Organizational Learning

Within A Learning Classifier System

Jason R. Wilcox
Department of Computer Science
University of Illinois At Urbana-Champaign

IlliGAL Report No. 95003
May 1995

Illinois Genetic Algorithms Laboratory (IlliGAL)
Department of General Engineering
University of Illinois at Urbana-Champaign
117 Transportation Building
104 South Mathews Avenue
Urbana, IL 61801

ORGANIZATIONAL LEARNING WITHIN A LEARNING CLASSIFIER SYSTEM

BY

JASON R. WILCOX

S.B., Massachusetts Institute of Technology

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Department of Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 1995

Urbana, Illinois

Abstract

This thesis recasts the debate between Michigan-style and Pitt-style classifier systems to a debate on appropriately sizing organizations within a learning classifier system. Motivated by the economic study of transaction costs, an organizational classifier system (OCS) combining explicit use of multiple reputation values and organization sizing operators better distinguishes parasitic (less than optimal) classifiers than a simple classifier system (SCS). The results show that by building a system that autonomously adjusts the degree of individual to collective behavior, it is possible for it to be both efficient and resilient to problem difficulty.

Acknowledgments

I thank my parents, although no amount of thanks could sufficiently repay their unfailing love and support.

I thank Dr. David Goldberg, my adviser, for his input, support, and creative ideas.

I thank Kevin Carmody and Jeffrey Horn for their friendship and support.

I thank Rebecca Robinson for taking care of things while I worked on this thesis.

Support for this work was provided by the U.S. Air Force Office of Scientific Research under Grant No. F49620-94-1-0103 and from the National Aeronautics and Space Administration with Grant No. NGT-50873.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

A learning classifier system (LCS) is a rule-based machine learning system that makes use of a genetic algorithm to discover new rules. John Holland first proposed the LCS in 1971 (Holland, 1971). Since then, many variations on the original system have appeared (Goldberg, 1989). Much of the debate has been between the Michigan-style, where individual classifiers coded as individual strings evolve, and the Pittsburgh-style, where the entire rule set coded as a single string evolves. In a sense, the Michigan and Pittsburgh methodologies approach the same problem from the endpoints of a continuum from individual to collective behavior.

The central thrust of this thesis is to investigate strategies aimed at improving the performance of classifier systems. In particular, through the exploitation of reputation and the formation of appropriately sized organizations, an organizational classifier system (OCS) is designed and implemented such that it autonomously finds an effective balance between individual and collective behavior. A study of transaction costs in economics provides a key to designing and implementing the necessary mechanisms.

An important transaction cost is the gathering of information (Coase, 1988). For example, a person wanting to buy a new car must incur a cost to decide which particular car to purchase. One part of the decision process is the searching for each car's reputation based on some set of criteria. Thus, the buyer might choose to purchase the car with the best reputation as defined by friends or by the local newspaper. Paying attention to easily accessible reputation is cheaper than performing an in-depth investigation of advantages and disadvantages for each possible car.

Another strategy for reducing transaction costs is for the agent to join an organization that provides a highly probable guarantee for some important aspect of decision making (Coase, 1988). For example, a carpenter must incur a cost to learn what to make and to whom to sell. By accepting employment with a furniture manufacturer, the carpenter not only knows what to build, but is reasonably sure of a steady income. Thus, becoming a member of an organization reduces some transaction costs by removing uncertainty.

Using similar strategies, an OCS can be constructed to balance the individual and collective behavior found in current classifier systems. That balancing is necessary can be determined by a simple thought experiment. If an individual classifier system (Michigan-style) is applied to a difficult sequential learning problem, past studies have demonstrated how performance of the systems becomes dominated by non-productive parasitic rules, often resulting in poor performance (Grefenstette, 1987; Smith, 1991). On the other hand, if a collective classifier system (Pitt-style) is applied to a problem where stimulus-response learning is largely effective, the collective nature of learning only serves to delay the creation of good rules. This effectiveness-speed tradeoff is the primary concern of this thesis; it seeks effective performance in minimal

time and uses reputation and appropriate sizing of organizations as the primary tools to achieve the desired effects.

To accomplish these goals, the thesis starts by presenting a study on the sizing of organizations. Specifically, Chapter 2 presents an overview of transaction costs. Using techniques to reduce costs, the chapter builds an abstract model which explores the sizing of organizations. Afterwards, Chapter 3 examines more closely current approaches to classifiers systems. In addition, Chapter 3 builds a simple classifier system (SCS) which captures the important dynamics found in classifier systems using an individualistic approach. Tying together techniques to reduce transaction costs and the SCS, the thesis builds an organizational classifier system (OCS) and compares it to the simple classifier system. Chapter 4 begins by defining a problem on which to test classifier systems. After testing the SCS, the chapter designs and tests an OCS. In closing, the thesis investigates future research avenues and presents conclusions.

Chapter 2

Organizational Growth

This chapter focuses on mechanisms to size organizations within a classifier system. In particular, the debate between Michigan-style and Pitt-style classifier systems should be recast to a debate on the optimal size of organizations of rules within the classifier system. The Michigan approach keeps track of individual classifiers, while the Pitt approach evolves a population of organizations each containing a fixed number of classifiers. Neither approach is completely satisfactory. Michigan-style classifiers, while converging more rapidly, fail to learn good solutions to complex problems, while Pitt-style classifiers solve more difficult problems at relatively high computation costs. By developing techniques to find appropriately sized groupings or organizations of classifiers, a classifier system will autonomously favor individual or collective behaviors as is necessary to solve particular problems.

Transaction costs theory from economics attempts to explain the formation of organizations within an economic setting. By viewing the interactions of classifiers as transactions, it is possible to use the theory of transaction costs to aid in the development of organizational structures within the classifier system. Because classifier systems are complex, researchers

face a myriad of implementation decisions, parameter settings, and experimental design issues. These characteristics make the isolated investigation of a single facet or small subset of facets difficult at best. Before attempting to build organizations within a full-fledged classifier system, this chapter builds an abstract model that focuses on organizational growth.

Section 2.1 introduces the topic of transaction costs. Section 2.2 presents the abstract model, and Section 2.3 examines the results of using the model.

2.1 An Introduction to Transaction Costs

One goal of this thesis is to invent an autonomous mechanism for forming the appropriately sized organizations within a classifier system. The field of economics provides a source of inspiration for such a mechanism through the study of transaction costs. As an early thinker on the subject, R. H. Coase explains the sizing and formation of organizations from the framework of transaction costs (Coase, 1988). The basic idea is that the organization exists because it reduces the overhead transaction costs associated with exchanging goods and services. The first part of this section works to provide an intuitive understanding of transaction costs within both an economic market and a classifier system. The second part of this section examines mechanisms to reduce transaction costs.

What Are Transaction Costs?

To better understand transaction costs, we examine two people in a market. The first person, the manufacturer, is in charge of building and selling widgets. The second person, the worker, constructs the actual widgets from raw materials. Whenever the manufacturer contracts with the worker, a transaction occurs. They exchange currency for labor. What overhead costs exist

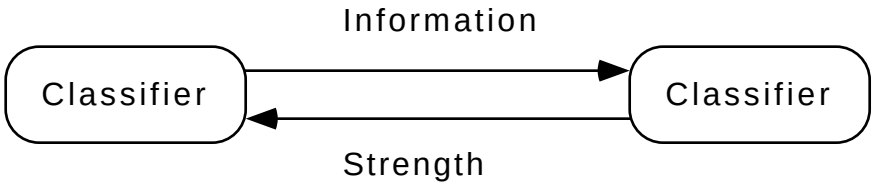

Figure 2.1: Two classifiers exchanging information and strength.

for the manufacturer? Some costs are the effort to find the worker, the expense of negotiating the price for the labor, and the risk of discovering poor craftsmanship. Similar to the manufacturer, the worker must put forth efforts to find employment and negotiate a price. In addition, the worker risks that the manufacturer will fail to pay the wages. Associated with all exchanges of goods and services are a similar set of costs.

Similar to two people in a market, two agents or classifiers in a computational system may exchange goods and services. Figure 2.1 shows two interacting classifiers exchanging information for a payment of strength (see Chapter 3 for details on the workings of classifier systems). Ideally, the classifier providing information would sell to that particular classifier whose downstream actions lead the system to receiving maximum reward. From the information provider's viewpoint, two costs are deciding to whom to sell and risking that the buyer fails to lead the system to gain the maximum reward. At the same time, the buying classifier faces two related costs. One is the cost of deciding from whom to buy, while the other cost is the uncertainty that the information correctly reflects a situation in which the buyer can help the system receive maximum reward.

The costs mentioned above can be discussed from the point of view of knowledge (Hayek, 1945). Before the manufacturer and the worker meet, they both have something to sell. However, neither the manufacturer nor the worker knows to whom to sell. After meeting, they

still lack the knowledge of each other's respective prices. The manufacturer wants to pay as little in wages as possible, while the worker wants as to receive as much in wages as possible. Negotiation is the process of learning each other's true or acceptable price. Finally, each must take a risk that the other is telling the truth and will deliver the goods. The more effectively each investigates the other, the less risk each will face. Thus, a portion of overhead costs is the cost of gaining knowledge about the marketplace.

Similarly, the two classifiers lack knowledge; each needs to discover with whom to transact. The more information that the system stores about each classifier (such as strength and specificity), the more information that each classifier can use to make a decision. Thus, a portion of the overhead costs is the costs of storing and analyzing a classifier's previous performance.

Mechanisms That Reduce Transaction Costs

According to some economists that subscribe to transaction theory, efforts to reduce transaction costs lead to the development of markets, organizations, and other economic structures (Coase, 1988). This subsection discusses some of the structures that develop while trying to reduce transaction costs. In particular, keeping track of reputation and forming organizations wherein transactions are repeated are two fundamental techniques. The organizational classifier system presented in the following chapters uses these techniques.

Joining an organization offers protection to members from many different overhead costs. To understand why organizations form, we first ask: why do people agree to long term contracts? Agreeing to sign such a contract reduces future transaction costs. Working with the economic market example, assume that both the manufacturer and the worker decide to sign a long term contract. The manufacturer no longer needs to worry about finding a new worker for

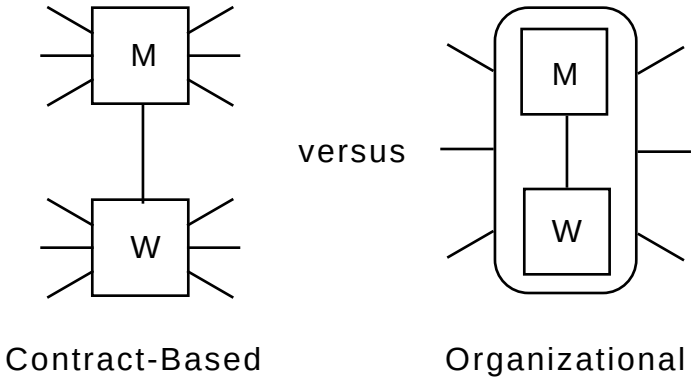

Figure 2.2: Transactions: As individuals, the manufacturer, M, and the worker, W, transact with the outside world alone. As members of an organization, they work as a team.

each widget. Also, the cost of negotiating wages reduces to a one time expense. At the same time, the worker faces a similar reduction in costs. As the manufacturer expands, he signs long term contracts with additional widget builders. The next step is to form an organization. The manufacturer now looks for permanent workers to replace temporary ones. Workers accept employment to join a group of people linked by a common goal. Figure 2.2 shows the relationship between the manufacturer and worker when they are not a part of an organization versus when they are part of an organization. The idea is that individual lines represent single transactions. When not part of an organization, a person must explicitly incur a cost for each transaction with the outside world (anyone differing from the manufacturer and worker). However, as a member of an organization, some transactions move from the individual's point of view to the organizational level. Thus, some tasks are jointly addressed, amortizing the costs among all employees.

Organizations of classifiers can reduce some of the same costs. For example, preferentially matching classifiers to messages posted by other classifiers within the same organization reduces the costs of finding 'good' classifiers with whom to transact. The implication is that conflict

resolution needs to worry less about selecting less-than-optimal classifiers when matching classifiers from the same organization.

Another cost-reducing strategy is to pay closer attention to reputation. Working from the earlier example, the manufacturer wants to hire a the worker who will produce the highest quality widgets. How does the manufacturer find such a worker? Although impossible to guarantee that a particular worker will fulfill expectations, the manufacturer might decide to expend a great deal of effort researching each prospective employee. Workers might be asked to take tests, build a prototype, or even work on a trial basis. Researching possible workers can be an expensive action. Thus, to reduce the searching costs, the manufacturer might ask for references from each worker. What role do the references play? In essence, the manufacturer is looking at the reputation of each worker as seen through the eyes of the references. If the manufacturer has previous experience with a worker, the information (a form of reputation) is also considered during the hiring process.

Reputation plays a key role within classifier systems. Within traditional individual-style (Michigan) classifier systems, classifiers carry a strength value. Credit allocation uses previous and current performance to assign strength, which (according to the above usage) implies that strength is a form of reputation for the carrying classifier. Conflict resolution uses the reputation of strength to make the decision with whom a classifier will transact. Chapter 4 provides treatment of reputation within classifier systems.

There are other ways of reducing transaction costs. Specialized markets reduce the cost of finding interested parties. For example, a farmer sells goods at a produce market because he knows that interested buyers will be present. Court systems provide a mechanism for protecting the rights of individuals. Employees join unions reducing negotiation costs with management.

Corporations join consortiums, reducing research costs. These examples only touch the surface of possible cost-reducing structures. Ultimately, we would like to learn from transaction theory a methodology to solve hard problems. The following section presents an abstract model used to explore techniques for forming organizations from a population of individuals. Later chapters address how to use the presented techniques to form organizations within classifier systems.

2.2 A Study of Organization Growth

Chapter 4 will focus on building an organizational classifier system (OCS) which uses mechanisms found through studying transaction costs. Here, the thesis constructs a simplified model that focuses on a facet of the OCS that controls organizational growth. Traditionally, a CS either evolves individual classifiers (Michigan-style) or evolves a collective population of classifiers (Pitt-style). The thesis will show that an OCS can simultaneously evolve both individuals and collections of individuals by allowing multiple organizations to form within the population of classifiers. Organizations compete for strength in much the same way as do individual rules. This approach involves the autonomous growth of competing organizations each trying to solve the same or related problems. Thus, an organization behaves both as an individual and as a collection.

This section radically simplifies the OCS by assuming all agents are identical. The following lists the main components of the simplified abstract model:

1. Agents and organizations

2. The fitness function

3. Organizational operators

4. Selection

Agents are individual classifiers, while organizations are containers of one or more agents. If organizations are viewed as variable-sized chromosomes, the system evolves a population of organizations in a GA-like fashion. The fitness function for each organization is a function of the number of agents. Operators generate new organization possibilities, while a tournament selection scheme selects organizations to be part of the next generation. The rest of this section provides greater details for each component.

Agents and Organizations

Here, organizations contain homogeneous agents. Agents do not perform any specific actions other than moving between organizations. The complexity of a normal classifier system is abstracted away; an organization's fitness becomes a function of the number of agents rather than a function of agent behavior.

The Fitness Function

This model assumes that the goal of each agent and organization is to maximize profit. The success of an organization is determined by the success of the individuals that make up the organization. Thus, the fitness function F represents the per capita profit for each organization. Each agent, as an employee, potentially earns revenue and incurs cost to the organizations. As an abstract model, the fitness function uses the number of agents within an organization as the sole determiner of fitness. Equation 2.1 shows the resulting fitness function for an organization: such that $R(M)$ equals the organizations total revenues, $C(M)$ equals total costs, and M equals the number of agents.

$$F = \frac{R(M) - C(M)}{M} \tag{2.1}$$

An organization's total revenue is determined by a function of the number of agents. While there are many possible revenue functions, three were examined that seem to represent a wide range of real situations. The following list describes each equation:

Constant - $R(M) = K$**.** If an organization's profit is not determined by the number of employees, its revenue would be constant.

Linear - $R(M) = K_0 + K_1 M$**.** The linear equation models organizations whose revenues increase by the same amount for each additional employee.

Polynomial - $R(M) = K_0 + K_1 M + K_2 M^2$**.** Many organizations are able to increase revenues by non-linear amounts with additional employees. A polynomial equation represents restricted growth, but it seems consistent with many organizations.

There are many costs associated each employee in a real-world organization. The model presented here focuses on the costs associated with communication between agents. Depending on the hierarchy of an organization, agents will have to communicate with different numbers of other agents. Here, costs increase as the quantity of communication increases. The following list describes the equations used to calculate total costs:

Constant - $C(M) = K$**.** Here, communication is trivial, and the addition of agents has no effect on an organization's total costs. While this case does not appear to correspond to any real world situation, it provides a base from which to compare other equations.

