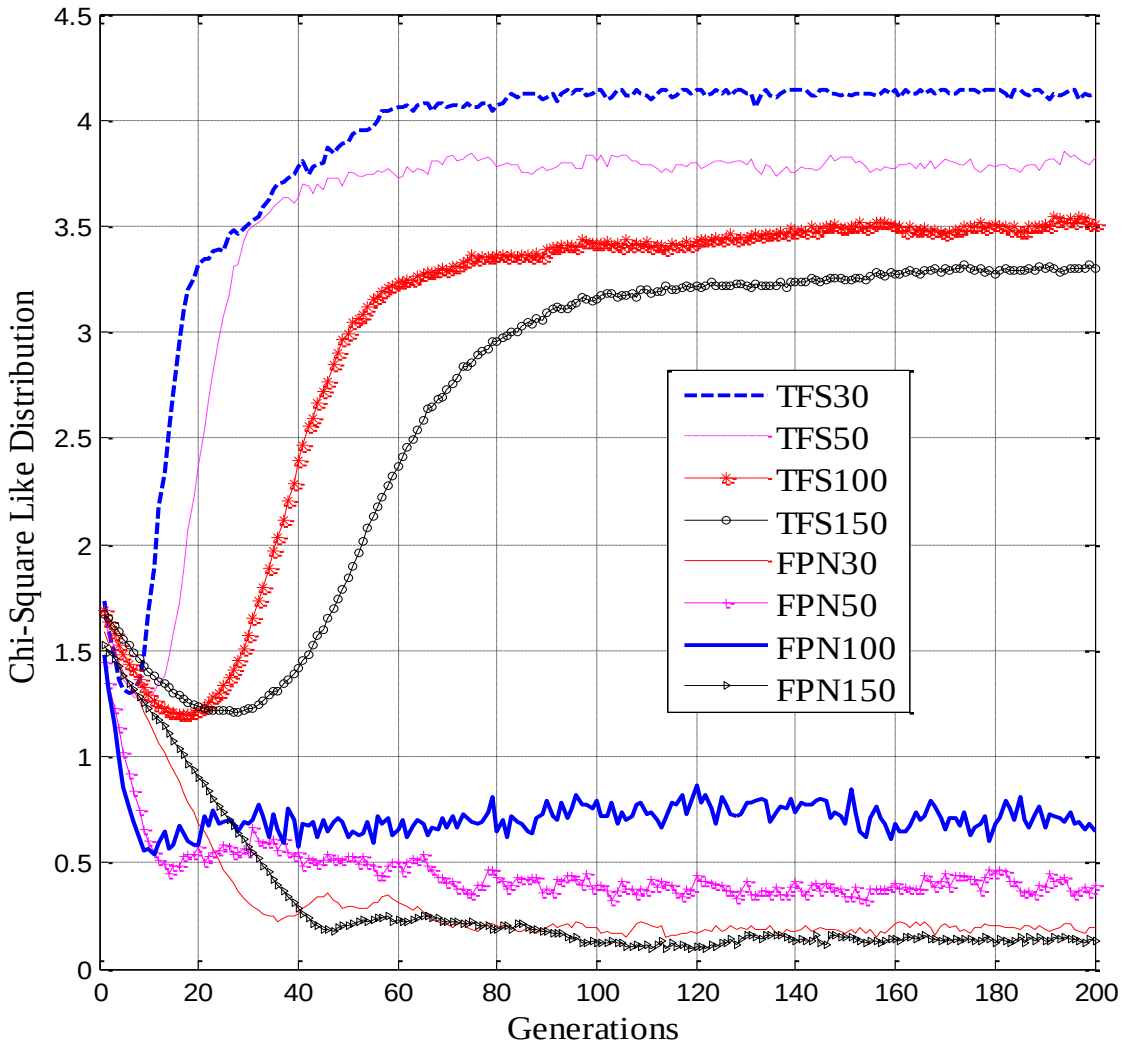

Figure 3.10. Chi-square like deviation for Fc(x).

CHAPTER 4

Evolving Hierarchical Cooperation in Classifiers

Classification is a supervised learning where the learning system, once sufficiently trained, seeks to categorize previously unseen instances in to correct classes or labels. A Learning Classifier System (LCS) accomplishes this task by evolving a population of classifiers using a reinforcement signal. It is a machine learning system based on reinforcement learning and Genetic Algorithms (GAs). Like an expert system, it utilizes a knowledge base of syntactically simple production rules that can be manipulated by GA (J. Holland, 1975, 1992). The use of a rule-based system allows an LCS to conveniently represent and refine complex control strategies (Wilson and Goldberg, 1989). Classifiers are rewarded every time an input is correctly classified. All classes (labels) are considered equally important (i.e., a classifier which correctly classifies an input from one class and another which correctly classifies a different input from another class are equally rewarded by the trainer). This essentially turns the classification problem into an optimization of a multimodal function with equal peaks. To achieve this task, an LCS has to build a set of rules that work in coordination to accurately model a given environment. This requires a mechanism to evolve and sustain a diverse, cooperative population of rules that together represent a concept or model a set of behaviors that solve a given problem. Building a hierarchical set of rules, where accurate and more specific rules respond to a subset of the situations covered by more general but less accurate rules is vital for a concise concept description, especially when dealing with an environment that has huge numbers of states. The LCS in its very nature has a unique power of discovering cooperating rules through the robust search ability of the GA by the guidance of reinforcement learning. However, the LCS would tend to lose diversity due to a strong selection pressure of the GA. Hence, to maintain a

cooperative diversity while applying a selection operator to the population of rules, it must incorporate some form of niching mechanism (Horn, 1993; Horn and Goldberg, 1996). Niching provides the LCS with the required restorative force to maintain diversity in the face of selection pressure. This chapter explores the impact of niching in LCS from the perspective of attaining a diverse set of cooperative hierarchical rules. The FPN niching technique introduced in chapter 3 of this dissertation is used as a resource sharing mechanism to reinforce classifiers that match to a given input.

In this work, we considered a stimulus-response (Nasraoui and Krishnapuram) based LCS system where an immediate reward or punishment is provided at each computational time step by the external environment. For such a system, there is no need of a complex credit assignment algorithm like the bucket brigade and hence the message list in Holland's formulation of LCS is omitted in our formulation. Though this work exclusively focuses on investigating the significance of implicit niching for evolving a multi-level hierarchical cooperation in a population of diverse rules for Boolean function learning, the algorithm developed here can be extended to any evolutionary algorithm that seeks to evolve a hierarchical set of cooperating rules for a concise concept description. A mathematical formulation for predicting the steady state strengths of subpopulations is also provided. Most of the content in this section is published in our recent work on the formation of default hierarchy in Boolean function learning using LCS (Workineh and Homaifar, 2012b).

4.1 Hierarchy in LCS

In a Michigan style LCS, an individual classifier in the population represents part of a solution to a given problem. There is no single rule that adequately models the environment instead the solution domain comprises of a set of rules that collectively give a better model of the

environment. Hence, a complete solution to the problem involves coordination among sets of rules in the population. And the search for cooperative rule sets takes place within a single population of competing and cooperating rules. Consider a scenario where the learning system needs to model an environment with huge number of states. An LCS that is to operate in such an environment can be modelled in either of two ways. The first is to build a model of the environment using a set of rules that never make mistakes. This homomorphic approach, however, is not practically feasible as it requires a vast number of rules to model realistic environments (Riolo, 1987). Besides, an environment exhibiting perpetual novelty combined with a limited sampling of it adds another order of complexity to this homomorphic approach (Booker, 1982, 1989). The other alternative is to build a hierarchical model where the task of the learning system is to categorize the states in to groups that can be treated in a similar way (Riolo, 1987; R. Smith and Goldberg, 1992). A hierarchical rule set provides a multi-level structure in which rules at the bottom of the hierarchy are very general and those at the top are very specific (refer to Figure 4.1).

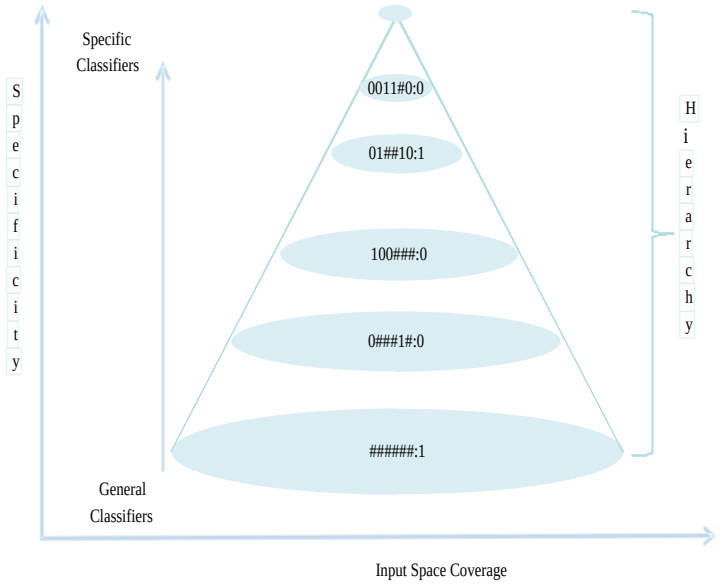

Figure 4.1. A hierarchical rule structure.

Take for instance, the 6-bit Boolean multiplexer (6-mux) problem whose disjunctive normal form is as follows:

$$Y=\bar{s_1}.\bar{s_2}.a_1+s_1.\bar{s_2}.a_2+\bar{s_1}.s_2.a_3+s_1.s_2.a_4 \quad (4.1)$$

Where $a_1 \ldots a_4$ are the data lines, s_1 and s_2 are the select inputs and Y is the output. The system's decision is correct when its output value is the same as the value of Y in equation (4.1) for a given input. Figure 4.2 shows both non-hierarchical and a hierarchical solution set for the 6-mux problem.

Non hierarchical set	Hierarchical set
000###:0	000###:0
001###:1	01#0##:0
01#0##:0	10##0#:0
01#1##:1	11###0:0
10##0#:0	######:1
10##1#:1	
11###0:0	
11###1:1	

Figure 4.2. Hierarchical and non-hierarchical solution set for 6-multiplexer problem.

The 8 rules in the non-hierarchical set in Figure 4-2 are the perfect solutions to the 6-mux problem. With default hierarchy, the same problem can be solved with a more compact hierarchical rule set. The last rule (with all hashes in its condition) in the hierarchical set is a default rule and the other four rules are exception rules. The default rule matches to all inputs covered by the exception rules but it makes a correct decision only half of the time. The default, while correct most of the time, is less accurate and makes mistakes for some cases it matches. Similarly, a hierarchical solution for the more complex problem (11-multiplexer) is shown in Figure 4.3.

Nonhierarchical Set	Hierarchical Set
0000#######:0	0000#######:0
0001#######:1	001#0######:0
001#0######:0	010##0#####:0
001#1######:1	011###0####:0
010##0#####:0	100####0###:0
010##1#####:1	101#####0##:0
011###0####:0	110######0#:0
011###1####:1	111#######0:0
100####0###:0	##########:1
100####1###:1	
101#####0##:0	
101#####1##:1	
110######0#:0	
110######1#:1	
111#######0:0	
111#######1:1	

Figure 4.3. Hierarchical and non-hierarchical solution set for 11-mux problem.

Generality and specificity of a classifier in an LCS is dependent on the number of hashes (i.e '#' symbols) in its condition. The more hashes a classifier has, the more general it is and covers more of the environmental niches. On the other hand, a specific classifier has fewer numbers of hashes in its condition. An exception rule is a specific rule with an action different from that of the default. Hierarchy can occur at any level within the rule sets. The term default hierarchy here refers to a hierarchical set of rules that contain a default rule for the default class along with other exception rules.

A working default hierarchy provides a great parsimony of the required rules to model the environment. In addition, the system's performance can be improved by adding more exception rules to the hierarchy. The essence behind achieving a working default hierarchy is therefore to build a more compact rule set with a reasonably fair accuracy as compared to the homomorphic model which aims to discover a set of rules that never make mistakes. This requires the coexistence of exception and default rules in the system.

In this regard, there are two major challenges to the formation of a viable default hierarchy in LCS. The first challenge is how to evolve the population to a hierarchical set. The learning process has to drive the system from an initial state of random population of rules to a state that embraces a hierarchical set of rules which models the environment at a desired accuracy. Evolving to a hierarchical set is one part of the challenge. Once the hierarchy is attained, it is also equally important to sustain it for generations in the face of deletion by the discovery component.

At times when a default and an exception rules match to an input, there should be a mechanism that favours the exception to fire. In other words, for a hierarchy to work properly, the exception rule has to cover the default at times when the default is wrong. These requirements urge the use of not only a proper niching scheme to maintain diversity but also a special bidding strategy to favor the exception rule over the default when both match to an input.

4.2 System Formulation

4.2.1 Classifier format. The classifier format in our implementation has 5 parameters: condition, action, strength, experience (Exp) and creation time (Ctime) (see Figure 4.4). The condition is a string from the ternary alphabet (0, 1 or #) and the action is binary (0 or 1). The hash symbol (#) in the condition is “don’t care” and matches to any input. The experience and creation time parameters are added for better understanding of the learning process. Experience (Exp) indicates the participation of a classifier in decision making process (i.e. match set) and the creation time refers to the iteration time at which the classifier is created. It helps to investigate whether a hierarchy once evolved can be survived for generations.

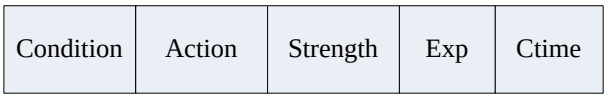

Figure 4.4. Classifier format.

The following notations are used in this dissertation: [P] refers to classifiers in the population, which is the bigger set, [M] refers to the match set and [AL] represents classifiers in the advocate list, which is a subset of [M].

4.2.2 Learning system. Learning in an LCS is an ongoing adaption to a partially known environment and not an optimization problem as in most reinforcement learning systems. The learning system includes the following major components: the auction, clearing house (CH), fitness proportionate resource sharing (FPRS) and the GA.

4.2.2.1 Auction. This is the part where classifiers in the match set participate in auctions by bidding a fixed proportion of their strength. The bid amount depends on the value of its current strength and the specificity. The deterministic potential bid (PB) of a classifier *i* during auction is given in equation (4.2).

$$PB_i = C_{bid} S_i \underbrace{\left(1 - \frac{NH}{CL}\right)} \tag{4.2}$$

Where NH is the number of hashes in the condition string, C_{bid} is the bid constant (see Table 4.1), S_i is the current strength and CL is the condition length. The specificity parameter is the ratio of the number of non-hash symbols to the condition length. The deterministic bid is not used directly to determine the auction winner. Instead, it is slightly perturbed by adding a random noise to promote exploration of the classifier space.

The effective bid (EB) is computed by adding a random noise to the bids submitted by each competing classifier using equation (4.3).

$$EB_i = PB_i \underbrace{(1 + rand() EBID))} \tag{4.3}$$

Where, EBID is the effective bid factor used during simulation to limit the random perturbation on the deterministic bid within a small range (10% for instance).

4.2.2.2 Fitness Proportionate Resource Sharing (FPRS). The learning system continuously interacts with its environment through its detectors and effectors. It uses a feedback about the impact of its action on the environment to learn from experience. The learning agent is blind without a proper guidance by a reward signal. A trainer is therefore necessary to determine whether the environmental modification was beneficial or detrimental. The reinforcement program (RP) determines the rule's fitness by generating a signal in the form of a reward or punishment. It determines the rule fitness and enables the system to learn from its environment based on a reward signal that implies the quality of its action. If the whole learning system is a water fall, the RP is the pipe that guides it to a point of interest. This work introduces a novel niching scheme termed fitness proportionate resource sharing, given in equation (4.4), for the formation of a viable default hierarchy.

$$r_i(t)=R\frac{S_i(t)}{\sum_{k=1}^{M(t)} S_k(t)} \tag{4.4}$$

Where, R is the total reward provided by the environment whose value is initialized once, M(t) is the number of classifiers in the advocate list at iteration *t,* $r_i(t)$ is the fraction of the total reward (R) that goes to classifier *i* at iteration *t* and $S_i(t)$ is the strength of classifier *i* at iteration *t*. The constant reward provided by the external environment is shared proportionally among classifiers in the advocate list.

4.2.2.3 Clearing House (CH). The CH is the part of the learning system that deals with the modifications in strength of classifiers as the classifier system learns. All classifiers pay existence tax and classifiers in the match set pay an additional overhead tax while classifiers in the advocate list has to pay also the bid amount. Assuming correct decision is taken by the system at iteration *t,* the strength of a classifier *i* in the advocate list at the next iteration is governed by equation (4.5).

$$S_i(t+1)=S_i(t)(1-C_{ext}-C_{oh}-C_{bid})+r_i(t) \quad (4.5)$$

Where, C_{ext} and C_{oh} are the existence and overhead tax constants respectively, C_{bid} is the bid constant, and $S_i(t)$ and $r_i(t)$ are the strength and reward for classifier *i* at iteration time *t*. A different paying and bidding policy (where the specificity factor is used during the bid computation to decide the winner, but it is left out during the calculation of the classifier's payout) is followed in this work. As can be seen from equations (4.2) and (4.5), the pay out of a classifier in equation (4.5) (i.e. $C_{bid}*S_i(t)$) is different from the potential bid amount given in equation (4.2) due to the specificity term used in bid computation.

Classifiers in [M] that are not in [AL], do not pay the bid and do not share a reward. Hence their strength is governed by equation (4.6).

$$S_i(t+1)=S_i(t)(1-C_{ext}-C_{oh}) \quad (4.6)$$

For classifiers in [P] that are not in [M], C_{oh}, C_{bid} and $r_i(t)$ are all zero and equation (4.5) is simplified as shown in equation (4.7).

$$S_i(t+1)=S_i(t)(1-C_{ext}) \quad (4.7)$$

4.2.2.4 Genetic Algorithm (GA). The GA discovers new rules among a population of candidate rules based on the experience of existing rules. It diversifies the population using mutation and cross over operators. A roulette wheel selection method is used to select parents for reproduction. The strength of new classifiers emerging from GA is initialized to a value that is neither too high (so that they do not dominate experienced classifiers) nor too low (to make them competent with the relatively more experienced classifiers in the system during auctions). In line with previous research work (Homaifar et al., 1988; Workineh and Homaifar, 2011), the strength of the new classifiers is initialized to a third of their parents' strength. Other initialization techniques (for instance initializing it to the average of the parents' strength) are also applied but

there is no significant difference in performance. Each GA operation brings two new classifiers that replace classifiers with lowest strength in the existing population. Figure 4.5 shows the interaction of the three major components of an LCS during the learning process when an environmental input is detected.

