The Detection and Characterization of Epistasis and Heterogeneity: A Learning Classifier System Approach

A Thesis
Submitted to the Faculty
in partial fulfillment of the requirements for the
degree of
Doctor of Philosophy
in
Genetics
by

Ryan John Urbanowicz
DARTMOUTH COLLEGE
Hanover, New Hampshire

March 2012

Examining Committee:

Professor Jason H. Moore (chair)

Professor Michael L. Whitfield

Professor Margaret J. Eppstein

Professor Robert H. Gross

Professor Tricia A. Thornton-Wells

Brian W. Pogue, Ph.D.
Dean of Graduate Studies

Abstract

As the ubiquitous complexity of common disease has become apparent, so has the need for novel tools and strategies that can accommodate complex patterns of association. In particular, the analytic challenges posed by the phenomena known as epistasis and heterogeneity have been largely ignored due to the inherent difficulty of approaching multifactor non-linear relationships. The term epistasis refers to an interaction effect between factors contributing to diesease risk. Heterogeneity refers to the occurrence of similar or identical phenotypes by means of independent contributing factors. Here we focus on the concurrent occurrence of these phenomena as they are likely to appear in studies of common complex disease. In order to address the unique demands of heterogeneity we break from the traditional paradigm of epidemiological modeling wherein the objective is the identification of a single best model describing factors contributing to disease risk. Here we develop, evaluate, and apply a learning classifier system (LCS) algorithm to the identification, modeling, and characterization of susceptibility factors in the concurrent presence of heterogeneity and epistasis. This work includes an examination of existing LCS algorithms, the development of new strategies to address the inherent problem of knowledge discovery in LCSs, the introduction of novel heuristics that improve LCS performance and allow for the explicit characterization of heterogeneity, and an application of these

collective advancements to an investigation of bladder cancer susceptibility. Successful analysis of simulated and real-world complex disease associations demonstrates the validity and unique potential of this LCS approach.

Citations to Previously Published Work

Chapter 1 has been published as:

> R. J. Urbanowicz, and J. H. Moore. Learning Classifier Systems: A Complete Introduction, Review, and Roadmap. *Journal of Artificial Evolution and Applications*. (2009):1, 2009.

Chapter 2 has been published as:

> R. J. Urbanowicz, and J. H. Moore. The Application of Michigan-Style Learning Classifier Systems to Address Genetic Heterogeneity and Epistasis in Association Studies. *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*. 195–202, 2010.

Chapter 3 has been published as:

> R. J. Urbanowicz, and J. H. Moore. The Application of Pittsburgh-Style Learning Classifier Systems to Address Genetic Heterogeneity and Epistasis in Association Studies. *Parallel Problem Solving from Nature–PPSN XI*. 404–413, 2011

Chapter 4 has been submitted as:

> R. J. Urbanowicz, A. Granizo-Mackenzie, and J. H. Moore. An Analysis Pipeline with Visualization-Guided Knowledge Discovery for Michigan-Style Learning Classifier Systems.

Chapter 5 has been submitted as:

> R. J. Urbanowicz, A. Granizo-Mackenzie, and J. H. Moore. Instance-Linked Attribute Tracking and Feedback for Michigan-Style Supervised Learning Classifier Systems.

Acknowledgments

This work would not have been possible without the financial support of the National Institute of Health (LM010098, LM009012, and AI59694), and the William H. Neukom Institute for Computational Science at Dartmouth College. I am grateful for the expertise of my thesis committee including Margaret Eppstein and Michael Whitfield, each of whom provided excellent professional guidance throughout this research. I would especially like to thank my advisor Jason Moore for his support, trust, and guidance throughout the development of this work. I would also like to acknowledge the contribution of my co-authors who have been instrumental in the publication of works included in this thesis. Ambrose Granizo-Mackenzie has contributed to chapters 4 and 5. Jeff Kiralis, Jonathan Fisher, Nicholas Sinnott-Armstrong, and Tamra Heberling contributed to the development of the model simulation strategy utilized in chapters 2, 3, 4, and 5 (submitted for publication outside of this thesis). I am grateful to Angeline Andrew and Margaret Karagas for their collaboration on chapter 6, and for making the bladder cancer susceptibility data available for analysis. I am fortunate to have worked alongside such a diverse and inquisitive group of students and post-docs. In particular I would like to thank Chantel Sloan, Anna Tyler, Casey Greene, Nicholas Sinnott-Armstrong, Delaney Granizo-Mackenzie, Ambrose Granizo-Mackenzie, Arvis Sulovari, Ting Hu, Richard Cowper, Cristian Dabaros, Josh Payne, and Dov Pechenick for their insights and day to day collaboration. I'd also like to

thank Peter Andrews, Jonathan Fisher, and Bill White for their programming expertise and Anna Green for having developed the LaTeX thesis template used here. Lastly I thank my all my teachers, mentors, family, and friends for their encouragement, love, and support. A special thanks to the Andrews family.

Contents

List of Tables

List of Figures

To my beloved father Gary who battled cancer for over a decade and still managed to be the best dad I could ask for.

To my wonderful mother Maureen for her unconditional support.

To my brother John who I'm just lucky to know.

And to the best of friends...

Chapter 1

An Introduction and Review of Learning Classifier Systems

Abstract

If complexity is your problem, learning classifier systems (LCSs) may offer a solution. These rule-based, multifaceted, machine learning algorithms originated and have evolved in the cradle of evolutionary biology and artificial intelligence. The LCS concept has inspired a multitude of implementations adapted to manage the different problem domains to which it has been applied (e.g. autonomous robotics, classification, knowledge discovery, and modeling). One field that is taking increasing notice of LCS is epidemiology, where there is a growing demand for powerful tools to facilitate etiological discovery. Unfortunately, implementation optimization

is non-trivial, and a cohesive encapsulation of implementation alternatives seems to be lacking. This paper aims to provide an accessible foundation for researchers of different backgrounds interested in selecting or developing their own LCS. Included is a simple yet thorough introduction, a historical review, and a roadmap of algorithmic components, emphasizing differences in alternative LCS implementations.

1.1 Introduction

As our understanding of the world advances, the paradigm of a universe reigned by linear models, and simple "cause and effect" etiologies becomes staggeringly insufficient. Our world and the innumerable systems that it encompasses are each composed of interconnected parts that as a whole exhibit one or more properties not obvious from the properties of the individual parts. These "complex systems" feature a large number of interacting components, whose collective activity is nonlinear. Complex systems become "adaptive" when they possess the capacity to change and learn from experience. Immune systems, central nervous systems, stock markets, eco-systems, weather, and traffic are all examples of complex adaptive systems (CASs). In the book "Hidden Order", John Holland specifically gives the example of New York City, as a system that exists in a steady state of operation, made up of *"buyers, sellers, administrations, streets, bridges, and buildings* [that] *are always changing. Like the standing wave in front of a rock in a fast-moving stream, a city is a pattern in time."* Holland conceptually outlines the generalized problem domain of a CAS and char-

acterizes how this type of system might be represented by rule-based "agents" [1]. The term "agent" is used to generally refer to a single component of a given system. Examples might include antibodies in an immune system, or water molecules in a weather system. Overall, CASs may be viewed as a group of interacting agents, where each agent's behavior can be represented by a collection of simple rules. Rules are typically represented in the form of "IF *condition* THEN *action*". In the immune system, antibody "agents" possess hyper-variable regions in their protein structure that allows them to bind to specific targets known as antigens. In this way the immune system has a way to identify and neutralize foreign objects such as bacteria and viruses. Using this same example, the behavior of a specific antibody might be represented by rules such as "IF the antigen-binding site fits the antigen THEN bind to the antigen", or "IF the antigen-binding site does not fit the antigen THEN do not bind to the antigen". Rules such as these use information from the system's environment to make decisions. Knowing the problem domain and having a basic framework for representing that domain, we can begin to describe the LCS algorithm. At the heart of this algorithm is the idea that, when dealing with complex systems, seeking a single best-fit model is less desirable than evolving a population of rules that collectively model that system. LCSs represent the merger of different fields of research encapsulated within a single algorithm. Figure 1.1 illustrates the field hierarchy that founds the LCS algorithmic concept. Now that the basic LCS concept and its origin have been introduced, the remaining sections are organized as follows: 1.2 summarizes

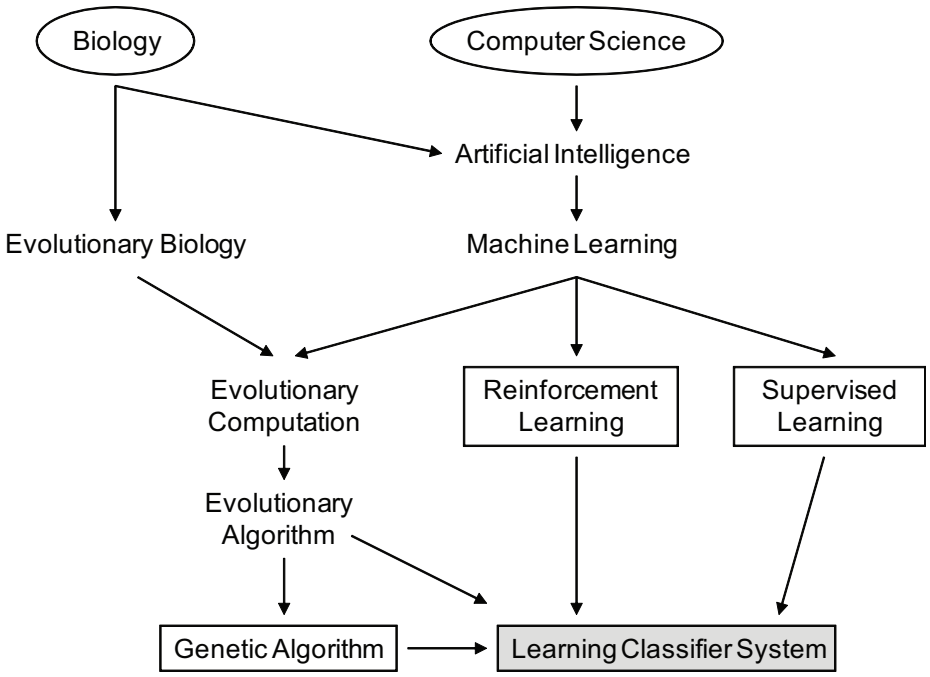

Figure 1.1: Field Tree - Foundations of the LCS community

the founding components of the algorithm, 1.3 discusses the major mechanisms, 1.4 provides an algorithmic walk through of a very simple LCS, 1.5 provides a historical review, 1.6 discusses general problem domains to which LCS has been applied, 1.7 identifies biological applications of the LCS algorithm, 1.8 briefly introduces some general optimization theory, 1.9 outlines a roadmap of algorithmic components, 1.10 gives some overall perspective on future directions for the field, and 1.11 identifies some helpful resources.

1.2 A General LCS

Let's begin with a conceptual tour of LCS anatomy. As previously mentioned, the core of an LCS is a set of rules (called the population of classifiers). The desired outcome of running the LCS algorithm is for those classifiers to collectively model an intelligent decision maker. To obtain that end, *"LCSs employ two biological metaphors; evolution and learning...* [where] *learning guides the evolutionary component to move toward a better set of rules."* [2] These concepts are respectively embodied by two mechanisms: the genetic algorithm, and a learning mechanism appropriate for the given problem (see sections 1.3.1 and 1.3.2 respectively). Both mechanisms rely on what is referred to as the "environment" of the system. Within the context of LCS literature, the environment is simply the source of input data for the LCS algorithm. The information being passed from the environment is limited only by the scope of the problem being examined. Consider the scenario of a robot being asked to navigate a maze environment. Here, the input data may be in the form of sensory information roughly describing the robot's physical environment [3]. Alternatively, for a classification problem such as medical diagnosis, the environment is a training set of pre-classified subjects (i.e. cases and controls) described by multiple attributes (e.g. genetic polymorphisms). By interacting with the environment, LCSs receive feedback in the form of numerical reward that drives the learning process. While many different implementations of LCS algorithms exist, Holmes et. al [4] outlines four practically universal components: (1) a finite population of classifiers that

represents the current knowledge of the system, (2) a performance component that regulates interaction between the environment and classifier population, (3) a reinforcement component (also called credit assignment component [5]) that distributes the reward received from the environment to the classifiers, and (4) a discovery component that uses different operators to discover better rules and improve existing ones. Together, these components represent a basic framework upon which a number of novel alterations to the LCS algorithm have been built. Figure 1.2 illustrates how specific mechanisms of LCS (detailed in section 1.9) interact in the context of these major components.

1.3 The Driving Mechanisms

While the above four components represent an algorithmic framework, two primary mechanisms are responsible for driving the system. These include discovery, generally by way of the genetic algorithm, and learning. Both mechanisms have generated respective fields of study, but it is in the context of LCS that we wish to understand their function and purpose.

1.3.1 Discovery - The Genetic Algorithm

Discovery refers to "rule discovery" or the introduction of rules that don't currently exist in the population. Ideally, new rules will be better at getting payoff

(i.e. making good decisions) than existing ones. From the start, this task has almost always been achieved through the use of a genetic algorithm (GA). The GA is a computational search technique that manipulates (evolves) a population of individuals (rules) each representing a potential solution (or piece of a solution) to a given problem. The GA, as a major component of the first conceptualized LCS [6], has largely surpassed LCS in terms of celebrity and common usage. GAs [7, 8] are founded on ideas borrowed from nature. Inspired from the neo-Darwinist theory of natural selection, the evolution of rules is modeled after the evolution of organisms using four biological analogies: (1) a code is used to represent the genotype/genome (condition), (2) a solution (or phenotype) representation is associated with that genome (action), (3) a phenotype selection process (survival of the fittest), where the fittest organism (rule) has a greater chance of reproducing and passing its "genetic" information on to the next generation, and (4) genetic operators are utilized to allow simple transformations of the genome in search of fitter organisms (rules) [9, 10]. Variation in a genome (rule) is typically generated by two genetic operators: mutation and crossover (recombination). Crossover operators create new genotypes by recombining sub-parts of the genotypes of two or more individuals (rules). Mutation operators randomly modify an element in the genotype of an individual (rule). The selection pressure that drives "better" organisms (rules) to reproduce more often is dependent on the fitness function. The fitness function quantifies the optimality of a given rule, allowing that rule to be ranked against all other rules in the population. In a simple classification

problem, one might use classification accuracy as a metric of fitness.

Running a genetic algorithm requires looping through a series of steps for some number of iterations (generations). Initially, the user must predefine a number of parameters such as the population size (N) and the number of generations, based on the user's needs. Additionally the GA needs to be initialized with a population of rules that can be generated randomly to broadly cover the range of possible solutions (the search space). The following steps will guide the reader through a single iteration of a simple genetic algorithm.

1. Evaluate the fitness of all rules in the current population.
2. Select "parent" rules from the population (with probability proportional to fitness).
3. Crossover and/or mutate "parent" rules to form "offspring" rules.
4. Add "offspring" rules to the next generation.
5. Remove enough rules from the next generation (with probability of being removed inversely proportional to fitness) to restore the number of rules to N.

As with LCSs, there are a variety of GA implementations that may vary the details underlying the steps described above (see section 1.9.5). GA research constitutes its own field that goes beyond the scope of this review. For a more detailed introduction to GAs we refer readers to Goldberg [8] and [11].