Linear - $C(M) = K_0 + K_1 M$**.** This would be the case if each agent communicates with a fixed number of other agents, regardless of the organization's size. An example of a hierarchical structure that applies to the linear equation is a tree with fixed branching.

Polynomial - $C(M) = K_0 + K_1 M + K_2 M^2$**.** For some organizations, the communication costs increase as a polynomial function of agents. For example, the total costs of an organization that requires each agent to communicate with every other agent is bounded by M^2.

By combining different total rewards and total costs functions, the fitness function can represent a wide range of situations. For example, an organization may produce a fixed quantity of widgets regardless of how many employees work. Additional employees add to the management costs without increasing the rewards. The reward equation (assuming that at least one employee can produce the above quantity) would be constant, while the cost equation might be linear or polynomial. Selection uses the fitness function to determine which organizations should be in new generations. Next, the section describes operators which create new organizations to be processed by selection.

Organizational Operators

Operators control the creation of new organizations from generation to generation. They are characterized by the number of agents moved between organizations and whether they increase or decrease the size of a selected organization. During each generation, the system applies operators to pairs of organizations within the current generation. Selection (described next) uses the output of the operators to choose which organizations form a new generation.

Each operator acts on a pair of organizations from the current population and outputs one or two new pairs. The following list describes each operator:

AddOne This operator moves a single agent between two organizations. Two parent organizations are randomly selected from the current population. If only one organization remains in the population, the operator in cooperation with selection directly passes the remaining organization to the next generation. Figure 2.3 shows how AddOne works. A and B are the pair of organizations selected from the current population. The operator forms two new pairs of organizations by moving a single agent between copies of A and B.

SubOne The counterpart to AddOne, this operator removes a single agent from an organization and places it in an empty organization. SubOne randomly selects an organization from the current population and uses an empty organization to complete the pair. If the selected organization only contains a single agent, the operator in cooperation with selection directly passes the organization to the next generation. Unlike AddOne, SubOne outputs a single organization pair, moving a single agent from a copy of the selected organization into the empty organization.

Join As with AddOne, the Join operator randomly selects two organizations from the current population. Again, if the current population contains a single organization, that organization moves directly into the next generation. Unlike AddOne, this operator generates a single organization pair consisting of one organization, containing all the agents from the parent pair and an empty organization.

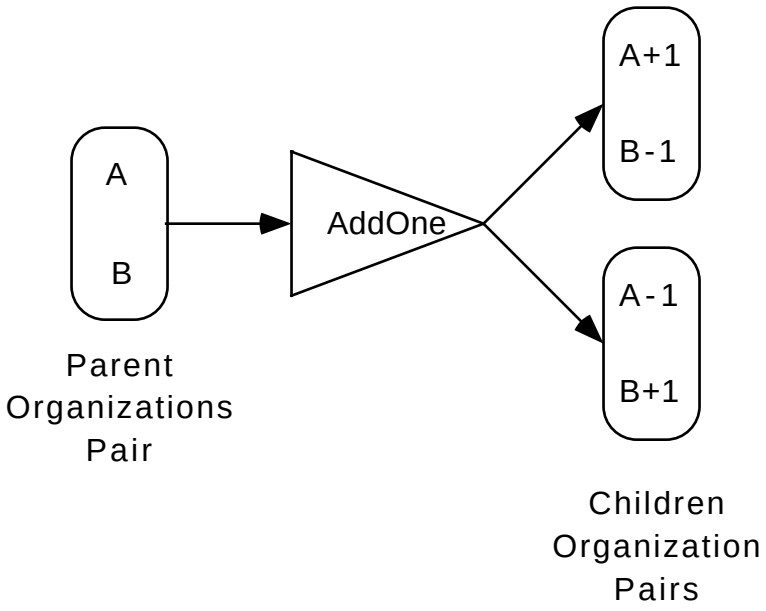

Figure 2.3: The operator AddOne forms two new pairs of organizations from a parent pair of organizations (A and B).

Cut Just as SubOne is the counterpart to AddOne, Cut is the counterpart to Join. It selects a single organization from the current population and pairs it with an empty organization. Cut then forms a new pair by moving a random number of agents from a copy of the selected organization into the empty organization.

The model uses operators in related pairs: AddOne/SubOne and Join/Cut. By examining the equations relating to speed of convergence for each operator pair, it should be possible to calculate the speed with which a system will converge. Because selection (discussed next) is a greedy algorithm (always selecting the most fit organization), the result of selection on the output of operators is to always improve or maintain the organization with the highest fitness. The time for the system to create one organization with the optimal size can be estimated by the approximate time to combine organizations containing one agent into an organization of the optimal size.

$$t = S_{optimal} \tag{2.2}$$

Equation 2.2 shows the estimated time for the system to learn one organization of optimal size, when using the AddOne and SubOne operators. Because these operators only change an organization by one agent, the time is equal to the optimal size.

$$\left(\frac{1}{2}\right)^{t} S_{optimal} < 1 \tag{2.3}$$

The Join and Cut operators change an organization by an unspecified number of agents. When used together, a coarse approximation of the time to find the first organization containing the optimal size is shown in Equation 2.3. The equation assumes that on average the operators change the size of an organization by a factor of two. Equation 2.4 solves Equation 2.3 for time.

The derivation occurs by first re-arranging Equation 2.4 and taking the log of each side yields:

$$t \log_2 \frac{1}{2} < \log_2 \frac{1}{S_{optimal}}$$

Reducing the resulting equation yields:

$$-t < -\log_2 \frac{1}{S_{optimal}}$$

Finally, solving for t yields Equation 2.4:

$$t > \log_2 S_{optimal} \tag{2.4}$$

For both operator pairs, we expect convergence to be somewhat slower (but of the same magnitude) than indicated because at times the system will not change the highest fit organization. The above operators work closely with selection to form a new generation. The process of selection is described next.

Selection

The selection mechanism maintains as invariant the number of agents found within the entire population of organizations. The basic idea is that offspring organizations, generated by an operator, compete with parent organizations. After an operator creates a set of pairs of new organizations, a tournament is held between the new pairs and the parent pair. The pair containing the organization with the highest fitness survives to the next generation, while all the other pairs are deleted. Because the operator creates pairs of offspring organizations that contain the same number of agents as the parent pair of organizations, the number of agents within the system remains constant.

The selection scheme used here is an extension of the selection method used in the GIGA system (Culberson, 1992). There, the selection scheme maintained genetic material from generation to generation by only allowing offspring to replace parent chromosomes when they had higher fitness. The motivation was to keep the genetic material invariant. Here, the same kind of scheme is used, but the motivation differs in that the number of agents is invariant. Putting together the components of the abstract model, the next section presents the results of running the model when varying different parameters.

2.3 Results of the Organizational Growth Model

This section presents the experimental results for the above model when varying fitness functions, operators, and initialization strategies. The fitness functions and operators are described above. The section begins by introducing three initialization strategies. It continues by presenting the results of three experiments, each exploring a different parameter. The section ends by summarizing the results of organizational growth model.

Initialization Strategies

At the start of each run, the model is initialized to contain a population of organizations and agents. Because the model represents an abstraction of a classifier system, it is important to represent different states of the production system. The following list describes three initialization strategies that capture various states that may be found in a production system:

1. Start with 100 organizations each containing one agent.

2. Start with one organization containing 100 agents.

3. Start with a random number of organizations such that the total number of agents equals 100.

The three strategies explore three possible states of an organizational classifier system. The first strategy represents a system of independent agents, while the second strategy represents a system containing a single organization. The third strategy represents an intermediate state between the first two strategies where organizations contain different numbers of classifiers. The rest of this section describes the results of running the abstract model given a number of different parameter settings.

Experiment 1: Comparing Operators

The first experiment examines differences between the AddOne/SubOne and Join/Cut operator pairs. Because AddOne and SubOne move one agent at a time, one expects that changes in the system will occur gradually. On the other hand, Join and Cut operators change the structure of organizations by an unspecified number of agents and should cause rapid learning. Equations 2.2 and 2.3 predict the approximate times to appropriately size one organization using the different operator pairs. After discussing some general results, two representative runs are compared.

Several different fitness functions and initialization strategies were examined while varying the operators. In each case, using AddOne and SubOne operators caused the system to gradually learn the optimally sized organizations. Once the system found the organization with the highest fitness, it remained stable. Also as expected, when using Join and Cut operators, the system found the ideal solutions more quickly. Because organizations often change by large degrees, the improvements often oscillated above and below the ideal solutions. Additionally, even though one organization reached an ideal size, the system often took many more generations to stabilize the rest of the population. However, given enough time, using either operator pair enabled the system to correctly size each organization.

The rest of the results for this experiment consists of a discussion of two representative runs. Several hundred runs were completed in total. One run uses the AddOne and SubOne operator pair, while the other uses the Join and Cut operator pair. The initialization strategy and fitness function remain constant between the two runs and are described below:

- The initialization strategy starts the system with 100 organizations each containing one agent.

- The fitness function uses linear rewards and polynomial costs based on an organization's size, M.
  - $R = 100M$
  - $C = 625 + M^2$
  - $F = 100M - (625 + M^2)$

The optimal size of an organization is determined by calculating the maximum of the fitness function. In this case, we can readily solve for the optimum value analytically.

Substituting total rewards and total costs in the fitness function:

$$F = \frac{(100M - (625 + M^2))}{M}$$

Setting to zero the first derivative of F with respect to M:

$$0 = \frac{625}{M^2} - 1$$

Solving for M:

$$M = 25$$

Thus, the optimal size of an organization is 25, allowing the system to form 4 organizations of optimal size. In addition to predicting the optimal size of an organization, the approximate time to learn one organization with the optimal size can be calculated using Equations 2.2 and 2.4. For the AddOne and SubOne operator pair, substituting $M = 25$ into Equation 2.2 yields a convergence time $t = 25$. For the Join and Cut operator pair, substituting in $M = 25$ yields a convergence time $t = 4.64$. Thus, we expect the system to converge in about 25

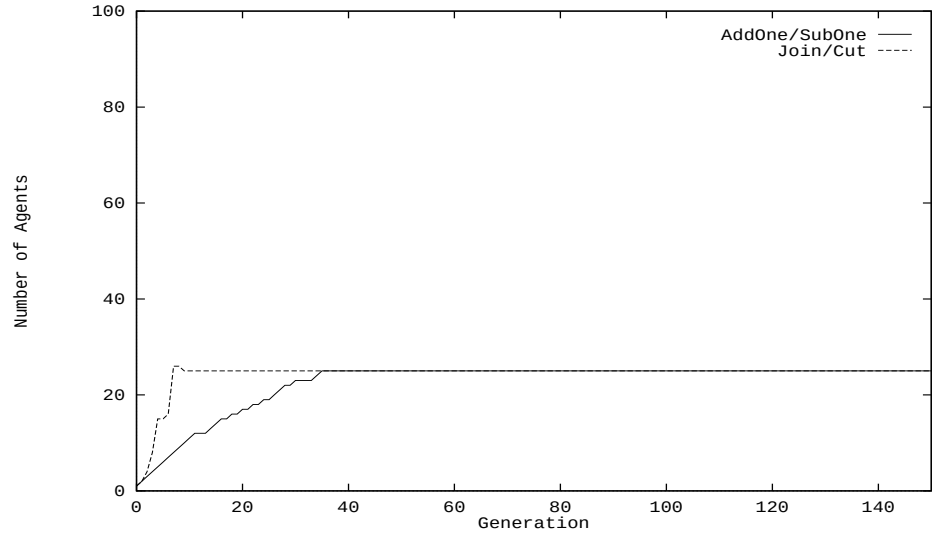

Figure 2.4: Organizational Growth: Fitness function uses linear rewards and polynomial costs. Each run is initialized with 100 organizations each containing one agent. One run uses AddOne/SubOne operator pair, while the other uses Join/Cut operator pair.

generations when using the AddOne/SubOne operator pair and in about 5 generations when using the Join/Cut operator pair.

Figure 2.4 shows the size of the highest fit organization during each run. As expected, both runs find an optimally sized organization near predicted times. Using the AddOne and SubOne pair, the system converged in about 30 generations. While using the Join and Cut, the system converged in about 7 generations. Also as expected, using AddOne and SubOne resulted in a more gradual change. The speed that the system learned the four optimal organizations is about twice as fast when using Join and Cut operators. Of course, the comparison does not necessarily hold for more complex models where agents are differentiated.

Experiment 2: Comparing Fitness Functions

Experiment 2 compares the organization growth model using different fitness functions. Varying the fitness function changes two characteristics of the search space. First, the fitness function controls the optimal size of an organization. Second, it determines the rate of change in fitness as the number of agents within an organization varies. Below, two representative runs show the results of using two different fitness functions.

Differing from the previous experiment, the runs here initialize the system with a random number of organizations of random size such that the total number of agents in the system equals 100. In addition, both runs use the Join and Cut operators. Each run uses a different fitness function. The first fitness function models an organization with linear rewards and linear costs. The equations are shown below:

- $R = 100M$
- $C = 20 + 50M$
- $F = \frac{100M-(20+50M)}{M}$

Because the rewards are greater than the costs, using the same kind of analysis from the previous experiment shows that the optimal size organization contains 100 agents (the most possible). Because the system is initialized with randomly sized organizations, it is impossible to directly plug into Equation 2.4 to determine the rate of convergence. However, using the worst case shows that the system should learn the best organization size in approximately 7 generations.

The second fitness function is based on linear rewards and polynomial costs. The equations are as follows:

- $R = 100M$
- $C = 625 + M^2$
- $F = \frac{50M - (625 + M^2)}{M}$

The analysis from the previous experiment shows that the optimal organization contains 25 agents. The system should learn the first organization containing 25 agents in approximately 5 generations.

Figure 2.5 shows the number of organizations in the system during each run. As expected, the run using the first fitness function quickly converges to one organization. The run using the second equation takes longer to fully converge. While not shown, the second run succeeds at quickly finding first organization containing the optimal number of agents. However, it then needs to size three other organizations also containing the optimal number of agents. The additional time used to appropriately size all the organizations explains the additional time needed to stabilize.

Experiment 3: Comparing Initialization Strategies

The final experiment examines the model's behavior when using different initialization strategies. Organizational sizing techniques need to be robust to differing initial organizational structures. Given the characteristics of the abstract model, the initialization strategy should not change how the system generally behaves. After describing the parameter settings, the results for three runs comparing initialization strategies are presented.

Each of the three runs uses the AddOne and SubOne operators. The fitness equation (the same one from experiment 1) uses linear rewards and polynomial costs. The list below describes the equations.

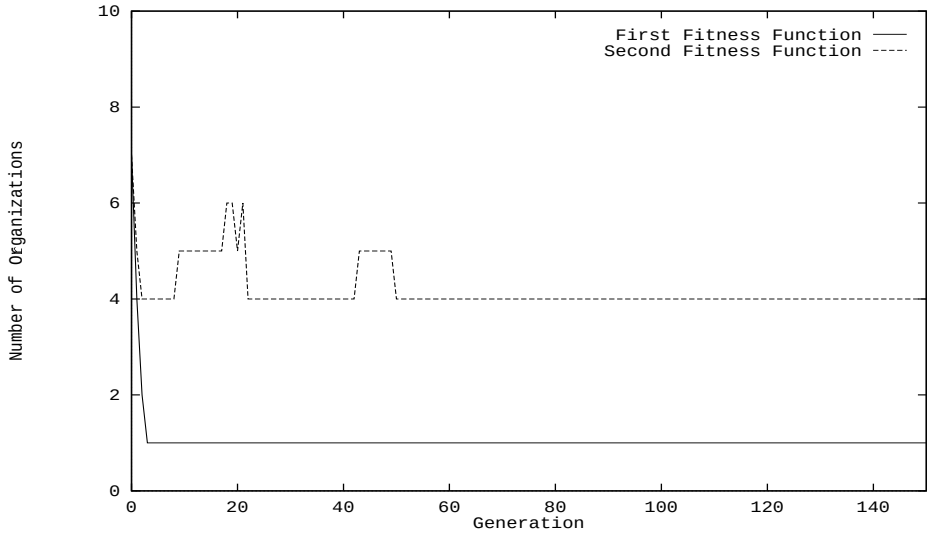

Figure 2.5: Organizational Growth: The operators are Join and Cut. Each run is initialized with randomly sized organizations such that the total number of agents equals 100.