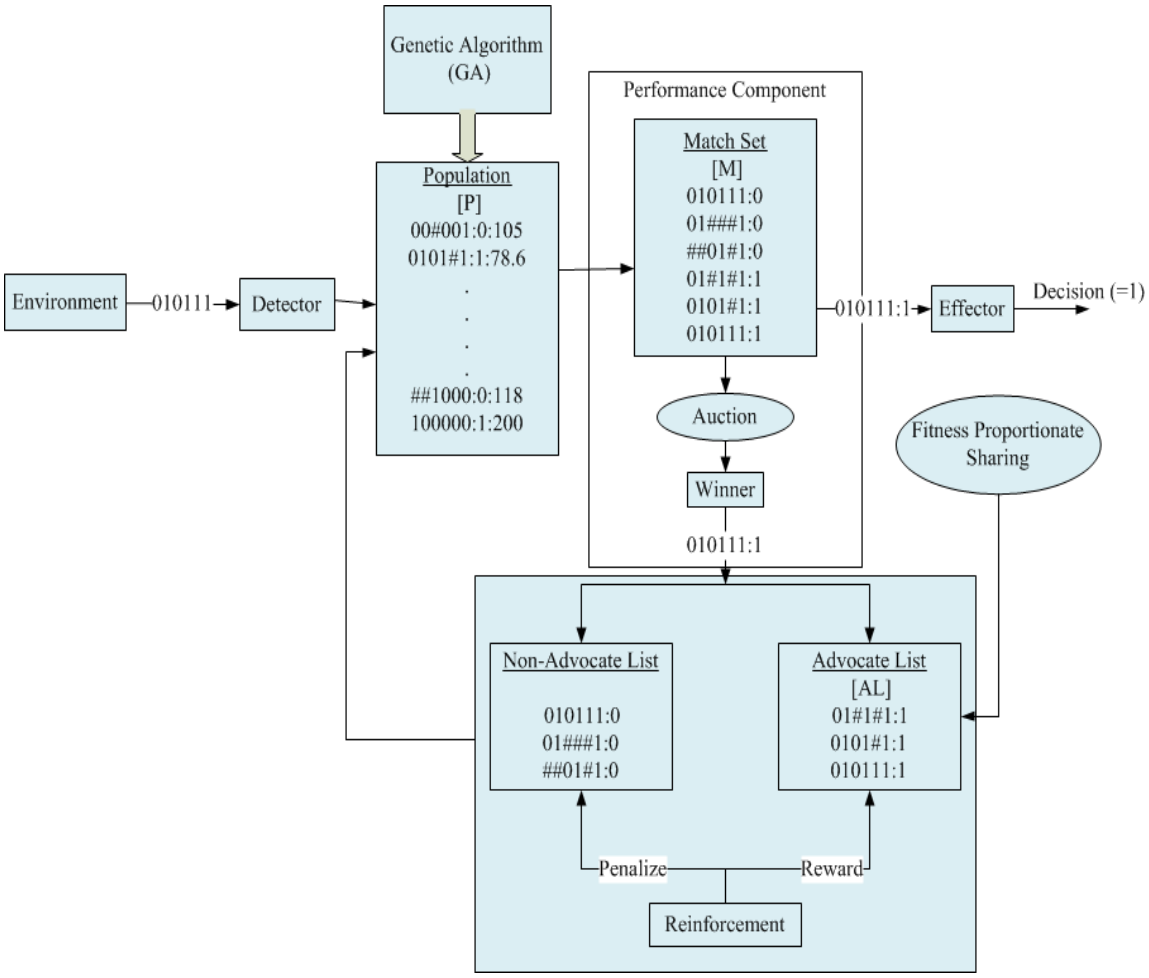

Figure 4.5. A Block diagram of an LCS in learning mode.

First, the performance component selects classifiers in the population whose conditions matched to the current input and forms the match set. Classifiers in the match participate in an auction to take action. Once the winner is classifier is identified, the reinforcement component provides a reward or punishment based on the action taken and the population is diversified by discovering new rules using the discovery component.

4.3 Learning Cycle

The learning cycle of the system implemented in this work involves the following sequences of computations at each iteration time.

1. Read an input from the environment.
2. Form the Match set [M] using all classifiers in the population [P] that match to the current input.
3. Classifiers in [M] bid in an auction by submitting a proportion of their strength using equation (4.3).
4. Declare a winner classifier based on highest effective bid submitted in step 3.
5. Form the advocate list [AL] out of classifiers in [M] proposing the same action as the winner classifier's action.
6. Execute action on the environment and possibly receive a reward (R).
7. If a reward is generated following the action in step 6, distribute the reward proportionally using equation (4.4) among classifiers in [AL].
8. Update the strength of classifiers in [P] using equations (4.5) through (4.7) accordingly.
9. Discover two new rules by applying GA on [P]
10. Repeat the above steps until a stopping criterion is met.

4.4 Steady State Analysis

LCSs are generally complex stochastic systems and a complete mathematical modeling of the learning dynamics is almost impossible even for the simplest scenario of a stimulus response LCS. To get a thorough understanding of the learning system, we presented a detailed simulation results that show the population dynamics, the performance accuracy and the variation of the strength at each epoch for learning Boolean function mapping in the multiplexer

problem. An epoch stands for one complete presentation of the environmental inputs to the system. The main purpose of our work is to introduce a novel sharing scheme that leads to the formation of a viable default hierarchy in LCS. To come up with an exhaustive mathematical formulation for predicting the emergence of subpopulations, the composition of the population at equilibrium and quantitative characterization of the maintenance of niches under the selection pressure of GA is not the primary intent of this work. The authors in (R. Smith et al., 1993) showed the impact of GA selection on the population composition under a fitness sharing scheme. Horn and Goldberg (Horn and Goldberg, 1996) also made an analysis of the niche dynamics for a much simpler LCS system (assuming the existence of only two niches and every classifier has equal specificity). In our case, classifiers can have different specificity and no assumption is made on the number of niches and the reward share of classifies is dependent on the current size of the advocate list, which we do not have any a priori knowledge. Under these circumstances, analysis of the niche dynamics for such a complex scenario is a daunting task.

But to have a further insight on the variation of the strength of classifiers under the selection pressure of GA, we made some simplifying assumptions and deduce mathematical expressions that can predict the steady state behavior of the total strength of subpopulations. It is interesting to express the impact of the various forms of taxes, the bid and the shared reward quantitatively on the steady state strength of classifiers in the population. We made an assumption that a default hierarchy will emerge at some epoch in the learning process and take control of the system. In other words, our steady state analysis gives the variation of the total strength of subpopulations (default and exception classifiers) disregarding the impact of other classifiers that are not part of the hierarchical set. The validity of this assumption is in fact experimentally proved by the series of simulation results conducted.

Also, as it is difficult to trace a specific classifier in the population and for the sake of simplicity, we treated classifiers in group and the total strength of the group is considered during the analysis. For instance, instead of dealing with each exception classifier separately, the whole set is treated as one group and the default set as another group. Two things are valid under this assumption. First, every group (the default set and the exception group) receives a reward at every other iteration. Second, considering classifiers in group avoids the dependency of strength modification on the size of the current advocate list (since the fraction term in the reward allocation adds up to unity for the group).

The default classifier matches to every input and is always subject to (C_{ext} and C_{oh}). It is correct only half of the time and hence would possibly receive a reward every other iteration. The other half of the inputs is covered by the exception classifiers. In effect, the exception group would receive a reward every other iteration for the cases where the default classifier is wrong.

Now, let *t* be an iteration time where there is no reward for the default classifier. This means at *t+1*, the default expects a reward and so on. The same is true for the exception group. Let K be the sum of all taxes and the bid amount, K_1 be the sum of the existence and overhead tax and K_2 be the existence tax, M_t the size of the advocate list at iteration *t*, $S_{DG}(t)$ represents the total strength of all instances of the default classifier at iteration *t*, $S_{EG}(t)$ represent the total strength of all classifiers in the exception group at iteration *t*.

4.4.1 Default classifier. The strength of the i^{th} default classifier (S_{Di}) at iteration *t+1* can be expressed in terms of its strength one iteration before using equation (4.8).

$$S_{Di}(t+1)=S_D(t)(1-K)+R\frac{S_{Di}(t)}{\sum_{j=1}^{M_t} S_j(t)} \tag{4.8}$$

Considering all instances of the default classifier in group, we know that the total reward R is distributed among different instances of the default and hence the fraction term adds up to

unity. Hence, the total strength for the default group at iteration *t+1* can be expressed in terms of the total strength one iteration earlier using equation (4.9). Similarly, the strengths at the *t+2* and *t+3* iterations are given by equations (4.10) and (4.11) respectively.

$$S_{DG}(t+1)=S_{DG}(t)(1-K)+R \tag{4.9}$$

$$\begin{aligned} S_{DG}(t+2)&=S_{DG}(t+1)(1-K_1) \\ &\equiv S_{DG}(t)(1-K)(1-K_1)+R(1-K_1) \end{aligned} \tag{4.10}$$

A reward comes at the $(t+3)^{th}$ iteration and the total strength at the $(t+3)^{th}$ iteration is given by equation (4.12)

$$S_{DG}(t+3)=S_{DG}(t+2)(1-K)+R \tag{4.11}$$

Back substituting the value of $S_{DG}(t+2)$ from equation (4.10) in to equation (4.11)

$$S_{DG}(t+3)=S_{DG}(t+1)(1-K_1)(1-K)+R \tag{4.12}$$

At steady state, we expect the following equations to hold.

$$\begin{aligned} S_{DG}(t)&=S_{DG}(t+2) \equiv S_{DG_{ss1}} \\ S_{DG}(t+1)&=S_{DG}(t+3) \equiv S_{DG_{ss2}} \end{aligned} \tag{4.13}$$

Solving these equations one at a time, we get equations (4.14) and (4.15).

$$\begin{aligned} S_{DG}(t)&=S_{DG}(t)(1-K)(1-K_1)+R(1-K_1) \\ &\gg S_{DG_{ss1}}\left(1-(1-K)(1-K_1)\right)=R(1-K_1) \\ S_{DG_{ss1}}&=R\,\frac{1-K_1}{K_1+K-K_1K} \end{aligned} \tag{4.14}$$

And the second steady state strength for the default group is given by equation (4.15).

$$\begin{aligned} S_{DG}(t+1)&=S_{DG}(t+1)(1-K_1)(1-K)+R \\ S_{DG_{ss2}}&=\frac{R}{K_1+K-K_1K} \end{aligned} \tag{4.15}$$

As the reward comes to the set at every other iteration, the steady state total strength of the default group oscillates between these two values. For a very small K_1, the two values are close to each other.

4.4.2 Exception classifiers. Following the same approach and considering the exception classifiers as one group, we can also derive the steady state equations for the total strength of exception classifiers.

$$S_{Ei}(t+1)=\frac{2^n-1}{2^n}S_{Ei}(t)\left(1-K_2\right)+\frac{1}{2^n}S_{Ei}(t)\left(1-K\right)+R\frac{S_{Ei}(t)}{\sum_{j=1}^{M_t}S_j(t)} \quad (4.16)$$

Where $S_{Ei}(t)$ is the strength of an exception classifier *i* at the t^{th} iteration, *n* is the number of select bits in the input (i.e. 2 for 6 mux, 3 for 11 mux etc). Again, considering all exception classifiers in group, the fraction term in the equation adds up to unity and we get simplified expression for the total group strength given in equation (4.17).

There are a total of 2^n classifiers in the exception group and only one receives a reward at a time. Hence the total strength for the group at the $(t+1)^{th}$ iteration is given by equation (4.17).

$$S_{EG}(t+1)=\frac{2^n-1}{2^n}S_{EG}(t)\left(1-K_2\right)+\frac{1}{2^n}S_{EG}(t)\left(1-K\right)+R \quad (4.17)$$

The first term on the right in equation (4.17) accounts for exception classifiers in the group that do not match to the given input, and the second term accounts for instances of the exception classifier that matches to the current input. Of a total of 2^n exception classifiers in the group, instances of only one exception classifiers matches to a given input and will be part of the advocate list and incur all forms of tax and bid.

There is no reward at $(t+2)$ and hence the total strength is given by equation (4.18).

$$S_{EG}(t+2)=S_{EG}(t+1)(1-K_2) \quad (4.18)$$

And the total strength at (t+3) is governed by equation (4.19).

$$S_{EG}(t+3)=\frac{2^n-1}{2^n}S_{EG}(t+2)(1-K_2)+\frac{1}{2^n}S_{EG}(t+2)(1-K)+R \tag{4.19}$$

After substitution and rearranging terms in equation (4.19), the group strength for the exception classifiers at the $(t+3)^{th}$ iteration is given by equation (4.20).

$$S_{EG}(t+3)=S_{EG}(t+1)\left(\frac{2^n-1}{2^n}(1-K_2)^2+\frac{1}{2^n}(1-K)(1-K_2)\right)+R \tag{4.20}$$

Again at steady state,

$$\begin{aligned} S_{EG}(t)&=S_{EG}(t+2)\equiv S_{EGss1}\\ S_{EG}(t+1)&=S_{EG}(t+3)\equiv S_{EGss2}\end{aligned} \tag{4.21}$$

Solving the first equality in equation (4.21) for steady state value, we get

$$S_{EG}(t)=\left(\frac{2^n-1}{2^n}S_{EG}(t)(1-K_2)+\frac{1}{2^n}S_{EG}(t)(1-K)+R\right)(1-K_2) \tag{4.22}$$

After substitution and rearranging terms in equation (4.22), we get

$$S_{EGss1}=R\frac{1-K_2}{1-\frac{2^n-1}{2^n}(1-K_2)^2-\frac{1}{2^n}(1-K)(1-K_2)} \tag{4.23}$$

And from the second equality in equation (4.21), we get equation (4.24).

$$S_{EG}(t+1)=S_{EG}(t+1)\left(\frac{2^n-1}{2^n}(1-K_2)^2+\frac{1}{2^n}(1-K)(1-K_2)\right)+R \tag{4.24}$$

After solving and rearranging terms in equation (4.24), the second steady state value is given by equation (4.25).

$$S_{EGss2}=\frac{R}{1-\frac{2^n-1}{2^n}(1-K_2)^2-\frac{1}{2^n}(1-K)(1-K_2)} \tag{4.25}$$

The steady state value of the total strength of exception classifiers oscillates between the two values given in equations (4.23) & (4.25). Usually, the existence tax is very small (i.e. $1-K_2 \cong 1$) and hence the two values are very close to each other. The optimum values of the

simulation parameters are given in Table 4.1. The mutation and crossover probabilities are varied during the experiments and the best results are obtained for all multiplexer problems using the values given in the Table 4.1.

Table 4.1

Simulation parameters with their optimum values

Parameter	Value			Meaning
	6-mux	11-mux	20-mux	
Pop size	200	400	800	Number of classifiers
C_{ext}	0.001	0.001	0.001	Existence tax
C_{oh}	0.005	0.005	0.005	Overhead tax
C_{bid}	0.1	0.1	0.1	Bid coefficient
Px	0.65	0.65	0.65	Probability of crossover
Pm	0.008	0.008	0.008	Probability of mutation
EBID	0.1	0.1	0.1	Ebid constant

CHAPTER 5

FPN for Dynamic Clustering

In this era of huge amount of data, clustering has a pivotal role in data mining with innumerable applications in a wide range of fields. The goal of clustering is to partition data in to categories or clusters so that objects in the same cluster are similar in a certain type of measure and different from those of other clusters (Nasraoui and Krishnapuram, 2000; Streichert et al., 2004; Zhang et al., 2006). Generally, there are two broad categories of clustering techniques: hierarchical and partitional. Hierarchical clustering techniques can be further divided in to agglomerative (begins with each entry as a cluster center and proceeds successively merging smaller clusters in to larger one) and divisive analysis (starts with one big cluster and proceeds by splitting the larger cluster). On the contrary, the partitional clustering techniques directly decompose the data set in to several disjoint clusters based on a defined criterion. K-mean clustering is one of the most known partitional clustering techniques (Sheng et al., 2004). This chapter explores the application of the proposed niching technique for clustering using both synthetic and real data.

The intent here, however, is not to compare its performance with or claim an improvement over a specific clustering algorithm. We want to demonstrate how FPN can be applied for clustering of multi-dimensional data. Most clustering algorithms rely on a priori knowledge (e.g. number of clusters, the distribution of the data etc) on the data. For instance, the K-mean algorithm assumes a predetermined number of clusters and its performance is dependent on the cluster initialization and as a result it may get trapped in local optima (Sheng et al., 2004). The simulation results show that the developed niching technique can be applied for dynamic clustering when such a priori information is not available ahead. FPN utilizes the robust global

search ability of GA to dynamically locate cluster centers without making any a priori assumption about the distribution of the data.

5.1 Mapping Multi-modal Optimization to Cluster Discovery

As emphasized in earlier chapters, the use of niching enables GAs to evolve a diverse set of populations and hence making them suitable to discover multiple optima in the fitness landscape. With proper formulation of the objective function, a clustering problem can be mapped in to the optimization of a multi-modal function with unequal peaks. The location of the unequal peaks corresponds to the cluster centers in the feature space. The highest peak values of the fitness landscape represent dense cluster areas while the lower peaks map to sparse cluster centers. The number of peaks of the multi-modal function corresponds to the number of cluster centers. The task of the GA is then to search for the location of optimum points which represent cluster centers. The solution space for possible cluster centers consists of *n*-dimensional prototype vectors. For clarity and computational speed, a real valued GA implementation is preferred over binary GA (Giráldez et al., 2003). Hence, an individual in the population is a sequence of *n* real valued numbers representing a candidate cluster center.

Figure 5.1 shows how a multimodal function of one variable can be mapped in to a clustering problem in 1-dimensional feature space. The locations of the optimum points in the multimodal fitness landscape (C_1 to C_5 in the figure) represent the values of the cluster centers. The variation in fitness values of the five peaks accounts for the distribution of the data (i.e. highest peaks represent dense areas in the data and lower peaks correspond to clusters of sparse data). Similarly, a higher dimension multimodal function can be mapped in to clusters of a higher dimensional feature space data. For instance, optimization of a two dimensional multi-modal function can be mapped in to finding cluster centers in a 2-D feature space.

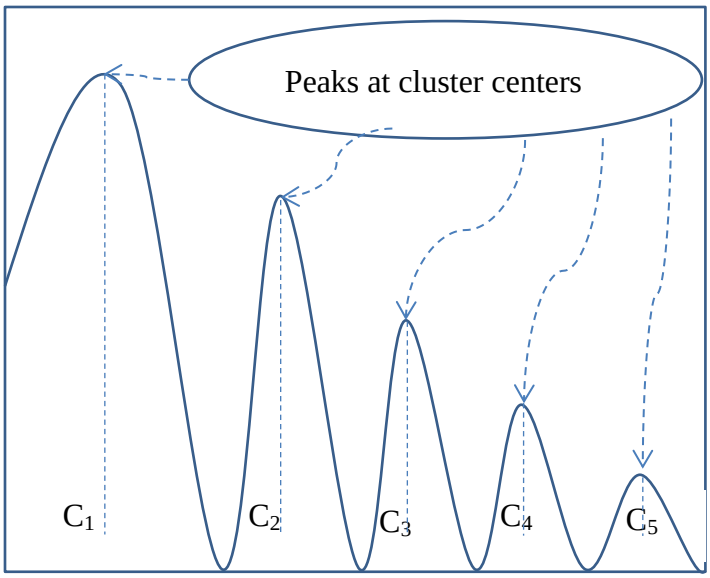

Figure 5.1. Mapping of a multimodal optimization problem to a clustering problem in 1-D.

Figure 5.2 shows a scenario where a two dimensional multimodal objective function defined over the data points can be map to clustering of data in 2-D feature space. For a one to one correspondence between the locations of the optimum points and cluster centers, the objective function needs to have local optima at or near the cluster centers. A density based objective function satisfies this criterion (Chang et al., 2010; Duan et al., 2007).

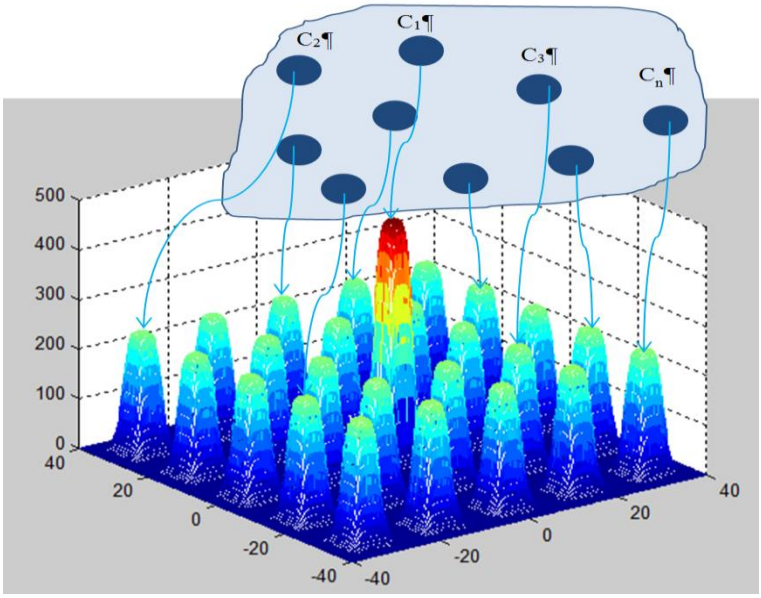

Figure 5.2. Mapping a 2-D multimodal function in to a clustering problem in 2-D.