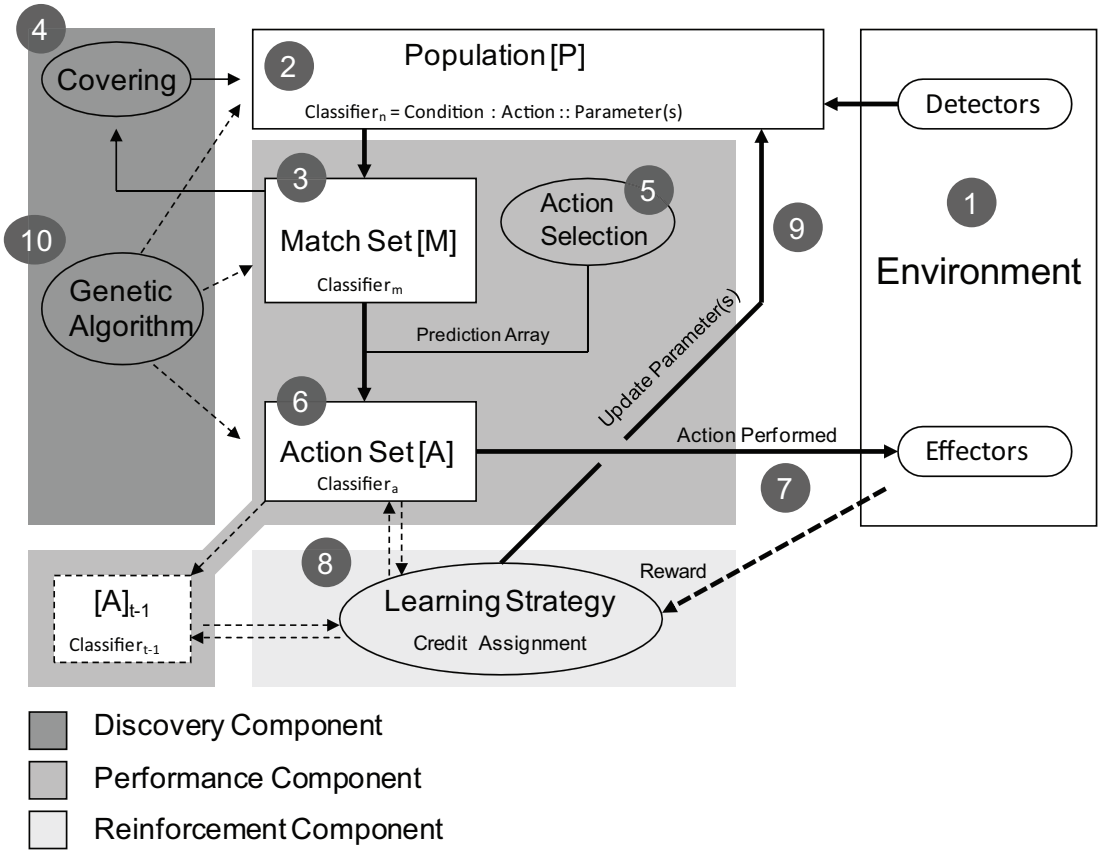

Figure 1.2: A Generic LCS - The values 1-10 indicate the typical steps included in a single learning iteration of the system. Thick lines indicate the flow of information, thin lines indicate a mechanism being activated, and dashed lines indicate either steps that do not occur every iteration, or mechanisms that might operate at different locations.

1.3.2 Learning

In the context of artificial intelligence, learning can be defined as, *"the improvement of performance in some environment through the acquisition of knowledge resulting from experience in that environment"* [12]. This notion of learning via reinforcement (also referred to as *credit assignment* [3]) is an essential mechanism of the LCS architecture. Often the terms learning, reinforcement, and credit assignment are used interchangeably within the literature. In addition to a condition and action, each classifier in the LCS population has one or more parameter values associated with it (e.g. fitness). The iterative update of these parameter values drives the process of LCS reinforcement. More generally speaking, the update of parameters distributes any incoming reward (and/or punishment) to the classifiers that are accountable for it. This mechanism serves two purposes: (1) to identify classifiers that are useful in obtaining future rewards and (2) to encourage the discovery of better rules. Many of the existing LCS implementations utilize different learning strategies. One of the main reasons for this is that different problem domains demand different styles of learning. For example, learning can be categorized based on the manner that information is received from this environment. *Off-line* or *"batch"* learning implies that all training instances are presented simultaneously to the learner. The end result is a single rule set embodying a solution that does not change with respect to time. This type of learning is often characteristic of data mining problems. Alternatively, *on-line* or *"incremental"* learning implies that training instances are presented to the learner

one at a time, the end result of which is a rule set that changes continuously with the addition of each additional observation [12–14]. This type of learning may have no pre-specified endpoint, as the system solution may continually modify itself with respect to a continuous stream of input. Consider, for example, a robot that receives a continuous stream of data about the environment it is attempting to navigate. Over time it may need to adapt its movements to maneuver around obstacles it has not yet faced. Learning can also be distinguished by the type of feedback that is made available to the learner. In this context, two learning styles have been employed by LCSs; *supervised* learning and *reinforcement* learning, of which the latter is often considered to be synonymous with LCS. Supervised learning implies that for each training instance, the learner is supplied not only with the condition information, but also with the "correct" action. The goal here is to infer a solution that generalizes to unseen instances based on training examples that possess correct input/output pairs. Reinforcement learning (RL), on the other hand, is closer to *unsupervised* learning, in that the "correct" action of a training instance is not known. However, RL problems do provide feedback, indicating the "goodness" of an action decision with respect to some goal. In this way, learning is achieved through trial-and-error interactions with the environment where occasional immediate reward is used to generate a policy that maximizes long term reward (delayed reward). The term 'policy' is used to describe a state-action map that models the agent-environment interactions. For a detailed introduction to RL we refer readers to Sutton and Barto (1998) [15], Harmon

(1996) [16], and Wyatt (2005) [17]. Specific LCS learning schemes will be discussed further in section 1.9.4.

1.4 A Minimal Classifier System

The working LCS algorithm is a relatively complex assembly of interacting mechanisms operating in an iterative fashion. We complete our functional introduction to LCSs with an algorithmic walk through. For simplicity, we will explore what might be considered one of the most basic LCS implementations, a minimal classifier system (MCS) [18]. This system is heavily influenced by modern LCS architecture. For an earlier perspective on simple LCS architecture see Goldberg's SCS [8]. MCS was developed by Larry Bull as a platform for advancing LCS theory. While it was not designed for real-world applications, MCS offers a convenient archetype upon which to better understand more complex implementations. Figure 1.3 outlines a single iteration of MCS. In this example the input data takes the form of a four digit binary number, representing discrete observations detected from an instance in the environment. MCS learns iteratively, sampling one data instance at a time, learning from it, and then moving to the next. As usual a population of classifiers represents the evolving solution to our given problem. Each of the (N) classifiers in the population are made up of a condition, an action, and an associated fitness parameter $\{F\}$. The condition is represented by a string of characters from the ternary alphabet $0, 1, \#$ where # acts as a wildcard such that the rule condition 00#1 matches both the input

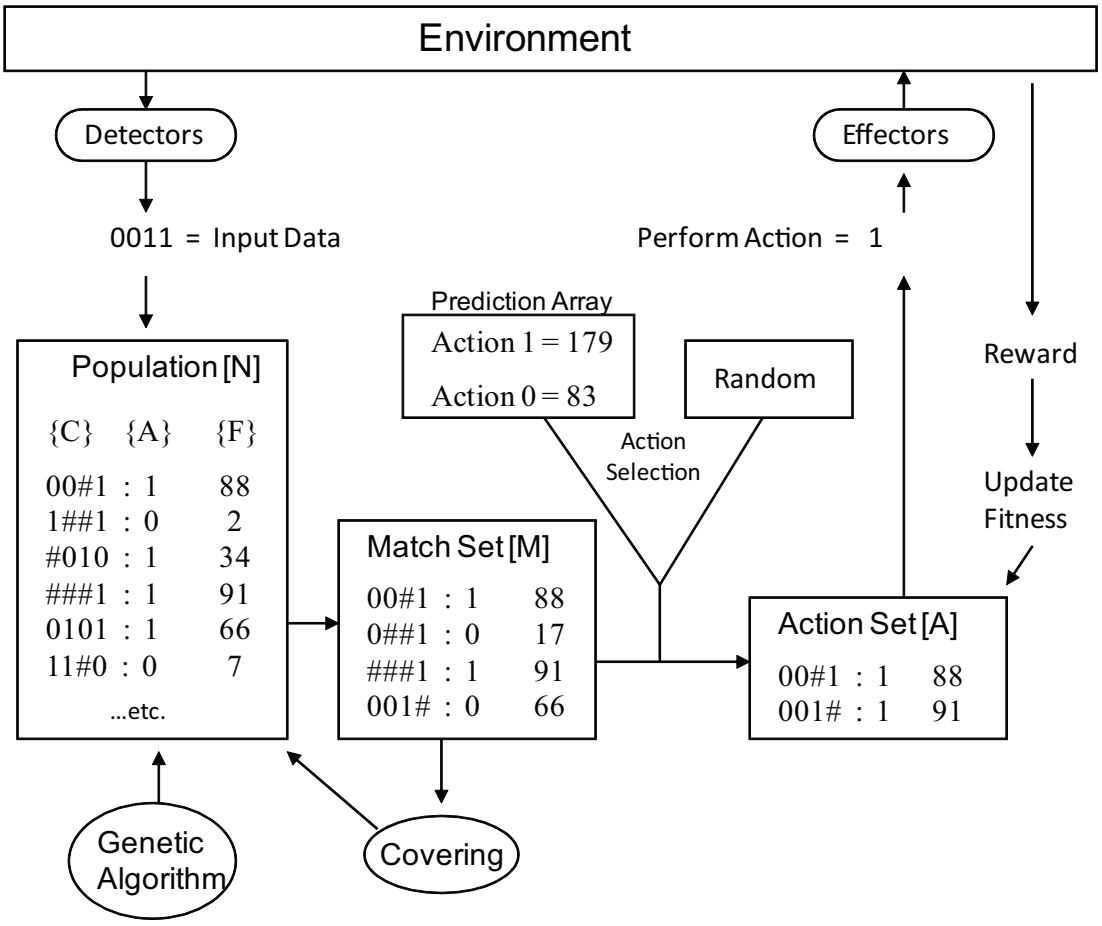

Figure 1.3: MCS algorithm - An example iteration.

0011 and the input 0001. The action is represented by a binary string where in this case only two actions are possible (0 or 1). The fitness parameter gives an indication of how good a given classifier is, which is important not only for action selection, but for application of the GA to evolve better and better classifiers. Before the algorithm is run, the population of classifiers is randomly initialized, and the fitness parameters are each set to some initial value f_0. Figure 1.3 depicts the MCS after having already been run for a number of iterations made evident by the diversity of fitness scores. With the receipt of input data, the population is scanned and any rule whose condition matches the input string at each position becomes a member of the current "match set" $[M]$. If none of the rules in the population match the input, a covering operator generates a rule with a matching condition and a random action [19]. The number of wildcards incorporated into the new rule condition is dependent on the rate ($p_{\#}$) set by the user. With the addition of a new rule, an existing rule must be removed from the population to keep (N) constant. This is done using roulette wheel selection where the probability of a rule being selected for replacement is inversely proportional to its fitness i.e., $1/(F_j+1)$ [20,21]. Once the match set is established, an action is selected using a simple explore/exploit scheme [22]. This scheme alternates between randomly selecting an action found within $[M]$ one round (explore), and selecting deterministically with a prediction array the next (exploit). The prediction array is a list of prediction values calculated for each action found in $[M]$. In MCS, the prediction value is the sum of fitness values found in the subset of $[M]$ advocating

the same action. The subset with the highest prediction value becomes the action set $[A]$, and the corresponding action is performed in the environment. Learning begins with receipt of an immediate reward (payoff = P) from the environment in response to the performed action. MCS uses a simple form of RL that uses the Widrow-Hoff procedure (see section 1.9.4.2) with a user defined learning rate of β. The following equation updates the fitness of each rule in the current $[A]$:

$$F_j \leftarrow F_j + \beta((P/|[A]|) - F_j)$$

The final step in MCS is the activation of a GA that operates within the entire population (panmictic). Together the GA and the covering operator make up the discovery mechanism of MCS. The GA operates as previously described where on each "explore" iteration, there is a probability (g) of GA invocation. This probability is only applied to "explore" iterations where action selection is performed randomly. Parent rules are selected from the population using roulette wheel selection. Offspring are produced using a mutation rate of (μ) (with a wildcard rate of ($p_{\#}$)) and a *single point* crossover rate of (χ). New rules having undergone mutation inherit their parent's fitness values, while those that have undergone crossover inherit the average fitness of the parents. New rules replace old ones as previously described. MCS is iterated in this manner over a user defined number of generations.

1.5 Historical Perspective

The LCS concept, now three decades old, has inspired a wealth of research aimed at the development, comparison, and comprehension of different LCSs. The vast majority of this work is based on a handful of key papers [3, 19, 22, 23, 63] that can be credited with founding the major branches of LCS. These works have become the founding archetypes for an entire generation of LCS algorithms that seek to improve algorithmic performance when applied to different problem domains. As a result, many LCS algorithms are defined by an expansion, customization, or merger of one of the founding algorithms. Jumping into the literature, it is important to note that the naming convention used to refer to the LCS algorithm has undergone a number of changes since its infancy. John Holland, who formalized the original LCS concept [136] based around his more well-known invention, the Genetic Algorithm (GA) [6], referred to his proposal simply as a classifier system, abbreviated either as (CS), or (CFS) [137]. Since that time, LCSs have also been referred to as adaptive agents [1], cognitive systems [3], and genetics-based machine learning systems [2, 8]. On occasion they have quite generically been referred to as either production systems [6, 138] or genetic algorithms [139] which in fact describes only a part of the greater system. The now standard designation of a "learning classifier system" wasn't adopted until the late 80's [140] after Holland added a reinforcement component to the CS architecture [27, 141]. The rest of this section provides a synopsis of some of the most popular LCSs to have emerged, and the contributions they made to the

field. This brief history is supplemented by Table 1.1 which chronologically identifies noted LCS algorithms/platforms and details some of the defining features of each. This table includes the LCS style (Michigan $\{M\}$, Pittsburgh $\{P\}$, Hybrid $\{H\}$, and Anticipatory $\{A\}$), the primary fitness basis, a summary of the learning style or credit assignment scheme, the manner in which rules are represented, the position in the algorithm at which the GA is invoked (panmictic $[P]$, match set $[M]$, action set $[A]$, correct set $[C]$, local neighborhood LN, and modified LN (MLN) and the problem domain(s) on which the algorithm was designed and/or tested.

1.5.1 The Early Years

Holland's earliest CS implementation, called Cognitive System One (CS-1) [3] was essentially the first learning classifier system, being the first to merge a credit assignment scheme with a GA in order to evolve a population of rules as a solution to a problem who's environment only offered an infrequent payoff/reward. An immediate drawback to this and other early LCSs was the inherent complexity of the implementation and the lack of comprehension of the systems operation [8]. The CS-1 archetype, having been developed at the University of Michigan, would later inspire a whole generation of LCS implementations. These "Michigan-style" LCSs are characterized by a population of rules where the GA operates at the level of individual rules and the solution is represented by the entire rule population. Smith's 1980 dissertation from the University of Pittsburgh [23] introduced LS-1, an alter-

Table 1.1: A summary of noted LCS algorithms.

System	**Year**	**Author/Cite**	**Style**	**Fitness**	**Learning/Credit Assignment**	**Rule Rep.**	**GA**	**Problem**
CS-1	1978	Holland [3]	M	Accuracy	Epochal	Ternary	[P]	Maze Navigation
LS-1	1980	Smith [23]	P	Accuracy	Implicit Critic	Ternary	[P]	Poker Decisions
CS-1 (based)	1982	Booker [24]	M	Strength	Bucket Brigade	Ternary	[M]	Environment Navigation
Animat CS	1985	Wilson [25]	M	Strength	Implicit Bucket Brigade	Ternary	[P]	Animat Navigation
LS-2	1985	Schaffer [26]	P	Accuracy	Implicit Critic	Ternary	[P]	Classification
Standard CS	1986	Holland [27]	M	Strength	Bucket Brigade	Ternary	[P]	Online Learning
BOOLE	1987	Wilson [28]	M	Strength	One-Step Payoff-Penalty	Ternary	[P]	Boolean Function Learning
ADAM	1987	Greene [29]	P	Accuracy	Custom	Ternary	[P]	Classification
RUDI	1988	Grefenstette [30]	H	Strength	Bucket-Brigade and Profit-Sharing Plan	Ternary	[P]	Generic Problem Solving
GOFER	1988	Booker [31]	M	Strength	Payoff-Sharing	Ternary	[M]	Environment Navigation
GOFER-1	1989	Booker [32]	M	Strength	Bucket-Brigade-like	Ternary	[M]	Multiplexer Function
SCS	1989	Goldberg [8]	M	Strength	AOC	Trit	[P]	Multiplexer Function
SAMUEL	1989-1997	Grefenstette [33–35]	H	Strength	Profit-Sharing Plan	Varied	[P]	Sequential Decision Tasks
NEW BOOLE	1990	Bonelli [36]	M	Strength	Symmetrical Payoff-Penalty	Ternary	[P]	Classification
CFCS2	1991	Riolo [37]	M	Strength/ Accuracy	Q-Learning-Like	Ternary	[P]	Maze Navigation