- $R = 100M$
- $C = 625 + M^2$
- $F = \frac{50M - \left(625 + M^2\right)}{M}$

As previously calculated, the organization with the maximum fitness contains 25 agents. The time to correctly size the first organization should vary depending on the initialization strategy. The first strategy initializes the system with 100 organizations each containing one agent. The expected time is approximately 25 generations. The second strategy initializes the system with one organization containing 100 agents. The expected time is also 25 generations. The third strategy initializes the system with randomly sized organizations. The expected time varies depending on the initial size of the organizations. In the worst case, all the organizations contain agents such that their size is different from the optimal size by 25 or more agents. The approximate time would be 25 generations. However, it is possible that an initial organization

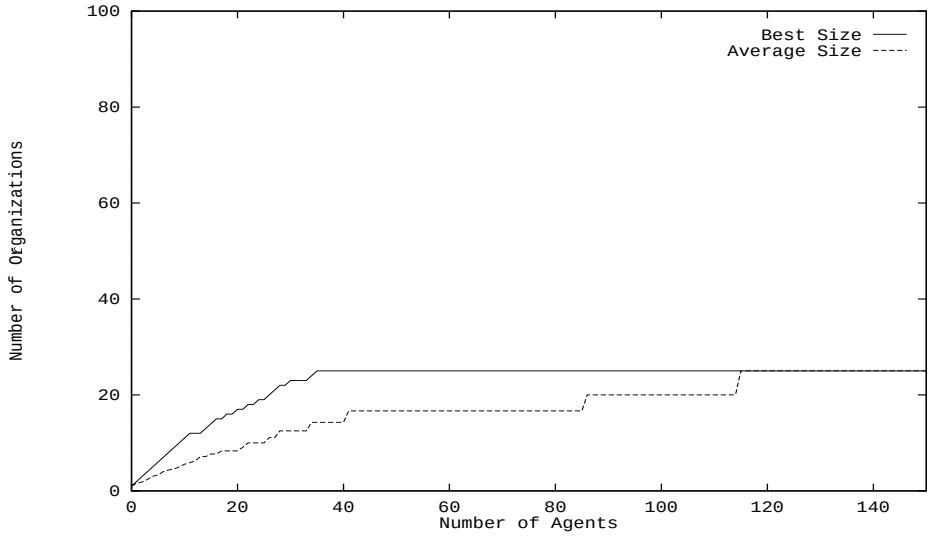

Figure 2.6: Organizational Growth: The fitness function uses linear rewards and polynomial costs. The operators are AddOne and SubOne. The run is initialized with 100 organizations each containing one agent.

will contain the optimal 25 agents. So, the estimated time is anywhere between 0 and 25 generations.

Figures 2.6, 2.7, and 2.8 show the size of both the most fit organization and the average fitness of all organizations during the respective runs using each initialization strategy.

Each run behaved as expected. The times to find the first appropriately sized organization for the first and second runs were both about 25 generations. In addition, the third substantially sized the optimal organizations faster than the previous two runs.

The above results show how organizational operators evolve appropriately sized organizations given the simplified environment. While using the Join and Cut operators enabled the system to more quickly learn the first organization containing the optimal size, the time for the entire population of organizations to stabilize was approximately the same as for the AddOne and SubOne operator pairs. In addition, the use of the Join and Cut operators caused the

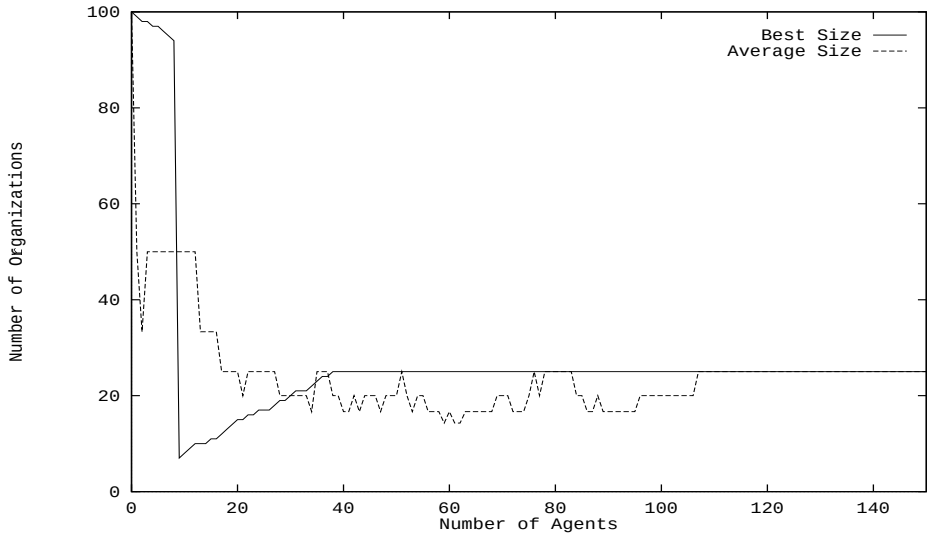

Figure 2.7: Organizational Growth: The fitness function uses linear rewards and polynomial costs. The operators are AddOne and SubOne. The run is initialized with one organization containing 100 agents.

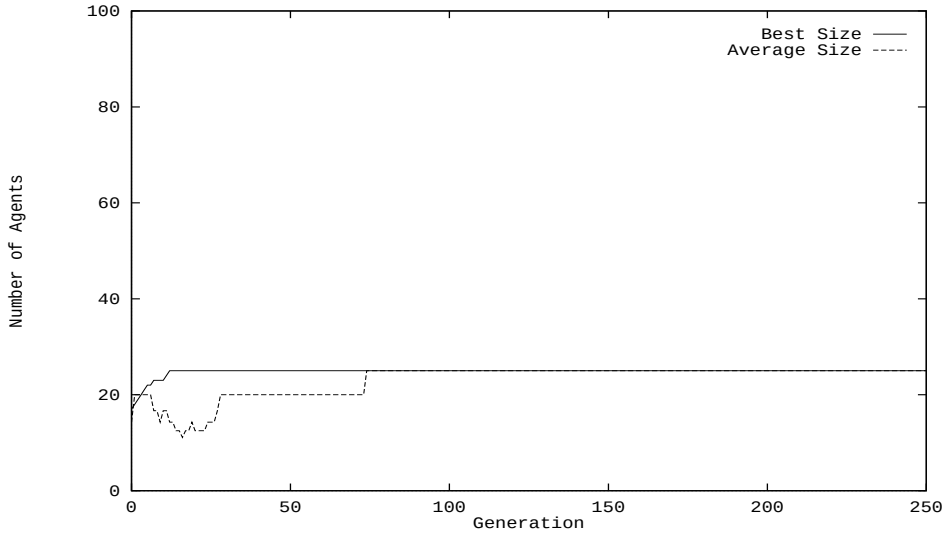

Figure 2.8: Organizational Growth: The fitness function uses linear rewards and polynomial costs. The operators are AddOne and SubOne. The run is initialized with randomly sized organizations such that the total number of agents equals 100.

system to oscillate organization sizes as it neared convergence. Using AddOne and SubOne led the system to gradually converge. While varying fitness function changed the optimal solutions, it did not affect the system's ability to converge. Finally, the system learned appropriately sized organizations regardless of the initialization strategy.

2.4 Summary

The chapter opens by providing an introduction to the theory of transaction costs from economics to explain the formation of organizations. By mapping the production component of a classifier system to an economic world, it is possible to find mechanisms that autonomously form hierarchies of classifiers. The main idea is an assumption that people try to reduce overhead costs associated with transactions, the exchange of goods or services. Several techniques reduce these costs. In particular, forming organizations provides a common interface to the outside world, protecting members from some overhead costs.

Afterwards, an abstract model of a classifier system of homogeneous agents is built to isolate the simplest organizational sizing factors. Complex organizational behavior is modeled by different fitness functions based on the size of the organizations. Organizational operators, acting on pairs of organizations, form new pairs of organizations with differing sizes, after which a form of tournament selection determines which pairs pass into new generations.

The results of running the abstract model under varying conditions show that, regardless of the fitness function and initialization strategies, the system is able to locate optimally sized organizations. The choice of operator changes how quickly the system sizes the first organization containing the optimal number of agents. Using the Join and Cut operator pair led the system to more quickly find the first optimally sized organization. However, in this system the time

for the system to completely converge was not sensitive to organization operator choice. In addition, using the AddOne and SubOne operator pair led the system to gradually converge, while using the Join and Cut operator pair led to a more oscillatory convergence.

Before describing the design of a practical classifier system augmented by organizational operators, it is important to understand the basic design of classifier systems without organization operators. The next chapter discusses relevant current work in classifier systems and details a simple classifier system.

Chapter 3

Classifier Systems

Classifier systems (CS) approach machine learning by combining a production system with a rule-discovery mechanism using a genetic algorithm. This chapter categorizes the two popular approaches, Michigan and Pitt, along a continuum of individual and collective behavior. It then examines some of the advantages and disadvantages of each approach and suggests that there is a benefit to autonomously adjusting the behavior along the continuum. A more comprehensive survey of classifier systems can be found by Wilson and Goldberg (Wilson & Goldberg, 1989; Goldberg, 1989). The chapter ends by introducing a simplified CS that forms the basis of the organizational classifier system (OCS), presented in the following chapter.

3.1 Individual and Collective Approaches to Classifier Systems

A useful way to categorize classifier systems is to examine the fundamental unit on which the genetic algorithm component works. Two methods, each at an extreme along a continuum, are the individual approach and the collective approach. At one end of the spectrum, individual

classifier systems map each classifier to an individual in a population of strings. The CS assigns strength (or fitness) through credit allocation schemes, such as the bucket brigade algorithm, while the GA component evolves individual classifiers. At the opposite end of the spectrum, collective classifier systems map the entire production system (the population of classifiers) to an individual in a population of strings. Here, the GA evolves the population of production systems, evaluating each individually against a problem environment. Traditionally, systems that follow the first method are called *Michigan-style* classifier systems (Holland, 1971), while systems associated with the second method are called *Pitt-style* classifier systems (Smith, 1980).

Both of the above approaches have benefits and difficulties. Classifier systems that exploit individual behavior have the advantage of a temporal credit allocation scheme (the bucket brigade algorithm) that enables the system to efficiently learn which rules to fire within the production system. Because credit is efficiently allocated, rule systems often adapt quickly. However, in practice, hard problems are not solvable because the bucket brigade fails to correctly capture the behavior of interacting classifiers. Complex behaviors such as building long rule chains, default hierarchies, and overcoming parasitic classifiers are difficult to learn (Riolo, 1987a; Riolo, 1987b; Westerdale, 1989; Smith, 1991).

On the other hand, classifier systems based on collective approaches make direct use of the GA to evolve entire production systems. In practice, these systems solve more difficult problems then the individual-based classifier systems. However, collective systems face two problems: large computation requirements needed to solve hard problems and the limited feedback bandwidth inherent to the GA (Grefenstette, 1987). Figure 3.1 shows a simple chart describing tradeoffs of the two approaches.

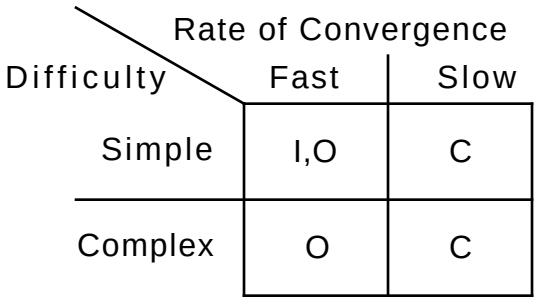

Figure 3.1: Tradeoffs between approaches to classifier systems. Individual (I) approaches converge quickly on simple problems. Collective (C) approaches converge slowly but are able to solve more complex problems. The proposed organizational (O) approach should solve complex problems and converge relatively fast.

Ideally, a classifier system would exploit both individual and collective learning behaviors. This thesis presents a concept for a classifier system capable of exhibiting both behaviors. The idea is to build a classifier system that simultaneously evolves groups of classifiers. These organizational groupings vary in size and interact with each other. Building from an individual-based approach, the organizational classifier system (OCS) presented in the next chapter allows individual classifiers to form organizations that are subject to similar evolutionary pressures as the classifiers themselves. The following section presents the simple classifier system that will form the point of departure for the OCS.

3.2 A Simple LCS

This section presents a detailed description of a simple classifier system (SCS) based on an individual approach. It captures many basic elements found within many Michigan-style classifier systems. It starts from models presented by both Goldberg and Smith (Goldberg, 1989; Smith, 1991).

The simple classifier system interacts with an outside environment which defines the learning problem. During each cycle, the environment outputs its current state, while accepting an input signal from the classifier system. Some actions will cause the environment to output an additional reward signal, while others will not. The goal of the classifier system is to learn a set of rules that over time cause the environment to signal the maximum reward.

The simple classifier system contains three components:

1. A production system

2. Credit allocation and conflict resolution schemes

3. Rule discovery through a genetic algorithm

The remainder of this section describes each component in detail and provides a summary of the steps performed when running the system.

The Production System

The production system consists of a collection of rules (termed classifiers) and a message board. Figure 3.2 shows the production system interacting with an environment. Each classifier contains a condition part and an action part. The condition part includes both internal and external (environmental) conditions, while the action part includes messages to be sent to both the message board and the environment. Classifiers take the following form:

IF *conditions* THEN *actions*

The message board contains two messages, internal and environmental. Posted by the classifier that most recently fired, the internal message represents internal memory. The external message, posted by the environment at the start of each cycle, represents the environment's

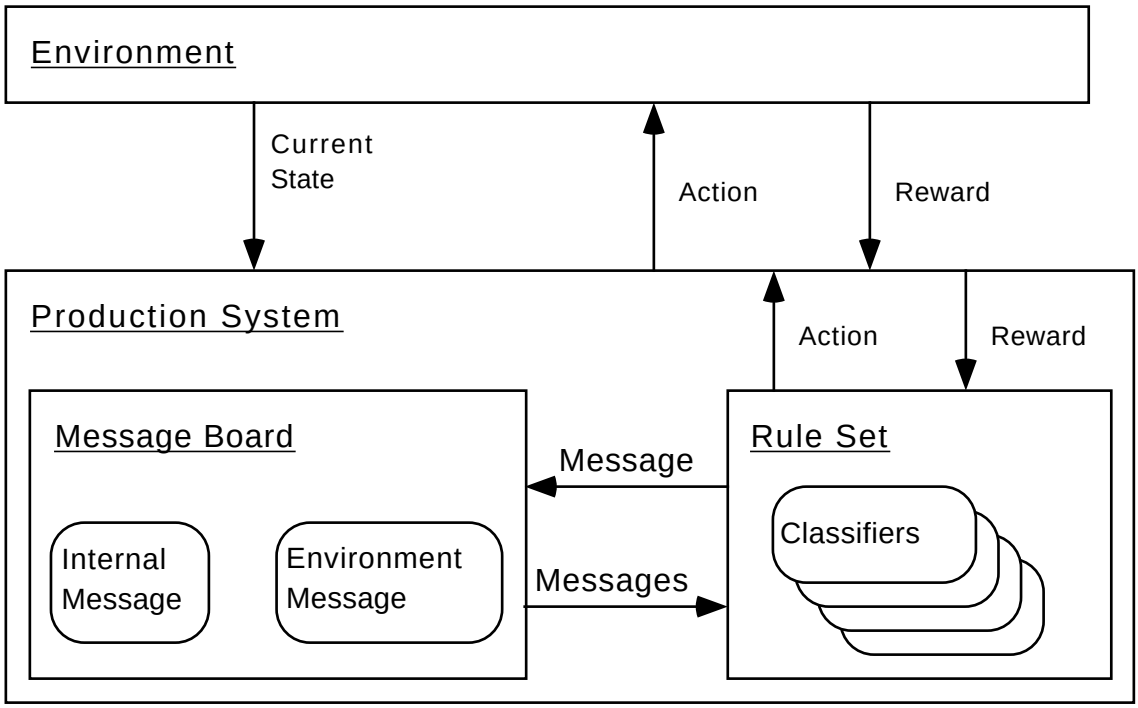

Figure 3.2: Overview of the simple classifier system (SCS). The goal is to maximize the reward signal output by the environment. Classifiers in the rule set match both messages that have been posted by other classifiers and those posted by the environment. Credit allocation and conflict resolution determine which matching classifiers fire. Classifiers selected tofire post internal messages and send action messages to the environment, resulting in reward signals.