5.2 Formulation of Fitness Function

It is generally assumed that dense areas of a feature space are identified as clusters. Defining a fitness function that takes this in to account is very crucial to the success of GA for discovering actual cluster centers in the data. Density based fitness functions are commonly used in literatures (Duan et al., 2007; Sander et al., 1998; Sheng et al., 2004; Zhang et al., 2006). In this work, a density based fitness function similar to the one given in (Chang et al., 2010) is applied. Unlike the fitness function defined in (Duan et al., 2007; Sander et al., 1998), the formulation of the fitness function used in our implementation does not require a fixed value of the neighborhood radius and the size of a cluster.

Let $X=\{x_1, x_2, \ldots, x_N\}$ represent a D-dimensional data and K be the size of the population which represents the initial number of clusters that prevail in the data. The goal is to find all cluster centers (C_i) that maximize the total similarity given by equation (5.1).

$$J(C)=\sum_{i=1}^{K}\sum_{j=1}^{N}\left(\exp\left(-\frac{\left\|x_j-C_i\right\|^2}{\beta}\right)\right)^{\alpha} \tag{5.1}$$

Where, α is a constant that determines the shape of the density function, $C= (C_1, C_2, \ldots C_K)$ represent the cluster centers, N is the number of instances and μ and β are the mean and variance of the data and are given by equation (5.2). The value of α varies with the data and its optimum value is determined using correlation comparison algorithm given in (Yang and Wu, 2004). The mean and the variance for the data are constant and hence computed only once before the start of the evolution.

$$\beta=\frac{\sum_{j=1}^{N}\left\|x_j-\mu\right\|^2}{N}$$
$$\mu=\frac{\sum_{j=1}^{N}x_j}{N} \tag{5.2}$$

Each individual in the population represents a candidate cluster center. The population is initialized to cluster candidates randomly selected from the data. The fitness of an individual (candidate cluster center) is computed using equation (5.3).

$$f(c)=\sum_{j=1}^{N}\left(e^{-\frac{\|x_j-c\|^2}{\beta}}\right)^{\alpha} \tag{5.3}$$

An individual in a dense area (surrounded by many data points in the search space) will have a higher fitness value. Initially, the number of clusters is equal to the population size. Hence, the goal of the niching technique is to evolve the population in to stable subpopulations that converge at the location of the cluster centers. In other words, if K actual cluster centers prevail in the dataset, we expect the emergence of K stable subpopulations at the end of evolution. The niche masters (the individual with the highest fitness in the group) of each subpopulation represents the final cluster centers.

5.2.1 Crossover operator. The crossover operator enables the GA to exploit already discovered candidate solutions (i.e. refining solutions). High crossover rate can lead to premature convergence, for instance getting trapped in a local optimum. If two parents (c_1 and c_2) are selected using a roulette wheel selection mechanism, then the two offsprings (c'_1 and c'_2) generated by the crossover operator are determined using equation (5.4).

$$\begin{aligned} c'_1 &= c_1 + r(c_1 - c_2) \\ c'_2 &= c_1 - r(c_1 - c_2) \end{aligned} \tag{5.4}$$

Where, r is a uniformly distributed random number over [0, 1].

5.2.2 Mutation operator. The mutation operator facilitates exploration. It helps the GA to get out of a local optimum and discover new regions in the search space. High mutation rate is undesired as it turns the GA in to a random search. Hence the mutation rate is usually set to a

very small value. A uniform neighborhood mutation is applied at each chromosome with a probability of p_m to generate the mutated value of a candidate cluster at the corresponding location. Let d^q_{min} and d^q_{max} represent the minimum and maximum value of the data along the q^{th} dimension respectively. Then the mutated value (d^q_m at the q^{th} dimension of a cluster center with value d^q is given by equation (5.5).

$$d_m^q = d^q + r_m R(d_{max}^q - d_{min}^q) \tag{5.5}$$

Where, r_m and R are uniformly distributed random numbers over the interval (0, 1) and [-1, 1] respectively.

The simulation is repeated R times using different seeds for the random number generator and the mean and standard error are computed using equations (5.6) and (5.7).

$$\langle v(t)\rangle = \frac{1}{R}\sum_{r=1}^{R} W_{r,t} \qquad \forall t \in (1, \ldots G) \tag{5.6}$$

$$\mathrm{E(t)} = \sqrt{\frac{1}{R} * (\frac{\sum_{r=1}^{R} (W_{r,t} - \langle v(t)\rangle)^2}{R\text{-}1}} \tag{5.7}$$

Where, v(t) and E(t) are the mean and standard error at generation t, $W_{r,t}$ is the number of niches discovered at the r^{th} experiment and t^{th} generation, G is the number of generations, R is the number of times the simulation is repeated.

5.3. Algorithmic Description

The FPN based clustering algorithm has the following computational steps. To counterbalance the effect of randomness in GA, the experiment is repeated 30 times using different seeds for the random number generator. The averages of these 30 runs are taken as the final cluster centers of the data.

1. Initialize the population using N randomly selected instances from the data set.
2. Evaluate the raw fitness of each individual in the population using equation (5.3).

3. Apply the dynamic niche identification algorithm (given in chapter 3) to identify the number of niches that prevails in the population.
4. Compute the shared fitness of individuals belonging to the same niche using FPN.
5. Apply real valued GA crossover and mutation given in equations (5.4) and (5.5).
6. Go to step 2 if the stopping criteria is not reached, otherwise proceed to step 7.
7. Select the niche masters of the population as a final cluster center and exit.

Figure 5.3 shows the distribution of the data and the population before the start and at the end of evolution. The figure shows a scenario where the size of the data is 40, population size is 20 and four clusters exist. The initial population is randomly selected from the data and the niche masters of the final population are picked as cluster centers. This particular setup demonstrates how the FPN based clustering algorithm evolves the population from a random initial state to a final state that is uniformly distributed among the cluster centers (i.e. the population divides evenly in to all clusters as expected ideally).

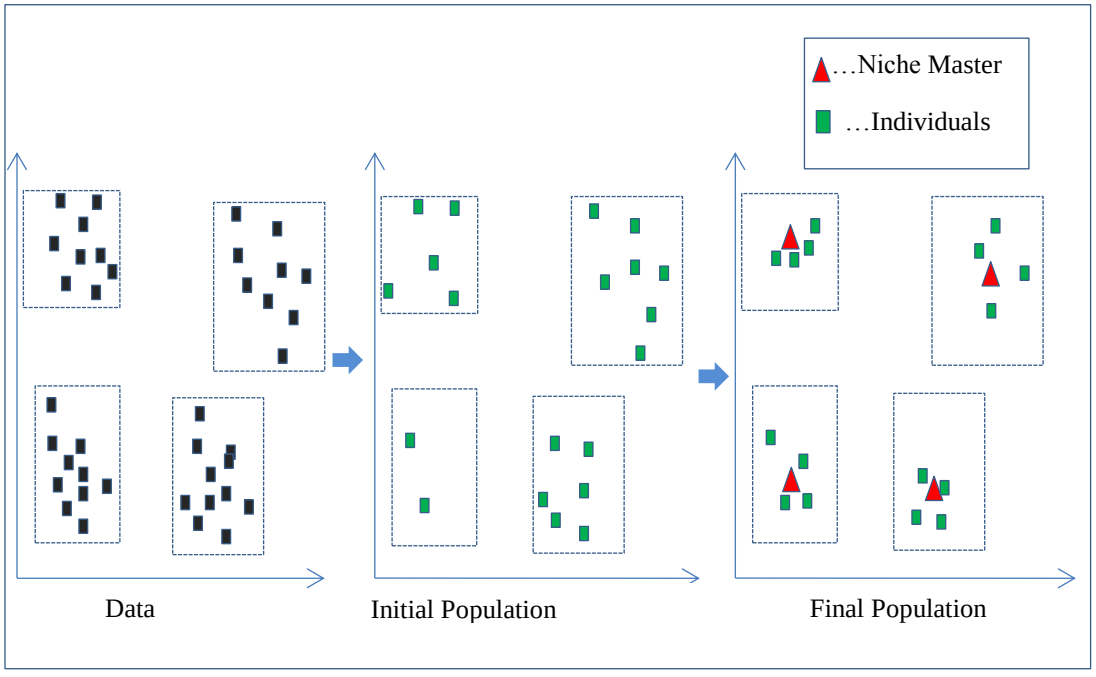

Figure 5.3. The distribution of the population at the start and end of evolution.

5.4. Simulation Results

To test the performance of the algorithm, we used both synthetic and actual data for clustering. The real data is obtained from the machine learning laboratory of the University of California at Irvine (*www.ics.uci.edu/~mlearn/*). Three real clustering data sets (iris, seed and skin) and two synthetic (one with well distributed and another with a sparse data) are used for testing the performance of FPN. The Fisher iris data set has a total of 150 instances of three different types of flowers (class setosa, class versicolor and class virginica). There are 50 instances per class and each instance has four attributes: sepal length, sepal width, petal length and petal width. The seed data set has 210 instances with 7 attributes: area, perimeter, compactness, length of kernel, width of kernel, asymmetry coefficient and length of kernel groove. The data set consists of kernels belonging to three different varieties of wheat: Kama, Rosa and Canadian, each containing 70 instances. The third data set used is the skin segmentation data set. This data set is collected by randomly sampling the R, G, B values from face images of various age, race and gender groups. The total sample size is 245,057 and it has two classes: skin and non-skin samples.

Figure 5.4 shows a data set that consists of 16 clusters. As can be seen from the same figure, the FPN based clustering algorithm discovered all the 16 clusters. In Figure 5.4 and Figure 5.5, the red squares represent the scatter plot of the data, the black circles represent the evolved GA population and the blue shaded circles refer to the niche masters in the final population. The number of niches is equal to the number of clusters that prevail in the data and their value represents the niche masters (cluster centers). This data set has well separated clusters and is relatively easy to identify the cluster centers. For faster convergence at the cluster locations, the population is initialized from the data.

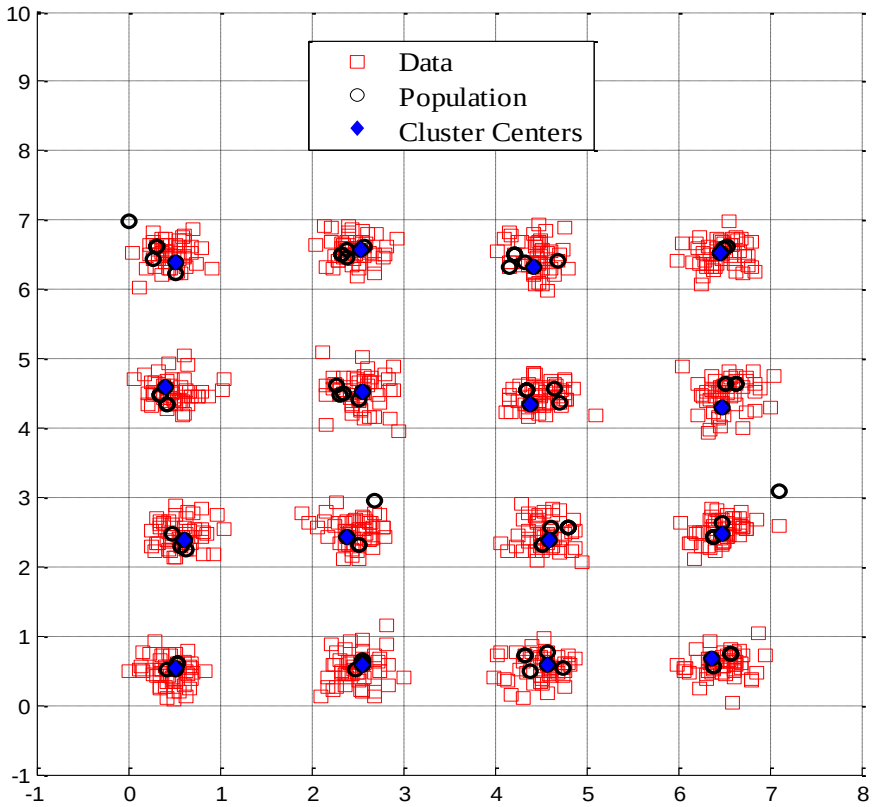

Figure 5.4. Synthetic data with well separated clusters.

Figure 5.5 shows a data set with a variable density of distribution (some regions are highly dense and others are sparse). Since the fitness function is defined based on density, individuals close to highly dense areas will have high fitness, while those close to less dense areas will have lower fitness. From the FPN perspective, this is equivalent to optimizing a multimodal function with variable peaks. As can be seen from the final population distribution in Figure 5.5, FPN discovers all the nine clusters in the data despite a significant difference in the density of the clusters. This is due to the fact that FPN is density unbiased as it considers both high and low peaks of the multimodal function equally important. A GA search based on traditional fitness sharing favors dense areas and as a result there is a high chance of missing clusters with sparse data. This also makes the FPN based clustering less sensitive to outlines in the data.

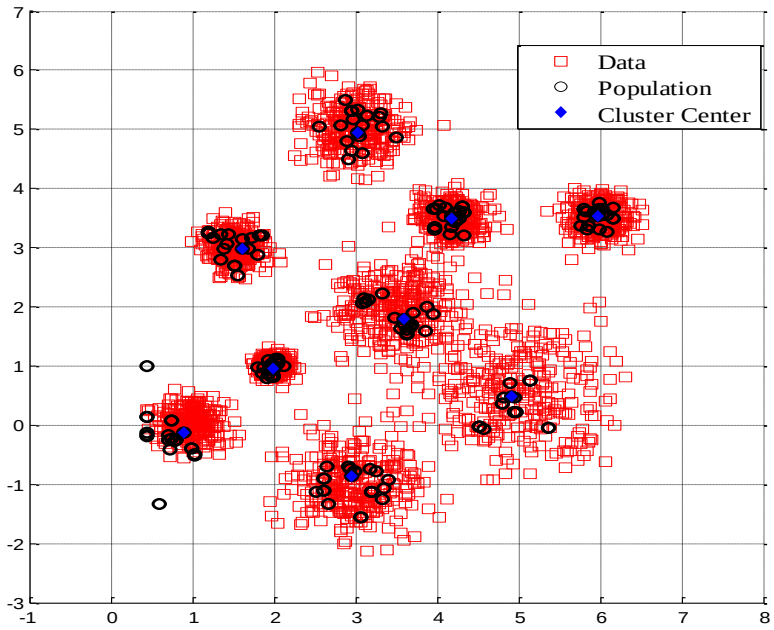

Figure 5.5. A synthetic data with less dense clusters.

The remaining data sets (iris, skin and seed) are higher dimensional and cannot be visualized in 3-D. Instead, a simulation result for the average number of niches and the standard error is depicted in Figure 5.6 and Figure 5.7 respectively. The number of niches corresponds to the number of cluster centers in the data. As can be seen, FPN discovers 2 clusters for the iris and skin data set and 3 clusters for the seed data set. The experiment is run 30 times and the number of niches discovered at each generation is averaged over the size of the experiment. The standard error measures the deviation of the number of niches (cluster centers) from the average number of niches as generation goes on. A constant value indicates that there is no variation in the number of niches discovered at each generation.

There is a high variation on the number of clusters discovered for the iris data set (see Figure 5.7) as compared to the other data sets. This is so, because the iris data contains only two linearly separable classes: one of the clusters contains Iris setosa, while the other cluster contains both Iris virginica and Iris versicolor. For the other data sets, the standard error or deviation is

constant, in accordance with the average number of clusters discovered by the FPN algorithm for those data sets as shown in Figure 5.6.

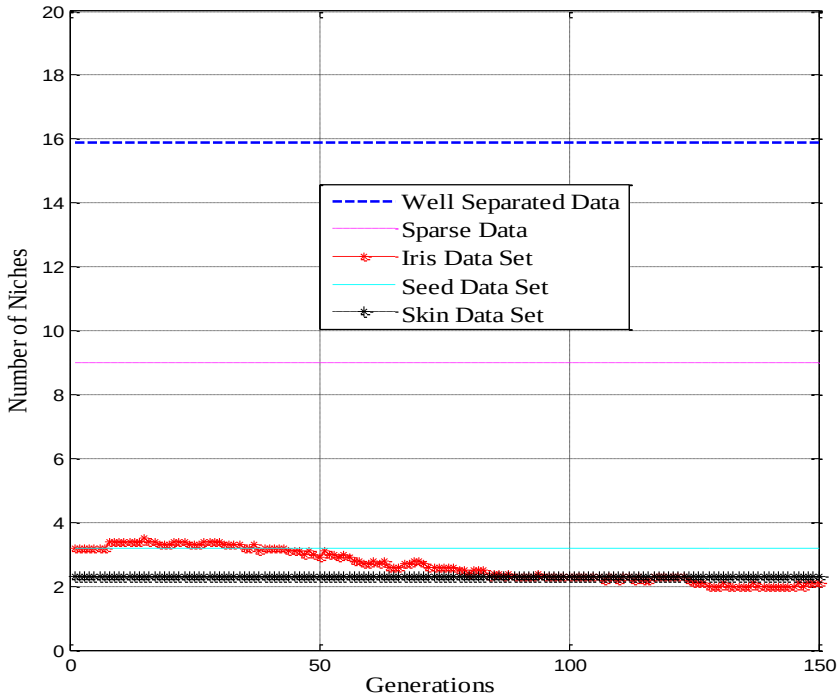

Figure 5.6. The average number of clusters discovered using FPN for the five data sets.

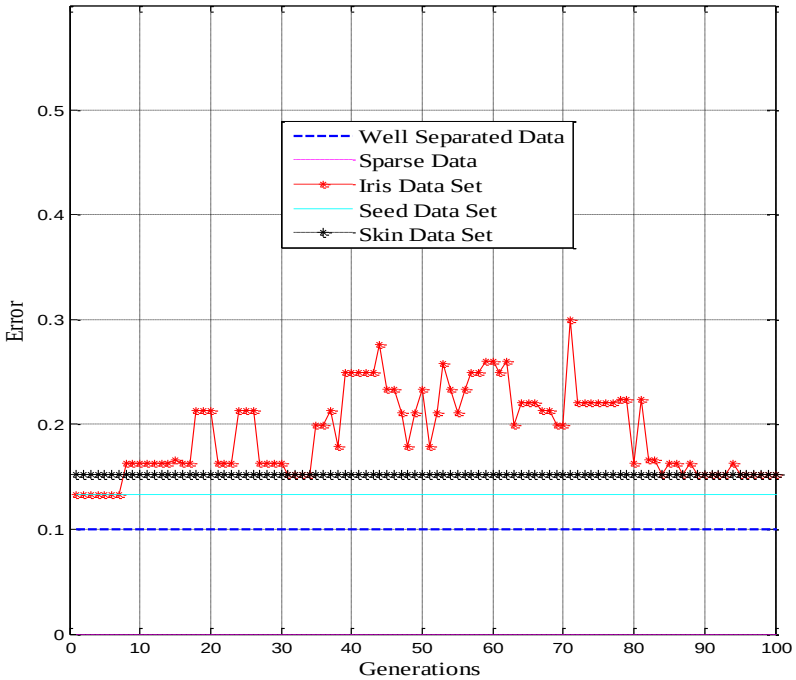

Figure 5.7. The standard error plot for two synthetic and three real datasets.