System	Year	Author/Cite	Style	Fitness	Learning/Credit Assignment	Rule Rep.	GA	Problem
HCS	1991	Shu [38]	H	Strength	Custom	Ternary	[P]	Boolearn Function Learning
Fuzzy LCS	1991	Valenzuela-Rendon [39]	M	Strength	Custom Bucket-Brigade	Binary - Fuzzy Logic	[P]	Classification
ALECSYS	1991-1995	Dorigo [40, 41]	M	Strength	Bucket Brigade	Ternary	[P]	Robotics
GABIL	1991-1993	De Jong [42, 43]	P	Accuracy	Batch - Incremental	Binary - CNF	[P]	Classification
GIL	1991-1993	Janikow [44, 45]	P	Accuracy	Supervised Learning - Custom	Multi-valued logic (VL1)	[P]	Multiple Domains
GARGLE	1992	Greene [46]	P	Accuracy	Custom	Ternary	[P]	Classification
COGIN	1993	Greene [47]	M	Accuracy/ Entropy	Custom	Ternary	[P]	Classification, Model Induction
REGAL	1993	Giordana [48–50]	H	Accuracy	Custom	Binary - First Order Logic	[P]	Classification
ELF	1993-1996	Bonarini [51–53]	H	Strength	Q-Learning-Like	Binary - Fuzzy Logic	[P]	Robotics, Cart-Pole Problem
ZCS	1994	Wilson [19]	M	Strength	Implicit Bucket Brigade	Ternary	[P]	Environment Navigation
ZCSM	1994	Cliff [54]	M	Strength	Implicit Bucket Brigade - Memory	Ternary	[P]	Environment Navigation
XCS	1995	Wilson [22]	M	Accuracy	Q-Learning-Like	Ternary	[A]	Mulitplexor Function and Environment Navigation
GA-Miner	1995-1996	Flockhart [55, 56]	H	Accuracy	Custom	Symbolic Functions	LN	Classification, Data Mining
BOOLE ++	1996	Holmes [57]	M	Strength	Symmetrical Payoff-Penalty	Ternary	[P]	Epidemiologic Classification

System	**Year**	**Author/Cite**	**Style**	**Fitness**	**Learning/Credit Assignment**	**Rule Rep.**	**GA**	**Problem**
EpiCS	1997	Holmes [58]	M	Strength	Symmetrical Payoff-Penalty	Ternary	[P]	Epidemiologic Classification
XCSM	1998	Lanzi [59, 60]	M	Accuracy	Q-Learning-Like	Ternary	[A]	Environment Navigation
ZCCS	1998-1999	Tomlinson [61, 62]	H	Strength	Implicit Bucket Brigade	Ternary	[P]	Environment Navigation
ACS	1998-2000	Stolzmann [63, 64]	A	Strength/ Accuracy	Bucket-Brigade-like	Ternary	–	Environment Navigation
iLCS	1999-2000	Browne [65, 66]	M	Strength/ Accuracy	Custom	Real-Value Alphabet	[P]	Industrial Applications - Hot Strip Mill
XCSMH	2000	Lanzi [67]	M	Accuracy	Q-Learning-Like	Ternary	[A]	Non-Markov Environment Navigation
CXCS	2000	Tomlinson [68]	H	Accuracy	Q-Learning-Like	Ternary	[A]	Environment Navigation
XCSR	2000	Wilson [69]	M	Accuracy	Q-Learning-Like	Interval Predicates	[A]	Real-Valued Multiplexor Problems
ClaDia	2000	Walter [70]	M	Strength	Supervised Learning - Custom	Binary - Fuzzy Logic	[P]	Epidemiologic Classification
OCS	2000	Takadama [71]	O	Strength	Profit Sharing	Binary	[P]	Non-Markov Multiagent Environments
XCSI	2000-2001	Wilson [72, 73]	M	Accuracy	Q-Learning-Like	Interval Predicates	[A]	Integer-Valued Data Mining
MOLeCS	2000-2001	Bernado-Mansilla [74, 75]	M	Accuracy	Multiobjective Learning	Binary	[P]	Multiplexor Problem
YACS	2000-2002	Gerard [76, 77]	A	Accuracy	Latent Learning	Tokens	-	Non-Markov Environment Navigation

System	Year	Author/Cite	Style	Fitness	Learning/Credit Assignment	Rule Rep.	GA	Problem
SBXCS	2001-2002	Kovacs [78, 79]	M	Strength	Q-Learning-Like	Ternary	[A]	Multiplexor Function
ACS2	2001-2002	Butz [80, 81]	A	Accuracy	Q-Learning-Like	Ternary	-	Environment Navigation
ATNo SFERES	2001-2007	Landau and Picault [82–86]	P	Accuracy	Custom	Graph-Based Binary-Tokens	[P]	Non-Markov Environment Navigation
GALE	2001-2002	Llora [87, 88]	P	Accuracy	Custom	Binary	LN	Classification, Data Mining
GALE2	2002	Llora [89]	P	Accuracy	Custom	Binary	MLN	Classification, Data Mining
XCSF	2002	Wilson [90]	M	Accuracy	Q-Learning-Like	Interval Predicates	[A]	Function Approximation
AXCS	2002	Tharakunnel [91]	M	Accuracy	Q-Learning-Like	Ternary	[A]	Multi-step Problems Environmental Navigation
TCS	2002	Hurst [92]	M	Strength	Q-Learning-Like	Interval Predicates	[P]	Robotics
X-NCS	2002	Bull [93]	M	Accuracy	Q-Learning-Like	Neural Network	[A]	Multiple Domains
X-NFCS	2002	Bull [93]	M	Accuracy	Q-Learning-Like	Fuzzy - Neural Network	[A]	Function Approximation
UCS	2003	Bernado-Mansilla [94]	M	Accuracy	Supervised Learning - Custom	Ternary	[C]	Classification - Data Mining
XACS	2003	Butz [95]	A	Accuracy	Generalizing State Value Learner	Ternary	–	Blocks World Problem
XCSTS	2003	Butz [96]	M	Accuracy	Q-Learning-Like	Ternary	[A]	Multiplexor Problem

System	Year	Author/Cite	Style	Fitness	Learning/Credit Assignment	Rule Rep.	GA	Problem
MOLCS	2003	Llora [97]	P	Multi-Objective	Custom	Ternary	[P]	Classification - LED Problem
YCS	2003	Bull [98]	M	Accuracy	Q-Learning-Like Widrow-Hoff	Ternary	[P]	Accuracy Theory - Multiplexor Problem
XCSQ	2003	Dixon [99]	M	Accuracy	Q-Learning-Like	Ternary	[A]	Rule-set Reduction
YCSL	2004	Bull [100]	A	Accuracy	Latent Learning	Ternary	-	Environment Navigation
PICS	2004	Gaspar [101, 102]	P	Accuracy	Custom - Artificial Immune System	Ternary	[P]	Multiplexor Problem
NCS	2004	Hurst [103]	M	Strength	Q-Learning-Like	Neural Network	[P]	Robotics
MCS	2004	Bull [104]	M	Strength	Q-Learning-Like Widrow-Hoff	Ternary	[P]	Strength Theory - Multiplexor Problem
GAssist	2004-2007	Bacardit [105–108]	P	Accuracy	ILAS	ADI - Binary	[P]	Data Mining UCI Problems
MACS	2005	Gerard [109]	A	Accuracy	Latent Learning	Tokens	–	Non-Markov Environment Navigation
XCSFG	2005	Hamzeh [110]	M	Accuracy	Q-Learning-Like	Interval Predicates	[A]	Function Approximation
ATNo SFERES-II	2005	Landau [111]	P	Accuracy	Custom	Graph-Based Integer-Tokens	[P]	Non-Markov Environment Navigation
GCS	2005	Unold [112, 113]	M	Accuracy	Custom	Context-Free Grammar CNF	[P]	Learning Context-Free Languages
DXCS	2005	Dam [114–116]	M	Accuracy	Q-Learning-Like	Ternary	[A]	Distributed Data Mining

System	Year	Author/Cite	Style	Fitness	Learning/Credit Assignment	Rule Rep.	GA	Problem
LCSE	2005-2007	Gao [117–119]	M	Strength and Accuracy	Ensemble Learning	Interval Predicates	[A]	Data Mining UCI Problems
EpiXCS	2005-2007	Holmes [120–122]	M	Accuracy	Q-Learning-Like	Ternary	[A]	Epidemiologic Data Mining
XCSFNN	2006	Loiacono [123]	M	Accuracy	Q-Learning-Like	Feedforward Multilayer Neural Network	[A]	Function Approximation
BCS	2006	Dam [124]	M	Bayesian	Supervised Learning - Custom	Ternary	[C]	Multiplexor Problem
BioHEL	2006	Bacardit [125, 126]	P	Accuracy	Custom	ADI - Binary	[P]	Larger Problems - Multiplexor, Protein Structure Prediction
XCSFGH	2006	Hamzeh [127]	M	Accuracy	Q-Learning-Like	Binary Polynomials	[A]	Function Approximation
XCSFGC	2007	Hamzeh [128]	M	Accuracy	Q-Learning-Like	Interval Predicates	[A]	Function Approximation
XCSCA	2007	Lanzi [129]	M	Accuracy	Supervised Learning - Custom	Interval Predicates	[M]	Environmental Navigation
LCSE	2007	Gao [119]	M	Accuracy	Q-Learning-Like	Interval Predicates	[A]	Medical Data Mining - Ensemble Learning
CB-HXCS	2007	Gershoff [130]	M	Accuracy	Q-Learning-Like	Ternary	[A]	Multiplexor Problem

System	Year	Author/Cite	Style	Fitness	Learning/Credit Assignment	Rule Rep.	GA	Problem
MILCS	2007	Smith [131]	M	Accuracy	Supervised Learning - Custom	Neural Network	[C]	Multiplexor, Protein Structure
rGCS	2007	Cielecki [132]	M	Accuracy	Custom	Real-Valued Context-Free Grammar Based	[P]	Checkerboard Problem
Fuzzy XCS	2007	Casilas [133]	M	Accuracy	Q-Learning-Like	Binary - Fuzzy Logic	[A]	Single Step Reinforcement Problems
Fuzzy UCS	2007	Orriols-Puig [134]	M	Accuracy	Supervised Learning - Custom	Binary - Fuzzy Logic	[C]	Data Mining UCI Problems
NAX	2007	Llora [135]	P	Accuracy	Custom	Interval Predicates	[P]	Classification - Large Data Sets
NLCS	2008	Dam [2]	M	Accuracy	Supervised Learning - Custom	Neural Network	[C]	Classification

native implementation that founded the fundamentally different "Pittsburgh-style" LCS. Also referred to as the "Pitt-approach", the LS-1 archetype is characterized by a population of variable length rule-sets (each rule-set is a potential solution) where the GA typically operates at the level of an entire rule-set. An early advantage of the Pitt-approach came from its credit assignment scheme, where reward is assigned to entire rule-sets as opposed to individual rules. This allows Pitt-style systems such as LS-1 to circumvent the potential problem of having to share credit amongst individual rules. But, in having to evolve multiple rule sets simultaneously, Pitt-style systems suffer from heavy computational requirements. Additionally, because Pitt systems learn iteratively from sets of problem instances, they can only work off-line, whereas Michigan systems are designed to work on-line, but can engage off-line problems as well. Of the two styles, the Michigan approach has drawn the most attention as it can be applied to a broader range of problem domains and larger, more complex tasks. As such, it has largely become what many consider to be the standard LCS framework. All subsequent systems mentioned in this review are of Michigan-style unless explicitly stated otherwise. Following CS-1, Holland's subsequent theoretical and experimental investigations [27,141–150] advocated the use of the *bucket brigade* credit assignment scheme (see section 1.9.4.1). The bucket brigade algorithm (BBA), inspired by Samuel [151] and formalized by Holland [148] represents the first learning/credit assignment scheme to be widely adopted by the LCS community [20,24,28]. Early work by Booker on a CS-1 based system suggested a number of modifications

including the idea to replace the panmictically acting GA with a niche-based one (i.e., the GA acts on $[M]$ instead of $[P]$) [24]. The reason for this modification was to eliminate undesirable competition between unrelated classifiers, and to encourage more useful crossovers between classifiers of a common "environmental niche". This in turn would help the classifier population retain diversity and encourage inclusion of problem sub-domains in the solution. However, it should be noted that niching has the likely impact of making the GA more susceptible to local maxima, a disadvantage for problems with a solution best expressed as a single rule. In 1985, Stewart Wilson implemented an Animat CS [25] that utilized a simplified version of the bucket brigade, referred to later as an *implicit bucket brigade* [8]. Additionally, the Animat system introduced a number of concepts that persist in many LCSs today including *covering* (via a "create" operator), the formalization of an action set $[A]$, an estimated time-to-payoff parameter incorporated into the learning scheme, and a general progression towards a simpler CS architecture [25, 152]. In 1986 Holland published what would become known as the standard CS for years to come [27]. This implementation incorporated a strength-based fitness parameter and BBA credit assignment as described in [148]. While still considered to be quite complex and susceptible to a number of problems [152], the design of this hallmark LCS is to this day a benchmark for all other implementations. The next year, Wilson introduced BOOLE, a CS developed specifically to address the problem of learning Boolean functions [28]. Characteristic of the Boolean problem, classifiers are immediately rewarded in re-

sponse to performing actions. As a result BOOLE omits sequential aspects of the CS, such as the BBA which allows reward to be delayed over a number of time steps, instead relying on a simpler "one-step" CS. Bonelli et. al. [36] later extended BOOLE to a system called NEWBOOLE in order to improve the learning rate. NEWBOOLE introduced a "symmetrical payoff-penalty" (SPP) algorithm (reminiscent of supervised learning), which replaced $[A]$ with a correct set $[C]$, and not-correct set $Not[C]$. In 1989, Booker continued his work with GOFER-1 [32] adding a fitness function based on both payoff and non-payoff information (e.g. strength and specificity), and further pursued the idea of a "niching" GA. Another novel system that spawned its own lineage of research is Valenzuela's fuzzy LCS, which combined fuzzy logic with the concept of a rule-based LCSs [39]. The fuzzy LCS represents one of the first systems to explore a rule representation beyond the simple ternary system. For an introduction to fuzzy LCS we refer readers to [153]. An early goal for LCSs was the capacity to learn and represent more complex problems using "internal models" as was originally envisioned by Holland [144, 148]. Work by Rick Riolo addressed the issues of forming "long action chains" and "default hierarchies" that had been identified as problematic for the BBA [140, 154, 155]. A long action chain refers to a series of rules that must sequentially activate before ultimately receiving some environmental payoff. They are challenging to evolve since *"there are long delays before rewards are received, with many unrewarded steps between some stage setting actions and the ultimate action those actions lead to"* [156]. Long chains are important for modeling

behaviors that require the execution of many actions before the receipt of a reward. A default hierarchy is a set of rules with increasing levels of specificity, where the action specified by more general rules is selected by "default" except in the case where overriding information is able to activate a more specific rule. *"Holland has long argued that default hierarchies are an efficient, flexible, easy-to-discover way to categorize observations and structure models of the world"* [156]. [157] describes hierarchy formation in greater detail. Over the years a number of methods have been introduced in order to allow the structuring of internal models. Examples would include internal message lists, non-message-list memory mechanisms, "corporate" classifier systems, and enhanced rule syntax and semantics [156]. Internal message lists, part of the original CS-1 [3] exist as a means to handle all input and output communication between the system and the environment, as well as providing a makeshift memory for the system. While the message list component can facilitate complex internal structures, its presence accounts for much of the complexity in early LCS systems. The trade-off between complexity and comprehensibility is a theme that has been revisited throughout the course of LCS research [152, 156, 158]. Another founding system is Riolo's CFCS2 [37], which addressed the particularly difficult task of performing "latent learning" or "look-ahead planning", where *"actions are based on predictions of future states of the world, using both current information and past experience as embodied in the agent's internal models of the world"* [156]. This work would later inspire its own branch of LCS research: anticipatory classifier systems

(ACS) [63]. CFCS2 used "tags" to represent internal models, claiming a reduction in the learning time for general sequential decision tasks. Additionally, this system is one of the earliest to incorporate a Q-learning-like credit assignment technique (i.e., a non-bucket brigade temporal difference method). Q-learning-based credit assignment would later become a central component of the most popular LCS implementation to date.