Phenotypical	Genotypical
IF AB THEN BD	0001\$0111

Table 3.1: Two ways of representing the same classifier. Note that the symbol \$ separates the conditions from the actions in the genotypical representation.

current state. Messages can be represented both by phenotype and genotype. For example, if a message's phenotype is one of four letters (A, B, C, D), the genotype would be the concatenation of two bits ({00}, {01}, {10}, {11}).

During each cycle, the production system performs pattern matching between the condition part of classifiers and messages already posted to the message board. Conflict resolution (described later) selects a single classifier to fire. When firing, a classifier sends an internal message to the message board and an action message to the environment. Table 3.1 shows, in both phenotypic and genotypic forms, an example of a classifier whose conditions match an internal message A and the current state B and whose actions post an internal message B and send the action message D. The environment uses the action message as input and may output a reward signal based on that message.

Next, the section examines credit allocation and conflict resolution schemes which control which classifiers are selected to fire.

Credit Allocation and Conflict Resolution Schemes

Credit allocation and conflict resolution schemes control how the production system selects which classifiers should fire. In addition, credit allocation assigns a utility measure to each classifier, which the GA component uses as a classifier's fitness.

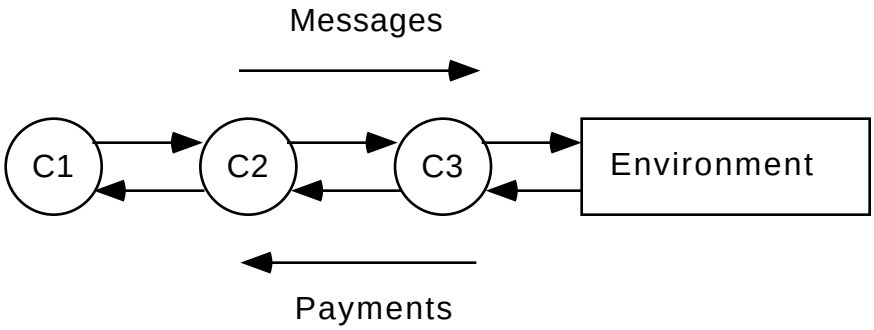

Figure 3.3: Representation of the bucket brigade process. Classifiers, represented by circles, post messages when their conditions are met by matching previously posted messages. The arrows from left to right show the flow of messages. At the same time, strength flows from right to left. In order to fire, a classifier pays a fraction of its strength to the previous classifier. The last classifier to fire in the chain receives strength as a reward from the environment.

Credit allocation distributes reward among those rules that either directly or indirectly cause the environment to signal a reward. The simple classifier system uses the bucket brigade algorithm, developed by Holland (Holland, 1971). The idea is similar to the notion of a chain of people passing buckets of water to each other with the goal of extinguishing a fire. Classifiers also form chains when solving a problem. Each classifier posts a message to be read by the next member of the chain, while matching a message posted by the previous member. Whenever a classifier fires, it must pay a fraction of its strength to the classifier that posted the previous internal message. In addition, when a classifier's action causes the environment to signal a reward, the classifier's strength increases by the amount of the reward.

Figure 3.3 shows the bucket brigade process. The three classifiers form a rule chain that lead to an environmental reward. In the figure, each circle represents a classifier. First, classifier C1 fires, posting a message that allows classifier C2 to fire during the next cycle. Before C2 fires, it pays a fraction of its strength to classifier C1.

Similarly, C3 must pay a fraction of its strength to C2 before firing. Classifier C3 sends an action message to the environment, which responds by outputting a reward signal that augments C3's strength. When the rule chain fires again, a fraction of the original reward moves from C3 to C2. If the system fires the rule chain repetitively, the reward will eventually be distributed among the three classifiers. Multiple reward signals cause the strength in the rule system to grow without bound. To limit strength, the SCS applies a tax to each classifier at the end of a cycle. This is likely to cause ineffective rules to have smaller strength. Credit allocation also plays an important role in the rule-discovery mechanisms described later in this section.

Conflict resolution uses the strength values assigned during credit allocation to determine which classifiers should fire when two or more classifiers match the same environment state and internal message. In order to reduce premature convergence, it calculates a noisy bid for each matching classifier and selects the one with the highest bid. The noisy bid, developed by Goldberg, combines a deterministic bid (a fraction of classifier's strength) and some Gaussian noise (Goldberg, 1989).

Rule Discovery Using a Genetic Algorithm

Rule discovery usually occurs through the use of a genetic algorithm. The classifiers contained in the production system make up the population of individuals that the GA evolves. A classifier's fitness has conventionally been defined to be its strength value, determined through credit allocation. Because this thesis focuses on learning within the production system, the SCS does not make use of a rule-discovery mechanism.

The Process of Running the Simplified Classifier System

The following summarizes the steps of running the SCS through one cycle.

- The environment posts the current state to the message board
- The production system creates a list of potential classifiers that match both the environment message and the internal message
- Conflict resolution selects one classifier from the potential list to fire
- Credit allocation transfers a fraction of strength from the selected classifier to the classifier that previously fired
- The selected classifier fires by posting an internal message and sending an action to the environment
- If the environment signals a reward, it augments the firing classifier's strength by the amount of the reward
- At the end of the cycle, the production system applies a tax to each classifier, reducing its strength

The classifier system completes the above steps during each cycle. The system runs until some stopping criterion is met. For example, the environment reaches a stopping state or the system completes a fixed number of cycles.

3.3 Summary

Most classifier system research efforts treat classifiers as individuals among a population or as part of a collection of classifiers. The Michigan-style classifiers take the individual approach,

using a GA to evolve distinct classifiers. While rapidly converging to promising solutions, these systems often fail to find optimal rule sets for hard problems. The Pitt-style classifier systems take the collective approach, using a GA to evolve a population of production systems each containing a fixed number of classifiers. While able to solve harder problems, these systems are computationally expensive. Ideally, a classifier system would autonomously evolve a production system containing both individual and collective structures. The degree to which individual or collective behavior exists would be determined by the nature of the problem.

Recognizing that classifier systems are very complex and difficult to analyze, the chapter ends by detailing a simple classifier system (SCS). Based on an individual approach, the SCS keeps track of a population of classifiers that interact with an outside environment. Classifiers post messages to a message board and send action messages to the environment when matching both an environment state and an internal message from the message board. Conflict resolution selects which classifiers fire by using a noisy bidding. Credit allocation assigns strength based on the bucket brigade algorithm.

The next chapter uses organizational operator techniques from Chapter 2 to modify the SCS to form organizations of classifiers autonomously. It compares the SCS to an organizational classifier system with a common problem environment.

Chapter 4

Autonomous Organizational Learning

This chapter contrasts learning within a simple classifier system (SCS) to learning within an organizational classifier system (OCS). Although able to solve simple stimulus-response problems, classifier systems evolving individual classifiers are often unable to learn ideal rule sets when problem difficulty increases. On the other hand, collective approaches to classifier systems may be able to solve harder problems, but are computationally expensive. Organizational learning is the process of autonomously forming groups of individual classifiers which act in a coordinated fashion.

Forming good organizations is easier said than done. When viewed from the perspective of a single rule, the question of whom to trust (whom to answer and whom to pay) is the crucial one. In Chapter 2, the question was viewed in the light of transaction cost theory; and those ideas are carried over to the classifier systems here. Specifically, we use reputation for organizational recruitment, and we pay attention to efficient organization sizing.

The work here builds on Smith's work on parasitic classifiers, using his problem as a test bed for new mechanisms (Smith, 1991). The problem requires classifier systems to remember previous environment states in order to send the correct actions to the environment. Careful analysis of this problem shows three types of parasitic classifiers. This study finds that, in a simple classifier system (SCS), as the number of parasites to ideal classifiers increases, performance rapidly decreases. Testing the organizational classifier system (OCS), using reputation values and organizational operators on the same problem, shows improvement, particularly in weeding out many of the parasites that stymied the SCS.

The chapter starts by introducing the notion of parasitic behavior intuitively. Section 4.2 then describes a test environment where parasitic classifiers are prevalent. Section 4.3 runs the SCS on the test environment and summarizes the results. Afterwards, the chapter analyzes deficiencies of using bucket brigade strength for conflict resolution. Two reputation values, short-term and long-term, are introduced to replace traditional strength values. Section 4.5 describes the design of the OCS, while the following section shows the results of running the OCS on the test problem. The chapter analyzes the results in Section 4.6. A further section indicates directions for further research. Finally, the chapter ends with a summary.

4.1 A Discussion on Parasitic Behavior

This section provides a discussion of parasitic behavior from two viewpoints. The first part looks at human society to find an intuitive definition for a parasite. We gain insight into the OCS design by noting the mechanisms society has evolved to deal with parasitic behavior. The second part looks at parasitic behavior within a classifier system.

Parasitic Behavior from Society's Viewpoint

A parasite benefits from an exchange of goods or services without making a useful or beneficial contribution. For example, using the manufacturer and worker example from Chapter 2, the manufacturer hires a worker to build a widget. If the manufacturer pays the worker and the worker does not build the widget, an exchange occurs where one party's contribution is not beneficial. Thus, this worker is a parasite (at least from the manufacturer's viewpoint). The problem of discovering parasitic behavior becomes more difficult as the number of parties involved increase. For example, if the manufacturer hires a team of workers to build widgets, a parasitic worker may be able to hide behind the success of the group.

Some parasites exhibit charlatan-like behavior. They advertise that they will provide high-quality goods or services when in fact they deliver less than advertised. For example, practitioners of mail fraud make products look very attractive on paper, when the product is either cheap or non-existent. Another example may occur when hiring a new employee into a company. The employee may have a wonderful resume and great references. However, once hired, the employee may not perform well or, worse, may cause someone else in the company to perform poorly. The former case can be handled by identifying and firing the poor performer, but the latter case is difficult to address.

How does society deal with parasites? In early societies, members of a tribe could bring complaints to a chieftain. Fear of punishment kept tribal members in line. Today, courts and laws act to protect previously established rights. Failure to abide by the rules can result in fines or jail sentences. Established markets, such as stock exchanges, incorporate rules that threaten to expel members when violated. Other mechanisms such as word-of-mouth reputation and consumer reports also prevent parasitic behavior by alerting people to a parasite's history.

Although the above mechanisms can prevent many acts of fraud, they will not prevent subtler forms of parasitic behavior. In these instances, selection may take place at the level of later organization competition.

Parasitic Behavior from the Classifier System Viewpoint

Classifier systems face similar parasitic behavior among classifiers as does society among people. If we consider the production system of a CS to be a market, classifiers transact with each other by exchanging information and strength. Classifiers selling information post an internal message to the message board, while classifiers buying the information use the posted message as indication that it is a good time to fire. An example of a parasite would be one that sells information causing useful rules (ones that help gain reward from the environment) to fire during inappropriate situations (ones that do not lead the system to gain environmental reward). Because we are interested in maximizing the reward received from the environment, it may also be useful to consider any a parasite classifier that does not lead the system to gain the maximum possible reward.

How well does a conventional individual-based CS discover parasitic classifiers? Classifier systems built so far do not contain many of the mechanisms used in society to determine parasitic behavior. If the classifier system makes use of the internal message board and the problem is not just stimulus-response, the system needs to develop rule chains. In this case, if the environment does not signal maximum reward, it becomes very difficult to determine which rule is performing poorly. The problem could be with the current rule or with the one that posted the previous message. Worse, it could be with any classifier that posted any previous message. To determine which classifier to fire, the system generally selects the classifier with the

most strength. However, charlatan parasitic classifiers often gain undeserved rewards. These classifiers fail to lead the system to maximum reward even though they are strong.

The next section provides examples of parasitic classifiers within a problem environment.

4.2 Testing Environment

This section presents a testing environment that will allow us to explore the ability of a classifier system to distinguish parasitic classifiers. In his dissertation, Smith (1991) used this testing environment to explore how classifier systems behave when tackling problems that require memory-depth greater than zero (i.e., not stimulus-response). The results show that parasitic classifiers can and do cause the traditional classifier system to fail to learn optimal solutions. This section begins by introducing memory-depth problems. It follows by describing the memory-depth-one problem used in this thesis. Finally, the section ends by introducing the concept of working sets and describing the parasitic nature of some classifiers.

An Introduction to Memory-Depth Problems

A memory-depth problem is one where the system must know something about the past in order to select an action for the present. The nature of these problems is that classifiers must interact in a coordinated fashion to achieve maximum reward. For example, one classifier might post an internal message which indicates which classifier should fire next. Other related problems require learning rule-chains, but do not require the system to keep track of previous environment states. For example, a system might require that one rule fires after the next without regard to the environment state in-between. Thus, an important aspect of memory-depth problems is

the need for classifiers to use both the environment state and an internal message to determine the next correct action.

The Memory-Depth-One Problem

The testing environment used in this thesis is a memory-depth-one problem, where the system need only keep track of the environment state from the previous time step. The CS stores the environment's state information in the internal message, enabling the system to remember the previous environment state.

Consider classifiers with the following form:

IF *current-state message* & *previous-state message*

THEN *next-state message* & *action message*

To fire, a classifier matches both the current-state message (posted by the environment) and the previous-state message (stored as an internal message). When a classifier fires, it posts a new internal message (next-state message) and sends an action message to the environment.

In the problem specification, the environment keeps track of two state variables, current and previous. At the start of each cycle, the environment randomly generates a new current state, setting the previous-state value to the old current-state value. If the production system sends an action message that corresponds to the value of the previous state, the environment signals a reward.

Specifically, the environment states are one of four values (in phenotype terms): a, b, c, or d. Similarly, messages are one of four corresponding values: A, B, C, or D. An example classifier is shown below:

IF A & B THEN A & B

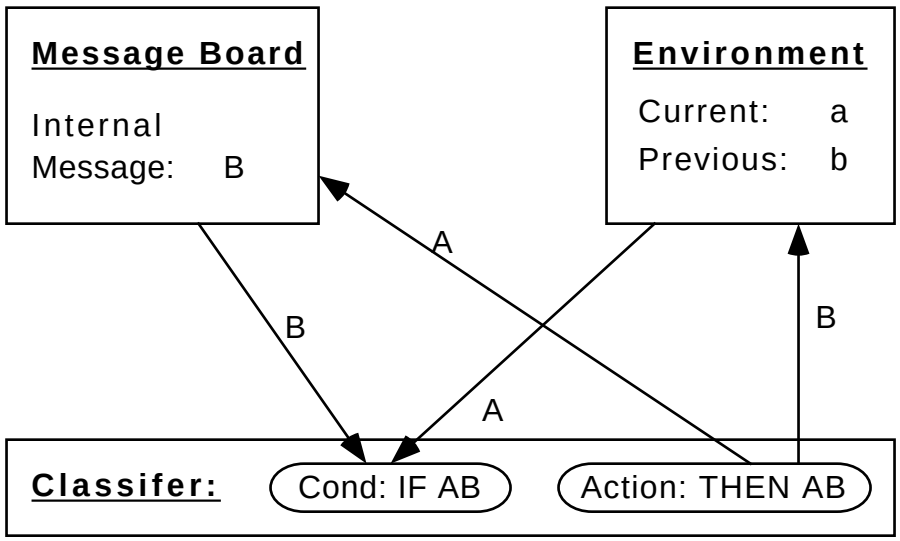

Figure 4.1: The classifier matches the current-state message A and the previous-state message B. When it fires, it posts the next-state message A (replacing the old previous-state message) and sends the action message B.

The rule says if the current-state message equals A and the internal message equals B then post a new internal message A and send the action message B to the environment. Figure 4.1 shows the example classifier's interaction with a message board and the environment. Because the environment's previous state 'b' corresponds to the action message B, the environment would signal a reward if the classifier were to fire.

A quick examination of a classifier's structure shows that there are 16 unique conditions (4 current-states combined with 4 previous-state messages) and 16 unique actions (4 next-state messages combined with 4 actions) leading to 256 unique classifiers.

Next, the section examines the nature of the above classifiers within a CS.

Working Sets and Types of Classifiers

As a classifier system interacts with an environment, it learns a subset or working set of the classifiers contained within the production system that are regularly chosen to fire. In other words, the system learns which rules should fire during conflict resolution situations. Unlike rule-chains, the order in which classifiers are fired within a working set is not specified. Ideally, a classifier system would learn the working set that causes the environment to signal the maximum reward. However, with rule-discovery turned off (as it is in this thesis), the goal of the system is to learn the working set that achieves the maximum reward, given a particular population of classifiers.

Using the above environment, a CS will ideally learn rules for each of the 16 unique conditions. A complete working set contains rules reacting to each unique condition. Table 4.1 shows one possible complete working set of 16 classifiers capable of obtaining the maximum reward from the environment. The order in which the classifiers are fired depends on the current-state messages posted by the environment.