CHAPTER 6

Results and Discussion

This chapter has two major sections. The first section discusses the simulation results for FPN technique introduced in chapter 3 of this dissertation. A comparison with other niching schemes is made using benchmark multimodal functions from literatures. The second section extends the application of the niching technique developed for evolving hierarchical cooperation in classifiers. A comparison on whether other existing sharing techniques can lead to the formation of default hierarchy in LCS and whether they can sustain it after it has emerged, is also made in this section.

6.1. Results for FPN

The FPN scheme is applied for the optimization of multimodal functions both with equal and unequal peaks and its performance is compared with the traditional fitness sharing scheme. Simulation is carried out with various population sizes to investigate how the niching techniques behave as the population size varies. The simulations are run 20 times and the average values are plotted. As the goal of a niching technique is to discover multiple peaks in parallel, one possible way of measuring system performance is displaying the number of peaks discovered as the search process goes on. In Figure 6.1, a performance comparison for a multimodal function having five equal peaks ($F_a(x)$) is shown. As can be seen from this figure, there is no significant difference in performance between the two algorithms for this function. For instance, for a population size of 50, both algorithms discovered almost all the five peaks (see Figure 6.1). This result is expected as FPN essentially degenerates in to traditional fitness sharing for multimodal functions having equal peaks. However, for multimodal functions having unequal peaks, there is a significant difference in performance (see Figure 6.2 and Figure 6.3). The first is for a function

with a small fitness ratio among the different peaks ($F_b(x)$ function). For this scenario, FPN has a reasonably fair performance even at a small population size. For a population size of 30, the traditional niching scheme discovered nearly 75% of the peaks whereas FPN discovered about 95% of the peaks on average. As the population size increases, there is an improvement in performance of both techniques. For a population size of 50, both algorithms discovered almost all of the peaks. In the simulation results (both tables and figures), TFS refers to the traditional fitness sharing scheme, FPN stands for the fitness proportionate niching and PS is the population size.

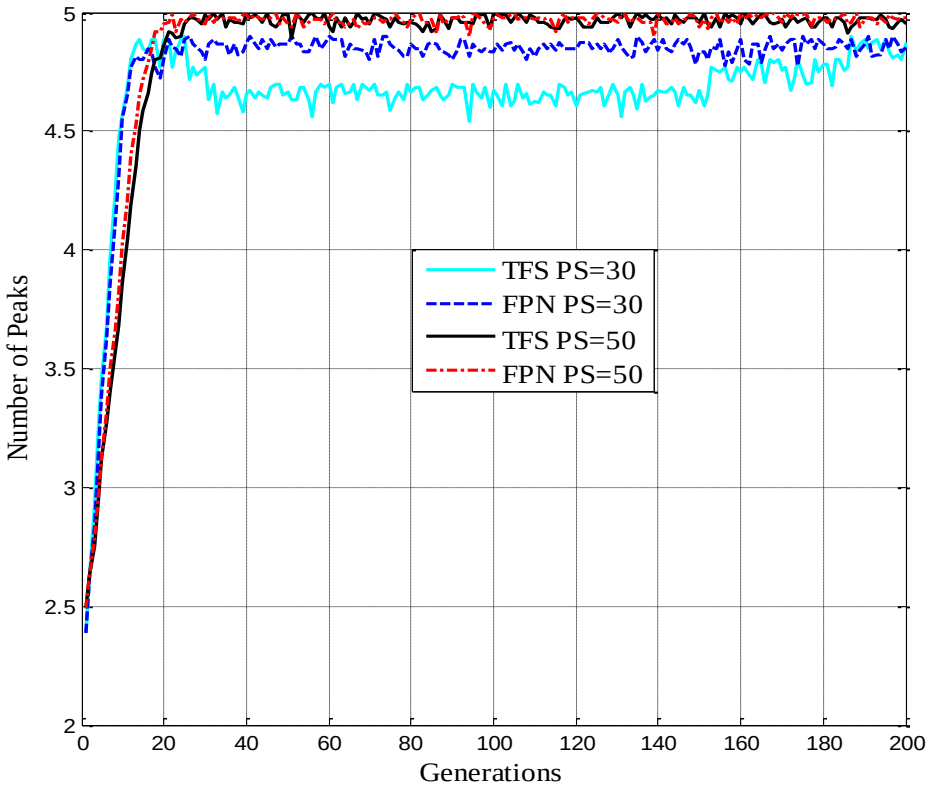

Figure 6.1. Number of peaks discovered for $F_a(x)$ out of a total of 5 peaks.

Figure 6.3 displays the simulation result for $F_c(x)$ for various population sizes. As can be seen from the first subplot in Figure 6.3, the traditional fitness sharing technique discovers only the highest peak (only 1 peak out of a total of 5 peaks) while FPN discovered almost all of them.

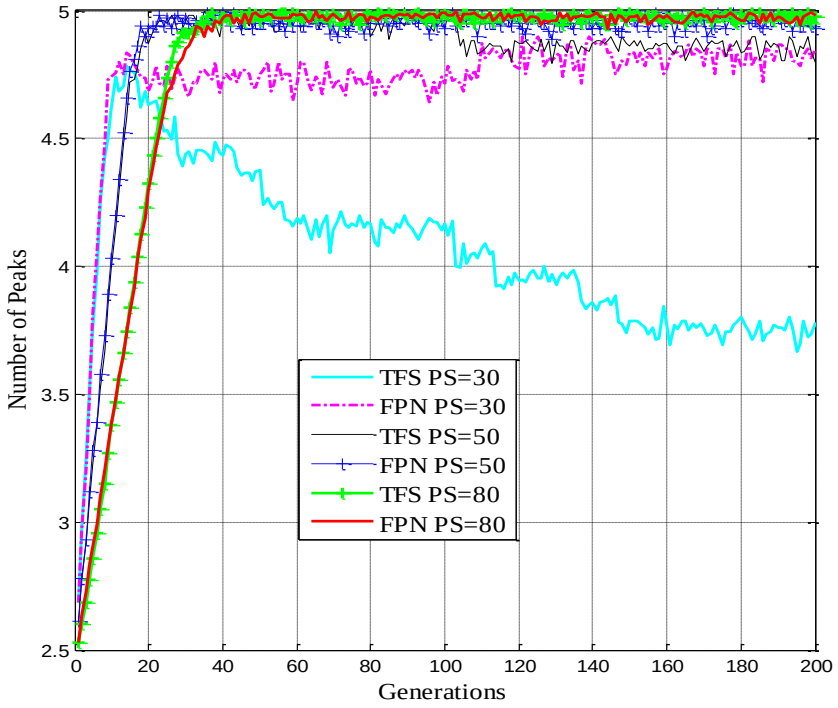

Figure 6.2. Number of peaks discovered for $F_b(x)$ out of a total of 5 peaks.

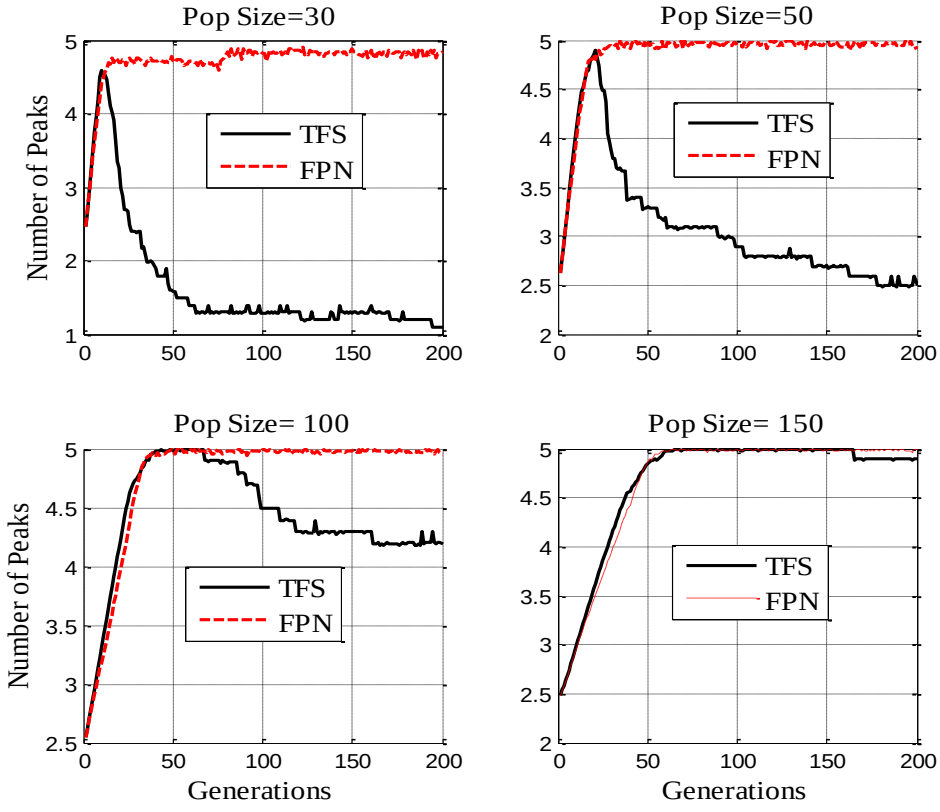

Figure 6.3. Number of peaks discovered for $F_c(x)$, out of a total of 5 peaks.

The results for $F_c(x)$ show how the traditional niching technique is sensitive to the difference in fitness of the peaks. To discover all the peak locations, it requires a very large population size which depends on the fitness ratio at the highest and lowest peaks. The performance dependency of TFS on the population size was discussed in chapter 3. In this particular simulation, the traditional niching technique requires a population size of 150 to discover all the peaks as compared to 50 or lower population size for the FPN scheme.

Figure 6.4 and Figure 6.5 demonstrated the performance of FPN and TFS for the more complex multimodal functions ($F_d(x)$ and $F_e(x)$). $F_d(x)$ has ten unevenly distributed equal peaks whereas, the shekel foxhole function ($F_e(x)$) has 25 uniformly distributed equal peaks. As can be seen in the results, FPN outperformed TFS in discovering the peaks.

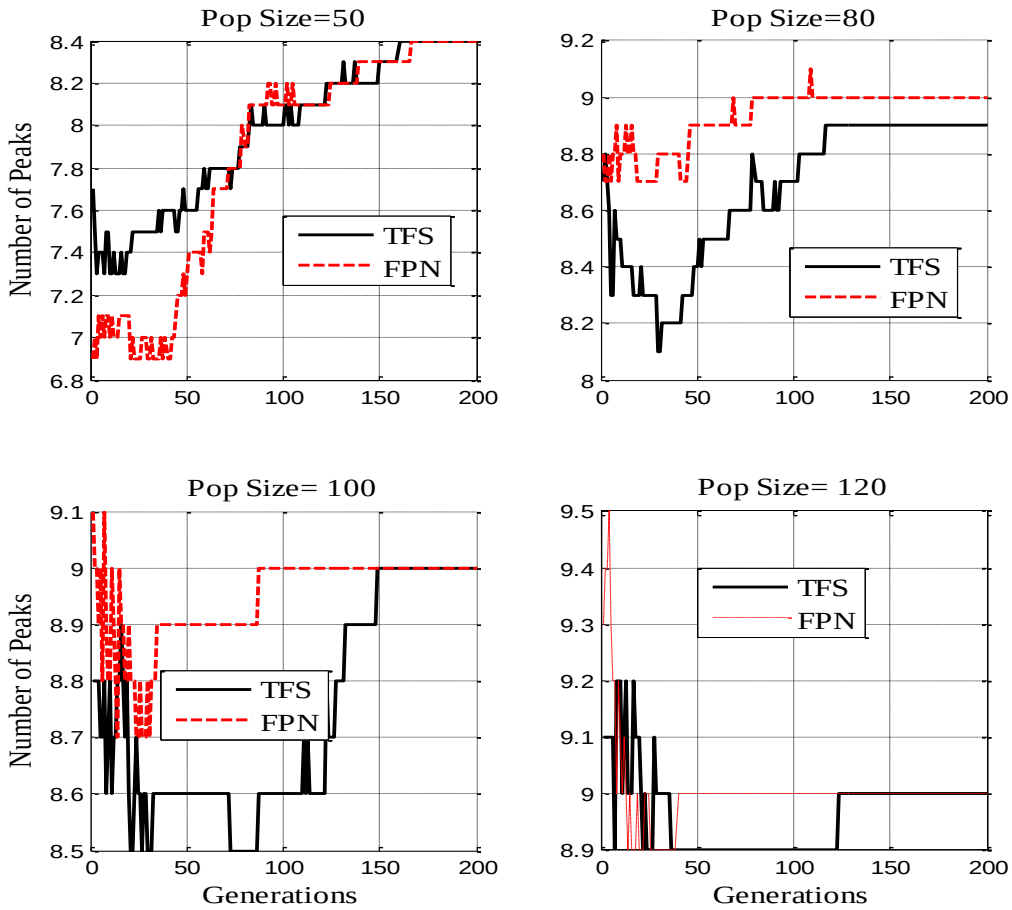

Figure 6.4. Number of peaks discovered for $F_d(x)$, out of a total of 10 peaks.

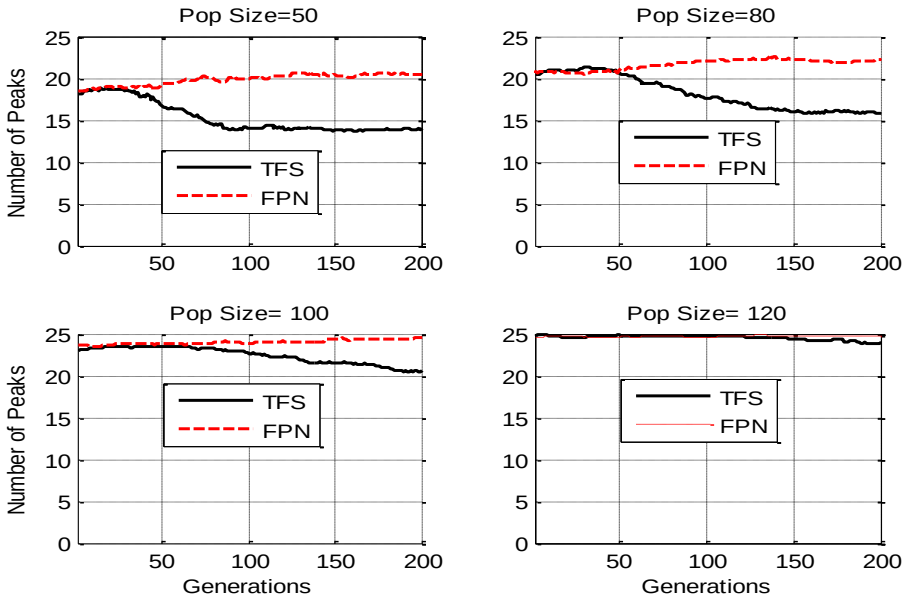

Figure 6.5. Number of peaks discovered for $F_e(x)$, out of a total of 25 peaks.

Another way of measuring the performance of a search technique is to observe the population distribution along the various peaks. Tables 6.1 and 6.2 show the distribution of the population among the various peaks for $F_b(x)$ and $F_c(x)$ test functions. The values portrayed in the tables are the average number of individuals at each of the five peaks over 20 runs.

Table 6.1

Population distribution at the five different peaks for $F_b(x)$

PS		Peak1	Peak2	Peak3	Peak4	Peak5
30	TFS	10	9.7	7.1	3	0
	FPN	7.3	6.8	6.7	5.3	3.6
50	TFS	15.9	14.4	10.9	5.7	2
	FPN	11	10.9	10.4	9.7	7.3
80	TFS	24.6	23.1	17	10.3	4.6
	FPN	18	16	16	15	15

As can be seen from both tables, FPN tends to distribute the population among the various peaks uniformly irrespective of the fitness difference among the peaks. In Table 6.2, for instance, TFS discovers only one of the five peaks using a population size of 30. But FPN discovered all the five peaks using the same population size.

Table 6.2

Population distribution at the five different peaks for Fc(x)

PS		Peak1	Peak2	Peak3	Peak4	Peak5
30	TFS	29.8	0.2	0	0	0
	FPN	7.9	6.9	6.3	5.5	3
50	TFS	47.1	2	0.7	0.1	0
	FPN	12.1	10.7	9.7	9.3	7.5
100	TFS	88	5.7	4.5	1.4	0.2
	FPN	21.6	20.6	20	19.9	17.1
150	TFS	128.4	9.8	7	3.5	1.3
	FPN	31.9	31.5	30.5	28.9	26.9

6.2. Results for LCS

For the sake of comparing results with previous research work, the proposed algorithm was applied to the multiplexer problem (Wilson's Boole function). Simulations were done on the 6-multiplexer problem and to show the scalability and consistency of the algorithm, the experiment is extended to higher multiplexer problems (11 & 20 multiplexers). For all simulations, classifiers were initialized to an initial strength of 100. The payoff for a correct decision (R) by the system was set to 1000 while absence of a reward was considered as

punishment for a wrong response. The specificity value for the default classifier was set to 0.1 in the case of 6-mux, 0.08 for 11-mux and 0.01 for the 20-mux. The values for the initial strength, reward and specificity are initialized in accordance with previous research work. The values for the other parameters are given in Table 4.1.

6.2.1 Performance evaluation. The performance of the learning system is measured by the accuracy of its response to a given input. Figure 6.6 shows the percentage of correctly identified environmental inputs by the system and the solution count as a function of the number of epochs. The upper curve represents the percentage of correct decision by the system and the lower curve is the percentage of the population that contains the perfect solution set. For the 6-mux problem, an epoch stands for one complete presentation of the environmental inputs to the system. So an epoch here represents the average system response on the past 64 inputs. In general, for an *n*-bit input representation, there are a total of 2^n different environmental inputs to the system. All simulations (for all 6, 11 & 20- multiplexers) were carried out 20 times using different seeds for the random number generator and the averaged results are portrayed.

As can be seen from Figure 6.6 , the percentage accuracy of the system averaged over the 20 runs is well above 95% after the 100^{th} epoch. The solution count is the percentage of population that contains instances of the perfect solution set (refer to Figure 4.2, left column) averaged over the size of an epoch. For instance, at the 350^{th} epoch 90% of the population contains instances of the perfect solution. The high percentage accuracy and solution count achieved is an indication of how well the system learns its environment. The same simulation was done using a half population size (100 classifiers) and resulted in nearly the same level of accuracy but with a lower percentage of solution count (85%). The effect of varying the mutation and cross over rates was also investigated.

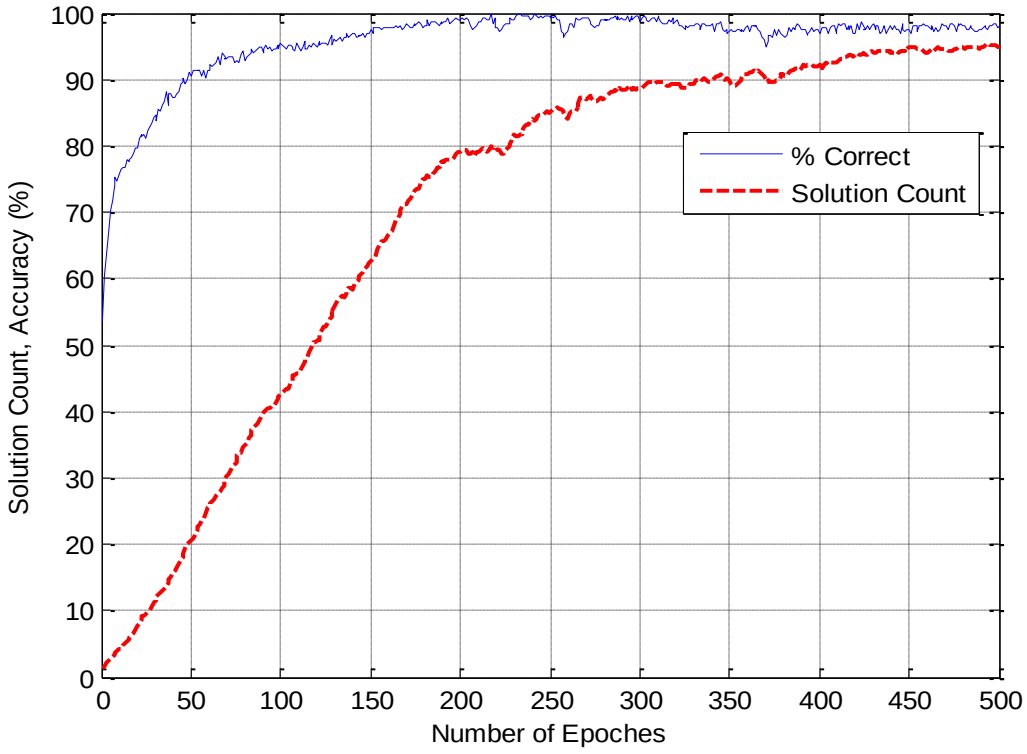

Figure 6.6. System performance for the 6-multiplexer problem.