1.5.2 The Revolution

From the late 80's until the mid-90's the interest generated by these early ideas began to diminish as researchers struggled with LCS's inherent complexity and the failure of various systems to reliably obtain the behavior and performance envisioned by Holland. Two events have repeatedly been credited with the revitalization of the LCS community, namely the publication of the "Q-Learning" algorithm in the RL community, and the advent of a significantly simplified LCS architecture as found in the ZCS and XCS (see Table 1.1). The fields of RL and LCSs have evolved in parallel, each contributing to the other. RL has been an integral component of LCSs from the very beginning [3]. While the founding concepts of RL can be traced back to Samuel's checker player [151], it wasn't until the 80's that RL became its own identifiable area of machine learning research [159]. Early RL techniques included Holland's BBA [148] and Sutton's *temporal difference* (TD) method [160], which were followed closely by Watkins's Q-Learning method [161]. Over the years a handful of studies

have confirmed the basic equivalence of these three methods, highlighting the distinct ties between the two fields. To summarize, the BBA was shown to be one kind of TD method [160], and the similarity between all three methods were noted by Watkins [161] and confirmed by Liepins, Dorigo, and Bersini [162, 163]. This similarity across fields paved the way for the incorporation of Q-learning-based techniques into LCSs. To date, Q-learning is the most well-understood and widely-used RL algorithm available. In 1994, Wilson's pursuit of simplification culminated in the development of the "zeroth-level" classifier system (ZCS) [19], aimed at increasing the understandability and performance of an LCS. ZCS differed from the standard LCS framework in that it removed the rule-bidding and internal message list, both characteristic of the original BBA (see section 1.9.3). Furthermore, ZCS was able to disregard a number of algorithmic components that had been appended to preceding systems in an effort to achieve acceptable performance using the original LCS framework (e.g., heuristics [25] and operators [164]). New to ZCS, was a novel credit assignment strategy that merged elements from the BBA and Q-Learning into the "QBB" strategy. This hybrid strategy represents the first attempt to bridge the gap between the major LCS credit assignment algorithm (i.e., the BBA) and other algorithms from the field of RL. With ZCS, Wilson was able to achieve similar performance to earlier, more complex implementations demonstrating that Holland's ideas could work even in a very simple framework. However, ZCS still exhibited unsatisfactory performance, attributed to the proliferation of over-general classifiers. The following

year, Wilson introduced an eXtended Classifier System (XCS) [22] noted for being able to reach optimal performance while evolving accurate and maximally general classifiers. Retaining much of the ZCS architecture, XCS can be distinguished by the following key features: an accuracy based fitness, a niche GA (acting in the action set [A]), and an adaptation of standard Q-Learning as credit assignment. Probably the most important innovation in XCS was the separation of the credit assignment component from the GA component, based on accuracy. Previous LCSs typically relied on a *strength* value allocated to each rule (reflecting the reward the system can expect if that rule is fired; a.k.a. reward prediction). This one strength value was used both as a measure of fitness for GA selection, and to control which rules are allowed to participate in the decision making (i.e., predictions) of the system. As a result, the GA tends to eliminate classifiers from the population that have accumulated less reward than others, which can in turn remove a low-predicting classifier that is still well suited for its environmental niche. *"Wilson's intuition was that prediction should estimate how much reward might result from a certain action, but that the evolution learning should be focused on most reliable classifiers, i.e., classifiers that give a more precise (accurate) prediction"* [165]. With XCS, the GA fitness is solely dependent on rule accuracy calculated separately from the other parameter values used for decision making. Although not a new idea [3, 136, 166], the accuracy-based fitness of XCS represents the starting point for a new family of LCSs, termed "accuracy-based" that are distinctly separable from the family of "strength-based" LCSs epitomized by

ZCS (see Table 1.1). XCS is also important, because it successfully bridges the gap between LCS and RL. RL typically seeks to learn a value function that maps out a complete representation of the state/action space. Similarly, the design of XCS drives it to form an all-inclusive and accurate representation of the problem space (i.e., a *complete map*) rather than simply focusing on higher payoff niches in the environment (as is typically the case with strength-based LCSs). This latter methodology, which seeks a rule set of efficient generalizations, tends to form a *best action map* (or a *partial map*) [79, 167]. In the wake of XCS, it became clear that RL and LCS are not only linked but inherently overlapping. So much so that analyses by Lanzi [168] led him to define LCSs as RL systems endowed with a generalization capability. *"This generalization property has been recognized as the distinguishing feature of LCSs with respect to the classical RL framework"* [9]. *"XCS was the first classifier system to be both general enough to allow applications to several domains and simple enough to allow duplication of the presented results"* [158]. As a result XCS has become the most popular LCS implementation to date, generating its own following of systems based directly on or heavily inspired by its architecture.

1.5.3 In the Wake of XCS

Of this following, three of the most prominent will be discussed: ACS, XCSF, and UCS. In 1998 Stolzmann introduced ACS [63] and in doing so formalized a new LCS family referred to as "anticipation-based". "[ACS] *is able to predict the perceptual*

consequences of an action in all possible situations in an environment. Thus the system evolves a model that specifies not only what to do in a given situation but also provides information of what will happen after a specific action was executed" [80]. The most apparent algorithmic difference in ACS is the representation of rules in the form of a condition-action-effect as opposed to the classic condition-action. This architecture can be used for multi-step problems, planning, speeding up learning, or disambiguating perceptual aliasing (where the same observation is obtained in distinct states requiring different actions). Contributing heavily to this branch of research, Martin Butz later introduced ACS2 [80] and developed several improvements to the original model [81,95,169–171] . For a more in depth introduction to ACS we refer the reader to [64] and [172]. Another brainchild of Wilson's was XCSF [90]. The complete action mapping of XCS made it possible to address the problem of function approximation. *"XCSF evolves classifiers which represent piecewise linear approximations of parts of the reward surface associated with the problem solution"* [158]. To accomplish this, XCSF introduces the concept of computed prediction, where the classifier's prediction (i.e., predicted reward) is no longer represented by a scalar parameter value, but is instead a function calculated as a linear combination of the classifier's inputs (for each dimension) and a weight vector maintained by each classifier. In addition to systems based on fuzzy logic, XCSF is of the minority of systems able to support continuous-valued actions. In complete contrast to the spirit of ACS, the sUpervised Classifier System (UCS) [94] was designed specifically to address single-step problem

domains such as classification and data mining where delayed reward is not a concern. While XCS and the vast majority of other LCS implementations rely on RL, UCS trades this strategy for supervised learning. Explicitly, classifier prediction was replaced by accuracy in order to reflect the nature of a problem domain where the system is trained, knowing the correct prediction in advance. UCS demonstrates that a best action map can yield effective generalization, evolve more compact knowledge representations, and can converge earlier in large search spaces.

1.5.4 Revisiting the Pitt

While there is certainly no consensus as to which style LCS (Michigan or Pittsburgh) is "better", the advantages of each system in the context of specific problem domains are becoming clearer [173]. Some of the more successful Pitt-style systems include GABIL [42], GALE [87], ATNoSFERES [83], MOLCS [97], GAssist [105], BioHEL [125] (a descendant of GAssist), and NAX [135] (a descendant of GALE). All but ATNoSFERES were designed primarily to address classification/data mining problems to which Pitt-style systems seem to be fundamentally suited. NAX and BioHEL both received recent praise for their human-competitive performance on moderately complex and large tasks. Also, a handful of "hybrid" systems have been developed that merge Michigan and Pitt-style architectures (e.g., REGAL [48], GA-Miner [55], ZCCS [61], and CXCS [68]).

1.5.5 Visualization

There is an expanding wealth of literature beyond what we have discussed in this brief history [174]. One final innovation that will likely prove to be of great significance to the LCS community is the design and application of visualization tools. Such tools allow researchers to follow algorithmic progress by (1) tracking on-line performance (i.e., by graphing metrics such as error, generality, and population size), (2) visualizing the current classifier population as it evolves (i.e., condition visualization), and (3) visualizing the action/prediction (useful in function approximation to visualize the current prediction surface) [121,131,175]. Examples include Holmes's EpiXCS Workbench geared towards knowledge discovery in medical data [121], and Butz and Stalph's cutting-edge XCSF visualization software geared towards function approximation [175, 176] and applied to robotic control in [177]. Tools such as these will advance algorithmic understandability and facilitate solution interpretation, while simultaneously fueling a continued interest in the LCS algorithm.

1.6 Problem Domains

The range of problem domains to which LCS has been applied can be broadly divided into three categories: function approximation problems, classification problems, and reinforcement learning problems [178]. All three domains are generally tied by the theme of optimizing prediction within an environment. *Function approxima-*

tion problems seek to accurately approximate a function represented by a partially overlapping set of approximation rules (e.g., a piece-wise linear solution for a sine function). *Classification problems* seek to find a compact set of rules that classify all problem instances with maximal accuracy. Such problems frequently rely on supervised learning where feedback is provided instantly. A broad sub-domain of the classification problem includes "data mining", which is the process of sorting through large amounts of data to extract or model useful patterns. Classification problems may also be divided into either Boolean or real-valued problems based on the problem type being respectively discrete, or continuous in nature. Examples of classification problems include Boolean function learning, medical diagnosis, image classification (e.g., letter recognition), pattern recognition, and game analysis. *RL problems* seek to find an optimal behavioral policy represented by a compact set of rules. These problems are typically distinguished by inconsistent environmental reward often requiring multiple actions before such reward is obtained (i.e., multi-step RL problem or sequential decision task). Examples of such problems would include robotic control, game strategy, environmental navigation, modeling time-dependant complex systems (e.g., stock market), and design optimization (e.g., engineering applications). Some RL problems are characterized by providing immediate reward feedback about the accuracy of a chosen class (i.e., single-step RL problem), which essentially makes it similar to a classification problem. RL problems can be partitioned further based on whether they can be modeled as a Markov decision process (MDP) or a partially

observable Markov decision process (POMDP) . In short, for Markov problems the selection of the optimal action at any given time depends only on the current state of the environment and not on any past states. On the other hand, Non-Markov problems may require information on past states to select the optimal action. For a detailed introduction to this concept we refer readers to [9, 179, 180].

1.7 Biological Applications

One particularly demanding and promising domain for LCS application involves biological problems (e.g. epidemiology, medical diagnosis, and genetics). In order to gain insight into complex biological problems researchers often turn to algorithms that are themselves inspired by biology (e.g., genetic programming [181], ant colony optimization [182], artificial immune systems [183], and neural networks [184]). Similarly, since the mid 90's biological LCS studies have begun to appear that deal mainly with classification-type problems. One of the earliest attempts to apply an LCS algorithm to such a problem was [139]. Soon after, John Holmes initiated a lineage of LCS designed for epidemiological surveillance and knowledge discovery that included BOOLE++ [57], EpiCS [58], and most recently EpiXCS [120]. Similar applications include [70, 72, 107, 119, 185–187], all of which examined the Wisconsin breast cancer data taken from the UCI repository [188]. LCSs have also been applied to protein structure prediction [108, 126, 131], diagnostic image classification [135, 189], and promoter region identification [190].

1.8 Optimizing LCS

There are a number of factors to consider when trying to select or develop an "effective" LCS. The ultimate value of an LCS might be gauged by the following: (1) *performance* - the quality of the evolved solution (rule set), (2) *scalability* - how rapidly the learning time or system size grows as the problem complexity increases, (3) *adaptivity* - the ability of on-line learning systems to adapt to rapidly changing situations, and/or (4) *speed* - the time it takes an off-line learning system to reach a "good" solution. Much of the field's focus has been placed on optimizing performance (as defined here). The challenge of this task is in balancing algorithmic pressures designed to evolve the population of rules towards becoming what might be considered an optimal rule set. The definition of an optimal rule set is subjective, depending on the problem domain, and the system architecture. Kovacs discusses the properties of an optimal XCS rule set $[O]$ as being correct, complete, minimal (compact), and non-overlapping [191]. Even for the XCS architecture it is not clear that these properties are always optimal (e.g., discouraging overlap prevents the evolution of default hierarchies, too much emphasis on correctness may lead to over-fitting in training, and completeness is only important if the goal is to evolve a complete action map). Some of the tradeoffs are discussed in [192, 193]. Instead, researchers may use the characteristics of *correctness, completeness, compactness*, and *overlap* as metrics with which to track evolutionary learning progress. LCS, being a complex multifaceted algorithm is subject to a number of different pressures driving the rule-set evolution.

Butz and Pelikan discuss five pressures that specifically influence XCS performance, and provide an intuitive visualization of how these pressures interact to evolve the intended complete, accurate, and maximally general problem representation [194, 195]. These include *set pressure* (an intrinsic generalization pressure), *mutation pressure* (that influences rule specificity), *deletion pressure* (included in set pressure), *subsumption pressure* (decreases population size), and *fitness pressures* (that generate a major drive towards accuracy). Other pressures have also been considered, including parsimony pressure for discouraging large rule sets (i.e. bloat) [196], and crowding (or niching) pressure for allocating classifiers to distinct sub-sets of the problem domain [24]. In order to ensure XCS success, Butz defines a number of learning bounds that address specific algorithmic pitfalls [197–200]. Broadly speaking, the number of studies addressing LCS theory are few in comparison to applications-based research. Further work in this area would certainly benefit the LCS community.

1.9 Component Roadmap

The following section is meant as a summary of the different LCS algorithmic components. Figure 1.2 encapsulates the primary elements of a generic LCS framework (heavily influenced by ZCS, XCS, and other Michigan-style systems). Using this generalized framework we identify a number of exchangeable methodologies, and direct readers towards the studies that incorporate them. Many of these elements have been introduced in section 1.5, but are put in the context of the working algo-

rithm here. It should be kept in mind that some outlying LCS implementations stray significantly from this generalized framework, and while we present these components separately, the system as a whole is dependent on the interactions and overlaps that connect them. Elements that don't obviously fit into the framework of Figure 1.2 will be discussed in section 1.9.6. Readers interested in a simple summary and schematic of the three most renowned systems (including Holland's standard LCS, ZCS, and XCS) are referred to [201].

1.9.1 Detectors and Effectors

The first and ultimately last step of an LCS iteration involves interaction with the environment. This interaction is managed by detectors and effectors [27]. Detectors sense the current state of the environment and encode it as a standard message (i.e. formatted input data). The impact of how sensors are encoded has been explored [202]. Effectors, on the other hand, translate action messages into performed actions that modify the state of the environment. For supervised learning problems, the action is supplanted by some prediction of class, and the job of effectors is simply to check that the correct prediction was made. Depending on the efficacy of the systems' predicted action or class, the environment may eventually or immediately reward the system. As mentioned previously, the environment is the source of input data for the LCS algorithm, dependant on the problem domain being examined. *"The learning capabilities of LCS rely on and are constrained by the way the agent perceives the*

environment, e.g., by the detectors the system employs" [156]. Also, the format of the input data may be binary, real-valued, or some other customized representation. In systems dealing with batch learning, the dataset that makes up the environment is often divided into a training and a testing set (e.g. [58]) as part of a cross-validation strategy to assess performance and ensure against over-fitting.

1.9.2 Population

Modifying the knowledge representation of the population can occur on a few levels. First and foremost is the difference in overall population structure as embodied by the Michigan and Pitt-style families. In Michigan systems the population is made up of a single rule-set that represents the problem solution, and in Pitt systems the population is a collection of multiple competing rule-sets, each of which represent a potential problem solution (see Figure 1.4). Next, is the overall structure of an individual rule. Most commonly, a rule is made up of a condition, an action, and one or more parameter values (typically including a prediction value and/or a fitness value) [1, 19, 22], but other structures have been explored e.g., the condition-action-effect structure used by ACSs [63]. Also worth mentioning are rule-structure-induced mechanisms, proposed to encourage the evolution of rule dependencies and internal models. Examples include: bridging-classifiers (to aid the learning of long action chains) [148, 154], tagging (a form of implicitly linking classifiers) [1, 203, 204], and classifier-chaining (a form of explicitly linking classifiers and the defining feature of

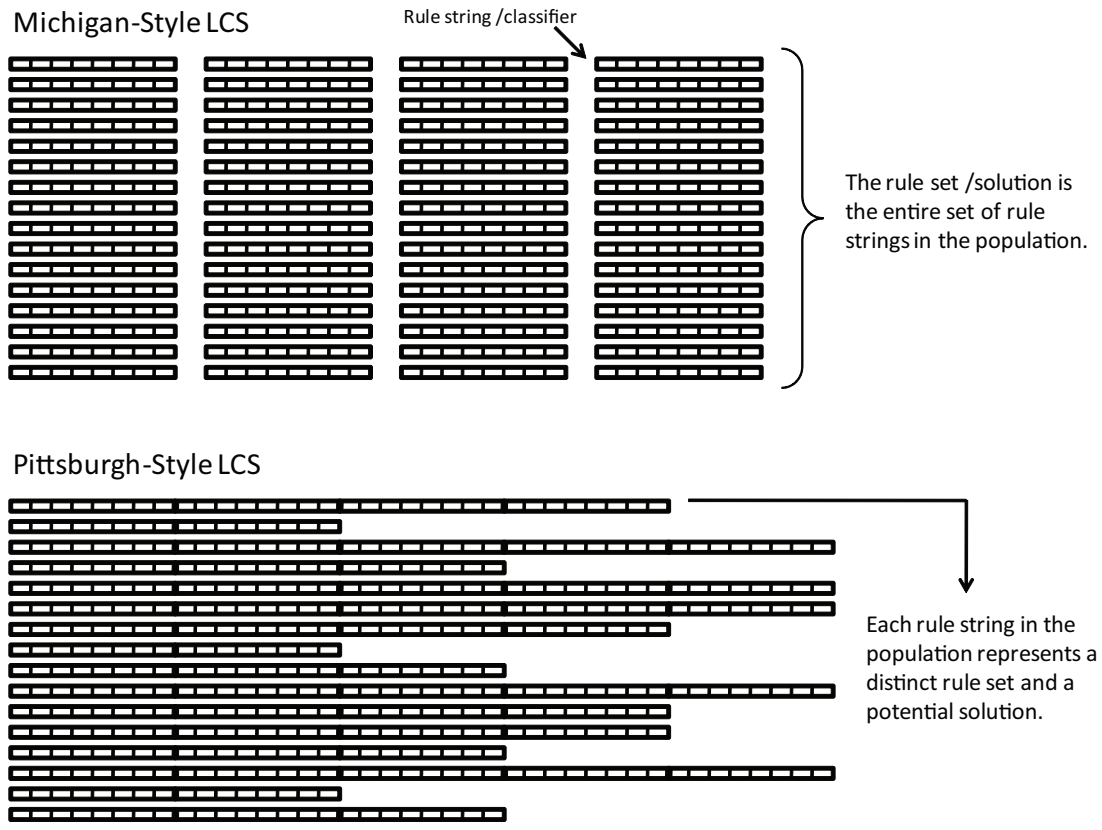

Figure 1.4: Michigan vs. Pitt-style systems.