Two factors influence a working set's ability to obtain reward. First, the working set may contain an incomplete set of rules. Table 4.2 shows a working set lacking rules that post next-state messages using C or D values. The working set is not complete because it does not address all possible situations. Second, the system may incorporate in the working set parasitic classifiers, which lead the system to obtain less than the maximum reward. Again, examining Table 4.2, the last four rules post next-state messages that cause the next classifier that fires to send an incorrect action message. Next, the section examines the nature of individual classifiers that make up a working set.

Conditions		Actions	
Current State Message	Previous State Message	Next State Message	Action Message
A	A	A	A
A	B	A	B
A	C	A	C
A	D	A	D
B	A	B	A
B	B	B	B
B	C	B	C
B	D	B	D
C	A	C	A
C	B	C	B
C	C	C	C
C	D	C	D
D	A	D	A
D	B	D	B
D	C	D	C
D	D	D	D

Table 4.1: Working set of 16 classifiers capable of obtaining the maximum reward from the environment.

Conditions		Actions	
Current State Message	Previous State Message	Next State Message	Action Message
A	A	A	A
A	B	A	B
B	A	B	A
B	B	B	B
C	A	A	A
C	B	B	B
D	A	A	A
D	B	B	B

Table 4.2: Working set of 8 classifiers, which fail to cover the 16 possible situations.

Conditions		Actions	
Current State Message	Previous State Message	Next State Message	Action Message
A	A	B	B
A	B	B	A
A	C	B	C
A	D	B	D
B	A	A	B
B	B	A	A
B	C	A	C
B	D	A	D
C	A	C	B
C	B	C	A
C	C	C	C
C	D	C	D
D	A	D	B
D	B	D	A
D	C	D	C
D	D	D	D

Table 4.3: Working set of 16 classifiers capable of obtaining the maximum reward from the environment. Notice that when previous-state messages indicates an A or B, the classifier sends a B or A action message respectively. The working set obtains the maximum reward, because whenever the current-state message is A or B, the rules post B or A next-state messages respectively.

A quick comparison between the classifiers of Table 4.1 and the classifiers of Table 4.2 shows that some classifiers differ in the quality of information posted to the message board. Examination of possible classifiers reveals three types of parasitic classifiers in relation to a particular working set. That is, parasitic behavior is context dependent. Whether a classifier is parasitic may depend on the working set to which the CS is closest to convergence. For example, Table 4.3 shows a complete working set which contains some classifiers that would be parasitic to the working set of Table 4.1. In particular, the classifiers that react to conditions containing A or B previous-state messages would cause some rules in the Table 4.2 to fire incorrectly.

Classifier Type	Conditions		Actions	
	Current State	Previous State Message	Next State Message	Action Message
Ideal	A	B	A	B
1	A	B	A	C
2	A	B	D	B
3	A	B	D	C

Table 4.4: Example types of classifiers relative to an ideal working set.

The following definitions of parasites are relative to a set of ideal classifiers (in that they are part of a complete working set that achieves the maximum reward) shown in Table 4.1:

Type 1 Parasites These classifiers post the correct next-state message but send an incorrect action message to the environment

Type 2 Parasites These classifiers post an incorrect next-state message, while sending the correct action message to the environment

Type 3 Parasites These classifiers post the incorrect next-state message and send an incorrect action message to the environment

Table 4.4 shows an example of the four possible classifications for classifiers. Within a classifier system, each of the presented classifiers would compete during conflict resolution because their conditions are the same. The system, however, may or may mot be able to locate the ideal rule to fire.

The next section describes the results of running the simple classifier system (SCS) on the test problem environment under a number of different initial settings.

4.3 Running the SCS on the Test Environment

This section presents the results of running the simple classifier system (SCS) presented in Chapter 3 on the test environment described in the previous section. The experiments consist of seven tests each initializing the system with a different set of classifiers. Three criteria are used to examine a system's ability to achieve reward and converge under these test situations. Ideally, the SCS would always find a complete working set of 16 ideal rules capable of achieving the maximum environmental reward. Indeed, as long as the ratio of parasitic to ideal classifiers is small, the system performs well. However, as the number of parasites increase, performance degrades rapidly. The section begins by describing the seven tests. Then, it presents the performance criteria. The section closes by discussing the results of running the SCS.

Description of Experiments

The purpose of developing a testing environment is to provide a common problem within which to evaluate the performance of different classifier systems. Because the SCS and upcoming OCS do not use a rule-discovery mechanism, it is necessary to simulate how the system behaves given different initial rule sets in order to evaluate their performance. In each of the tests below, the members of the complete working set shown in Table 4.1 are part of the initial rule set. By ensuring that a complete working set (capable of achieving the maximum reward) is part of each test, a classifier system's performance is comparable to an ideal standard. The following list describes seven tests using different numbers of ideal and parasitic classifiers in the population:

Test 1 The 16 individual ideal classifiers necessary to achieve the maximum reward

Test 2 16 ideal classifiers and 1 Type 2 parasite

Test 3 16 ideal classifiers and 8 Type 2 parasites

Test 4 16 ideal classifiers and 5 Type 1, 6 Type 2 and 5 Type 3 parasites

Test 5 16 ideal classifiers and 16 Type 2 parasites

Test 6 32 ideal classifiers and 32 Type 2 parasites

Test 7 16 ideal classifiers and 32 Type 2 parasites

The tests were designed to explore three aspects of problem difficulty. The following list describes the three problem aspects:

Parasite-to-Ideal Ratio The proportion of parasitic classifiers to ideal ones indicates both the amount of noise in a system and the degree to which a classifier system must overcome the parasite problem. Tests 1, 2, 3, 5, and 7 vary the ratio of parasites to ideal classifiers from zero to two. We expect that performance of both the SCS and OCS will degrade as the ratio increases. However, the OCS may degrade less quickly.

Parasitic Mix Three types of parasitic classifiers are defined in the previous section. Test 4 examines how the system behaves when faced with parasites from each type. One might naively expect that Type 3 parasites cause the most difficulty because they fail to both post the correct next-state message and send the correct action. However, because of the greater signal difference between a parasite and an ideal classifier, the system should be able to more easily detect and isolate the Type 3 parasitic classifiers. Thus, depending on system parameters, either the Type 1 or Type 2 parasites should be most difficult to distinguish.

Scale Adding classifiers to the system increases the amount of noise that must be overcome. Comparing Test 5 and 6 will show how a system behaves when the number of classifiers doubles. In addition to distinguishing between ideal and parasitic classifiers, the system must select a subset of ideal classifiers to be part of the working set. Because competing organizations isolate and weaken groups of under-performing classifiers, the OCS may handle the increase in numbers more gracefully than the SCS.

Next, the section examines different ways of measuring the performance of a CS on the above tests. This will serve as a baseline for examining OCS improvements.

Measuring Performance

The criteria used to measure the performance of CS on the above tests is described in the list below:

Percent Correct This is the percentage of actions sent to the environment that resulted in a reward signal. It measures a classifier system's ability to achieve the goal of maximizing environmental reward.

Number of Mistakes This is the number of parasitic classifiers that have substantial strength (strength > 1.0) at the end of a run. If the CS has converged and contains a working set, number of mistakes indicates the number of parasitic classifiers that succeeded in fooling the system. However, if the system has not converged, this number could be artificially high or low and loses some meaning.

Convergence Time This is an approximate evaluation of the amount of time for the system to converge on a steady percent-correct score. Time is measured by the generation number,

Test	Percent Correct	Number Of Mistakes	Convergence
1	100	0	0
2	100	0	20
3	63	6	4000
4	72	5	3000
5	51	6	7000
6	47	4	40000
7	37	6	10000

Table 4.5: Summary of SCS performance - Notice that for later tests, percent correct decreases without a corresponding increase in the number of mistakes.

the number of opportunities the production system has to send an action message to the environment. An alternative criteria not examined here would be the convergence time for the system to find a stable working set. However, it is possible that the an unstable working set leads to stable percent-correct performance (imagine two identical classifiers competing to be selected).

The above criteria allow us to evaluate a classifier system's performance both by the quality of solutions and speed of convergence. The section closes by examining the performance of the SCS on the test environment. In a later section, these measures are also used to evaluate the performance of the OCS.

SCS Performance

Ideally, the SCS would always find a set of 16 ideal rules necessary to achieve the maximum reward from the environment. However, as the number of parasites increases, performance degrades. Table 4.5 summarizes the results in terms of the performance criteria mentioned above. Below we examine the SCS's performance on all three aspects of problem difficulty discussed earlier.

The first problem aspect examines changing the ratio of parasitic to ideal classifiers. The hypothesis is that increasing the ratio would degrade performance. As expected, the system achieved lower percent-correct scores as the ratio increased in tests 1, 2, 3, 5, and 7. The time to convergence also appears to increase with the ratio. Interestingly, the number of mistakes does not change among tests 3, 5, and 7. Close examination of the classifiers at the end of the runs shows the percent-correct score is somewhat dependent on which classifiers are parasites. If two Type 2 classifiers generally fire right after one another, the effect is less damaging than if they each separately fire before ideal classifiers. Thus, the decrease in percent correct between Test 3 and 5 occurs because the parasites learned during Test 5 are more damaging. However, this phenomenon does not explain the poor performance resulting during Test 7. There, the system fails to learn a stable working set even though the percent correct converges.

The second problem aspect examines the mix of different types of parasitic classifiers. The hypothesis is that Type 1 or 2 parasites would be more difficult to recognize. The results of Test 4 shows the performance of the SCS on a set of classifiers with nearly equal numbers of each type of parasite. The system learns a complete working set containing only ideal and Type 2 parasitic classifiers. That is, it correctly weeds out Type 1 and 3, but not Type 2, parasites. Upon further inspection, it appears that Type 2 parasites are harder to distinguish because Type 1 and Type 3 parasites fail to obtain reward from the environment. When a classifier fires for the first time, its strength is determined by environmental reward and payment from the next firing classifier. The amount received through payments from the next classifier is independent (at least early in the run) of which Type of classifier is firing. Type 1 and Type 3 classifiers do not gain reward from the environment, while Type 2 and ideal classifiers do gain reward directly from the environment. Therefore, at the beginning of a run, Type 2

parasites and ideal classifiers appear to be equally useful. Type 2 parasites survive because their disruption of downstream classifiers is hard to identify. It appears that Type 2 parasites act with charlatan-like behavior.

The third problem aspect examines the performance of the CS when scaling up the number of classifiers. The hypothesis is that the addition of classifiers will increase noise, resulting in worse system performance. As expected, doubling the number of classifiers while maintaining the proportion of parasitic to ideal classifiers from Test 5 to Test 6 decreases performance. The most dramatic difference between the two runs is the large increase in convergence time. The SCS under Test 6 used over 4 times the number of generations to converge as in Test 5. Closer observation of the run shows that the SCS also failed to find a stable working set, explaining the poor percent-correct score and the relatively low number of mistakes.

The following list summarizes the performance of the SCS across the different problem aspects:

- Increasing the ratio of parasitic classifiers to ideal classifiers decreases performance
- Type 2 parasitic classifiers are more likely to become part of the working set than Type 1 and Type 3 classifiers because of their charlatan-like behavior
- Increasing the number of classifiers results in a decrease in system's ability to converge on a working set

The performance of the SCS on the problem environment shows that parasitic classifiers and increases in noise levels (due to scaling) do prevent the SCS from achieving ideal performance. The rest of this chapter is devoted to examining an organizational classifier system (OCS) designed to overcome some of the shortcomings found in the SCS through autonomously learning

collective structures. The next section looks at improving the use of reputation within a classifier system.

4.4 Using Reputation to Distinguish Parasitic Classifiers

Under a number of different initial conditions, the simple classifier fails to locate parasitic classifiers. This section suggests a mechanism based on reputation to differentiate parasitic classifiers from ideal ones. Traditional classifier systems often use a single value, strength, to predict a classifier's future performance. By using multiple values reflecting different time scales, a CS may be able to distinguish the charlatan-like behavior of some parasites. The OCS, presented in the next section, uses two reputation values, a long-term and short-term strength, to help overcome the parasite problem. Before examining parasitic detection, this section provides an intuitive explanation of reputation.

One definition for reputation is any stored information about previous performance. Thus, the reputation of a car manufacturer can be captured by people's opinion about previously manufactured vehicles. As stated in Chapter 2, people are motivated to use reputation values to reduce transaction costs. Reputation, acting as a predictor of future performance, can reduce uncertainties during a transaction. For example, instead of researching the quality of each particular car, a buyer relies on published consumer reports. Different types of reputation attributes capture different types of information. In particular, some values examine the same quality over different lengths of time. For example, a buyer may be interested in a car's achievable gas mileage during the first, second, and tenth years. A car with a high first year value may or may not perform well over time. The buyer must carefully examine the meanings

of different reputation values to make the best decisions. Next, the section examines how classifier systems use reputation during conflict resolution.

A useful way of studying classifier systems is to view a classifier's strength as a type of reputation indicating future performance. Conflict resolution uses strength to resolve which classifiers should fire, while credit allocation through the bucket brigade algorithm updates strength values. Strength is a reputation value indicating the amount of reward that a classifier is predicted to bring into the system. Thus, conflict resolution uses reputation to reduce the risk of firing less productive classifiers. Unfortunately, the long-term nature of strength values conflicts with the short-term goals of conflict resolution, as described next.

Conflict resolution attempts to select those classifiers that maximize reward from the environment. Ideally, conflict resolution would select those classifiers that maximize performance over the lifetime of the system. To this effort, conflict resolution uses a classifier's strength as an indication of performance over many generations. However, as the system changes (both through credit allocation and rule-discovery), there is a need to explore alternative selections. Conflict resolution has a short-term goal to find the highest fit classifier for the current situation, including the current classifier in the working set. At the same time, the system has a long-term goal to find a set of classifiers that achieves the maximum reward possible given the entire population of classifiers. Unfortunately, strength often changes so gradually that it is a poor indicator of short-term performance. Thus, conflict resolution may select a strong rule that is unable to cope with the current state. In particular, charlatans (a type of parasite) gain strength early while performing poorly over the long-term. Ideally, conflict resolution would select those classifiers that would maximize the short-term rewards and provide good perfor-

mance over the long-term. Next, an improvement to credit allocation and conflict resolution to better distinguish charlatan-like behavior within an OCS is introduced.

The OCS presented in the next section uses two reputation values based on short-term and long-term performance. Both organizations and classifiers carry the two reputation values. From the classifier viewpoint inside an organization, conflict resolution selects which classifiers should fire based on the most recent performance. A short-term reputation that will be called *ST reputation* allows conflict resolution to use a greedy scheme, maximizing performance for current state of the system. A long-term strength that will be called *LT reputation* is used to determine which classifiers stay members of a particular organization. Using the LT reputation increases the long-term quality of classifiers found within good organizations. Thus, conflict resolution can make greedy decisions using the ST reputation with greater confidence that the selected classifiers will not be parasitic. By using a long-term value to determine members of an organization, the system has greater trust or confidence that the greedy-like decisions by conflict resolution will select classifiers among those that will likely have sustained performance. From the organizational viewpoint, conflict resolution uses a long-term reputation, also called LT reputation, to determine which organizations should affect the environment. Thus, only those organizations that have sustained performance affect the environment. Similarly, organizations grow based on their short-term reputation, ST reputation. Those organizations which are currently performing well have the opportunity to grow and expand their working set.

The next section describes the OCS that combines both the additional reputation values and organizational structures to improve performance over the SCS.

4.5 Organizational Classifier System

The organizational classifier system (OCS) autonomously learns organizational structures by using both organizational techniques introduced in Chapter 2 and reputation values introduced in the previous section. The main idea of the OCS model is to separate the 'good' rules, those that lead to optimal decisions, from the 'bad' rules, those that lead to sub-optimal decisions. In essence, the system simultaneously learns competing working sets. This section provides the implementation details of the OCS, while the next section discusses the experimental results of running the OCS on the test environment from Section 4.2.

The OCS consists of the following three components:

- The production system
- Credit allocation and conflict resolution schemes
- Organizational growth component

The first two are similar to the components of the SCS, while the third is an additional learning mechanism relating to organizational growth. The rest of this section describes each component and the way it differs from the SCS.