Figure 6.7 displays the average bid amount of winner classifiers versus the number of epochs for the 20 runs. The bid interaction helps to get an insight on how the strength of the population varies with time. It gives a quantitative clue on the steady state strength of classifiers that influence the system's decision. At the start of the iteration, the population is more likely to be packed with specific classifiers. This is so because in a ternary alphabet system with a random initialization, there is a higher chance ($2/3^{rd}$) for each condition bit to be initialized to a non-hash symbol. But as iteration goes on, the hierarchical set begins to dominate the population resulting in a decline of the bid amount (note that the bid amount is proportional to the specificity of the classifier). This trend is clearly evident from the plots in Figure 6.7, Figure 6.10 and Figure 6.13 having a high pick at the start and declining abruptly until it finally settles to some steady state value.

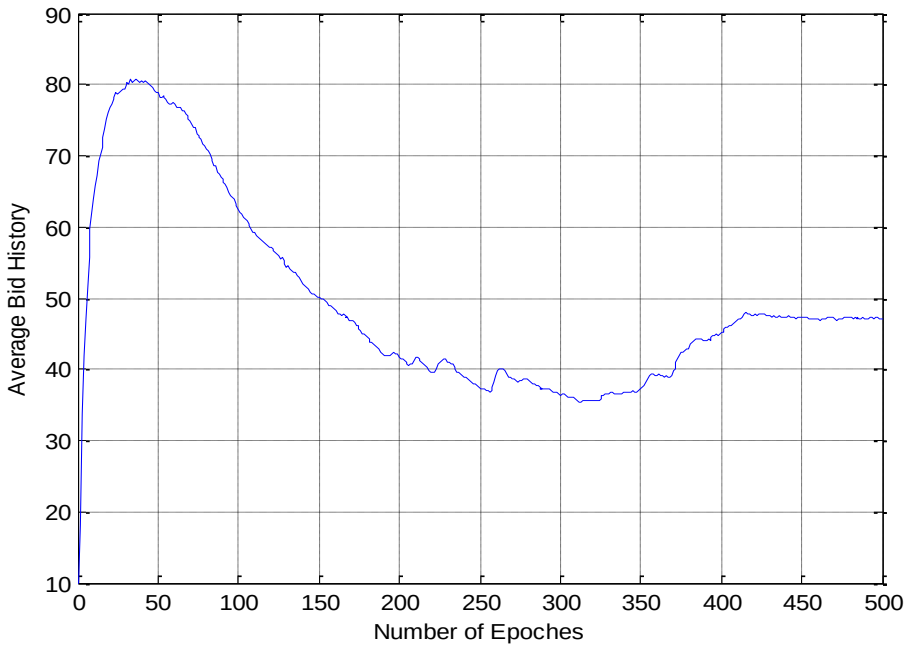

Figure 6.7. The average bid amount of winner classifiers at each epoch for 6-mux.

To check whether the system has evolved to a default hierarchy and its ability to sustain it once formed, a tabular result showing a sorted list of classifiers in the final population with their creation time and numerosity is displayed in Table 6.3 to Table 6.7 for the three multiplexer problems. The entries in the columns (C, A, S, NC and CT) refer to the condition, action, maximum strength, numerosity and the creation time of that specific classifier respectively. The tables show the final population statistics for one particular run.

For the 6-mux, a perfect default hierarchy has been achieved 19 (9 of them with a default of action 0 and the remaining 10 with a default of action 1) times out of a total of 20 runs. In one of the 20 runs, a default hierarchy was also attained but one of the 4 perfect solutions was missed. Table 6.3 shows a scenario where a default of action 0 and the 4 perfect solutions of action 1 dominating the total population. As can be seen from the number of copies in the column of the table, these classifiers comprise 96% of the population (192 instances out of a total population of 200 classifiers).

Table 6.3

A sample pattern of the final population for 6-mux with default of action of 0

No	C	A	S	NC	CT
1	001###	1	2959	29	5
2	10##1#	1	1935	32	3795
3	11###1	1	1310	33	4330
4	01#1##	1	927	33	2460
5	######	0	471	65	1315
6	###0##	0	193	1	23600
7	#####0	0	184	1	3695
8	##0###	0	175	1	18655
9	#0##1#	1	115	1	24475

Table 6.4 displays the same statistics for a default of action 1 and the other 4 perfect solutions of action 0 case. Again, these hierarchical set contains 193 instances of the total population. The creation time (CT column in the tables) gives an insight on the time of emergence of a hierarchical set and whether the learning system was able to maintain it. It was measured in terms of iteration, not epoch. A creation time of 0 indicates that particular classifier was part of the initial population. In one iteration, an input was presented to the learning system and its response to it was evaluated. In 500 epochs, there are a total of 32000 (i.e. 500*64) iterations. The time of formation of the default hierarchy can be inferred by looking at the creation times of individual classifiers that comprise the hierarchical set. In Table 6.3 for instance, considering the top 5 classifiers that comprise a default hierarchy, the highest creation

time value is 4330 (nearly at the 9th epoch), which means that the latest classifier that joined the hierarchical set is "11###1/1". And from Table 6.4, the hierarchical set has emerged at iteration 9340 (nearly at the 19th epoch) and survived afterwards. In both cases, the default classifier has the highest numerosity indicating that it was well protected and flourished.

Table 6.4

A sample pattern of the final population for 6-mux with default of action of 1

No	C	A	S	NC	CT
1	11###0	0	3087	25	3335
2	01#0##	0	1419	28	0
3	10##0#	0	1265	35	9340
4	000###	0	481	39	6255
5	######	1	445	66	1490
6	####1#	1	190	1	21690
7	##1###	1	189	1	0
8	###1##	1	185	1	1575
9	#####1	1	182	1	24290

To get a more in depth insight, Figure 6.8 shows the distribution of the population, the variation in strength and the potential bid amount as the learning process continues. Through guidance of the system using a fitness proportionate resource sharing scheme and discovery of new rules by the GA, the learning process evolves the population from a random start to three big subpopulations (the default hierarchical set, the perfect classifiers that are not part of the hierarchy and the rest). In the sub plots the labels DHS, PSND and 'Others' stands for the default

hierarchical set, perfect solutions that are not part of DHS (for 6-mux for instance, these are instances of any of the four classifiers that are not part of the hierarchical set), and classifiers that are neither in the DHS nor in the PSND respectively. The DHS emerges only when the default and all other exception classifiers emerge in the population. Before the formation of the DHS, the exceptions are part of the PSND and the default is part of the 'Others' group. The 'Others' subpopulation group contains a broad variety of classifiers (classifiers with no or several hashes which can be correct or wrong) and usually expected to have a larger size at the start of the epoch for a random initialization.

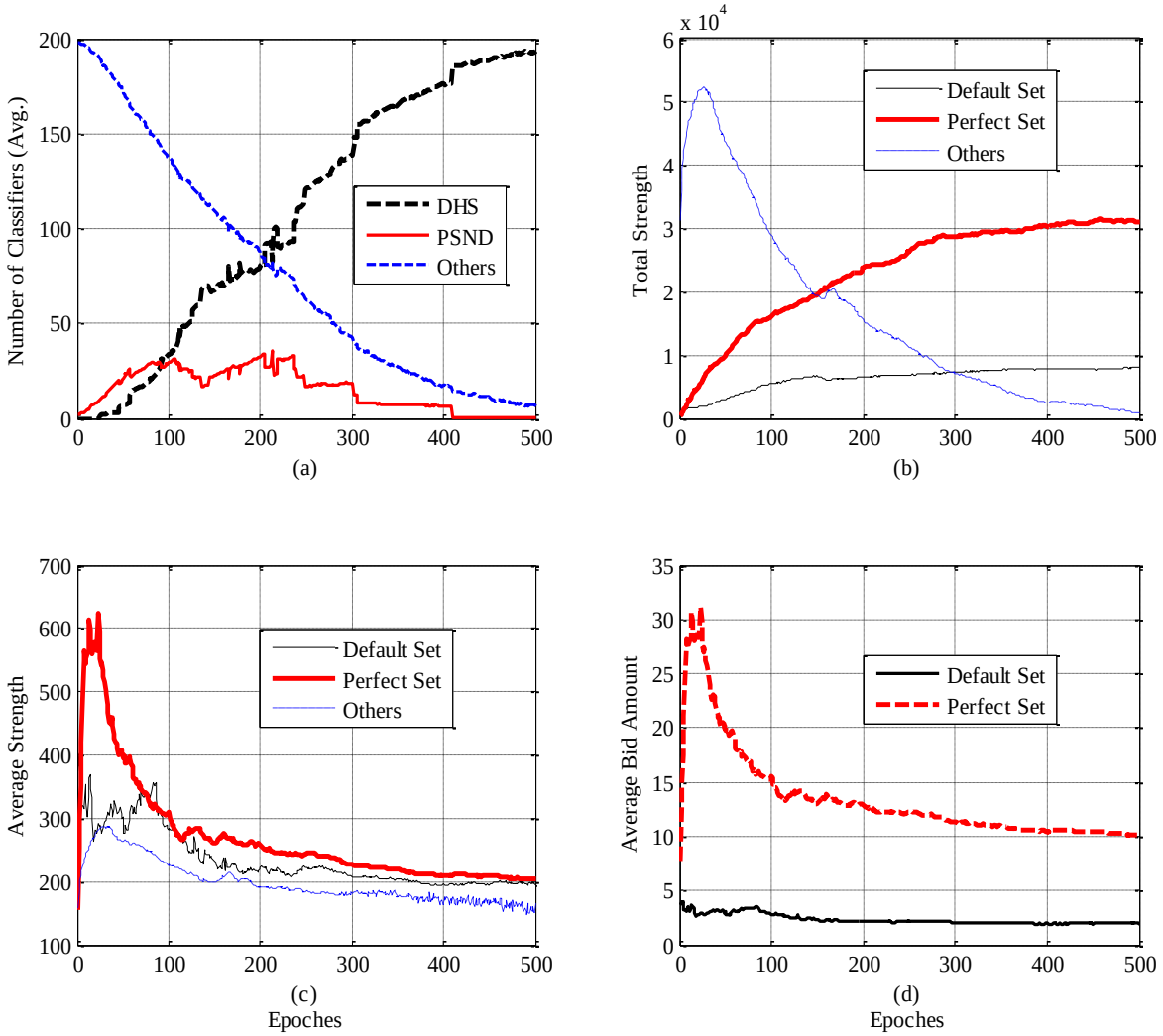

Figure 6.8. Subpopulation distribution for 6-mux using a FPRS scheme.

Figure 6.8 (a) shows the distribution of the population among these three subpopulations. The plot shows at what epoch the hierarchical set has emerged and whether the learning system is able to maintain it for long. As expected, both DHS and PSND are very small at the start and the DHS starts to dominate the population as the learning process continues. Figure 6.8(b) and Figure 6.8(c) show how the total and average strength of classifiers in the default, perfect and 'Others' set vary with epoch. The 'Default Set' contains all instances of a default classifier and the 'Perfect Set' includes all perfect classifiers in the population. Here, we are particularly interested to make a comparison of the strength between the default and exception classifiers. As can be seen from the graph, the perfect set has maintained a higher strength as compared to the default rule. A difference in strength between the default and the exception classifiers helps to attain a steady state bid separation between these sets (see Figure 6.8(d)) and hence allowing the exception classifiers to protect the default when it is wrong. Similar simulations are also done for 11 and 20 multiplexers (see Figure 6.11 and Figure 6.14).

6.2.2 Scalability and robustness. For testing the scalability and robustness of the proposed niching scheme, a similar simulation was also conducted on more complex and large input problems (11 & 20 multiplexers). For these multiplexers, the number of input combinations are very large (2^{11} for 11-mux and over a million for 20-mux, 2^{20}). Hence, considering all the inputs at each epoch would be computationally cumbersome. Instead, only a fraction of randomly selected inputs were used out of the total possible combinations of environmental inputs at each epoch. For instance, for the 11-mux, an epoch represented only 512 (25% of the inputs) iterations or input presentations and the simulation results are shown in Figures 6.9 to 6.11. The upper curve in Figure 6.9 represents the percentage of correct decision by the system; the lower curve is the percentage of the population that contains the perfect solution set.

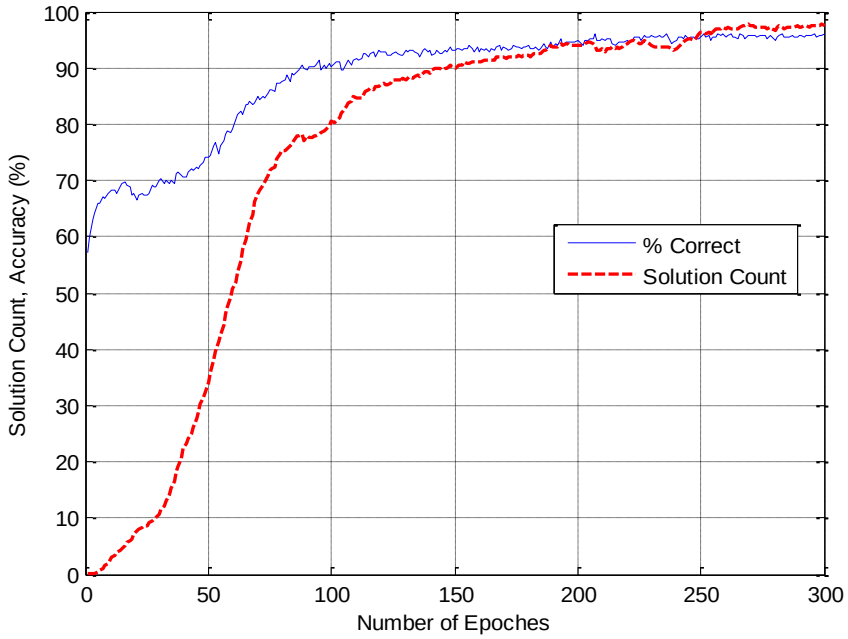

Figure 6.9. System performance for the 11-mux using FPRS scheme.

A bid history plot for the winner classifiers is displayed in Figure 6.10. As can be seen, the bid history settles to a steady state value after some iteration. This is expected because the bid amount is a fraction of the strength of winner classifiers which converges to a certain steady state value after sufficient iterations (see steady state analysis section of Chapter 4).

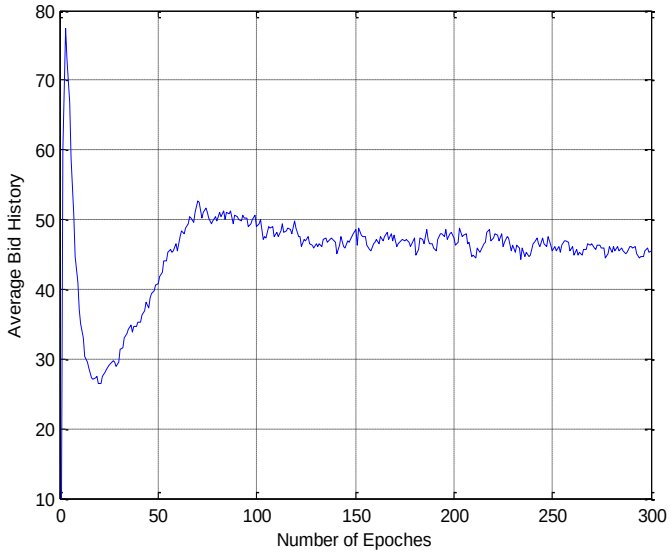

Figure 6.10. The average bid amount of winner classifiers at each epoch for 11-mux.

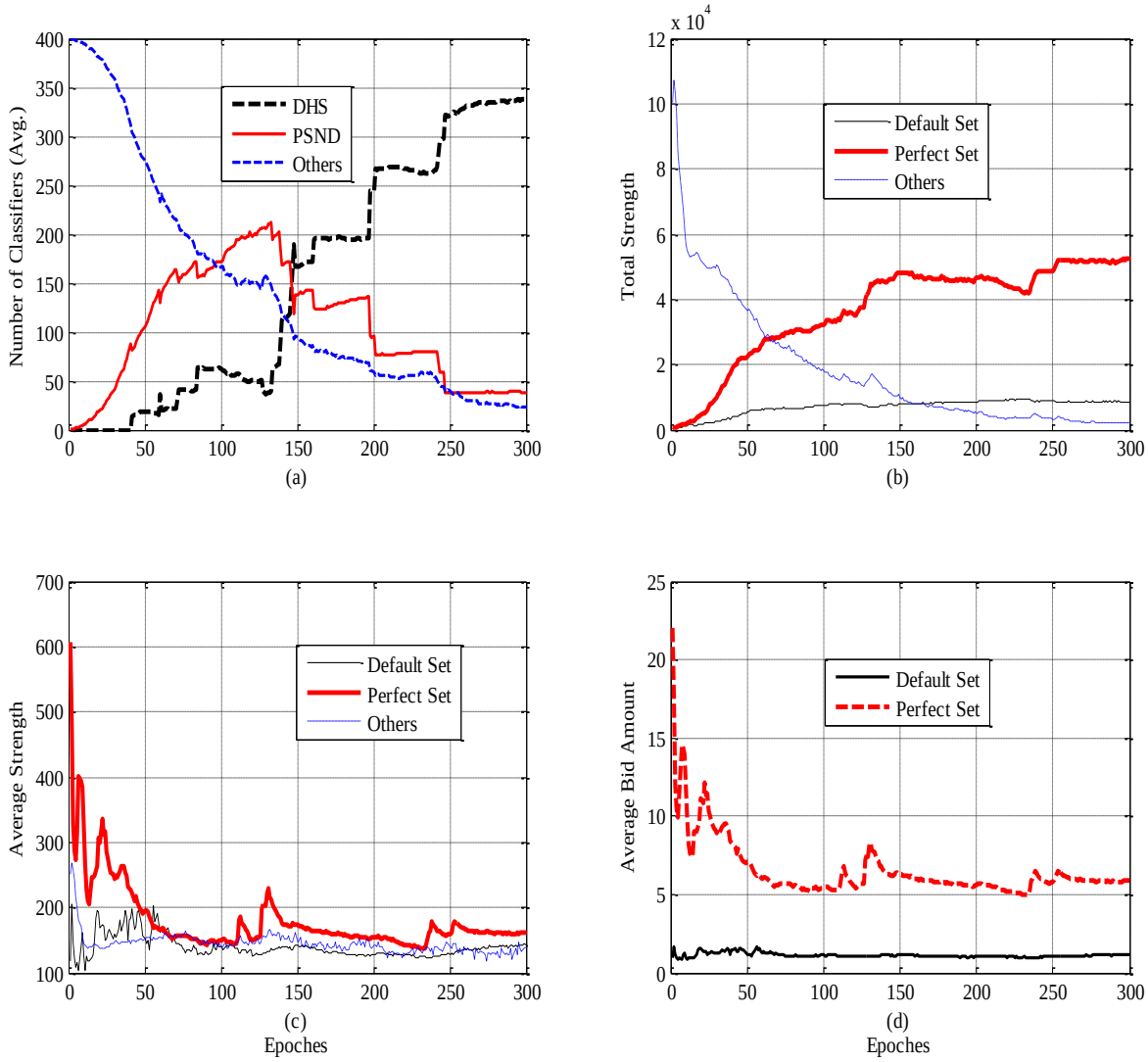

Figure 6.11. Subpopulation distribution for 11-mux using FPRS scheme.