a "corporate" classifier system) [61, 68]. The most basic level of rule representation is the *syntax*, which depicts how either the condition or action is actually depicted. Many different syntaxes have been examined for representing a rule condition. The first, and probably most commonly used syntax for condition representation was fixed length bit-strings of the ternary alphabet $(0, 1, \#)$ corresponding with the simple binary encoding of input data [1, 3, 19, 22]. Unfortunately, it has been shown that this type of encoding can introduce bias as well as limit the system's ability to represent a problem solution [205]. For problems involving *real-valued* inputs the following condition syntaxes have been explored: real-valued alphabet [65], center-based interval predicates [69], min-max interval predicates [72], unordered-bound interval predicates [206], min-percentage representation [207], convex hulls [208], real-valued context-free grammar [132], ellipsoids [209], and hyper-ellipsoids [210]. Other condition syntaxes include: partial matching [211], value representation [203], tokens, [76, 82, 109, 111], context-free grammar [112], first-order logic expressions [212], messy conditions [213], GP-like conditions (including s-expressions) [55,214–218], neural networks [2,93,219,220], and fuzzy logic [39,51,53,133,134,221]. Overall, advanced representations tend to improve generalization and learning, but require larger populations to do so. Action representation has seen much less attention. Actions are typically encoded in binary or by a set of symbols. Recent work has also begun to explore the prospect of computed actions, also known as computed prediction, which replaces the usual classifier action parameter with a function (e.g. XCSF function

approximation) [90, 129]. Neural network predictors have also been explored [123]. Backtracking briefly, in contrast to Michigan-style systems, Pitt-style implementations tend to explore different rule *semantics* and typically rely on a simple binary syntax. Examples of this include: VL1 [44], CNF [42], and ADI [105, 125]. Beyond structural representation, other issues concerning the population include: (1) population initialization, (2) deciding whether to bound the population size (N), and if it is bound, (3) what value of (N) to select [106, 200].

1.9.3 Performance Component and Selection

This section will discuss different performance component structures and the selection mechanisms involved in covering, action selection, and the GA. The *message list*, a component found in many early LCSs (not included on Figure 1.2), is a kind of blackboard that documents the current state of the system. Acting as an interface, the message list temporarily stores all communications between the system and the environment (i.e. inputs from the detector, and classifier-posted messages that culminate as outputs to the effector) [27, 31, 147, 201]. One potential benefit of using the message list is that the LCS *"can emulate memory mechanisms when a message is kept on the list over several time steps"* [9]. The role of message lists will be discussed further in the context of the BBA in section 1.9.4. While the match set [M] is a ubiquitous component of Michigan-style systems the action set [A] only appeared after the removal of the internal message list. [A] provided a physical location with which

to track classifiers involved in sending action messages to the effector. Concurrently, a *previously active action set* $[A]_{t-1}$ was implemented to keep track of the last set of rules to have been placed in $[A]$. This temporary storage allows reward to be implicitly passed up the activating chain of rules and was aptly referred to as an implicit bucket brigade. For LCSs designed for supervised learning (e.g., NEWBOOLE [36] and UCS [94]), the sets of the performance component take on a somewhat different appearance, with $[A]$ being replaced with a correct set $[C]$, and not-correct set $Not[C]$ to accommodate the different learning style. Going beyond the basic set structure, XCS also utilized a *prediction array* added to modify both action selection and credit assignment [22]. In brief, the prediction array calculates a system prediction $P(a_j)$ for each action a_j represented in $[M]$. $P(a_j)$ represents the strength (the likely benefit) of selecting the given a_j based on the collective knowledge of all classifiers in $[M]$ that advocate a_j. Its purpose will become clearer in section 1.9.4. Modern LCS selection mechanisms serve three main functions: (1) using the classifiers to make an action decision, (2) choosing parent rules for GA "mating", and (3) picking out classifiers to be deleted. Four selection mechanisms are frequently implemented to perform these functions. They include: (1) *purely stochastic* (random) *selection*, (2) *deterministic selection* - the classifier with the largest fitness or prediction (in the case of action selection) is chosen, (3) *proportionate selection* (often referred to as roulette-wheel selection) - where the chances of selection are proportional to fitness, and (4) *tournament selection* - a number of classifiers (s) are selected at random and the one with

the largest fitness is chosen. Recent studies have examined selection mechanisms and noted the advantages of tournament selection [96, 222–225]. It should be noted that when selecting classifiers for deletion, any fitness-based selection will utilize the inverse of the fitness value so as to remove less-fit classifiers. Additionally, when dealing with action selection, selection methods will rely on the prediction parameter instead of fitness. Also, it is not uncommon, especially in the case of action selection, to alternate between different selection mechanisms (e.g. MCS alternates between stochastic and deterministic schemes from one iteration to the next). Sometimes this method is referred to as the pure explore/exploit scheme [19]. While action selection occurs once per iteration, deletion occurs in the following circumstance; the global population (N) is bound, and new classifiers are being added to a population that has reached (N). At this point, a corresponding number of classifiers must be deleted. This may occur following covering (explained in section 1.4) or after the GA has been triggered. Different GA triggering mechanisms are discussed in section 1.9.5. Of final note is a bidding mechanism. Bidding was used by Holland's LCS to select and allow the strongest n classifiers in $[M]$ to post their action messages to the message list. Additionally either bidding or a conflict resolution module [156] may be advocated for action selection from the message list. A classifier's "bid" is proportional to the product of its strength and specificity. The critical role of bidding in the BBA is discussed in the next section.

1.9.4 Reinforcement Component

Different LCS credit assignment strategies are bound by a similar objective (to distribute reward), but the varying specifics regarding where/when they are called, what parameters are included and updated, and what formulas are used to perform those updates have lead to an assortment of methodologies, many of which have only very subtle differences. As a result, the nomenclature used to describe an LCS credit assignment scheme is often vague (e.g. Q-Learning-Based [22]) and occasionally absent. Therefore to understand the credit assignment used in a specific system, we refer readers to the relevant primary source. Credit assignment can be as simple as updating a single value (as is implemented in MCS), or it may require a much more elaborate series of steps (e.g. BBA). We briefly review two of the most historically significant credit assignment schemes i.e., the BBA and XCS's Q-Learning-based strategy.

1.9.4.1 Bucket Brigade

"The bucket brigade [BBA] *may most easily be viewed as an information economy where the right to trade information is bought and sold by classifiers. Classifiers form a chain of middlemen from information manufacturer (*[detectors of] *the environment) to information consumer (the effectors)" - Goldberg.* [8] The BBA, as described below, involves both performance and reinforcement components. The following steps outline its progression over a single time iteration (t): It should be noted

that within a given (t), the message list can receive only a limited number of input messages as well as a limited number of classifier postings. Also, when a classifier posts a message to the current message list it is said to have been "activated" during (t).

1. Post one or more messages from the detector to the current message list $[ML]$.
2. Compare all messages in $[ML]$ to all conditions in $[P]$ and record all matches in $[M]$.
3. Post "action" messages of the highest bidding classifiers of $[M]$ onto $[ML]$.
4. Reduce the strengths of these activated classifiers $\{C\}$ by the amount of their respective bids B(t) and place those collective bids in a "bucket" B_{total}. (Paying for the privilege of posting a new message).
5. Distribute B_{total} evenly over the previously activated classifiers $\{C'\}$. (Suppliers $\{C'\}$ are rewarded for setting up a situation usable by $\{C\}$).
6. Replace messages in $\{C'\}$ with those in $\{C\}$ and clear $\{C\}$. (Updates record of previously activated classifiers).
7. $[ML]$ is processed through the output interface(effector) to provoke an action.
8. This step occurs if a reward is returned by the environment. The reward value is added to the strength of all classifiers in $\{C\}$ (the most recently activated classifiers receive the reward).

"Whenever a classifier wins a bidding competition, it initiates a transaction in which it pays out part of its strength to its suppliers and then receives similar payments from its consumers. [This] *strength is a kind of capital. If a classifier receives more from its consumers than it paid out, it has made a profit, that is its strength is increased" - Holland* [27]. The update for any given classifier can be summarized by the following equation where $S(t)$ is classifier strength, $B(t)$ is the bid of the classifier (see step 4), $P(t)$ is the sum of all payments made to this classifier by $\{C\}$ (see step 5), and $R(t)$ is any reward received (see step 8):

$$S(t+1) = S(t) - B(t) + P(t) + R(t)$$

The desired effect of this cycle is to enable classifiers to pass reward (when received) along to classifiers that may have helped make that reward possible. See [148] and [8] for more details.

1.9.4.2 Q-Learning-based

The Q-learning-based strategy used by XCS is an archetype of modern credit assignment. First off, it should be noted that the performance component of XCS is similar to that described for MCS (although XCS adds a prediction array and an $[A]_{t-1}$, both imperative to the credit assignment strategy). Each classifier (j) in XCS tracks four parameters: prediction (p), prediction error (ϵ), fitness (F), and experience (e). The update of these parameters takes place in $[A]_{t-1}$ as follows:

1. Each rule's ϵ is updated: $\epsilon_j \leftarrow \epsilon_j + \beta(|P - p_j|) - \epsilon_j)$

2. Rule predictions are updated: $p_j \leftarrow p_j + \beta(P - p_j)$

3. Each rule's accuracy is determined: $\kappa_j = exp[(ln\alpha)(\epsilon_j - \epsilon_0)/\epsilon_0)]$ for $\epsilon_j > \epsilon_0$ otherwise 1.

4. A relative accuracy κ'_j, is determined for each rule: $\kappa'_j = \kappa_j / \Sigma\kappa_{[A]t-1}$

5. Each rule's F is updated using κ'_j: $F_j \leftarrow F_j + \beta(\kappa'_j - F_j)$

6. Increment e for all classifiers in $[A]$.

β is a learning rate constant ($0 \leq \beta \leq 1$) while ϵ_0 and α are accuracy function parameters. The procedure used to calculate p, ϵ, and F is the widely implemented Widrow-Hoff formula [226] (also known as Least Mean Square) seen here:

$$x \leftarrow x + \beta(y - x)$$

An important caveat is that initially p, ϵ, and F are actually updated by respectively averaging together their current and previous values. It is only after a classifier has been adjusted at least $1/\beta$ times that the Widrow-Hoff procedure takes over parameter updates. This technique, referred to as "moyenne adaptive modifee" [227], is used to make early parameter values move more quickly to their "true" average values in an attempt at avoiding the arbitrary nature of early parameter values. The direct influence of Q-learning on this credit assignment scheme is found in the update of p_j that takes the maximum prediction value from the prediction array, discounts it by a factor, and adds in any external reward received in the previous time. The

resulting value, which Wilson calls P (see steps 1 and 2), is somewhat analogous to Q-Learning's Q-values. Also observe that a classifier's fitness is dependent on its ability to make accurate predictions, but is not proportional to the prediction value itself. For further perspective on basic modern credit assignment strategy see [19, 22, 228].

1.9.4.3 More Credit Assignment

Further differences in credit assignment schemes also exist. For example Pitt-style systems track credit at the level of entire rule sets as opposed to assigning parameters to individual rules. Supervised learning systems like UCS have basically eliminated the reinforcement component (as it is generally understood) and instead maintains and updates a single accuracy parameter [94]. Of course, many other credit assignment and parameter update strategies have been suggested and implemented. Here we list some of these strategies: epochal [3], implicit bucket brigade [25], one-step payoff-penalty [28], symmetrical payoff-penalty [36], hybrid bucket brigade-backward averaging (BB-BA) algorithm [229], non-bucket brigade temporal difference method [37], action-oriented credit assignment [230, 231], QBB [19], average reward [91], gradient descent [232, 233], eligibility traces [234], Bayesian update [124], least squares update [235], and Kalman filter update [235].

1.9.5 Discovery Components

A standard discovery component is comprised of a GA and a covering mechanism. The primary role of the covering mechanism is to ensure that there is at least one classifier in $[P]$ that can handle the current input. A new rule is generated by adding some number of #'s (wild cards) at random to the input string and then selecting a random action (i.e., the new rule "0#110#0 − 01" might be generated from the input string 0011010) [25]. The random assignment of an action has been noted to be helpful escaping *loops* [22]. The parameter value(s) of this newly generated classifier are set to the population average. Covering might also be used to initialize classifier populations on the fly, instead of starting the system with an initialized population of maximum size. The covering mechanism can be implemented differently by modifying the frequency that #'s are added to the new rule [22,25,28], altering how a new rule's parameters are calculated [19], and expanding the instances in which covering is called (e.g. ZCS will "cover" when the total strength of $[M]$ is less than a fraction of the average seen in $[P]$ [19]). Covering does more than just handle an unfamiliar input by assigning a random action. *"Covering allows the system* [to] *test a hypothesis (the condition-action relation expressed by the created classifier) at the same time"* [19]. The GA discovers rules by building upon knowledge already in the population (i.e. the fitness). The vast majority of LCS implementations utilize the GA as its primary discovery component. Specifically, LCSs typically use *steady state* GAs, where rules are changed in the population individually without any defined notion of a generation.

This differs from *generational* GAs where all or an important part of the population is renewed from one generation to the next [9]. GAs implemented independent of an LCS are typically *generational*. In selecting an algorithm to address a given problem, an LCS algorithm that incorporates a GA would likely be preferable to a straightforward GA when dealing with more complex decision making tasks, specifically ones where a single rule cannot effectively represent the solution, or in problem domains where adaptive solutions are needed. Like the covering mechanism, the specifics of how a GA is implemented in an LCS may vary from system to system. Three questions seem to best summarize these differences: (1) Where is the GA applied? (2) When is GA triggered? and (3) What operators does it employ? The set of classifiers to which the GA is applied can have a major impact on the evolutionary pressure it produces. While early systems applied the GA to $[P]$ [3], the concepts of *restricted mating* and niching [24] moved its action to $[M]$ and then later to $[A]$, where it is typically applied in modern systems (see Table 1.1). For more on niching see [22, 24, 236, 237]. The triggering of the GA can simply be controlled by a parameter (g) that represents the probability of trigering the GA on a given time step (t), but in order to more fairly allocate the application of the GA to different developing niches, it can be triggered by a tracking parameter [22, 32]. Crossover and mutation are the two most recognizable operators of the GA. Both mechanisms are controlled by parameters representing their respective probabilities of being called. Historically, most early LCSs used simple one-point crossover, but interest in discovering complex "building

blocks" [8, 238, 239] has led to examining two-point, uniform, and informed crossover (based on estimation of distribution algorithms) as well [239]. Additionally, a smart crossover operator for a Pitt-style LCS has also been explored [240]. The GA is a particularly important component of Pitt-style systems that relies on it as its only adaptive process. Oftentimes it seems more appropriate to classify Pitt-style systems simply as an evolutionary algorithm as opposed to what is commonly considered to be a modern LCS [9, 156]. Quite differently, the GA is absent from ACSs, instead relying on non-evolutionary discovery mechanisms [63, 76, 80, 95, 109].