The Production System

The production system has similarities to aspects of both to Pitt-style and Michigan-style classifier systems. Like Pitt-style systems, the OCS contains a population of organizations containing classifiers. However, OCS organizations differ in two ways: they are of variable size and they interact with each other. The organizational growth component controls how organizations are sized, while the production system uses credit allocation and conflict resolution

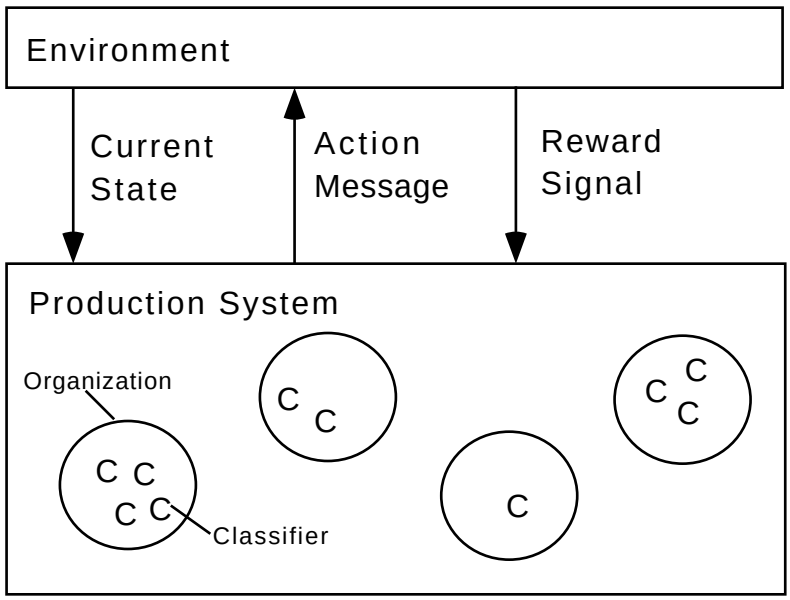

Figure 4.2: The OCS production system interacting with the environment.

to control classifier and organization interactions. The production system, similar to Michigan-style systems, pattern matches classifiers to internal message messages and environment states. Unlike the Michigan-style systems, each organization contains its own message board to which classifiers contained in the organization post. While classifiers only post to the organization containing them, they can read internal messages from any organization. Figure 4.2 shows an overview of the production system.

Credit Allocation and Conflict Resolution

Credit allocation assigns reputation to both classifiers and organizations, while conflict resolution uses reputation to determine the interactions of the classifiers and organizations. The schemes used here differ from the traditional Michigan-style systems in that each classifier and organization carries two values, ST reputation and LT reputation. Below the section examines

how credit allocation assigns reputation to each classifier and organization. Later, the section examines how conflict resolution uses the reputation values.

Credit Allocation

First, a classifier's reputation is determined by its success at obtaining reward, both through direct environment signals and through payments from other classifiers. An important part of a classifier's reputation is the bucket brigade payment of LT strength made by other classifiers using the original classifier's posted internal message. In the equations below, the curly braces enclose a set of LT reputation values for classifiers that fire after reading the posted internal message.

A *classifier's ST reputation* measures a classifier's most recent performance. The value for the ith classifier to fire is calculated as follows:

$$V^i_{STC} = R + b\sum\left\{V^{i+1}_{LTC}\right\}$$

The R is the reward received from the environment and $b\sum\left\{V^{i+1}_{LTC}\right\}$ is a fixed fraction of the sum of LT reputation values for each classifier using the posted internal message.

A *classifier's LT reputation* is similar to the strength values found in Michigan-style systems. Overlooking taxes for the moment, the LT value for the ith classifier to fire is calculated as follows:

$$V^i_{LTC} = (1-b)V^i_{LTC} + b\sum\left\{V^{i+1}_{LTC}\right\} + R$$

Thus, a classifier's LT reputation is determined by the amount that it pays to the previous classifier, the reward received by the set of classifiers using its posted internal message. In addition, the LT reputation is reduced every cycle by a tax T_c (a fraction of LT reputation) paid to the organization containing the classifier; the effect is to diminish the strength of those classifiers that do not fire often.

Next, an organization's reputation values are indirectly determined by the success of its classifiers. The reputation values of an organization differ from a classifier's values in that the values are indications of longer term performance. Thus, the short-term reputation of an organization is akin to the long-term reputation of the classifier.

An *organization's ST reputation* is updated each cycle as follows:

$$V_{STO}^{t+1} = (1 - T_O)V_{STO}^{t} + \sum_{c \in O} T_c V_{LTC}^{c} - F$$

T_O is a system tax, T_c is a tax on each classifier c within the organization O, and F is a fee paid whenever the organization is selected to affect the environment.

An organization's LT reputation is an assessment of performance over the lifetime of the organization. The value is calculated as follows:

$$V_{LTO}^{i} = S_{success}/S_{attempts}$$

The $S_{success}$ is the number of times the organization has caused a reward signal when selected to affect the environment and $S_{attempts}$ is the number of times the organization was selected to affect the environment.

Reputation for	Use
Classifier	
ST	Conflict resolution selects a classifier
LT	Organizational growth decides which classifiers remain in an organization
Organization	
ST	Organizational growth selects which organizations grow
LT	Conflict resolution selects which organization affects the environment

Table 4.6: A summary of the uses for ST reputation and LT reputation for both classifiers and organizations.

Conflict Resolution

Conflict resolution uses the ST reputation for classifiers and LT reputation for organizations. For classifiers, it uses ST reputation to decide which classifier should fire; for organizations, it uses LT reputation to decide which organization should affect the environment. In both cases, conflict resolution selects the classifier or organization that contains the highest reputation. Because classifiers are contained within organizations, the system optimizes the organization's performance in a greedy manner using the short-term value of classifiers. At the same time, conflict resolution limits the organizations that affect the environment to those containing working sets that have proven to perform well over time.

Next, the organizational growth component will use a classifier's LT reputation and an organization's ST reputation when sizing an organization. Table 4.6 summarizes the use for each reputation value.

Organizational Growth Component

The organizational growth operators control the sizing of organizations. The above description of credit allocation and conflict resolution describes how organizations and classifiers interact when selecting a classifier to fire. Here, the section describes how organizations vary their size

and membership. The OCS randomly selects one of two operators, Grow, or Shrink, to apply to each organization in the current population. Each operator attempts to change the size of an organization by either by adding or subtracting classifiers. Now, the section examines the general nature of the operators and then describe each.

Examining the results in Chapter 2 on the organizational growth model shows that using the AddOne and SubOne operators results in a more gradual and less volatile search for appropriately sized organizations. In addition, the results from running the SCS on the test environment show that increasing noise levels through scaling dramatically increases the time needed to converge. Thus, with the hope of keeping disruptions to a minimum, the Grow operator, which increases the size of organizations, moves a single classifier between organizations. On the other hand, the Shrink operator can potentially remove all the classifiers of the organization. This ensures that any classifiers marked as parasites are quickly removed. The resulting bias in favor of shrinkage reflects the difficulty of removing parasitic classifiers as the key challenge for the OCS. Below is a description of each operator.

The Grow operator attempts to increase the size of an organization. It considers an organization's fitness to be a function of both LT reputation and number of employees in much the same was as the organizational growth model uses the profit. It first selects a classifier from the entire population of classifiers not contained in the growing organization. This thesis examines two methods for selecting this recruited classifier. The first method, random growth, randomly selects a classifier from the potential pool. The second method, vertical growth, tries to grow more selectively. Each organization keeps a list of the classifiers that posted internal messages that were matched by its members. From this list, a classifier is selected. Regardless of the method used to select the classifier, the operator then runs a tournament between the two

organizations, the growing one and the container of the selected classifier to be recruited. The organization with the highest salary, calculated as the organization's ST reputation divided by the number of its employees, wins the competition and keeps the selected classifier.

The Shrink operator reduces an organization by removing those classifiers that are performing poorly. Shrink compares each employee's LT reputation to a threshold set near zero. Those classifiers whose strength is below the threshold are removed from the organizations. Each removed classifier joins a separate empty organization.

The following section shows the results of running the OCS on the test environment presented in Section 4.2.

4.6 Running the OCS on the Test Environment

This section presents the results of running a series of experimental OCS models on the tests described in Section 4.2. The results are compared to the performance of the SCS using the same criteria. The section begins by describing the three model variables that the results examine. It then describes the parameter settings for two versions of the OCS that are used to examine OCS performance. It continues by analyzing the results of the OCS on the three problem aspects described in Section 4.2. Finally, the section summarizes the results.

Important Parameters

As with most classifier systems, the OCS is complex and has many parameters to configure. While many variables were examined in the course of the development of the OCS, most changes resulted in similar behavior. The following list describes three parameters which, when changed, resulted in significant performance differences:

Period The period is the number of generations (number of times actions are sent to the environment) between organizational growth cycles. During each test, five period values were examined: 1, 5, 25, 50 and 100. At issue is the time necessary to optimize and evaluate a current organizational structure, enabling the organization growth component to make 'good' decisions. The hypothesis is that increasing the period should improve the OCS's ability to distinguish parasitic classifiers.

Initial LT Reputation When organizational growth removes a classifier from an organization, it places the classifier in a new organization. The initial LT reputation for organizations is the LT reputation assigned to the new organizations. Because conflict resolution selects (to affect the environment) the organization with the greatest LT reputation, the initial setting of the value determines the degree to which these organizations get a chance to affect the environment in the future. If a low value is used, the system is less likely to select the organization. However, if a high value is used, the system will more likely select the organization. Because conflict resolution uses a deterministic approach, it is important to ensure that the system continue exploring alternate paths. Thus, the hypothesis is that increasing the initial LT reputation for organizations will improve performance.

Sizing Strategy The organizational growth component uses two techniques, random growth and vertical growth, to find classifiers to add to organizations. The hypothesis is that using the more selective vertical growth mechanism will improve performance.

Next, the section describes two models of the OCS that are used to demonstrate performance of the OCS on the test environment.

Constant	Value	Definition
R	1.0	Amount of reward signaled by the environment
b	0.1	The fraction of strength passed between classifiers
T_O	0.01	System tax applied to organizations each cycle
T_c	0.01	Classifier tax applied to classifiers each cycle
F	0.1	Fee each organization pays when affecting the environment
Initial LTO	0.95	The initial LT reputation for new organizations

Table 4.7: Summary of the constants and their values used in each run.

Two OCS Models

Because of the plethora of possible versions of the OCS, two models have been selected to demonstrate overall performance. Table 4.7 shows a list of the relative constants and their values used in the two runs. The *Random OCS* has the Grow operator use random growth, while the *Vertical OCS* has the Grow operator use vertical growth.

Next, the section examines the performance of the OCS on the three problem aspects discussed in Section 4.2.

Results

To compare the OCS to the SCS using the performance criteria set in Section 4.2, several versions of the OCS were examined on the tests also described in Section 4.2. Three aspects of problem difficulty were examined in the tests. Below is a brief review of the seven tests and the problem aspects.

The tests are as follows:

Test 1 The 16 ideal classifiers necessary to achieve the maximum reward

Test 2 16 ideal classifiers and 1 Type 2 parasite

Test 3 16 ideal classifiers and 8 Type 2 parasites

Test 4 16 ideal classifiers and 5 Type 1, 6 Type 2 and 5 Type 3 parasites

Test 5 16 ideal classifiers and 16 Type 2 parasites

Test 6 32 ideal classifiers and 32 Type 2 parasites

Test 7 16 ideal classifiers and 32 Type 2 parasites

The problem aspects are as follows:

Parasite-to-Ideal Ratio The proportion of parasitic classifiers to ideal ones indicates both the amount of noise in a system and the degree to which a classifier system must overcome the parasite problem. Tests 1, 2, 3, 5, and 7 vary the ratio of parasites to ideal classifiers from zero to two. We expect that performance of both the SCS and OCS will degrade as the ratio increases. However, the OCS may degrade less quickly.

Parasitic Mix Three types of parasitic classifiers are defined in the previous section. Test 4 examines how the system behaves when faced with parasites from each type. One might naively expect that Type 3 parasites cause the most difficulty because they fail to both post the correct next-state message and send the correct action. However, because of the greater signal difference between a parasite and an ideal classifier, the system should be able to more easily detect and isolate the Type 3 parasitic classifiers. Thus, depending on system parameters, either the Type 1 or Type 2 parasites should be most difficult to distinguish.

Scale Adding classifiers within the system increases the amount of noise that must be overcome. Comparing Test 5 and 6 will show how a system behaves when the number of classifiers doubles. In addition to distinguishing between ideal and parasitic classifiers, the system

	Vertical OCS			SCS		
Test	Percent Correct	Number of Mistakes	Convergence	Percent Correct	Number of Mistakes	Convergence
1	100	0	0	100	0	0
2	100	0	5000	100	0	20
3	93	1	10000	63	6	4000
5	72	5	90000	51	6	7000
7	70	3	90000	37	6	10000

Table 4.8: Summary of Vertical OCS and SCS performance.

must select a subset of ideal classifiers to be part of the working set. Because competing organizations isolate and weaken groups of under-performing classifiers, the OCS may handle the increase in numbers more gracefully than the SCS.

Next, the performance of the OCS on each problem aspect is analyzed.

Parasite-to-Ideal Ratio

In general, the OCS achieved greater percent-correct scores than did the SCS. Although occasionally the OCS fails to distinguish the one parasitic classifier in Test 1. At the same time, the OCS used much more time to converge. However, in most cases, substantial performance (high percent-correct scores) were reached during the first few hundred generations. Table 4.8 summarizes the results of the Vertical OCS and the SCS on the five relative tests. This time, the OCS finds the parasite in Test 2. Figures 4.3 and 4.4 show example runs for Test 3 and 5. In the figures, overall percent correct shows the percent correct achieved up to a particular generation. The temporal percent correct shows the percent correct achieved between organizational growth cycles. In addition, Figure 4.4 shows the performance of the SCS on the same test.

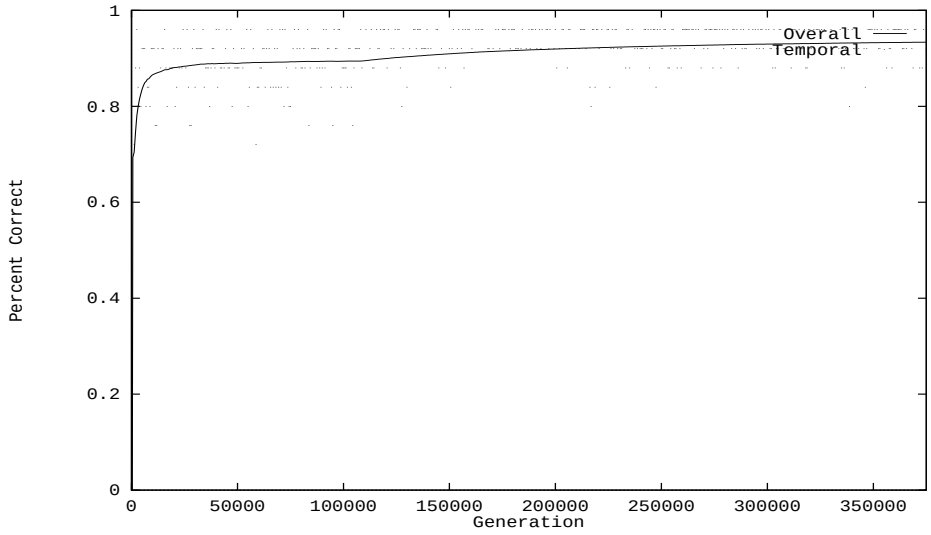

Figure 4.3: Combined OCS on Test 3 with Period 25 - Mistakes 1 - Convergence 10000.

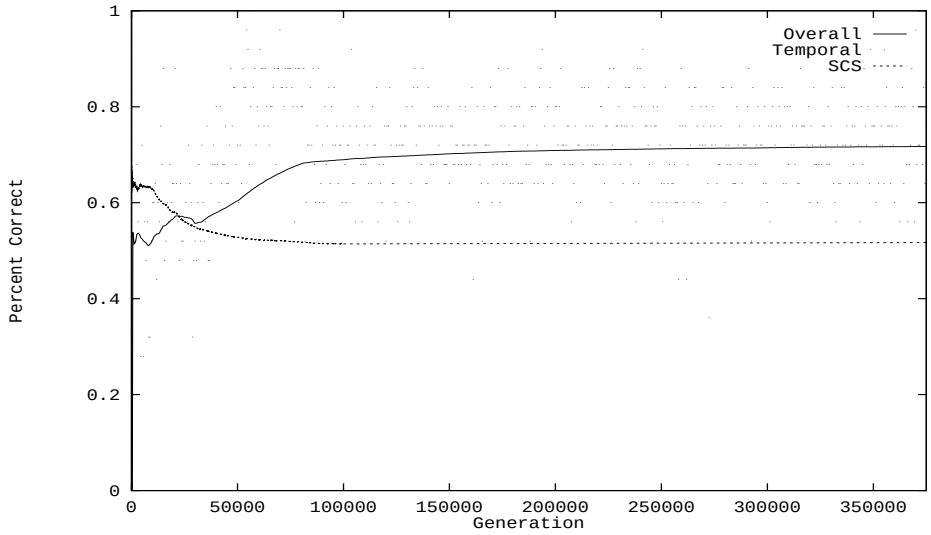

Figure 4.4: Combined OCS on Test 5 with Period 25 - Mistakes 5 - Convergence 90000.