A typical scenario showing the population statistics for the top 10 classifiers at steady state is shown in Tables 6.5 & 6.6. As can be seen from these tables, the hierarchical set comprises of 98% of the population (i.e. 394 instances out of a population size of 400 in Table 6.5 and 392/400 in Table 6.6). From the creation time, it can also be seen that the hierarchy once formed in the learning process is maintained for generations. For instance, observing the top 9 classifiers that comprise a hierarchical set in Table 6.6, the default hierarchy was achieved at the 42000^{th} iteration (about the 82^{th} epoch).

Table 6.5

A sample pattern of the final population for 11-mux with default action of 0

No	C	A	S	NC	CT
1	111#######1	1	5499	15	43880
2	110######1#	1	4851	24	18210
3	010##1#####	1	4281	30	29290
4	101#####1##	1	3010	44	17920
5	100####1###	1	2841	44	13440
6	011###1####	1	2296	44	13620
7	0001#######	1	1750	52	9750
8	001#1######	1	1317	62	8790
9	##########	0	108	79	81530
10	#########1#	0	99	1	153540

Figure 6.12 shows the classification accuracy of the learning system for the 20-mux problem. To reduce computation time, the solution count plot is not included in this simulation. In here an epoch represents 50000 input presentations out of a total of 2^{20} inputs. The plot shows the percentage accuracy of the system over the past 50000 randomly selected environmental inputs averaged over 20 runs. The classification accuracy of the system here is slightly lower (close to 90%) as compared to the 6-mux and 11-mux problems. This is expected taking into account the complexity of the system and the randomness in GA which brings new classifiers into the system that can possibly perturb its performance (i.e. the GA operation can bring inaccurate classifiers for a transient time and degrade the overall system performance).

An LCS system is complex and difficult to fully understand and analyze its dynamics even for the simpler scenario of a stimulus response system. We have conducted different simulations that help to better understand the learning dynamics (emergence of subpopulations, strength variation, bid amount) and give insight on the formation of hierarchically cooperative subpopulations. The average bid history plot shown in Figure 6.13 indicates the steady state bid amount for winner classifiers. As can be seen from the plot, the average bid history finally converged to some steady state value. The simulation results for the 20-mux have also shown the scalability of the sharing technique for achieving a viable default hierarchy of cooperative rules in a competing environment.

Table 6.6

A sample pattern of the final population for 11-mux with default action of 1

No	C	A	S	NC	CT
1	111#######0	0	2892	38	31990
2	001#0######	0	2739	34	42000
3	101#####0##	0	2698	32	31220
4	011###0####	0	2246	37	24080
5	110######0#	0	1913	45	17390
6	0000#######	0	955	44	26790
7	100####0###	0	790	48	12490
8	010##0#####	0	635	48	7750
9	###########	1	261	66	19990
10	########0##	0	223	1	137630

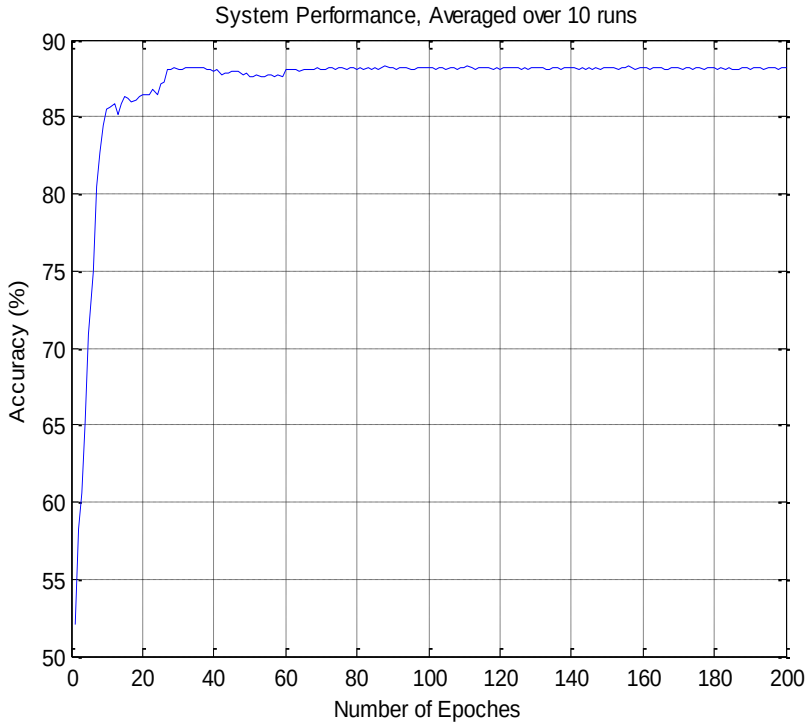

Figure 6.12. The system performance (in percentage accuracy) for 20-mux problem.

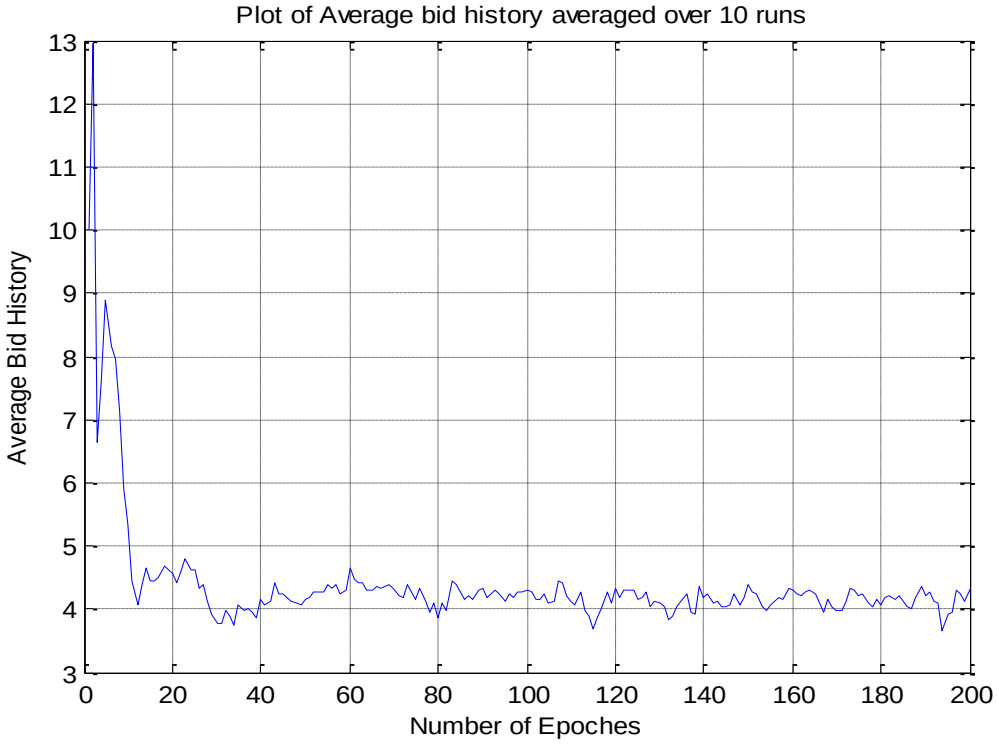

Figure 6.13. The average bid amount of winner classifiers at each epoch for 20-mux.

Figure 6.14(a) shows the distribution of the population along the three clusters. A population size of 1000 was used in this simulation and the default hierarchy has emerged after the 50th epoch. From the average bid amount subplot, it can also be inferred that there is enough bid separation between the default and exception classifiers at steady state.

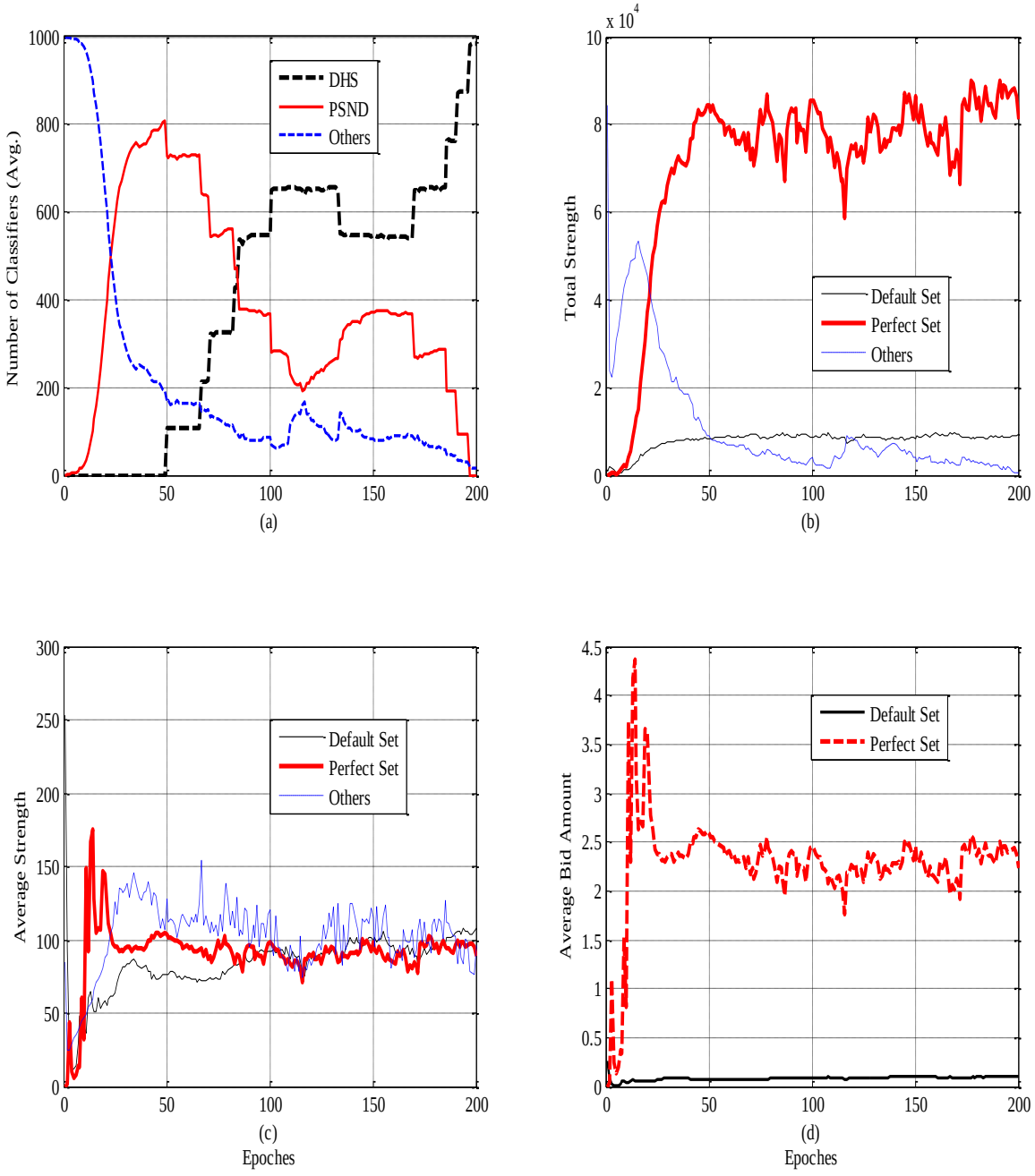

Figure 6.14. Subpopulation distribution for 20-mux using FPRS scheme.

In Table 6.7, the composition of the final population for one typical scenario of default rule of action one is shown. As shown in Table 6.7, the learning system has evolved to a hierarchical set (the top 17 classifiers) nearly after the 146th epoch which dominates the population (99% of the total population) afterwards. In here, a default rule of action 0 (18th row in Table 6.7) also emerged and co-existed with a default rule of action 1. But, this classifier has a lower strength and much fewer numbers of instances as compared to the other default. Also, the other few classifiers emerged nearly at the end of the iteration (see the creation time) due to GA indicating that the hierarchical set has successfully taken control of the system decision.

6.2.3 Control experiments. To make a comparison with other rewarding schemes and verify the soundness effectiveness of the fitness proportionate sharing scheme, we conducted two sets of control experiments using the same simulation set up (parameter values and random seeds) with our technique for the 6 and 11 multiplexer problems. The first set of experiment applies uniform sharing and no sharing techniques on a randomly initialized population. The uniform sharing scheme distributes the total reward R uniformly among all classifiers in the advocate list. This type of sharing is known in literatures as implicit sharing (Horn and Goldberg, 1996; Wilson, 1989). In the no sharing scheme, each classifier in the advocate list receives a full reward R from the environment every time a correct action is predicted. In other words, every classifier in the advocate list receives a constant reward independent of the presence or absence of other classifiers in its vicinity. This type of sharing assumes the resource of a given environmental niche is infinite and can accommodate all the classifiers in the population. Under the selection pressure of the GA used during discovery of rules, this sharing scheme leads to premature convergence as the whole population tends to converge to the location of an early discovered optimum point (Workineh and Homaifar, 2012a).

Table 6.7

A sample pattern of the final population for 20-mux with default action of 1

No	C	A	S	NC	CT (x103)
1	0100####0##########	0	337	59	101
2	0110######0########	0	327	55	167
3	0111#######0########	0	325	58	179
4	1100############0###	0	314	57	106
5	1001#########0######	0	313	58	187
6	0001#0##############	0	312	59	143
7	0011###0###########	0	299	59	219
8	1101#############0##	0	296	61	344
9	0101#####0##########	0	295	59	198
10	1110##############0#	0	295	56	219
11	1010##########0#####	0	294	58	283
12	1111###############0	0	293	58	151
13	1011###########0####	0	289	57	289
14	1000########0#######	0	277	62	215
15	00000###############	0	271	59	164
16	0010##0############	0	252	60	183
17	###################	1	230	57	7341
18	###################	0	228	2	9413
19	1###################	1	106	1	9962
20	1111###############0	1	90	1	9999

The simulation results for this control run are shown in Figure 6.15 and Figure 6.16. As can be seen from the population distribution in subplots a and b of Figure 6.15 and Figure 6.16, a viable default hierarchy does not emerge under both reward allocation techniques, at least for the same simulation set up with the fitness proportionate resource sharing scheme. The results are also consistence with previous research where there is no claim on the formation of a viable default hierarchy using any of the reward allocation schemes mentioned.

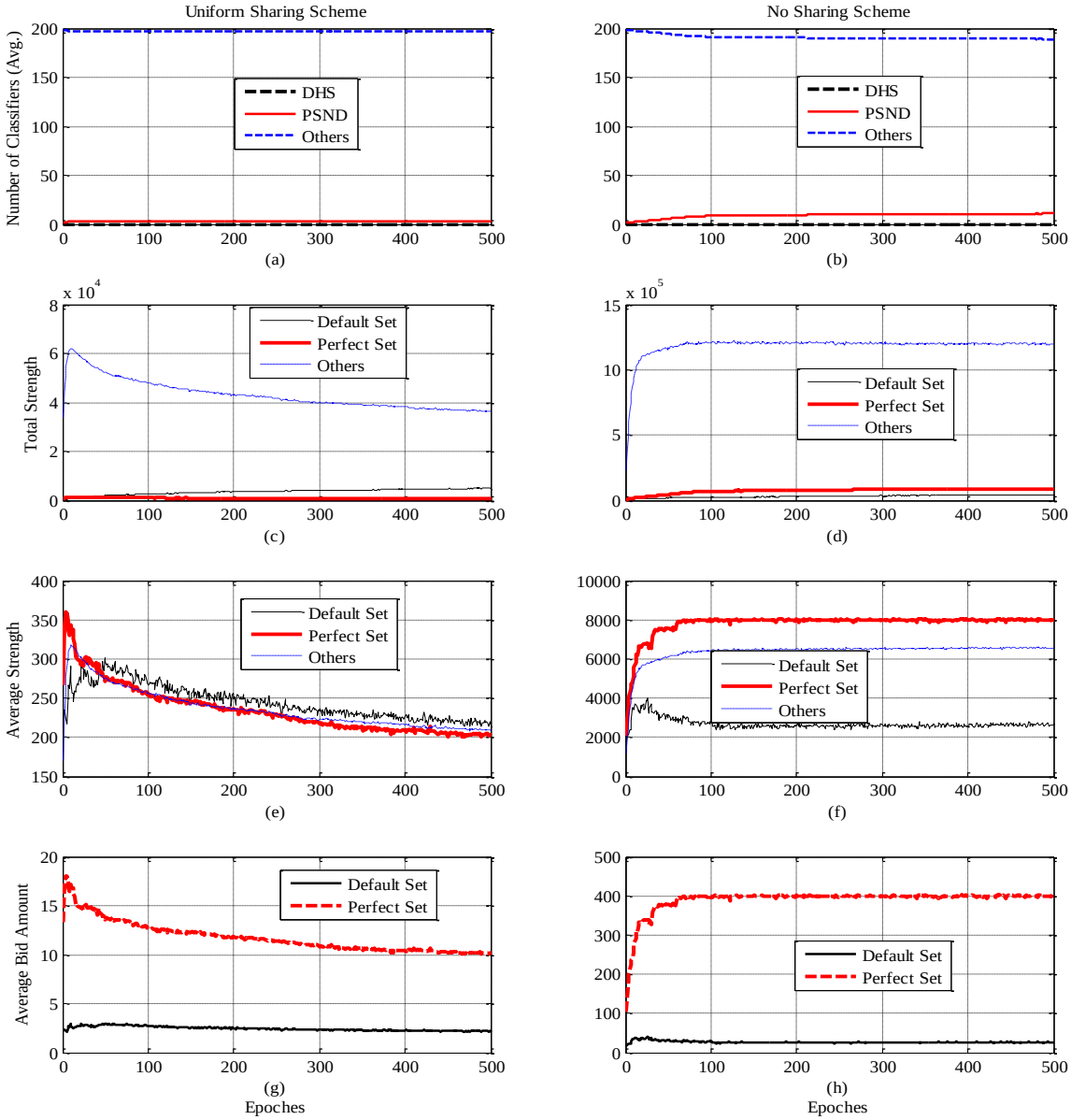

Figure 6.15. Result for 6-mux with uniform and no sharing schemes.