1.9.6 Beyond the Basics

This section briefly identifies LCS implementation themes that extend beyond the basic framework such as the addition of memory, multi-learning classifier systems, multi-objectivity, and data concerns. While able to deal optimally with Markov problems, the major drawback of simpler systems like ZCS and XCS was their relative inability to handle non-Markov problems. One of the methodologies that were developed to address this problem was the addition of memory via an internal register (i.e. a non-message-list memory mechanism) that can store a limited amount of information reguarding a previous state. Systems adopting memory include ZCSM [54], XCSM [59], and XCSMH [67]. Another area that has drawn attention is the development of what we will call "multi-learning classifier systems" (MLCSs) (i.e., multi-agent LCSs, ensemble LCSs, and distributed LCSs) that run

more than one LCS at a time. Multi-agent LCSs were designed to model multi-agent systems that intrinsically depend on the interaction between multiple intelligent agents (e.g. game-play) [71, 241, 242]. Ensemble LCSs were designed to improve algorithmic performance and generalization via parallelization [117–119, 130, 243–245]. Distributed LCSs were developed to assimilate distributed data (i.e. data coming from different sources) [114, 246, 247]. Similar to the concept of MLCS, Ranawana and Palade published a detailed review and roadmap on *multi-classifier systems* [248]. Multi-objective LCSs discussed in [249] explicitly address the goals implicitly held by many LCS implementations (i.e. accuracy, completeness, minimalism) [74, 97]. A method that has been explored to assure minimalism is the application of a rule compaction algorithm for the removal of redundant or strongly overlapping classifiers [73, 99, 250, 251]. Some other dataset issues that have come up especially in the context of data mining include missing data [107,252], unbalanced data [253], dataset size [254], and noise [97]. Some other interesting algorithmic innovations include partial matching [211], endogenous fitness [255], self-adapted parameters [219, 256–259], abstraction, [260], and macro-classifiers [22].

1.10 Conclusion

"Classifier Systems are a quagmire - a glorious, wondrous, and inventing quagmire, but a quagmire nonetheless" - Goldberg [261]. This early perspective was voiced at a time when LCSs were still quite complex and nebulously understood. Structurally

speaking, the LCS algorithm is an interactive merger of other stand-alone algorithms. Therefore, its performance is dependent not only on individual components but also on the interactive implementation of the framework. The independent advancement of GA and learning theory (in and outside the context of LCS) has inspired an innovative generation of systems that no longer merrit the label of a "quagmire". The application of LCSs to a spectrum of problem domains has generated a diversity of implementations. However it is not yet obvious which LCSs are best suited to address a given domain, nor how to best optimize performance. The basic XCS architecture has not only revitalized interest in LCS research, but has become the model framework upon which many recent modifications or adaptations have been built. These expansions are intended to address inherent limitations in different problem domains, while sticking to a trusted and recognizable framework. But will this be enough to address relevant real-world applications? One of the greatest challenges for LCS might inevitably be the issue of scalability as the problems we look to be solved increase exponentially in size and complexity. Perhaps, as was seen pre-empting the development of ZCS and XCS, the addition of heuristics to an accepted framework might again pave the way for some novel architecture(s). Perhaps there will be a return to Holland-style architectures as the limits of XCS-based systems are reached. A number of theoretical questions should also be considered: What are the limits of the LCS framework? How will LCS take advantage of advancing computational technology? How can we best identify and interpret an evolved population (solution)? And how

can we make using the algorithm more intuitive and/or interactive? Beginning with a gentle introduction, this paper has described the basic LCS framework, provided a historical review of major advancements, and provided an extensive roadmap to the problem domains, optimization, and varying components of different LCS implementations. It is hoped that by organizing many of the existing components and concepts, they might be recycled into or inspire new systems that are better adapted to a specific problem domain. The ultimate challenge in developing an optimized LCS is to design an implementation that best arranges multiple interacting components, operating in concert, to evolve an accurate, compact, comprehensible solution, in the least amount of time, making efficient use of computational power. It seems likely that LCS research may culminate in one of two ways. Either there will be some dominant core platform, flexibly supplemented by a variety of problem specific modifiers, or will there be a handful of fundamentally different systems that specialize to different problem domains. Whatever the direction, it is likely that LCS will continue to evolve and inspire methodologies designed to address some of the most difficult problems ever presented to a machine.

1.11 Resources

For various perspectives on the LCS algorithm, we refer readers to the following review papers [4, 7, 9, 152, 156, 158, 165, 178, 262]. Together, [152], and [165] represent two decade-long consecutive summaries of current systems, unsolved problems and

future challenges. For comparative system discussions see [79, 192, 263, 264]. For a detailed summary of LCS community resources as of (2002) see [265]. For a detailed examination of the design and analysis of LCS algorithms see [266].

1.12 Bibliography

[1] Holland J (1996) Hidden Order: How Adaptation Builds Complexity. Addison Wesley Publishing Company.

[2] Dam H, Abbass H, Lokan C, Yao X (2008) Neural-Based Learning Classifier Systems. Transactions on Knowledge and Data Engineering 20: 26–39.

[3] Holland J, Reitman J (1978) Cognitive systems based on adaptive agents. In D A Waterman and F Hayes-Roth (Eds) Pattern-directed inference systems .

[4] Holmes J, Lanzi P, Stolzmann W, Wilson S (2002) Learning classifier systems: New models, successful applications. Information Processing Letters 82: 23–30.

[5] Minsky M (1961) Steps toward artificial intelligence. Proc IRE 49: 8–30.

[6] Holland J (1975) Adaptation in natural and artificial systems. University of Michigan Press Ann Arbor.

[7] Bull L, Kovacs T (2005) Foundations of Learning Classifier Systems. Springer.

[8] Goldberg D (1989) Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA.

[9] Sigaud O, Wilson S (2007) Learning classifier systems: a survey. Soft Computing-A Fusion of Foundations, Methodologies and Applications 11: 1065–1078.

[10] Holmes J (2000) Learning Classifier Systems Applied to Knowledge Discovery in Clinical Research Databases. Learning Classifier Systems, From Foundations to Applications : 243–262.

[11] Goldberg D (2002) The Design of Innovation: Lessons from and for Competent Genetic Algorithms. Kluwer Academic Publishers.

[12] Langley P (1996) Elements of Machine Learning. Morgan Kaufmann.

[13] Harries M, Sammut C, Horn K (1998) Extracting Hidden Context. Machine Learning 32: 101–126.

[14] Ben-David S, Kushilevitz E, Mansour Y (1997) Online Learning versus Offline Learning. Machine Learning 29: 45–63.

[15] Sutton R, Barto A (1998) Reinforcement Learning: An Introduction. MIT Press.

[16] Harmon M, Harmon S (1996) Reinforcement Learning: A Tutorial. View or download on http://eureka1 aa wpafb af mil/rltutorial/, december .

[17] Wyatt J (2005) Reinforcement Learning: a brief overview. Foundations of Learning Classifier Systems : 179–202.

[18] Bull L (2005) Two Simple Learning classifier Systems. Foundations of Learning Classifier Systems : 63–89.

[19] Wilson S (1994) ZCS: A Zeroth Level Classifier System. Evolutionary Computation 2: 1–18.

[20] Goldberg D (1983) Computer-aided gas pipeline operation using genetic algorithms and machine learning. Ph.D. thesis, Doctoral dissertation, Department Civil Engineering, University of Michigan, Ann Arbor.

[21] Baker J (1987) Reducing bias and inefficiency in the selection algorithm. Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application table of contents : 14–21.

[22] Wilson S (1995) Classifier Fitness Based on Accuracy. Evolutionary Computation 3: 149–175.

[23] Smith S (1980) A learning system based on genetic adaptive algorithms. Ph.D. thesis, University of Pittsburgh.

[24] Booker L (1982) Intelligent behavior as an adaptation to the task environment .

[25] Wilson S (1985) Knowledge Growth in an Artificial Animal. Proceedings of the 1st International Conference on Genetic Algorithms table of contents : 16–23.

[26] Schaffer J, Grefenstette J (1985) Multi-objective learning via genetic algorithms. Proceedings of the Ninth International Joint Conference on Artificial Intelligence : 593–595.

[27] Holland J (1986) Escaping brittleness: The possibilities of general-purpose learning algorithms applied to parallel rule-based systems. Machine learning, an artificial intelligence approach 2: 593–623.

[28] Wilson S (1987) Classifier Systems and the Animat Problem. Machine Learning 2: 199–228.

[29] Greene D (1987) Automated knowledge acquisition: Overcoming the expert system bottleneck. i¿ Proceedings of the Eighth International Conference on Information Systems,¡/i¿ J DeGross and C Kriebel (Eds), Pittsburgh : 107–117.

[30] Grefenstette J (1988) Credit assignment in rule discovery systems based on genetic algorithms. Machine Language 3: 225–245.

[31] Booker L (1988) Classifier Systems that Learn Internal World Models. Machine Learning 3: 161–192.

[32] Booker L (1989) Triggered rule discovery in classifier systems. Proceedings of the third international conference on Genetic algorithms table of contents : 265–274.

[33] Grefenstette J (1989) Incremental learning of control strategies with genetic algorithms. Proceedings of the sixth international workshop on Machine learning table of contents : 340–344.

[34] Grefenstette J (1992) The Evolution of Strategies for Multiagent Environments. Adaptive Behavior 1: 65.

[35] Grefenstette J (1997) The Users Guide to SAMUEL-97: An Evolutionary Learning System. Navy Center for Applied Research in Artificial Intelligence, Naval Research Laboratory, Washington, DC, USA .

[36] Bonelli P, Parodi A, Sen S, Wilson S (1990) NEWBOOLE: a fast GBML system. Proceedings of the seventh international conference (1990) on Machine learning table of contents : 153–159.

[37] Riolo R (1991) Lookahead planning and latent learning in a classifier system. Proceedings of the first international conference on simulation of adaptive behavior on From animals to animats table of contents : 316–326.

[38] Shu L, Schaeffer J (1991) HCS: Adding Hierarchies to Classifier Systems. Proceedings of the 4th International Conference on Genetic Algorithms : 339–345.

[39] Valenzuela-Rendon M (1991) The fuzzy classifier system: a classifier system for continuously varying variables. Proceedings of the Fourth International Conference on Genetic Algorithms : 346–353.

[40] Dorigo M, Sirtori E (1991) Alecsys: A Parallel Laboratory for Learning Classifier Systems. Proceedings of the Fourth International Conference on Genetic Algorithms .

[41] Dorigo M (1995) Alecsys and the AutonoMouse: Learning to control a real robot by distributed classifier systems. Machine Learning 19: 209–240.

[42] De Jong K, Spears W (1991) Learning Concept Classification Rules using Genetic Algorithms. Proceedings of the Twelth International Conference on Artificial Intelligence IJCAI-91 2.

[43] De Jong K, Spears W, Gordon D (1993) Using Genetic Algorithms for Concept Learning. Machine Learning 13: 161–188.

[44] Janikow C (1991) Inductive Learning of decision rules in attribute-based examples: a knowledge-intensive genetic algorithm approach. Ph.D. thesis, PhD thesis, University of North Carolina.

[45] Janikow C (1993) A knowledge-intensive genetic algorithm for supervised learning. Machine Learning 13: 189–228.

[46] Greene D (1992) Inductive knowledge acquisition using genetic adaptive search. Ph.D. thesis, Doctoral dissertation, The.

[47] Greene D, Smith S (1993) Competition-Based Induction of Decision Models from Examples. Machine Learning 13: 229–257.

[48] Giordana A, Saitta L (1993) REGAL: An integrated system for learning relations using genetic algorithms. Proceedings of the 2nd International Workshop on Multistrategy Learning : 234–249.

[49] Giordana A, Saitta L, Zini F (1994) Learning disjunctive concepts with distributed genetic algorithms. Evolutionary Computation, 1994 IEEE World Congress on Computational Intelligence, Proceedings of the First IEEE Conference on : 115–119.

[50] Giordana A, Neri F (1995) Search-Intensive Concept Induction. Evolutionary Computation 3: 375–416.

[51] Bonarini A (1993) ELF: learning incomplete fuzzy rule sets for an autonomous robot. Proc of EUFIT '93 ELITE Foundation, Aachen, Germany : 69–75.

[52] Bonarini A (1994) Some methodological issues about designing autonomous agents which learn their behaviors: the ELF experience. R Trappl (ed) Cybernetics and Systems Research '94 : 1435–1442.

[53] Bonarini A (1996) Evolutionary learning of fuzzy rules: competition and cooperation. Fuzzy Modelling: Paradigms and Practice : 265–284.

[54] Cliff D, Ross S (1994) Adding memory to ZCS. Adaptive Behavior 3: 101–150.

[55] Flockhart I, Radcliffe N GA-MINER: parallel data mining with hierarchical genetic algorithms-final report. EPCC AIKMS GA-Miner-Report 1.

[56] Flockhart I, Radcliffe N, Simoudis E, Han J, Fayyad U (1996) A Genetic Algorithm-Based Approach to Data Mining. Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96) : 299–302.

[57] Holmes J (1996) A Genetics-Based Machine Learning Approach to Knowledge Discovery in Clinical Data. AMIA ANNUAL SYMPOSIUM : 883–883.

[58] Holmes J (1997) Discovering Risk of Disease with a Learning Classifier System. Proceedings of the Seventh International Conference of Genetic Algorithms (ICGA97) : 426–433.

[59] Lanzi P, e Inf D (1998) Adding memory to XCS. Evolutionary Computation Proceedings, 1998 IEEE World Congress on Computational Intelligence, The 1998 IEEE International Conference on : 609–614.

[60] Lanzi P (1998) An analysis of the memory mechanism of XCSM. Genetic Programming 98: 643–651.

[61] Tomlinson A, Bull L (1998) A Corporate Classifier System. LECTURE NOTES IN COMPUTER SCIENCE : 550–559.

[62] Tomlinson A, Bull L (1999) A Zeroth Level Corporate Classifier System. Proceedings of the 1999 Genetic and Evolutionary Computation Conference Workshop Program : 306–313.

[63] Stolzmann W (1998) Anticipatory classifier systems. Proceedings of the third annual genetic programming conference : 658–664.

[64] Stolzmann W (2000) An Introduction to Anticipatory Classifier Systems. LECTURE NOTES IN COMPUTER SCIENCE : 175–194.

[65] Browne W (1999) The Development of an Industrial Learning Classifier System for Application to a Steel Hot Strip Mill. Division of Mechanical Engineering and Energy Studies Cardiff, Wales, University of Wales 242.

[66] Browne W, Holford K, Moore C, Bullock J (2000) An industrial Learning Classifier System: the importance of pre-processing real data and choice of alphabet. Engineering Applications of Artificial Intelligence 13: 25–36.

[67] Lanzi P, Wilson S (2000) Toward Optimal Classifier System Performance in Non-Markov Environments. Evolutionary Computation 8: 393–418.

[68] Tomlinson A, Bull L (2000) A Corporate XCS. LECTURE NOTES IN COMPUTER SCIENCE : 195–208.

[69] Wilson S (2000) Get Real! XCS with Continuous-Valued Inputs. LECTURE NOTES IN COMPUTER SCIENCE : 209–222.

[70] Walter D, Mohan C (2000) ClaDia: a fuzzy classifier system for disease diagnosis. In: Evolutionary Computation, 2000. Proceedings of the 2000 Congress on. volume 2.

[71] Takadama K, Terano T, Shimohara K (2000) Learning Classifier Systems Meet Multiagent Environments. In: Third International Workshop on Learning Classifier Systems (IWLCS-2000), L. Lanzi, W. Stolzmann et SW Wilson (Eds.). Springer, Paris, France. Springer, pp. 192–210.

[72] Wilson S (2000) Mining Oblique Data with XCS. Revised Papers from the Third International Workshop on Advances in Learning Classifier Systems : 158–176.

[73] Wilson S (2001) Compact Rulesets from XCSI. Revised Papers from the 4th International Workshop on Advances in Learning Classifier Systems : 197–210.

[74] Bernado-Mansilla E, Garrell-Guiu J (2000) MOLeCS: A MultiObjective Learning Classifier System. In: Proceedings of the 2000 Conference on Genetic and Evolutionary Computation. volume 1.

[75] Mansilla E, Guiu J (2001) MOLeCS: Using Multiobjective Evolutionary Algo-

rithms for Learning. LECTURE NOTES IN COMPUTER SCIENCE : 696–710.

[76] Gérard P, Sigaud O (2000) YACS: Combining dynamic programming with generalization in classifier systems. Advances in learning classifier systems: Third international workshop, IWLCS : 52–69.

[77] Gérard P, Stolzmann W, Sigaud O (2002) YACS: a new learning classifier system using anticipation. Soft Computing-A Fusion of Foundations, Methodologies and Applications 6: 216–228.

[78] Kovacs T (2001) A Comparison of Strength and Accuracy-Based Fitness in Learning Classifier Systems. Ph.D. thesis, PhD thesis, University of Birmingham.

[79] Kovacs T (2002) Two Views of Classifier Systems. LECTURE NOTES IN COMPUTER SCIENCE : 74–87.