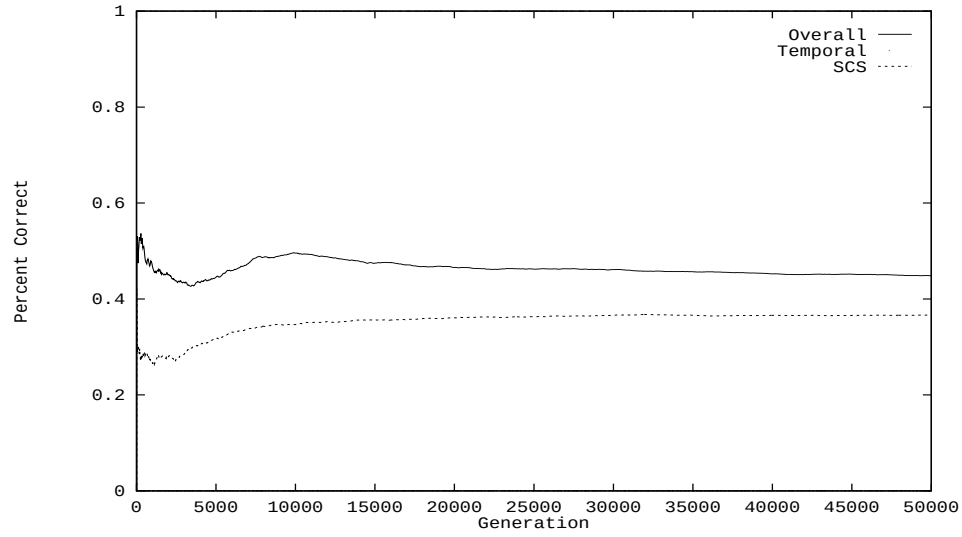

Figure 4.5: Combined OCS on Test 7 with Period 1 - Mistakes 1 - Convergence 10000. Also included is the results of the SCS on Test 7.

As mentioned above, varying some parameters dramatically affected performance of the OCS. Increasing the initial LT strength of an organization generally improved performance, especially during the later runs. During the runs on tests with low ratios of parasitic classifiers to ideal classifiers, varying the period between organizational growth did not improve performance. However, as the parasitic ratio increased, increasing the period between use of organizational Grow and Shrink operators generally improved performance. Figures 4.5 through 4.9 show representative runs of the Vertical OCS on Test 7. In addition, Figure 4.5 shows the corresponding result of the SCS on Test 7. Using vertical growth or random growth resulted in similar behavior.

An unexpected result was the formation of default hierarchies among the interacting organizations during several runs. Default hierarchies are hierarchical relationship of rules. Some rules act as general rules, matching a broad number of conditions. Other rules act as exceptions

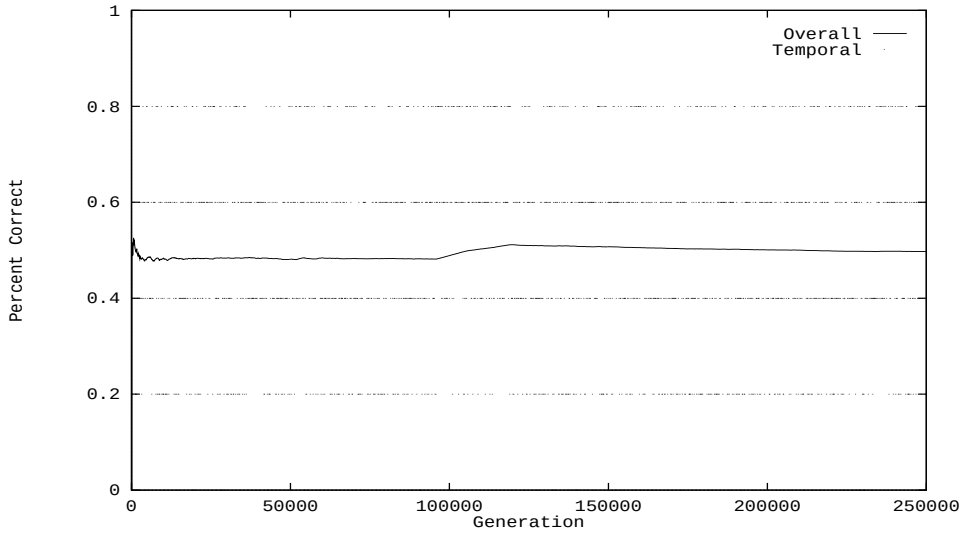

Figure 4.6: Combined OCS on Test 7 with Period 5 - Mistakes 3 - Convergence 1000.

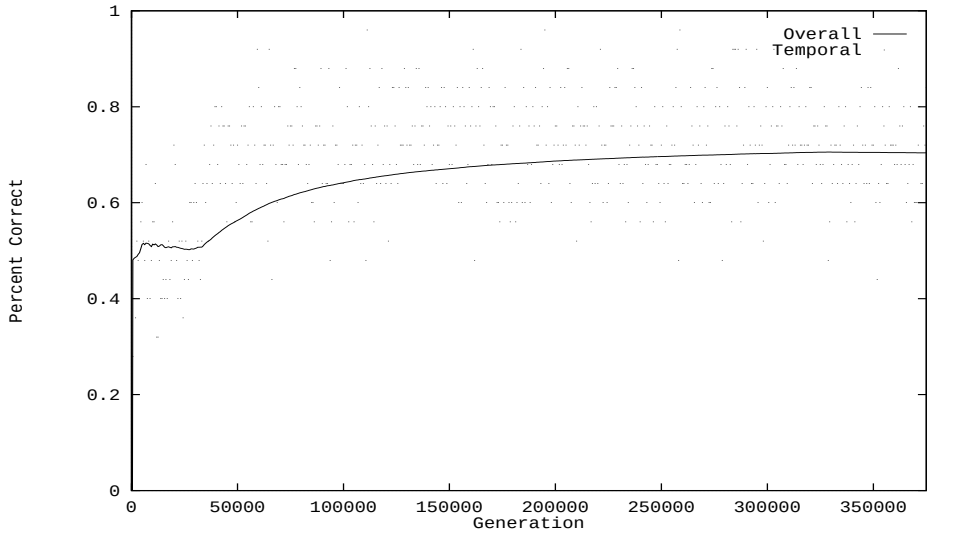

Figure 4.7: Combined OCS on Test 7 with Period 25 - Mistakes 3 - Convergence 90000.

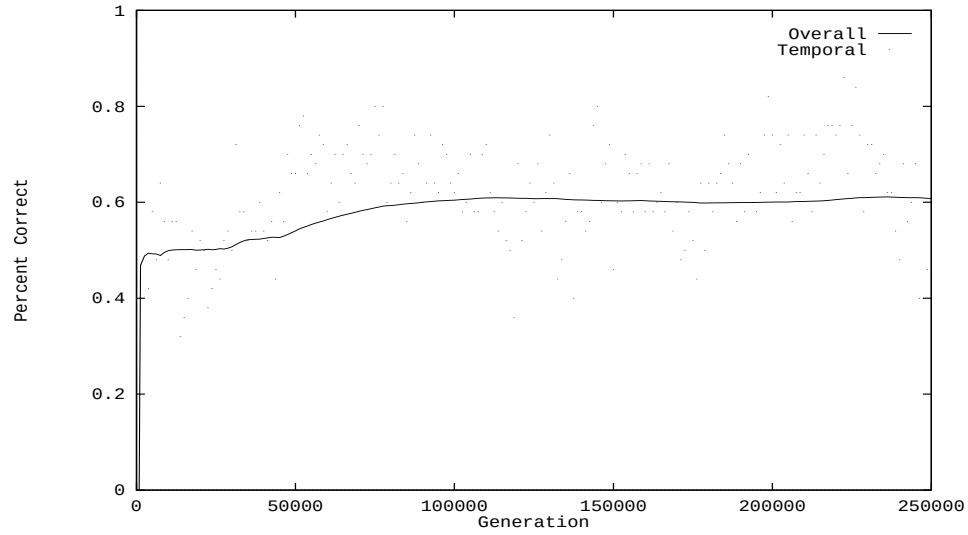

Figure 4.8: Combined OCS on Test 7 with Period 50 - Mistakes 1 - Convergence 75000.

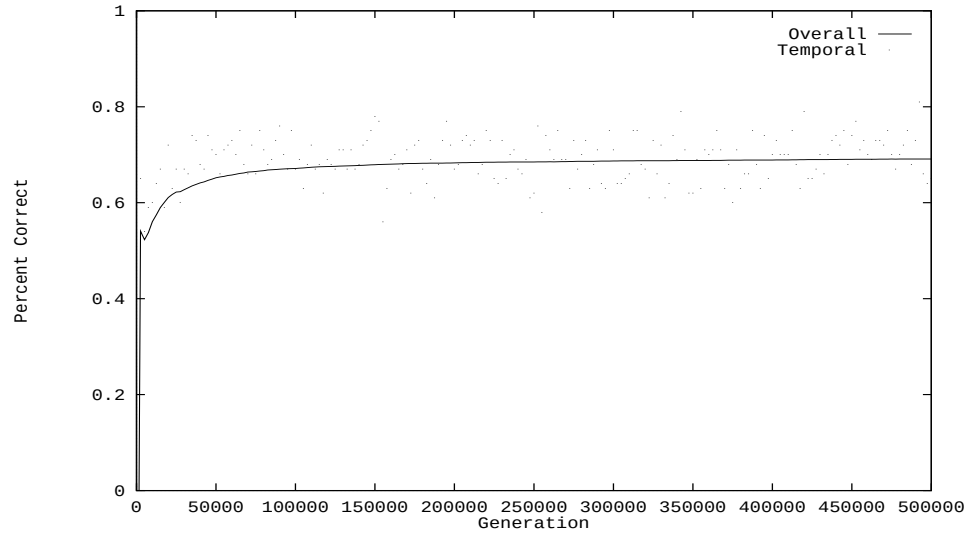

Figure 4.9: Combined OCS on Test 7 with Period 100 - Mistakes 3 - Convergence 20000.

to the general rules, matching a subset of conditions. During some runs, the OCS evolved one large organization containing a complete working set including some parasitic classifiers. For the most part, this organization is selected to affect the environment. A smaller organization containing a lower ratio of parasitic classifiers to ideal classifiers also evolves. Because the smaller organization has fewer parasites, its LT reputation is higher than the larger organization. Thus, when a classifier from the smaller organization can fire, conflict resolution chooses the smaller organization to affect the environment. Essentially, the smaller organization acts as an exception to the more general behavior of the large organization. The effect is to maintain a high percent-correct score, while allocating significant strength a number of parasites.

Next, the section examines the performance of the OCS when faced with varying types of parasites.

Parasitic Mix

The results of running the OCS on Test 4 (containing various types of parasitic classifiers) varied depending on whether random growth or vertical growth were used during organizational growth. The SCS outperformed the Random OCS regardless of the initial LT reputation for organizations and the period. The Vertical OCS achieved significantly higher percent-correct scores than the SCS. This argues that intelligent recruitment is a key determinant of organizational success. Similar to the SCS, the only parasites that achieved significant strength (were members of the most successful organizations) were Type 2 parasites. The results of the Vertical OCS are summarized in the Table 4.9.

Next, the section examines the performance of the OCS when scaling up the number of classifiers.

Period	Percent Correct	Number of Mistakes	Convergence
1	68	5	8000
5	83	5	15000
25	82	3	65000
50	86	2	30000
100	78	4	50000

Table 4.9: Summary of the Vertical OCS performance on Test 4.

Scale

Similar to the SCS, OCS performance dropped when doubling the number of classifiers from Test 5 to Test 6. However, the OCS did generally achieve higher percent-correct scores than did the SCS. In addition, a very interesting pattern of improvement occurs during the Vertical OCS's run when increasing the period between use of organizational operators. Figure 4.10 and Figure 4.11 show the performance of the Vertical OCS on Test 7 using periods of 50 and 100 respectively. Notice that at the ends of the runs, the temporal scores increase dramatically. This can best be seen in Figure 4.11. It appears that after hundreds of thousands of generations, the OCS improves the most successful organization. A close examination of the organizational structure of the system at the end of these runs shows that the system sustains percent-correct scores near 0.7. It appears that the OCS contained two main competing organizations, one of which slowly lost strength as the run progressed. In the end, the one containing the higher-quality working set prevailed.

The rest of this section summarizes the results of the OCS-SCS comparison.

Result Summary

The following list summarizes the results of the OCS across the three aspects of problem difficulty.

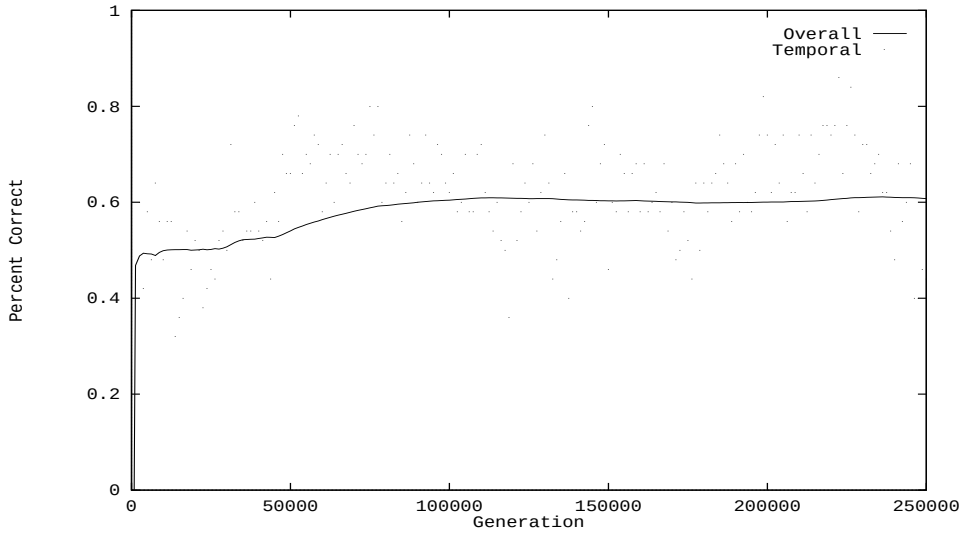

Figure 4.10: Combined OCS on Test 7 with Period 50 - Mistakes 1 - Convergence 5000.

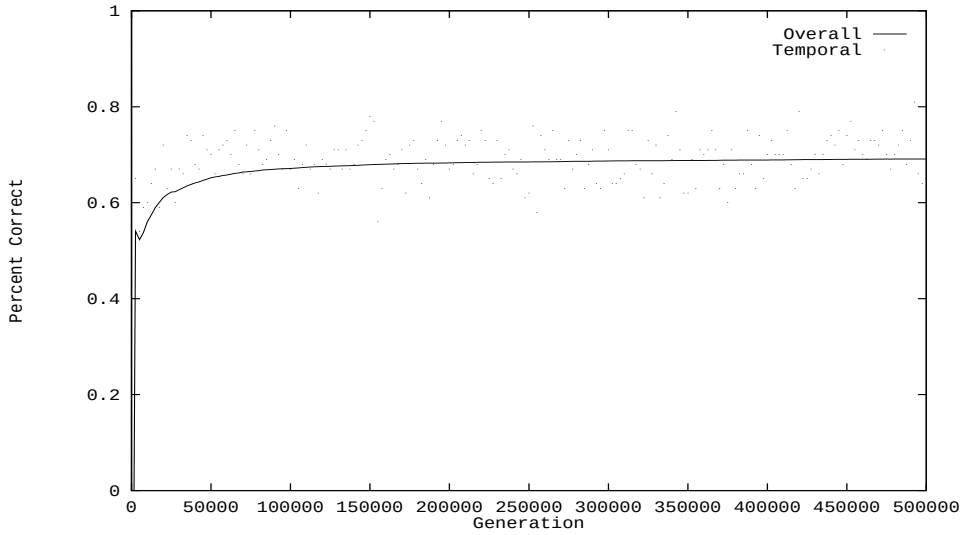

Figure 4.11: Combined OCS on Test 7 with Period 100 - Mistakes 3 - Convergence 65000.