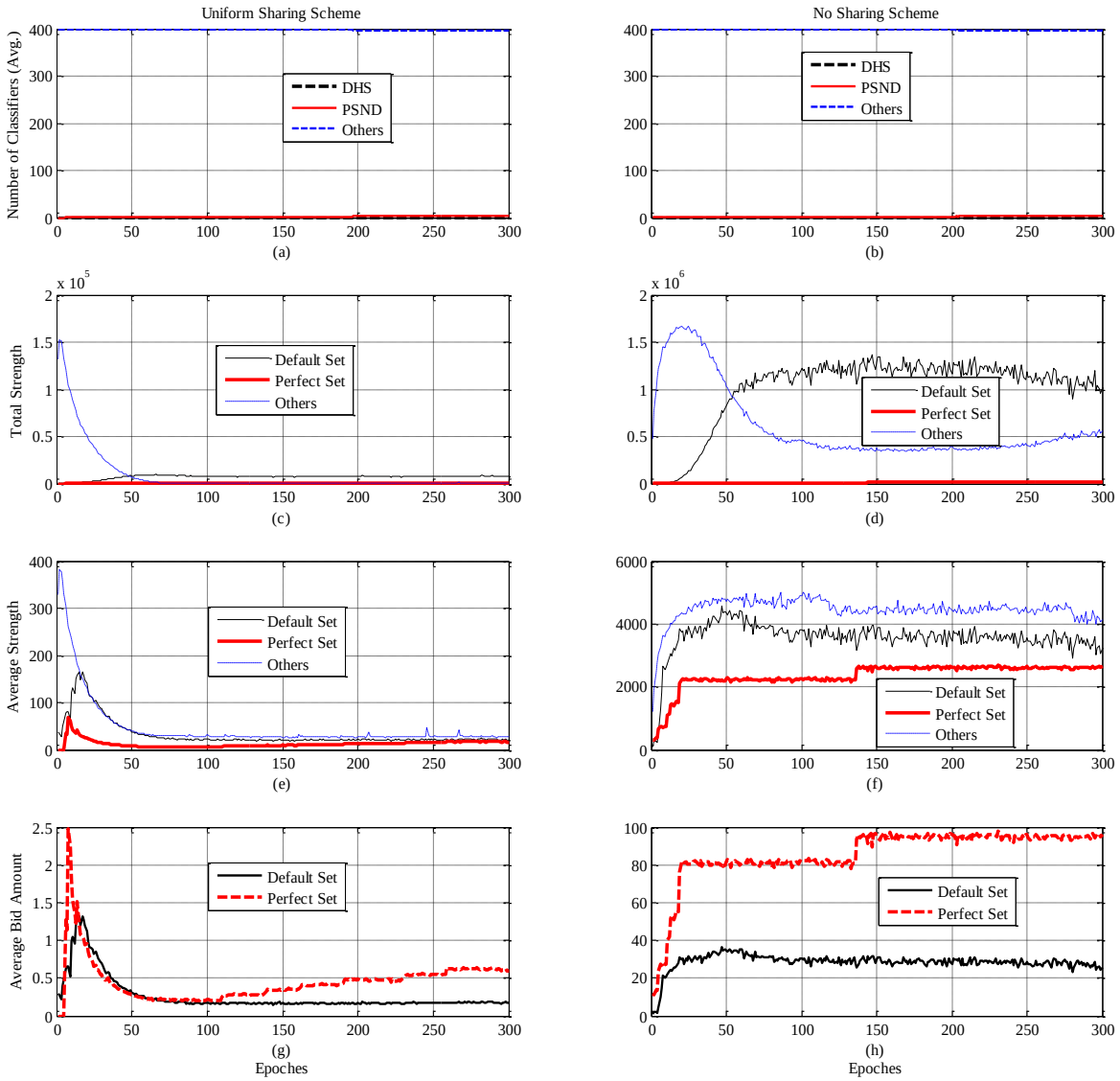

Figure 6.16. Result for 11-mux with uniform and no sharing schemes.

It is also interesting to explore whether an already emerged default hierarchy can be maintained under these reward allocation techniques (i.e. uniform sharing and no sharing schemes). To verify this, we conducted another set of control experiments using the output of our sharing technique as an initial population for the two techniques. In other words, the learning cycle begins with our sharing technique and once the default hierarchy has evolved and controlled the system, the fitness proportionate resource sharing is turned off and the other schemes are applied onwards. Again, the experiment is conducted for the 6 and 11 multiplexer

and the results obtained are shown in Figure 6.17 and Figure 6.18 respectively. The uniform sharing technique has a better performance in maintaining the default hierarchical set as compared to the no sharing techniques. This is evident from the population distribution subplots a and b of Figure 6.17 and Figure 6.18 (both techniques maintain a larger DHS throughout the learning process). However, the experiments are in no way conclusive that the two techniques are able to sustain a default hierarchy once it has emerged for a random initialization.

6.2.4 Theoretical prediction vs. simulation results. The steady state analysis section provided a theoretical formulation for the expected value of the total strength at steady state for the default and exception classifiers under the specified assumptions. It is logical to compare the similarity between the actual values of the total strength obtained using simulations (see part (b) of Figure 6.8, Figure 6.11 and Figure 6.14) with that of the theoretical estimates given in Chapter 4. For the specified values of the parameters used during simulation (R=1000, K=0.106, K_1=0.006 and K_2=0.001), the steady state value of the total strength for the default group using equation (4.15) would be ~=$8.98*10^3$. Also, for the exception group, using these values in equation (4.25), the steady state total strength value would be ~=$3.54*10^4$ for 6-mux, $6.62*10^4$ for 11-mux and $1.17*10^5$ for 20-mux. The corresponding actual steady state values from simulations in Figure 6.8(b), Figure 6.11(b) and Figure 6.14(b) respectively, are very close to these estimated values. The small deviations (i.e. the estimated values are slightly higher than the actual values) are expected due to the simplifying assumption we have made during the formulation. Under the assumption we made, the steady state formulation disregards the impact of other classifiers (classifiers that are not in the default hierarchical set). But in practice, other classifiers, though very small in number, always exist in the system due to discovery by GA and there is a chance that they can reduce the reward share of both the default and exception group.

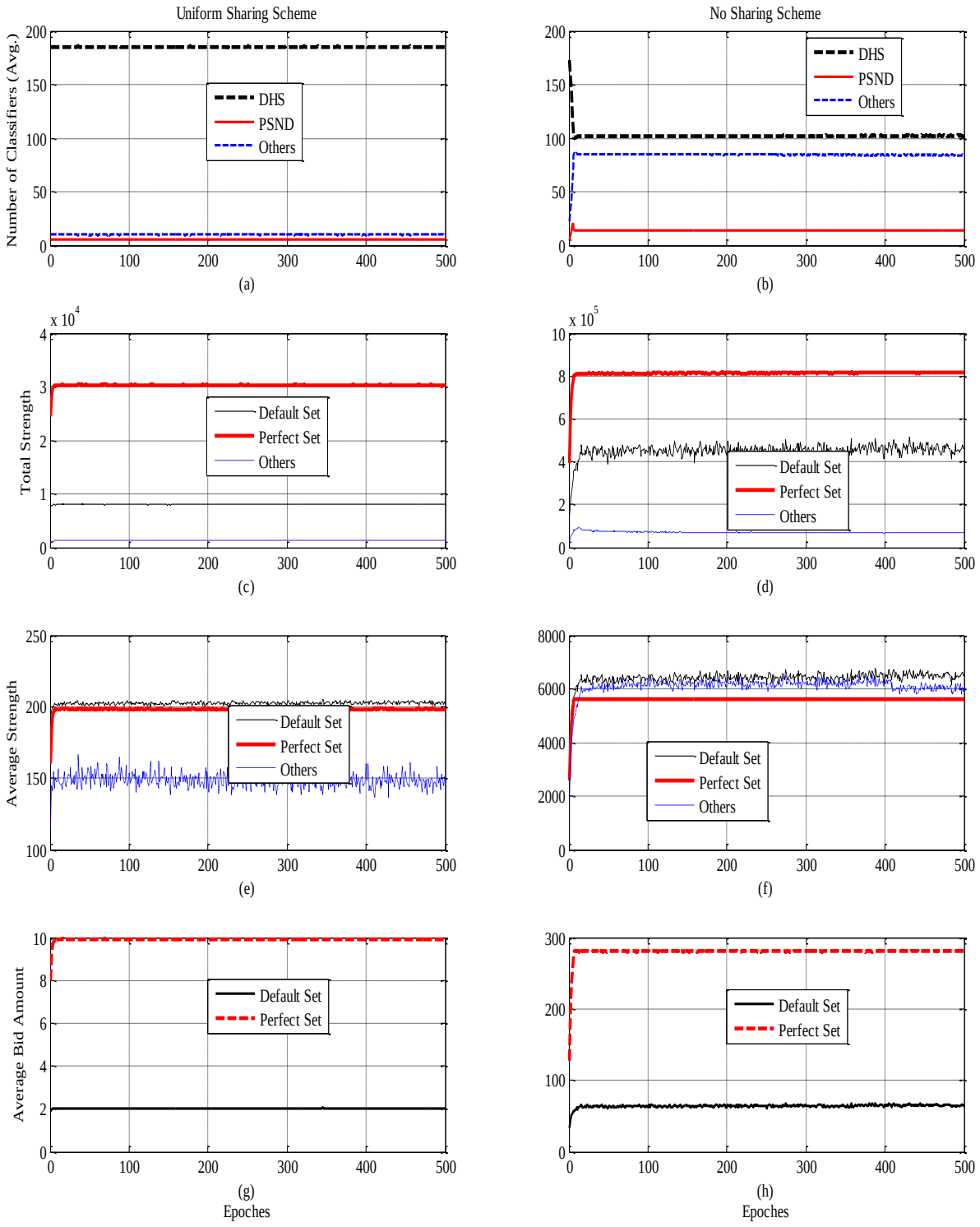

Figure 6.17. A result to check whether a default hierarchy can be sustained under other sharing schemes for 6-mux

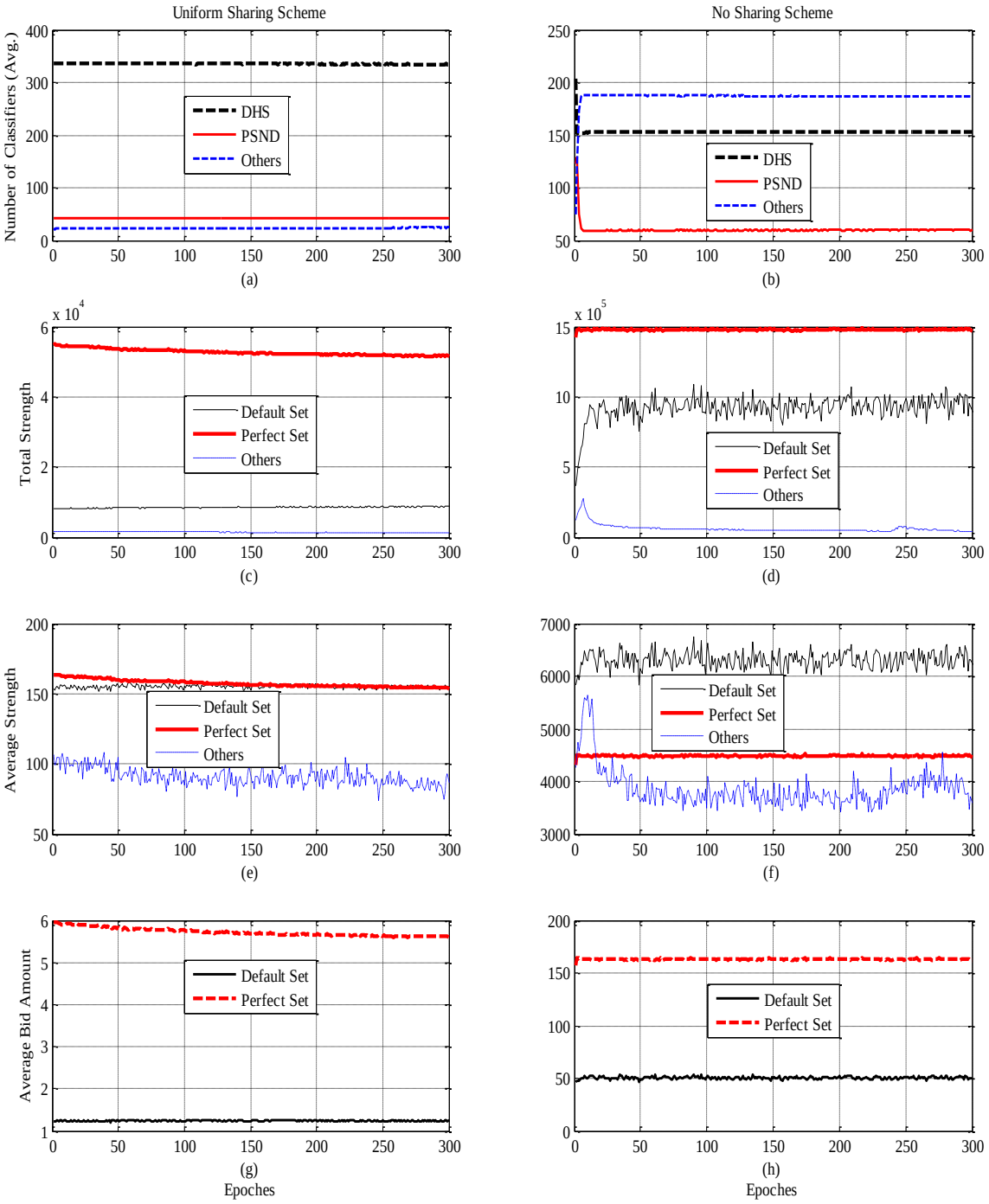

Figure 6.18. Simulations to check whether a default hierarchy can be sustained using other sharing schemes for 11-mux.

CHAPTER 7

Conclusion and Future Work

The "beauty of diversity" makes more real sense to evolutionary algorithms than to any other areas of discipline. Embracing useful diversity enables evolutionary algorithms to discover multiple solutions to a problem and hence extend their global search ability for broad applications such as classification and dynamic clustering. This chapter concludes the dissertation by emphasizing the major achievements made and suggesting ideas for future research path. The major component of this dissertation was the introduction of a novel, ecologically inspired niching technique for evolutionary algorithms. We emphasized its application for evolving a cooperative population of rules in classifiers, multimodal optimization and dynamic clustering. A substantial contribution to the field of learning classifier systems (LCSs) was made through a novel resource sharing technique for discovering and maintaining default hierarchies.

7.1. Summary

Chapter 1 gave a brief introduction on evolutionary algorithms. A bigger picture of the problem addressed in this work, major contributions made and the scope of the work were presented. Before addressing the major thrust of this work, a survey of previous research in this area was a logical step. Chapter 2 did just that by giving a detailed review of previous research on issues in multimodal function optimization, various niching techniques, challenges in niche radius estimation and hierarchical cooperation in classifiers. The major contribution of this work was presented in Chapter 3. Fitness Proportionate Niching (FPN) was introduced. When the objective function has several unequal peaks with a large peak ratio, the traditional niching techniques tend to discover only the location of the highest peak or require a very large

population size in order to discover all the peaks. This demand of large population size added to the distance comparison between individuals makes the traditional sharing techniques computationally cumbersome. The performance of FPN was compared with that of existing sharing method for optimizing multimodal functions with unequal peaks. A technique for estimating the niche radius, a mathematical formulation, complexity analysis and an ecological analogy were also given. Both simulation results and mathematical analyses showed that the performance of the proposed niching technique is insensitive to the fitness difference of the peaks. When individuals share the resource of a given niche in proportion to their fitness, the population distributes among all the peaks uniformly irrespective of the fitness variation at the niches. In other words, high peaks in the multimodal fitness landscape are no longer strong population attractors. This enabled the emergence of stable subpopulations at all the optimum points in the search space and avoids the population size threshold requirement.

Chapter 4 presented a breakthrough in LCS. A new reinforcement technique using FPN based resource sharing was applied. Developing algorithms that lead to the emergence and maintenance of a default hierarchy in an LCS has remained an unachieved goal for researchers in this community for decades. The resource sharing technique presented in this work has filled the gap in the research by enabling the co-evolution of default and exception classifiers in such a way that exception classifiers protect the default from making mistakes without starving it.

To model an environment with a reasonable accuracy, an LCS needs to have a mechanism to build a cooperative set of diverse rules in the population. Learning in an LCS is an ongoing process of discovering cooperative and competitive rules by the GA through the guidance provided by reinforcement. For the best exploration, an LCS requires an intensive search by applying the GA vigorously while embracing useful diversity in the population. The

need to maintain diverse subpopulations compels the use of a restorative force to counterbalance the selection pressure with some sort of diversity maintaining mechanism. Depending on the complexity of the working environment, adequate modelling of the environment might require a huge number of rules that collectively provide a better model of the environment. Building a hierarchical set of rules, where accurate and more specific rules respond to a subset of the situations covered by more general but less accurate default rules is vital to achieve a compact rule set size, especially when dealing with an environment that has huge numbers of states. This requires the co-existence of exception and default rules in the system so that the exception rules can protect the default rule from making mistakes without starving them. To the best of our knowledge, the techniques proposed so far have failed to provide protection without a subsequent starvation of the default. The proposed niching scheme was applied for learning a Boolean function. The robustness and scalability of the algorithm was tested by solving multiplexer problems with various numbers (6, 11 and 20) of inputs. The results obtained for all the simulations proved the effectiveness of the proposed niching technique.

In Chapter 5, the feasibility of FPN for dynamic clustering was demonstrated using both real and synthetic data. The chapter showed how an optimization of a multimodal function with unequal peaks can be mapped into a clustering problem. A formulation of the fitness function for the GA was provided. It was shown that FPN based clustering can be an alternative clustering method when knowledge about the data (e.g. distribution of the data, number of clusters etc) is not known in advance. A detailed discussion of the simulation results was given in Chapter 6. The simulation results obtained are consistent with the mathematical formulations of the corresponding approaches given in previous chapters.

7.2. Future Work

The results presented herein can inspire a number of research directions to extend the ideas presented in this work. This section pinpoints some of them.

7.2.1 Multi-label classification. In Chapter 4, we demonstrated a successful application of FPN for evolving hierarchical cooperation in classifiers. The classification problem solved was a single-label classification problem, where an instance belongs to only one label or class. This is foundational research and can serve as point of departure for more sophisticated biologically inspired computation techniques required for multi-label classification. Two promising research lines that can benefit from this work are protein sequence classification and semantic scene identification. There has been quite a lot of research on single label classification of data. In multi-label classification, an instance in the training set is associated with a set of classes, and the task is to output a set of classes whose size is unknown a priori for each unseen instance (Tsoumakas and Katakis, 2007; Tsoumakas et al., 2010). For example, in bioinformatics, a given protein sequence can be associated with different functions (Jiang and McQuay, 2012). In medicine a patient may be suffering from multiple diseases at the same time. In semantic scene identification a given picture can belong to different categories (for instance both beach and sunset) (Shen et al., 2004). Effective application of multi label techniques can also be very crucial to understanding complex biological systems. For instance, knowing the protein mapping from sequence to structure and then structure to function can help in discovering medical drugs for diseases of no known cure.

Existing multi-label classification approaches follow two general trends: problem transformation and algorithmic adaptation (Tsoumakas and Katakis, 2007; Tsoumakas et al., 2010). There are four major intuitive approaches followed in traditional training techniques. The

first ignores instances belonging to several classes. Another option uses a subjective approach where an instance is assigned to the most obvious class during training. The third option is to extend the number of classes (labels) by forming a hybrid class to accommodate the multi-label data. The drawback of this approach is that it substantially increases the number of classes to be considered and the data in such combined classes are usually sparse. The fourth option is to decompose each multi-label instance into multiple independent binary classification problems (one per category). But this approach does not consider the correlations between the different labels of each instance. In reality, different functional classes are naturally dependent on one another. Thus, this approach ignores the inherent correlations among different classes, which often could be an important indicator for deciding the class memberships, especially when a severe unbalanced data problem occurs. Instead, in multi-label learning, class memberships can be inferred through label correlations, which provide an opportunity to improve the classification accuracy. Evaluation of the learning technique is also another challenge as standard single label evaluation metrics such as precision, recall and accuracy can be vague for multi-label classification. Due to the overlap of the classes the output of the classifier can be perfectly correct, partially correct or fully wrong depending on the number of associations of a particular instance.

7.2.2 Extension to real-valued LCS. We have considered a stimulus-response (Nasraoui and Krishnapuram) learning system for binary Boolean function learning. When the inputs are real valued representation becomes an issue. This is particularly because the inputs can have any value and the search space grows significantly. LCSs with real-valued inputs would be a good area to investigate further as it helps to expand the application of the technique for real-valued data classification.