[80] Butz M (2001) Biasing Exploration in an Anticipatory Learning Classifier System. Lecture Notes In Computer Science; Vol 2321 : 3–22.

[81] Butz M, Hoffmann J (2002) Anticipations Control Behavior: Animal Behavior in an Anticipatory Learning Classifier System. Adaptive Behavior 10: 75.

[82] Picault S, Landau S ATNoSFERES: a Darwinian Evolutionary Model for In-

dividual or Collective Agent Behavior. Technical report, Technical report (in press), LIP6, Paris, 2001.

[83] Landau S, Picault S, Drogoul A (2001) ATNoSFERES: a Model for Evolutive Agent Behaviors. Proceedings of the AISB 1.

[84] Landau S, Picault S, Sigaud O, Gérard P (2002) A Comparison Between ATNoSFERES And XCSM. Proceedings of the Genetic and Evolutionary Computation Conference table of contents : 926–933.

[85] Landau S, Picault S, Sigaud O, Gerard P (2003) Further Comparison between ATNoSFERES and XCSM. LECTURE NOTES IN COMPUTER SCIENCE : 99–117.

[86] Landau S, Sigaud O, Picault S, Gerard P (2007) An Experimental Comparison Between ATNoSFERES and ACS. LECTURE NOTES IN COMPUTER SCIENCE 4399: 144.

[87] Llora X, Garrell J, et al. (2001) Knowledge-independent data mining with fine-grained parallel evolutionary algorithms. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001), Morgan Kaufmann. pp. 461–468.

[88] Kovacs T (2002) Genetic based machine learning using fine-grained parallelism

for data mining. Ph.D. thesis, PhD thesis, Enginyeria i Arquitectura La Salle. Ramon Llull University.

[89] Llorà X, i Guiu J (2002) Coevolving Different Knowledge Representations With Fine-grained Parallel Learning Classifier Systems. In: Proceedings of the Genetic and Evolutionary Computation Conference table of contents. Morgan Kaufmann Publishers Inc. San Francisco, CA, USA, pp. 934–941.

[90] Wilson S (2002) Classifiers that approximate functions. Natural Computing 1: 211–234.

[91] Tharakunnel K, Goldberg D (2002) XCS with Average Reward Criterion in Multi-step Environment. Technical report, Technical Report, Illinois Genetic Algorithms Laboratory (IlliGAL) Department of General Engineering University of Illinois at Urbana-Champaign, A. 1.

[92] Hurst J, Bull L, Melhuish C (2002) TCS Learning Classifier System Controller on a Real Robot. LECTURE NOTES IN COMPUTER SCIENCE : 588–600.

[93] Bull L, OHara T (2002) Accuracy-based neuro and neuro-fuzzy classifier systems. Proceedings of the Fourth Genetic and Evolutionary Computation Conference (GECCO-2002) : 905–911.

[94] Bernado-Mansilla E, Garrell-Guiu J (2003) Accuracy-Based Learning Classifier

Systems: Models, Analysis and Applications to Classification Tasks. Evolutionary Computation 11: 209–238.

[95] Butz M, Goldberg D (2003) Generalized State Values in an Anticipatory Learning Classifier System. LECTURE NOTES IN COMPUTER SCIENCE : 282–302.

[96] Butz M, Sastry K, Goldberg D (2003) Tournament Selection: Stable Fitness Pressure in XCS. LECTURE NOTES IN COMPUTER SCIENCE : 1857–1869.

[97] Llora X, Goldberg D (2003) Bounding the Effect of Noise in Multiobjective Learning Classifier Systems. Evolutionary Computation 11: 279–298.

[98] Bull L (2003) A Simple Accuracy-based Learning Classifier System. Learning Classifier Systems Group Technical Report UWELCSG03-005, University of the West of England, Bristol .

[99] Dixon P, Corne D, Oates M (2003) A Ruleset Reduction Algorithm for the XCS Learning Classifier System. LECTURE NOTES IN COMPUTER SCIENCE : 20–29.

[100] Bull L (2004) Lookahead and Latent Learning in a Simple Accuracy-Based Classifier System. LECTURE NOTES IN COMPUTER SCIENCE : 1042–1050.

[101] Gaspar A, Hirsbrunner B PICS: Pittsburgh Immune Classifier System .

[102] Gaspar A, Hirsbrunner B From Optimization to Learning in Changing Environments: The Pittsburgh Immune Classier System .

[103] Hurst J, Bull L (2004) A Self-adaptive Neural Learning Classifier System with Constructivism for Mobile Robot Control. LECTURE NOTES IN COMPUTER SCIENCE : 942–951.

[104] Bull L (2004) A Simple Payoff-Based Learning Classifier System. LECTURE NOTES IN COMPUTER SCIENCE : 1032–1041.

[105] Bacardit J (2004) Pittsburgh genetic-based machine learning in the data mining era:representations, generalization, and run-time. Ph.D. thesis, PhD thesis, Enginyeria i Arquitectura La Salle, Ramon Llull University, Barcelona, European Union(Catalonia, Spain).

[106] Bacardit J (2005) Analysis of the initialization stage of a Pittsburgh approach learning classifier system. Proceedings of the 2005 conference on Genetic and evolutionary computation : 1843–1850.

[107] Bacardit J, Butz M (2007) Data Mining in Learning Classifier Systems: Comparing XCS with GAssist. LECTURE NOTES IN COMPUTER SCIENCE 4399: 282.

[108] Stout M, Bacardit J, Hirst J, Smith R, Krasnogor N (2007) Prediction of topological contacts in proteins using learning classifier systems. Soft Com-

puting, Special Issue on Evolutionary and Metaheuristic-based Data Mining (EMBDM),(in press) .

[109] Gérard P, Meyer J, Sigaud O (2005) Combining latent learning with dynamic programming in the modular anticipatory classifier system. European Journal of Operational Research 160: 614–637.

[110] Hamzeh A, Rahmani A (2005) An Evolutionary Function Approximation Approach to Compute Prediction in XCSF. LECTURE NOTES IN COMPUTER SCIENCE 3720: 584.

[111] Landau S, Sigaud O, Schoenauer M (2005) ATNoSFERES revisited. Proceedings of the 2005 conference on Genetic and evolutionary computation : 1867–1874.

[112] Unold O (2005) Context-free grammar induction with grammar-based classifier system. ARCHIVES OF CONTROL SCIENCE 15: 681.

[113] Unold O, Cielecki L (2005) Grammar-based Classifier System. Issues in Intelligent Systems: Paradigms, EXIT, Warsaw : 273–286.

[114] Dam H, Abbass H, Lokan C (2005) DXCS: an XCS system for distributed data mining. Proceedings of the 2005 conference on Genetic and evolutionary computation : 1883–1890.

[115] Dam H, Abbass H, Lokan C (2005) Investigation on DXCS: An XCS system for distribution data mining, with continuous-valued inputs in static and dynamic environments. Proceedings of IEEE Cogress on Evolutionary Computation .

[116] Dam H, Abbass H, Lokan C (2005) The Performance of the DXCS System on Continuous-Valued Inputs in Stationary and Dynamic Environments. Evolutionary Computation, 2005 The 2005 IEEE Congress on 1.

[117] Gao Y, Huang J, Rong H, Gu D (2005) Learning classifier system ensemble for data mining. Proceedings of the 2005 workshops on Genetic and evolutionary computation : 63–66.

[118] Gao Y, Wu L, Huang J (2006) Ensemble Learning Classifier System and Compact Ruleset. LECTURE NOTES IN COMPUTER SCIENCE 4247: 42.

[119] Gao Y, Huang J, Rong H, Gu D (2007) LCSE: Learning Classifier System Ensemble for Incremental Medical Instances. LECTURE NOTES IN COMPUTER SCIENCE 4399: 93.

[120] Holmes J, Sager J (2005) Rule Discovery in Epidemiologic Surveillance Data Using EpiXCS: An Evolutionary Computation Approach. LECTURE NOTES IN COMPUTER SCIENCE 3581: 444.

[121] Holmes J, Sager J (2007) The EpiXCS Workbench: A Tool for Experimentation and Visualization. LECTURE NOTES IN COMPUTER SCIENCE 4399: 333.

[122] Holmes J (2007) Detection of Sentinel Predictor-Class Associations with XCS: A Sensitivity Analysis. LECTURE NOTES IN COMPUTER SCIENCE 4399: 270.

[123] Loiacono D, Lanzi P Evolving Neural Networks for Classifier Prediction with XCSF. In: ECAI 2006, Workshop on Evolutionary Computation. pp. 36–40.

[124] Dam H, Abbass H, Lokan C (2006) BCS: a Bayesian Learning Classifier System. Technical report, Technical Report TR-ALAR-200604005, The Artificial Life and Adaptic Robotics Laboratory, School of Information Technology and Electrical Engineering, University of New South Wales.

[125] Bacardit J, Krasnogor N (2006) Biohel: Bioinformatics-oriented hierarchical evolutionary learning (Nottingham ePrints). University of Nottingham .

[126] Bacardit J, Stout M, Hirst J, Sastry K, Llorà X, et al. (2007) Automated alphabet reduction method with evolutionary algorithms for protein structure prediction. In: Proceedings of the 9th annual conference on Genetic and evolutionary computation. ACM Press New York, NY, USA, pp. 346–353.

[127] Hamzeh A, Rahmani A (2006) Extending XCSFG beyond Linear Approximation. In: Evolutionary Computation, 2006. CEC 2006. IEEE Congress on. pp. 2246–2253.

[128] Hamzeh A, Rahmani A (2007) A New Architecture of XCS to Approximate

Real-Valued Functions Based on High Order Polynomials Using Variable-Length GA. In: Natural Computation, 2007. ICNC 2007. Third International Conference on. volume 3.

[129] Lanzi P, Loiacono D (2007) Classifier systems that compute action mappings. In: Proceedings of the 9th annual conference on Genetic and evolutionary computation. ACM Press New York, NY, USA, pp. 1822–1829.

[130] Gershoff M, Schulenburg S (2007) Collective behavior based hierarchical XCS. In: Proceedings of the 2007 GECCO conference companion on Genetic and evolutionary computation. ACM Press New York, NY, USA, pp. 2695–2700.

[131] Smith R, Jiang M (2007) MILCS: a mutual information learning classifier system. In: Proceedings of the 2007 GECCO conference companion on Genetic and evolutionary computation. ACM Press New York, NY, USA, pp. 2945–2952.

[132] Cielecki L, Unold O (2007) GCS with Real-Valued Input. LECTURE NOTES IN COMPUTER SCIENCE 4527: 488.

[133] Casillas J, Carse B, Bull L (2007) Fuzzy-XCS: A Michigan Genetic Fuzzy System. Fuzzy Systems, IEEE Transactions on 15: 536–550.

[134] Orriols-Puig A, Casillas J, Bernadó-Mansilla E (2007) Fuzzy-UCS: preliminary results. Proceedings of the 2007 GECCO conference companion on Genetic and evolutionary computation : 2871–2874.

[135] Llorà X, Reddy R, Matesic B, Bhargava R (2007) Towards better than human capability in diagnosing prostate cancer using infrared spectroscopic imaging. In: Proceedings of the 9th annual conference on Genetic and evolutionary computation. ACM Press New York, NY, USA, pp. 2098–2105.

[136] Holland J (1976) Adaptation. Progress in theoretical biology 4: 263–293.

[137] Robertson G, Riolo R (1988) A tale of two classifier systems. Machine Learning 3: 139–159.

[138] Smith S (1983) Flexible learning of problem solving heuristics through adaptive search. Proceedings of the Eighth International Joint Conference on Artificial Intelligence : 422–425.

[139] Congdon C (1995) A comparison of genetic algorithms and other machine learning systems of a complex classification task from common disease research. Ph.D. thesis, University of Michigan.

[140] Riolo R (1988) Empirical studies of default hierarchies and sequences of rules in learning classifier systems. Unpublished doctoral dissertation, University of Michigan, Ann Arbor .

[141] Holland J (1986) A mathematical framework for studying learning in classifier systems. Physica D 2: 307–317.

[142] Holland J (1980) Adaptive algorithms for discovering and using general patterns in growing knowledge-bases. International Journal of Policy Analysis and Information Systems 4: 245–268.

[143] Holland J (1980) Adaptive knowledge acquisition. Unpublished research proposal .

[144] Holland J (1981) Genetic algorithms and adaptation(Technical Report No. 34). Ann Arbor: University of Michigan, Department of Computer and Communication Sciences .

[145] Holland J (1983) Induction in artificial intelligence(Technical Report). Ann Arbor: University of Michigan, Department of Computer and Communication Sciences .

[146] Holland J (1983) A more detailed discussion of classifier systems(Technical Report). Ann Arbor: University of Michigan, Department of Computer and Communication Sciences .

[147] Holland J (1984) Genetic algorithms and adaptation. Adaptive Control of Ill-Defined Systems : 317–333.

[148] Holland J (1985) Properties of the Bucket Brigade. Proceedings of the 1st International Conference on Genetic Algorithms table of contents : 1–7.

[149] Holland J (1985) A mathematical framework for studying learning in classifier systems(Research Memo RIS-25r). Cambridge, MA: the Rowland Institute for Science .

[150] Holland J (1987) Genetic algorithms and classifier systems: foundations and future directions. Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application table of contents : 82–89.

[151] Samuel A (1959) Some studies in machine learning using the game of checkers. IBM Journal of Research and Development : 211–232.

[152] Wilson S, Goldberg D (1989) A critical review of classifier systems. Proceedings of the third international conference on Genetic algorithms table of contents : 244–255.

[153] Bonarini A (2000) An Introduction to Learning Fuzzy Classifier Systems. Lecture Notes in Artificial Intelligence, 1813 : 83–104.

[154] Riolo R (1987) Bucket brigade performance: I. Long sequences of classifiers. In: Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application table of contents. Lawrence Erlbaum Associates, Inc. Mahwah, NJ, USA, pp. 184–195.

[155] Riolo R (1987) Bucket brigade performance: II. Default hierarchies. In: Pro-

ceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application table of contents. Lawrence Erlbaum Associates, Inc. Mahwah, NJ, USA, pp. 196–201.

[156] Lanzi P, Riolo R (2003) Recent trends in learning classifier systems research. Natural Computing Series : 955–988.

[157] Barry A (1996) Hierarchy Formation within Classifier Systems A Review. In: Proceedings of the First International Conference on Evolutionary Algorithms and their Application EVCA'96. pp. 195–211.

[158] Lanzi P (2008) Learning classifier systems: then and now. Evolutionary Intelligence 1: 63–82.

[159] Sutton R (1992) Introduction: The challenge of reinforcement learning. Machine Learning 8: 225–227.

[160] Sutton R (1988) Learning to predict by the methods of temporal differences. Machine Learning 3: 9–44.

[161] Watkins C (1989) Learning from Delayed Rewards .

[162] Liepins G, Hilliard M, Palmer M, Rangarajan G (1989) Alternatives for Classifier System Credit Assignment. In: Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI-89). pp. 756–761.

[163] Dorigo M, Bersini H (1994) A comparison of Q-learning and classifier systems. In: Proceedings of the third international conference on Simulation of adaptive behavior: from animals to animats 3: from animals to animats 3 table of contents. MIT Press Cambridge, MA, USA, pp. 248–255.

[164] Dorigo M (1993) Genetic and Non-Genetic Operators in ALECSYS. Evolutionary Computation 1: 151–164.

[165] LANZI P, RIOLO R (2000) A roadmap to the last decade of learning classifier system research(from 1989 to 1999). Lecture notes in computer science : 33–61.

[166] Frey P, Slate D (1991) Letter recognition using Holland-style adaptive classifiers. Machine Learning 6: 161–182.

[167] Kovacs T (2002) A Comparison of Strength and Accuracy-Based Fitness in Learning Classifier Systems. Ph.D. thesis, PhD thesis, University of Birmingham.

[168] Lanzi P (2002) Learning classifier systems from a reinforcement learning perspective. Soft Computing-A Fusion of Foundations, Methodologies and Applications 6: 162–170.

[169] Butz M, Goldberg D, Stolzmann W (2000) Introducing a genetic generalization pressure to the anticipatory classier system: Part 1-theoretical approach. In:

submitted to the Genetic and Evolutionary Computation Conference (GECCO-2000).

[170] Butz M, Goldberg D, Stolzmann W (2000) Investigating Generalization in the Anticipatory Classifier System. LECTURE NOTES IN COMPUTER SCIENCE : 735–744.

[171] Butz M, Goldberg D, Stolzmann W (2000) Probability-Enhanced Predictions in the Anticipatory Classifier System. In: Proceedings of the International Workshop on Learning Classifier Systems (IWLCS-2000), in the Joint Workshops of SAB 2000 and PPSN 2000. Springer.