- Generally, the OCS achieves higher percent-correct scores than the SCS as the ratio of parasitic classifiers to ideal classifiers increase. While convergence appears to take much longer for the OCS, substantial improvements often occur during the first few thousand generations.

- Type 2 parasites are harder to distinguish for both the OCS and the SCS than other types. Using vertical growth, the OCS achieves significantly higher percent-correct scores than the SCS, while using random growth, the OCS performs slightly worse than the SCS.

- When increasing the number of classifiers, the OCS continued to achieve higher percent-correct scores over the SCS. Additionally, dramatic improvement occurred at the ends of the runs using periods of 50 and 100.

The above results show that the OCS succeeded in isolating a greater number of parasitic classifiers from the most successful organization. As the system learned to size organizations it also occasionally learned default hierarchical relationships between the organizations appropriately. Additionally, the OCS succeeded in running competitions between organizations (shown most dramatically in the runs relating to Test 7). However, there seems to be a large computational price for much of the improvements. The next section looks at ways to improve the OCS for future use.

4.7 Improving the OCS

While the OCS succeeded in learning appropriately sized organizations during these tests, it retains a number of shortcomings in its current incarnation. The following list describes potential improvements:

- Better organizational sizing operators
- Using a non-deterministic conflict resolution scheme
- Using a rule-discovery mechanism such as an appropriately modified GA
- Re-examining reputation

The rest of this section looks at each of the above strategies.

Better Organizational Sizing Operators

The organizational growth component of the OCS controls the size of organizations by applying organizational operators to each organization. Two methods, random and vertical growth, were used by the Grow operator when increasing the size of successful organizations. Under 'easy' conditions, both methods resulted in similar performance. However, when faced with large numbers of different types of parasitic classifiers, using vertical growth was the superior choice. Since results were so sensitive to the choice of Growth operator, a number of other methods should be explored. Examples include selecting classifiers with high reputation values and keeping a pool of free agents (those classifiers that are not part of any organization). Classifiers that have either been removed from an organization by the Shrink operator or newly created might be put into the pool.

Similarly, other methods for shrinking an organization should be tried. Currently, the OCS waits until classifiers lose most of their LT reputation before ejecting them from an organization. However, waiting implies losing the reputation information gained over the previous firings. By removing classifiers as soon as conflict resolution regularly selects alternative classifiers, the system may make better use of the LT reputation in the context of a different working set.

Using a Non-Deterministic Conflict Resolution Scheme

To reduce some complexity, the current OCS uses a deterministic conflict resolution scheme by always selecting the classifier or organization with the greatest reputation. However, this means that the OCS may not explore good alternatives. The effect of using a non-deterministic scheme may be two-fold. First, there may be an improvement in the ability of the OCS to simultaneously evaluate multiple organizations and classifiers. Second, the dependence of the OCS on using high initial LT reputation values for organizations may decrease. The second case is important because the new organizations often contain known parasitic classifiers which generally should not be allowed to fire.

Using a Rule-discovery Mechanism

A quality that makes classifier systems appealing is that with rule-discovery they can learn new rules to improve the system. The above OCS does not use rule-discovery and is therefore limited to only learning how to better use given rules. Currently, traditional classifiers often use the strength of the classifier for fitness to be used by a genetic algorithm. An alternative, appropriate for an OCS, would be to remove known parasitic classifiers from the system altogether and rely on the GA to form replacements. To ensure sustained performance, the system could maintain the highest fit organizations by retaining parent classifiers.

Re-examining Reputation

In addition to introducing organizational structures, this thesis examined the use of multiple reputation values. Of significant value is the idea that keeping track of reputation on different time scales (long-term versus short-term) and at different levels (classifiers and organizations)

provides a way of discovering charlatan-like behavior. One improvement to the current OCS may be to increase the number of time scales examined even further, providing greater control to conflict resolution.

4.8 Summary

This chapter presents a study of organizational learning within a simple classifier system (SCS). It begins by introducing the parasitic problem faced by all classifier systems trying to solve problems that require memory. By nature, parasites appear attractive, while delivering less than advertised services. Considering the parasite problem, the chapter presents a test environment appropriate for testing the performance of different classifier systems. It examines the nature of CS solutions which include learning a working set of classifiers that are regularly selected by conflict resolution.

Using the test environment, the chapter explores the performance of the SCS along three problem aspects: ratio of parasitic classifiers to ideal ones, parasitic mix, and scaling. The results show that parasitic classifiers do reduce performance.

Before building an organizational classifier system (OCS), the chapter introduces new credit allocation and conflict resolution schemes that use two reputation values, LT reputation and ST reputation, each keeping track of performance over different time scales. The chapter then describes the implementation differences between the OCS and the SCS. Afterwards, the chapter explores the performance of the OCS in the test environment. The OCS succeeds in isolating a greater number of parasitic classifiers from the most successful organization. As the system learns to appropriately size organizations, it also occasionally learns default hierarchical

relationships between the organizations. The chapter ends by providing a discussion of possible future improvements to the OCS.

The next chapter summarizes the thesis and presents concluding remarks.

Chapter 5

Conclusion

The primary focus of this study has been the investigation of autonomous formation of appropriately sized organizations within classifier systems. The study of transaction costs from the field of economics has led to insights into the formation and sizing of organizations. Building on this foundation, the thesis has used different forms of reputation together with organizational sizing techniques to create an organizational classifier system (OCS) capable of autonomously adjusting the degree of individual and collective behaviors found within Michigan-style and Pitt-style classifier systems. Results have shown that the OCS is better suited to distinguishing parasitic (less-than-ideal) classifiers than is a simple Michigan-style classifier system. The rest of this chapter summarizes the presented work and presents final comments.

5.1 Summary

The first part of this thesis has devoted itself to gaining a better understanding of organizational growth through the study of transaction costs. Essentially, the theory of transaction costs explains organizational growth through efforts to reduce the overhead costs associated with

any exchange of goods or services. Two mechanisms, using reputation (knowledge of past performance) and forming organizations of trusted partners, are common ways to reduce these costs.

The design of an OCS has proceeded in three stages: (1) isolate organizational facets in an abstract model, (2) analyze the performance of a simple classifier system (SCS), and (3) design and test an organizational classifier system (OCS). During the first stage, germane to organizational growth, an abstract model has been built that isolates facets relating to sizing mechanisms. The results have shown that correct sizing can occur through either incremental or large steps using operators to both increase and decrease the size of an organization. However, using incremental steps has led to a more gradual and less volatile search.

The second stage has analyzed some deficiencies of simple classifier systems that must be overcome in a successful OCS. Before building the OCS, pertinent current work on classifier systems has been explored. Most systems fall into either the Michigan or Pitt approaches to classifier systems. The thesis also has presented a simple classifier system (SCS) which captures some common features of learning classifier systems following the Michigan-style.

To compare CS performance, a memory-depth-one problem which requires the SCS to use internal memory to obtain the maximum reward has been presented. When solving problems, an SCS learns a working set of classifiers that conflict resolution regularly selects to fire. The thesis has shown that from the perspective of a particular working set, some classifiers appear to be parasitic in that they lead the system to achieve less than optimal performance. Three types of parasites have been described based on how they lead the system to achieve less than optimal reward.

Three aspects of problem difficulty facing successful applications of the SCS that have been examined are: (1) the ratio of parasitic classifiers to ideal ones, (2) the mix of types of parasites, and (3) the scaling of the number of classifiers. The runs of the SCS have been evaluated based on percent correct, number of mistakes, and convergence times. The results have shown that the SCS fails both to distinguish parasitic classifiers and learn a stable working set when there are more than a few parasitic classifiers in the initial population. When examining a mix of types of parasites, parasites that receive environmental rewards but post inappropriate internal messages are the most difficult to distinguish and weed out. These classifiers exhibit charlatan-like behavior by fooling the system into assigning them high strength even though they fail to lead the system to optimal rewards. Finally, the SCS has taken over four times longer to converge when the number of classifiers doubled from 32 to 64.

After examining the performance of the SCS on the testing environment, the third stage has designed and tested an OCS. The thesis has analyzed the use of reputation values in conventional conflict resolution and credit allocation schemes. Specifically, using traditional strength values (assigned over the lifetime of a classifier) interferes with conflict resolution's short-term goals of selecting the best classifier to take immediate action. By explicitly using both short-term and long-term reputation values, an OCS should be better able to distinguish parasitic classifiers. Organizational operators then remove classifiers with poor long-term reputations (LT reputations) from organizations, allowing conflict resolution to use greedy-like schemes using short-term reputations (ST reputations) when selecting classifiers to post messages. Thus, a classifier's long-term performance determines membership within a working set (the organization) and its short-term performance determines whether it will win internal conflicts. Organizations also carry two reputation values. An organization's ST reputation de-

termines whether it can grow in size, while its LT reputation determines whether it can affect the environment.

Next, the organizational classifier system (OCS) has been presented. Like the Pitt-style systems, the OCS contains a population of organizations containing classifiers. However, OCS organizations differ in that they are of variable size and they interact with each other. Similar to Michigan-style approaches, a single population of classifiers interacts with a problem environment. Unlike the Michigan systems, however, the OCS contains separate message boards for each organization. Interaction between organizations occurs when classifiers match messages from other organizations, assuming that no internal classifier can match an internal message from its organization's message board. The formation of organizations occurs through the use of organizational operators based on the incremental add and subtract operators explored in the organizational growth model.

After describing the implementation details of the OCS, the thesis has applied the OCS to the same tests used for the SCS. The OCS succeeds in distinguishing more parasitic classifiers under difficult conditions. While the OCS quickly finds 'good' solutions, it often takes longer to finally converge. Additionally, the OCS proves sensitive to several parameters including: (1) the period between use of organizational operators, (2) the initial long-term reputation value for new organizations, and (3) the method used to size organizations.

The next section closes this thesis with concluding remarks.

5.2 Conclusions

This thesis demonstrates the effectiveness of autonomously sizing organizations within a classifier system. By viewing classifier systems as an economic system, it is possible to use the theory

of transaction costs to develop effective mechanisms to form organizations. The importance of such mechanisms is in the potential to bridge the gap between individual and collective approaches to classifier systems. Individual approaches can efficiently process classifiers to solve simple problems, but are unable to tackle more complex problems. Collective approaches appear to be able to eventually solve harder problems, but are computationally too expensive. By building a system that autonomously adjusts the degree of individual to collective behavior, it is possible to have a system that is both efficient and resilient to problem difficulty.

The conclusions of this thesis fall into three categories: (1) the abstract organizational model, (2) the OCS, and (3) organization-like computational models in machine learning. Below is a discussion of each.

First, conclusions relating to the abstract organizational growth model are as follows:

- The abstract organizational growth model with idealized operators can efficiently learn the optimal size of organizations when deciding the fitness of organizations is easy and the fitness function is unimodal.

- The incremental operators AddOne and SubOne from the organizational growth model provide a gradual mechanism for searching for optimally sized organizations. The crossover-like operators, Join and Cut, also form optimally sized organizations; however, the search process is more oscillatory.

The abstract model provided the groundwork for further exploration of organization sizing by showing that operator pairs such as AddOne/SubOne and Join/Cut can effectively size abstract organizations. Several recommendations can be made: (1) Further research into organizational modeling should use abstract models to simplify and isolate specific behaviors and (2) organization sizing operators should be modeled on the abstract organization operators.

The former recommendation suggests a method for understanding complex problems: divide and conquer. The latter recommendation has been tried here; the organization operators used in the OCS successfully aid in the sizing of organizations.

Second, conclusions relating to the OCS are as follows:

- The OCS improves the quality of solutions compared to a simple Michigan-style CS with modest degradation of convergence times.
- The transfer of operator behavior from the abstract organizational growth model to the OCS provides useful organization operators.
- The analysis of parasitic classifier behavior can lead to insights for model designs overcoming parasitic behavior.
- Reputation values representing performance over different time periods can be used to distinguish and isolate charlatan-like parasitic classifiers.
- Default hierarchies can naturally form among organizations within an OCS, improving the quality of solutions.

Expanding the SCS model, the OCS demonstrates that using organizations can improve the quality of solutions. In particular, competing organizations of classifiers improves the ability of the CS to distinguish and isolate parasitic classifiers from ideal ones. Essentially, each organization contains a potential working set, allowing the OCS to simultaneously evolve different solutions. In addition, cooperating organizations form default hierarchies with one organization acting as a general rule, while the others behave as exceptions. The OCS presented in this thesis only touches upon the mechanisms necessary to appropriately structure organizations. However, the potential rewards for further work in this area appear significant. The end of

Chapter 4 recommends a number of improvements relating to the specific OCS presented in that chapter. These include the following:

- Better organizational sizing operators
- Using a non-deterministic conflict resolution scheme
- Using a rule-discovery mechanism such as an appropriately modified GA
- Re-examining reputation

Third, the following conclusion relates to the field of organization-like computational models in machine learning.

- The study of transaction costs from the field of economics provides useful tools for studying organization-like computational models in machine learning.

The study of organization-like computational models provides a way of combining the efficient search of individual approaches and the robust search of collective approaches to classifier systems. Through the study of economics, a better understanding of both traditional classifier systems and the organizational classifier system is obtained. Beyond directly improving the OCS design presented here, there are important directions for further research for both the OCS and any organization-like computation model in machine learning. The following list describes a number of recommendations for future research.

- Bounding CS performance by both Michigan-style and Pitt-style classifier systems
- Examining convergence issues
- Analyzing classifier behavior within a larger problem set

- Exploring other cost-reducing techniques from economics

Because the OCS exhibits behavior from both individual and collective approaches to classifier systems, solution quality should be bound by both Michigan-style and Pitt-style systems. This thesis has explored simple Michigan-style performance. To complete the analysis, solution quality of Pitt-style systems on the same test problem should be examined.

An important quality of a classifier system's solution is the time for convergence. This thesis has provided approximate convergence times for percent-correct scores (the effectiveness of the CS to obtain environment reward). However, to truly evaluate performance, convergence times need to be based on purely objective criteria. In addition, there are several convergence times that are important. One is the convergence time for percent-correct scores. Another is the time for the CS to learn a stable working set.

An important area to research is types of behavior found among individual classifiers. This thesis has looked at a simple depth-one-memory problem and found three parasitic classifiers relative to a particular working set. In particular, charlatan-like parasitic classifiers were much more difficult to distinguish than the others. However, for other test problems, there may be variations in the types of classifiers found. By researching the types of behavior, improved model designs may be invented.

Another area that needs further research is the use of other cost-reducing techniques from economics. This thesis has focused on the organizational growth and reputation to reduce transaction costs. However, there are other structures, such as a court system and a governing body, that might be as useful. A more careful analysis of markets should also prove useful in the design of capable classifier systems.

References

Coase, R. H. (1988). *The firm, the market, and the law.* Chicago: University of Chicago Press.

Culberson, J. C. (1992). *GIGA program description and operation.* Unpublished manuscript, University of Alberta, Department of Computing Science, Edmonton, Alberta, Canada.

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization, and machine learning.* Reading, MA: Addison–Wesley.

Grefenstette, J. J. (1987). Multilevel credit assignment in a genetic learning system. *Genetic algorithms and their applications: Proceedings of the Second International Conference on Genetic Algorithms*, 202–209.

Hayek, F. A. (1945). The use of knowledge in society. *The American Economic Review, 35*(4), 519–530.

Holland, J. H. (1971). Processing and processors for schemata. In E. L. Jacks (Ed.), *Associative information processing* (pp. 127–146). New York: American Elsevier.

Riolo, R. L. (1987a). Bucket brigade performance: I. Long sequences of classifiers. *Genetic algorithms and their application: Proceedings of the Second International Conference on Genetic Algorithms*, 184–195.

Riolo, R. L. (1987b). Bucket brigade performance: II. Default hierarchies. *Genetic algorithms and their application: Proceedings of the Second International Conference on Genetic Algorithms*, 196–201.

Smith, R. E. (1991). *Default heirarchy formation and memory exploitation in learning classifier systems.* (TCGA Report No. 91003 and doctoral dissertation). Tuscaloosa: University of Alabama.

Smith, S. F. (1980). A learning system based on genetic adaptive algorithms. *Dissertation Abstracts International, 41*, 4582B. (University Microfilms No. 81-12638).

Westerdale, T. H., (1989). A defense of the bucket brigade. *Proceedings of the Third International Conference on Genetic Algorithms*, 282–290.

Wilson, S. W., & Goldberg, D. E. (1989). A critical review of classifier systems. *Proceedings of the Third International Conference on Genetic Algorithms*, 244–255.