7.2.3 More clustering applications. The application of FPN for dynamic clustering was given in Chapter 5. However, it was by no means thorough and hence further exploration by comparing the performance of FPN for dynamic clustering with other existing algorithms is essential. Investigating whether the clustering algorithm is able to discover clusters of arbitrary shape and whether it is robust to outliers is a good starting point to pursue.

7.2.4 Evolution dynamics. The empirical results given in Chapter 6 have shown the emergence of subpopulations (default and exception classifiers) as the learning process goes on. The steady state analysis addressed action-reward dynamics by examining the steady state behaviour of subpopulation of classifiers for the typical scenario of a stimulus-response LCS. A more general and detailed analysis of the evolution dynamics on the emergence of subpopulations, the composition of the population at equilibrium and the maintenance of niches under the selection pressure of the GA are interesting areas to explore in future research.

References

Ando, S., Sakuma, J., & Kobayashi, S. (2005). *Adaptive isolation model using data clustering for multimodal function optimization.* Paper presented at the Proceedings of the 2005 conference on Genetic and evolutionary computation.

Asoh, H., & Mühlenbein, H. (1994). On the mean convergence time of evolutionary algorithms without selection and mutation. *Parallel Problem Solving from Nature—PPSN III*, 88-97.

Bacardit , J. (2004). *Pittsburgh genetic-based machine learning in the data mining era: Representations, generalization, and run-time.* Universitat Ramon Llull.

Bacardit, J., Goldberg, D., & Butz, M. (2007). Improving the performance of a Pittsburgh learning classifier system using a default rule. In T. Kovacs, X. Llorà, K. Takadama, P. Lanzi, W. Stolzmann & S. Wilson (Eds.), *Learning Classifier Systems* (Vol. 4399, pp. 291-307): Springer Berlin Heidelberg.

Back, T. (1996). *Evolutionary algorithms in theory and practice: Evolution strategies, evolutionary programming, genetic algorithms*: Oxford University Press, USA.

Back, T., & Schwefel, H. (1993). An overview of evolutionary algorithms for parameter optimization. *Evolutionary computation, 1*(1), 1-23.

Booker, L. (1982). *Intelligent behavior as an adaptation to the task environment.* University of Michigan.

Booker, L. (1989). *Triggered rule discovery in classifier systems.* Paper presented at the Proceedings of the 3rd International Conference on Genetic Algorithms.

Booker, L., Goldberg, D., & Holland, J. (1990). Classifier systems and genetic algorithms. In J. G. Carbonell (Ed.), *Machine learning: paradigms and methods* (pp. 235-282): Elsevier North-Holland, Inc.

Butz, M., Kovacs, T., Lanzi, P., & Wilson, S. (2004). Toward a theory of generalization and learning in XCS. *Evolutionary Computation, IEEE Transactions on, 8*(1), 28-46.

Casillas, J., Carse, B., & Bull, L. (2007). Fuzzy-XCS: A Michigan genetic fuzzy system. *Fuzzy Systems, IEEE Transactions on, 15*(4), 536-550.

Chang, D., Zhang, X., Zheng, C., & Zhang, D. (2010). A robust dynamic niching genetic algorithm with niche migration for automatic clustering problem. *Pattern recognition, 43*(4), 1346-1360.

Chang, D., Zhao, Y., & Zheng, C. (2011). *A real-valued quantum genetic niching clustering algorithm and its application to color image segmentation.* Paper presented at the Intelligent Computation and Bio-Medical Instrumentation (ICBMI), 2011 International Conference on.

Cioppa, A., Stefano, C., & Marcelli, A. (2004). On the role of population size and niche radius in fitness sharing. *Evolutionary Computation, IEEE Transactions on, 8*(6), 580-592.

Cioppa, A., Stefano, C., & Marcelli, A. (2007). Where are the niches? Dynamic fitness sharing. *evolutionary computation, IEEE transactions on, 11*(4), 453-465.

Davidor, Y. (1991). *A naturally occurring niche & species phenomenon: The model and first results.* Paper presented at the Proceedings of the Fourth International Conference on Genetic Algorithms.

Deb, K., & Goldberg, D. (1989). *An investigation of niche and species formation in genetic function optimization.* Paper presented at the Proceedings of the 3rd International Conference on Genetic Algorithms.

Dick, G. (2010). *Automatic identification of the niche radius using spatially-structured clearing methods.* Paper presented at the Evolutionary Computation (CEC), 2010 IEEE Congress.

Dick, G., & Whigham, P. (2006). Spatially-structured evolutionary algorithms and sharing: Do they mix? *Simulated Evolution and Learning*, 457-464.

Dick, G., & Whigham, P. (2008). Spatially-structured sharing technique for multimodal problems. *Journal of Computer Science and Technology, 23*(1), 64-76.

Duan, L., Xu, L., Guo, F., Lee, J., & Yan, B. (2007). A local-density based spatial clustering algorithm with noise. *Information Systems, 32*(7), 978-986.

Floudas, C., & Pardalos, P. (1996). *State of the art in global optimization: Computational methods and applications. Nonconvex optimization and its applications*: Kluwer Academic, Dordrecht.

Forrest, S., Javornik, B., Smith, R., & Perelson, A. (1993). Using genetic algorithms to explore pattern recognition in the immune system. *Evolutionary computation, 1*(3), 191-211.

Frey, P., & Slate, D. (1991). Letter recognition using Holland-style adaptive classifiers. *Machine Learning, 6*(2), 161-182.

Giráldez, R., Aguilar-Ruiz, J., & Riquelme, J. (2003). *Natural coding: A more efficient representation for evolutionary learning.* Paper presented at the Genetic and Evolutionary Computation—GECCO 2003.

Goldberg, D. (1983). Computer-aided gas pipeline operation using genetic algorithms and rule learning.

Goldberg, D. (1989). *Genetic algorithms in search, optimization, and machine learning*: Addison-Wesley.

Goldberg, D., Deb, K., & Horn, J. (1992). Massive multimodality, deception, and genetic algorithms. *Urbana, 51*, 61801.

Goldberg, D., & Richardson, J. (1987). *Genetic algorithms with sharing for multimodal function optimization.* Paper presented at the Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application.

Hilliard, M., Liepins, G., Palmer, M., Morrow, M., & Richardson, J. (1987). *A classifier based system for discovering scheduling heuristics.* Paper presented at the Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application.

Holland, J. (1975). Adaptation in natural and artificial systems, University of Michigan press. *Ann Arbor, MI, 1*(97), 5.

Holland, J. (1976). Adaptation progress in theoretical biology (Vol. 4).

Holland, J. (1980). Adaptive algorithms for discovering and using general patterns in growing knowledge bases. *International Journal of Policy Analysis and Information Systems, 4*(3), 245-268.

Holland, J. (1985). *Properties of the bucket brigade*. Paper presented at the Proceedings of the 1st International Conference on Genetic Algorithms.

Holland, J. (1986). Escaping brittleness: The possibilities of general purpose learning algorithms applied to parallel rule-based system. *Machine Learning*, 593-623.

Holland, J. (1992). *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence*: MIT Press.

Holland, J., Booker, L., Colombetti, M., Dorigo, M., Goldberg, D., Forrest, S., Stolzmann, W. (2000). What is a learning classifier system? *Learning Classifier Systems*, 3-32.

Holland, J., & Holyoak, K. (1988). J., Nisbett RE, Thagard PR (1988). Induction-Processes of inference, learning, and discovery: Cambridge (Mass): MIT Press.

Holland, J., & Reitman, J. (1977). Cognitive systems based on adaptive algorithms. *SIGART Bull.*(63), 49-49.

Holland, J. H. (1995). Escaping brittleness: The possibilities of general-purpose learning algorithms applied to parallel rule-based systems. In F. L. George (Ed.), *Computation & intelligence* (pp. 275-304): American Association for Artificial Intelligence.

Homaifar, A., Goldberg, D., & Carroll, C. (1988). Boolean function learning with a classifier system. 264-272.

Horn, J. (1993). Finite Markov chain analysis of genetic algorithms with niching. *Forrest, 727*, 110-117.

Horn, J., & Goldberg, D. (1996). *Natural niching for evolving cooperative classifiers.* Paper presented at the Proceedings of the First Annual Conference on Genetic Programming, Stanford, California..

Horn, J., Goldberg, D., & Deb, K. (1994). Implicit niching in a learning classifier system: Nature's way. *Evolutionary computation, 2*(1), 37-66.

Jiang, J., & McQuay, L. (2012). Predicting protein function by multi-label correlated semi-supervised learning. *Computational Biology and Bioinformatics, IEEE/ACM Transactions on, 9*(4), 1059-1069.

Jong, K. (1975). Analysis of the behavior of a class of genetic adaptive systems.

Kovacs, T. (2004). *Strength or accuracy: Credit assignment in learning classifier systems*: Springer.

Lanzi, P., & Riolo, R. (2000). A roadmap to the last decade of learning classifier system research (from 1989 to 1999). *Learning Classifier Systems*, 33-61.

Lee, C., Cho, D., & Jung, H. (1999). Niching genetic algorithm with restricted competition selection for multimodal function optimization. *Magnetics, IEEE Transactions on, 35*(3), 1722-1725.

Li, J., Balazs, M., Parks, G., & Clarkson, P. (2002). A species conserving genetic algorithm for multimodal function optimization. *Evolutionary computation, 10*(3), 207-234.

Mahfoud, S. (1994a). *Crossover interactions among niches.* Paper presented at the Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on.

Mahfoud, S. (1994b). Population size and genetic drift in fitness sharing. *Urbana, 51,* 61801.

Mahfoud, S. (1995). Niching methods for genetic algorithms. Urbana, 51, 61801.

Mengshoel, O., & Goldberg, D. (1999). *Probabilistic crowding: Deterministic crowding with probabilistic replacement.* Paper presented at the Proc. of the Genetic and Evolutionary Computation Conference (GECCO-99).

Miller, B., & Shaw, M. (1996). *Genetic algorithms with dynamic niche sharing for multimodal function optimization.* Paper presented at the Evolutionary Computation, 1996., Proceedings of IEEE International Conference on.

Nasraoui, O., & Krishnapuram, R. (2000). *A novel approach to unsupervised robust clustering using genetic niching.* Paper presented at the Fuzzy Systems, 2000. FUZZ IEEE 2000. The Ninth IEEE International Conference on.

Pardalos, P., & Romeijn, H. (2002). *Handbook of global optimization* (Vol. 2): Springer.

Petrowski, A. (1996). *A clearing procedure as a niching method for genetic algorithms.* Paper presented at the Evolutionary Computation, 1996., Proceedings of IEEE International Conference on.

Riolo, R. (1987). *Bucket brigade performance: II. Default hierarchies*. Paper presented at the Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application, Cambridge, Massachusetts, USA.

Robertson, G., & Riolo, R. (1988). A tale of two classifier systems. *Machine Learning, 3*(2), 139-159.

Sander, J., Ester, M., Kriegel, H., & Xu, X. (1998). Density-based clustering in spatial databases: The algorithm GDBSCAN and its applications. *Data Mining and Knowledge Discovery, 2*(2), 169-194.

Sareni, B., & Krahenbuhl, L. (1998). Fitness sharing and niching methods revisited. *Evolutionary Computation, IEEE Transactions on, 2*(3), 97-106.

Shen, X., Boutell, M., Luo, J., & Brown, C. (2004). *Multilabel machine learning and its application to semantic scene classification.* Paper presented at the Proceedings of SPIE.

Sheng, W., Tucker, A., & Liu, X. (2004). *Clustering with niching genetic K-means algorithm.* Paper presented at the Genetic and Evolutionary Computation–GECCO 2004.

Shir, O., & Back, T. (2005). *Dynamic niching in evolution strategies with covariance matrix adaptation.* Paper presented at the Evolutionary Computation, 2005. The 2005 IEEE Congress on.

Shir, O., & Back, T. (2006). Niche radius adaptation in the CMA-ES niching algorithm. *Parallel Problem Solving from Nature-PPSN IX*, 142-151.

Shir, O., Emmerich, M., & Back, T. (2007). *Self-adaptive niching CMA-ES with Mahalanobis metric.* Paper presented at the Evolutionary Computation, 2007. CEC 2007. IEEE Congress on.

Smith, R., Forrest, S., & Perelson, A. (1993). Searching for diverse, cooperative populations with genetic algorithms. *Evolutionary computation, 1*(2), 127-149.

Smith, R., & Goldberg, D. (1990, 26-27 Mar 1990). *Reinforcement learning with classifier systems.* Paper presented at the AI, Simulation and Planning in High Autonomy Systems, 1990., Proceedings.

Smith, R., & Goldberg, D. (1992). Reinforcement learning with classifier systems: Adaptive default hierarchy formation. *journal of Applied Artificial Intelligence, 6*(1), 79-102.

Smith, R., & Valenzuela-Rendón, M. (1989). *A study of rule set development in learning classifier system.* Paper presented at the Proceedings of the third international conference on Genetic algorithms.

Smith, S. (1980). *A learning system based on genetic adaptive algorithms.* University of Pittsburgh.

Streichert, F., Stein, G., Ulmer, H., & Zell, A. (2004). *A clustering based niching EA for multimodal search spaces.* Paper presented at the Artificial Evolution.

Tsoumakas, G., & Katakis, I. (2007). Multi-label classification: An overview. *International Journal of Data Warehousing and Mining (IJDWM), 3*(3), 1-13.

Tsoumakas, G., Katakis, I., & Vlahavas, I. (2010). Mining multi-label data. *Data mining and knowledge discovery handbook*, 667-685.

Weise, T. (2008). Global optimization algorithms–Theory and application. *URL: http://www. it-weise. de, Abrufdatum, 1*, 24.

Wilson, S. (1985). *Knowledge growth in an artificial animal.* Paper presented at the Proceedings of the 1st International Conference on Genetic Algorithms.

Wilson, S. (1989). Bid competition and specificity reconsidered. *Journal of Complex Systems, 2*(6), 705-723.

Wilson, S. (1994). ZCS: A zeroth level classifier system. *Evolutionary computation, 2*(1), 1-18.

Wilson, S. (1995). Classifier fitness based on accuracy. *Evolutionary computation, 3*(2), 149-175.

Wilson, S., & Goldberg, D. (1989). A critical review of classifier systems *Proceedings of the third international conference on Genetic algorithms*: Morgan Kaufmann Publishers Inc.

Workineh, A., & Homaifar, A. (2011). Robust bidding in learning classifier systems using loan and bid history. *Complex Systems, 19*(3), 287.

Workineh, A., & Homaifar, A. (2012a). *Fitness proportionate niching: Maintaining diversity in a rugged fitness landscape.* Paper presented at the 2012 International Conference on Genetic and Evolutionary Methods, Las Vegas.

Workineh, A., & Homaifar, A. (2012b). *Fitness proportionate reward sharing: A viable default hierarchy formation strategy in LCS*. Paper presented at the the 2012 International Conference on Genetic and Evolutionary Methods, Las Vegas.

Workineh, A., & Homaifar, A. (2012c). *A new bidding strategy in LCS using a decentralized loaning and bid history.* Paper presented at the Aerospace Conference, 2012 IEEE.

Yang, M., & Wu, K. (2004). A similarity-based robust clustering method. *Pattern Analysis and Machine Intelligence, IEEE Transactions on, 26*(4), 434-448.

Zhang, G., Yu, L., Shao, Q., & Feng, Y. (2006). *A clustering based GA for multimodal optimization in uneven search space.* Paper presented at the Intelligent Control and Automation, 2006. WCICA 2006. The Sixth World Congress on.

Appendix A

The DNINE algorithm is given below:

Initialize: the population pop, the population size N, the niche radius (σ_o).

$N_A(t)=0$ (the number of actual niches at generation t)

$N_M(t)=0$ (the number of niche master candidates)

NC=0 (the niche master candidate set)

DN =0 (the dynamic niche set)

The niche master identification step:

For i=1 to N do

If the ith individual is not marked then

$N_M(t)= N_A(t)+1$;

$n(N_M(t))=1$ (the number of individuals in the $N_M(t)^{th}$ niche candidate set)

For j=i+1 to P do

If($d(i,j)<\sigma_o$) and (j^{th} individual is not marked)

Insert j^{th} individual in to the niche master candidate set NC,

$n(N_M(t))=n(N_M(t))+1$

End if

End for

If ($n(N_M(t))>1$) then

$N_A(t)= N_A(t)+1$

Mark i^{th} individual as the niche master of the $N_A(t)^{th}$ niche

Insert the pair (i^{th} individual, $n(N_M(t))$) in DN

End if

End if

End for

And the niche expansion (niche refining step) is as shown below:

For l=1 to u(t)

find the nearest neighbor of each niche master candidate.

determine whether the two niches communicate

if there is communication between them

merge the two niches

otherwise

both remain in the population

End for

End for

Appendix B

Journals

1. Workineh, A. and Homaifar, A. "Evolving Hierarchical Cooperation in Classifiers using Fitness Proportionate Niching", Journal of Complex Systems *(submitted, Jan 02).*
2. Workineh, A. and Homaifar, A. "Robust Bidding in Learning Classifier Systems using Loan and Bid History ", Journal of Complex Systems, Volume 19, Issue 3,pps 287-303, 2011 *(published).*
3. Dugda, M.T., Workineh, A. T., Homaifar, A. and Kim, J.H. (2012). "Receiver Function Inversion Using Genetic Algorithms", Bulletin Seismological Society of America *(published).*

Peer Reviewed Conference Papers

1. Workineh, A. and Homaifar, A. "Fitness Proportionate Reward Sharing: a Viable Default Hierarchy Formation Strategy in LCS", The 2012 International Conference on Genetic and Evolutionary Methods, GEM 2012, Las Vegas, July 16-19, 2012.
2. Workineh, A. and Homaifar, A., "Fitness Proportionate Niching: Maintaining Diversity in a Rugged Fitness Landscape", The 2012 International Conference on Genetic and Evolutionary Methods, GEM'12, Las Vegas, July 16-19, 2012.
3. Workineh, A. and Homaifar, A. "A New Bidding Strategy in LCS using Decentralized Loaning ", IEEE aerospace conference, March 06-13, Big Sky, Montana, 2012.
4. Workineh, A., Dugda, M., Homaifar, A. and Lebby, G., "GMDH and RBFGRNN Networks for Multi-Class Data Classification", The 2012 International Conference on Artifical Intelligence, ICAI'12, Las Vegas, July 16-19, 2012.

5. Workineh, A. and Homaifar, A. “Robust Bidding in LCS using Loan and Bid History”, IEEE aerospace conference, March 06-13, Big Sky, Montana, 2010.
6. Workineh, A., Homaifar, A. (Extended Abstract), “Fitness Proportionate Niching: A Different Perspective on Co-evolution of Diverse Population”, ALife13, MIT Press, July 19-22, 2012.

Poster Presentations (peer reviewed and blog posts)

1. Workineh, A., Dugda,M., Homaifar, A., and Lebby, G., “GMDH and RBFGRNN Networks –A Comparative Study for Multi-Class Data Classification”, The 8th International Conference & Expo on Emerging Technologies for a Smarter World (CEWIT2011)”, Hauppauge, New York, November 2-3, 2011.
2. Workineh, A. Opoku, D., and Homaifar, A. “Evolutionary Learning, Navigation and Target Identification for Assistive Robotic Application”, Poster presentation at the BEACON Annual Congress, Michigan State University, August 2011.
3. Workineh, A., and Homaifar, A, “Evolving Hierarchical Cooperation in Classifiers: Nature’s Way”, Poster Competition at North Carolina A&T State University, Spring 2012.
4. Workineh, A, and Homaifar, A., “Bidding Strategy in Learning Classifier Systems Using Loan and Niching GA”, blog post at BEACON Researcher’s website, August 2011.