[172] Butz M (2002) Anticipatory Learning Classifier Systems. Kluwer Academic Publishers.

[173] Bacardit J, Butz M (2004) Data Mining in Learning Classifier Systems: Comparing XCS with GAssist. Seventh International Workshop on Learning Classifier Systems (IWLCS-2004) .

[174] Kovacs T, Lanzi P, of Birmingham U, Centre CSR (1999) A Learning Classifier Systems Bibliography. SCHOOL OF COMPUTER SCIENCE RESEARCH REPORTS-UNIVERSITY OF BIRMINGHAM CSR .

[175] Stalph P, Butz M Documentation of XCSF-Ellipsoids Java plus Visualization .

[176] Butz M Documentation of XCSFJava 1.1 plus visualization. MEDAL Report 2007008.

[177] Butz M, Herbort O (2008) Context-dependent predictions and cognitive arm control with XCSF. In: Proceedings of the 10th annual conference on Genetic and evolutionary computation. ACM New York, NY, USA, pp. 1357–1364.

[178] Butz M (2007) Combining Gradient-Based With Evolutionary Online Learning: An Introduction to Learning Classifier Systems. In: Hybrid Intelligent Systems, 2007. HIS 2007. 7th International Conference on. pp. 12–17.

[179] Russell S, Norvig P, Canny J, Malik J, Edwards D (1995) Artificial intelligence: a modern approach. Prentice Hall Englewood Cliffs, NJ.

[180] Butz M (2006) Rule-based Evolutionary Online Learning Systems: A Principled Approach to Lcs Analysis And Design. Springer.

[181] Koza J (1992) Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press.

[182] Dorigo M, Stützle T (2004) Ant Colony Optimization. Mit Press.

[183] De Castro L, Timmis J (2002) Artificial Immune Systems: A New Computational Intelligence Approach. Springer.

[184] Haykin S (1998) Neural Networks: A Comprehensive Foundation. Prentice Hall PTR Upper Saddle River, NJ, USA.

[185] Bernado E, Llora X, Garrell J (2001) XCS and GALE: a Comparative Study of Two Learning Classifier Systems with Six Other Learning Algorithms on Classification Tasks. In: Proceedings of the 4th International Workshop on Learning Classifier Systems (IWLCS-2001). pp. 337–341.

[186] Kharbat F, Bull L, Odeh M (2007) Mining breast cancer data with XCS. Proceedings of the 9th annual conference on Genetic and evolutionary computation : 2066–2073.

[187] Unold O, Tuszyński K (2008) Mining knowledge from data using Anticipatory Classifier System. Knowledge-Based Systems .

[188] Blake C, Merz C (1998) UCI repository of machine learning databases .

[189] Alayón S, Estévez J, Sigut J, Sánchez J, Toledo P (2006) An evolutionary Michigan recurrent fuzzy system for nuclei classification in cytological images using nuclear chromatin distribution. Journal of Biomedical Informatics 39: 573–588.

[190] Unold O (2007) Grammar-Based Classifier System for Recognition of Promoter Regions. LECTURE NOTES IN COMPUTER SCIENCE 4431: 798.

[191] Kovacs T (1996) Evolving Optimal Populations with XCS Classier Systems. Master's thesis, School of Computer Science, University of Birmingham, Birmingham, UK .

[192] Kovacs T (2001) What should a classifier system learn? In: Evolutionary Computation, 2001. Proceedings of the 2001 Congress on. volume 2.

[193] Kovacs T (2002) What should a classifier system learn and how should we measure it? Soft Computing-A Fusion of Foundations, Methodologies and Applications 6: 171–182.

[194] Butz M, Pelikan M (2001) Analyzing the evolutionary pressures in XCS. In: Proceedings of the Third Genetic and Evolutionary Computation Conference (GECCO-2001). pp. 935–942.

[195] Butz M, Kovacs T, Lanzi P, Wilson S (2004) Toward a theory of generalization and learning in XCS. Evolutionary Computation, IEEE Transactions on 8: 28–46.

[196] Bassett J, De Jong K (2000) Evolving Behaviors for Cooperating Agents. LECTURE NOTES IN COMPUTER SCIENCE : 157–165.

[197] Butz M, Goldberg D (2003) Bounding the Population Size in XCS to Ensure Reproductive Opportunities. LECTURE NOTES IN COMPUTER SCIENCE : 1844–1856.

[198] Butz M, Goldberg D, Lanzi P, Sastry K Bounding the Population Size to Ensure Niche Support in XCS. Urbana 51: 61801.

[199] Butz M, Goldberg D, Lanzi P (2004) Bounding Learning Time in XCS. Urbana 51: 61801.

[200] Butz M (2004) Rule-based evolutionary online learning systems: learning bounds, classification, and prediction .

[201] Bull L (2004) Learning Classifier Systems: A Brief Introduction. Applications Of Learning Classifier Systems .

[202] Booker L (1991) Representing attribute-based concepts in a classifier system. Foundations of Genetic Algorithms : 115–127.

[203] Sen S (1994) A tale of two representations. In: Proceedings of the 7th international conference on Industrial and engineering applications of artificial intelligence and expert systems. Gordon and Breach Science Publishers, Inc. Newark, NJ, USA, pp. 245–254.

[204] Riolo R (1989) The Emergence of Coupled Sequences of Classifiers. In: Proceedings of the 3rd International Conference on Genetic Algorithms table of contents. Morgan Kaufmann Publishers Inc. San Francisco, CA, USA, pp. 256–264.

[205] Schuurmans D, Schaeffer J (1988) Representational Difficulties with Classifier Systems. Dept. of Computing Science, University of Alberta.

[206] Stone C, Bull L (2003) For Real! XCS with Continuous-Valued Inputs. Evolutionary Computation 11: 299–336.

[207] Dam H, Abbass H, Lokan C (2005) Be real! XCS with continuous-valued inputs. In: Proceedings of the 2005 workshops on Genetic and evolutionary computation. ACM New York, NY, USA, pp. 85–87.

[208] Lanzi P, Wilson S (2006) Using convex hulls to represent classifier conditions. In: Proceedings of the 8th annual conference on Genetic and evolutionary computation. ACM New York, NY, USA, pp. 1481–1488.

[209] Butz M (2005) Kernel-based, ellipsoidal conditions in the real-valued XCS classifier system. Proceedings of the 2005 conference on Genetic and evolutionary computation : 1835–1842.

[210] Butz M, Lanzi P, Wilson S (2006) Hyper-ellipsoidal conditions in XCS: rotation, linear approximation, and solution structure. In: Proceedings of the 8th annual conference on Genetic and evolutionary computation. ACM New York, NY, USA, pp. 1457–1464.

[211] Booker L (1985) Improving the Performance of Genetic Algorithms in Classifier Systems. In: Proceedings of the 1st International Conference on Genetic Algorithms table of contents. Lawrence Erlbaum Associates, Inc. Mahwah, NJ, USA, pp. 80–92.

[212] Mellor D (2005) A first order logic classifier system. In: Proceedings of the 2005 conference on Genetic and evolutionary computation. ACM New York, NY, USA, pp. 1819–1826.

[213] Lanzi P (1999) Extending the Representation of Classifier Conditions Part I: From Binary to Messy Coding. Proceedings of the Genetic and Evolutionary Computation Conference 1: 337–344.

[214] Tufts P Dynamic classifiers: genetic programming and classifier systems. In: Proceedings of the Genetic Programming. Papers from the 1995 AAAI Fall Symposium. pp. 114–119.

[215] Ahluwalia M, Bull L, Banzhaf W, Daida J, Eiben A, et al. (1999) A Genetic Programming-based Classifier System. Proceedings of the Genetic and Evolutionary 1: 11–18.

[216] Lanzi P, Perrucci A (1999) Extending the Representation of Classifier Conditions Part II: From Messy Coding to S-Expressions. Proceedings of the Genetic and Evolutionary Computation Conference 1: 345–352.

[217] Lanzi P Mining interesting knowledge from data with the xcs classier system. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001). pp. 958–965.

[218] Lanzi P (2007) An analysis of generalization in XCS with symbolic conditions.

In: Evolutionary Computation, 2007. CEC 2007. IEEE Congress on. pp. 2149–2156.

[219] Bull L, Hurst J (2003) A neural learning classifier system with self-adaptive constructivism. In: Evolutionary Computation, 2003. CEC'03. The 2003 Congress on. volume 2.

[220] O'Hara T, Bull L (2005) A memetic accuracy-based neural learning classifier system. In: Proceedings of the IEEE Congress on Evolutionary Computation. pp. 2040–2045.

[221] Carse B, Fogarty T (1994) A Fuzzy Classifier System Using the Pittsburgh Approach. Parallel Problem Solving from Nature–PPSN III: International Conference on Evolutionary Computation, the Third Conference on Parallel Problem Solving from Nature, Jerusalem, Israel, October 9-14, 1994: Proceedings .

[222] Butz M, Sastry K, Goldberg D (2002) Tournament selection in XCS. In: Proceedings of the Fifth Genetic and Evolutionary Computation Conference (GECCO-2003). volume 1869.

[223] Butz M, Goldberg D, Tharakunnel K (2003) Analysis and Improvement of Fitness Exploitation in XCS: Bounding Models, Tournament Selection, and Bilateral Accuracy. Evolutionary Computation 11: 239–277.

[224] Kharbat F, Bull L, Odeh M (2005) Revisiting genetic selection in the XCS

learning classifier system. In: Evolutionary Computation, 2005. The 2005 IEEE Congress on. volume 3.

[225] Butz M, Sastry K, Goldberg D (2005) Strong, Stable, and Reliable Fitness Pressure in XCS due to Tournament Selection. Genetic Programming and Evolvable Machines 6: 53–77.

[226] Widrow B, Hoff M (1960) Adaptive switching circuits. 1960 IRE WESCON Convention Record. New York: IRE 4: 96–104.

[227] Venturini G (1994) Apprentissage adaptatif et apprentissage supervise par algorithme genetique. Ph.D. thesis, These de Docteur en Science (Informatique), Universite de Paris-Sud.

[228] Butz M, Kovacs T, Lanzi P, Wilson S (2001) How XCS evolves accurate classifiers. In: Proceedings of the Third Genetic and Evolutionary Computation Conference (GECCO-2001). pp. 927–934.

[229] Liepins G, Wang L (1991) Classifier System Learning of Boolean Concepts. In: Proceedings of the Fourth International Conference on Genetic Algorithms. Morgan Kaufmann Publishers, pp. 318–323.

[230] Weiss G (1991) The Action oriented Bucket Brigade. Inst. für Informatik.

[231] Weiss G (1996) An action-oriented perspective of learning in classifier systems. Journal of Experimental & Theoretical Artificial Intelligence 8: 43–62.

[232] Butz M, Goldberg D, Lanzi P (2005) Gradient descent methods in learning classifier systems: improving XCS performance in multistep problems. Evolutionary Computation, IEEE Transactions on 9: 452–473.

[233] Lanzi P, Butz M, Goldberg D (2007) Empirical analysis of generalization and learning in XCS with gradient descent. In: Proceedings of the 9th annual conference on Genetic and evolutionary computation. ACM Press New York, NY, USA, pp. 1814–1821.

[234] Drugowitsch J, Barry A (2005) XCS with eligibility traces. In: Proceedings of the 2005 conference on Genetic and evolutionary computation. ACM New York, NY, USA, pp. 1851–1858.

[235] Lanzi P, Loiacono D, Wilson S, Goldberg D (2006) Prediction update algorithms for XCSF: RLS, Kalman filter, and gain adaptation. In: Proceedings of the 8th annual conference on Genetic and evolutionary computation. ACM New York, NY, USA, pp. 1505–1512.

[236] Horn J, Goldberg D, Deb K (1994) Implicit Niching in a Learning Classifier System: Nature's Way. Evolutionary Computation 2: 37–66.

[237] Horn J, Goldberg D, Koza J, Goldberg D, Fogel D, et al. (1996) Natural Niching for Cooperative Learning in Classifier Systems. In: Genetic Programming 1996: Proceedings of the First Annual Conference. MIT Press, pp. 553–564.

[238] Butz M, Pelikan M, Llorà X, Goldberg D (2005) Extracted global structure makes local building block processing effective in XCS. In: Proceedings of the 2005 conference on Genetic and evolutionary computation. ACM New York, NY, USA, pp. 655–662.

[239] Butz M, Pelikan M, Llora X, Goldberg D (2006) Automated Global Structure Extraction for Effective Local Building Block Processing in XCS. Evolutionary Computation 14: 345–380.

[240] Bacardit J, Krasnogor N (2006) Smart crossover operator with multiple parents for a Pittsburgh learning classifier system. In: Proceedings of the 8th annual conference on Genetic and evolutionary computation. ACM New York, NY, USA, pp. 1441–1448.

[241] Serendynski F, Cichosz P, Klebus G (1995) Learning classifier systems in multi-agent environments. In: Genetic Algorithms in Engineering Systems: Innovations and Applications, 1995. GALESIA. First International Conference on (Conf. Publ. No. 414). pp. 287–292.

[242] Sen S, Sekaran M (1996) Multiagent Coordination with Learning Classifier Systems. LECTURE NOTES IN COMPUTER SCIENCE : 218–233.

[243] Bull L, Studley M, Bagnall T, Whittley I (2005) On the use of Rule-Sharing in Learning Classifier System Ensembles. In: Evolutionary Computation, 2005. The 2005 IEEE Congress on. volume 1.

[244] Bull L, Studley M, Bagnall A, Whittley I (2007) Learning Classifier System Ensembles With Rule-Sharing. Evolutionary Computation, IEEE Transactions on 11: 496–502.

[245] Bacardit J, Krasnogor N (2008) Empirical evaluation of ensemble techniques for a pittsburgh learning classifier system. In: Proceedings of the 9th International Workshop on Learning Classifier Systems.(to appear), LNAI, Springer-Verlag.

[246] Dam H, Rojanavasu P, Abbass H, Lokan C (2008) Distributed Learning Classifier Systems .

[247] Lokan C (2008) Distributed Learning Classifier Systems. Learning Classifier Systems in Data Mining .

[248] Ranawana R, Palade V (2006) Multi-Classifier Systems: Review and a roadmap for developers. International Journal of Hybrid Intelligent Systems 3: 35–61.

[249] Bernado-Mansilla E, Llora X, Traus I Multiobjective Learning Classifier Systems: An Overview. Technical report, Technical report, University of Illinois at Urbana Champaign, Urbana, IL, 2005.

[250] Fu C, Davis L (2002) A Modified Classifier System Compaction Algorithm. In: Genetic and evolutionary computation conference (GECCO-2002). pp. 920–925.

[251] Butz M, Lanzi P, Wilson S (2008) Function Approximation With XCS: Hyperellipsoidal Conditions, Recursive Least Squares, and Compaction. Evolutionary Computation, IEEE Transactions on 12: 355–376.

[252] Holmes J, Sager J, Bilker W (2004) A Comparison of Three Methods for Covering Missing Data in XCS. IWLCS-2004 (GECCO 2004), Seattle, Washington, June .

[253] Orriols-Puig A, Bernadó-Mansilla E (2006) Bounding XCS's parameters for unbalanced datasets. In: Proceedings of the 8th annual conference on Genetic and evolutionary computation. ACM New York, NY, USA, pp. 1561–1568.

[254] Dam H, Shafi K, Abbass H (2005) Can Evolutionary Computation Handle Large Dataset? Technical report, Technical Report TR-ALAR-2005070001, http://seal. tst. adfa. edu. au/alar/techrep. html, 2005.

[255] Booker L (2000) Classier systems, endogenous fitness, and delayed rewards: A preliminary investigation. In: Proceedings of the International Workshop on Learning Classier Systems (IWLCS-2000), in the Joint Workshops of SAB.

[256] Hurst J, Bull L (2000) A Self-Adaptive Classifier System. In: Revised Papers from the Third International Workshop on Advances in Learning Classifier Systems. Springer, pp. 70–79.

[257] Bull L, Hurst J (2000) Self-Adaptive Mutation in ZCS Controllers. LECTURE NOTES IN COMPUTER SCIENCE : 339–346.

[258] Bull L, Hurst J, Tomlinson A (2000) Self-adaptive mutation in classifier system controllers. In: From Animals to Animats 6: Proceedings of the Sixth International Conference on Simulation of Adaptive Behavior. MIT Press.

[259] Hurst J, Bull L (2002) A Self-Adaptive XCS. LECTURE NOTES IN COMPUTER SCIENCE : 57–73.

[260] Browne W (2008) Improving Evolutionary Computation Based Data-Mining for the Process Industry: The Importance of Abstraction. Learning Classifier Systems in Data Mining .

[261] Goldberg D, Horn J, Deb K What Makes a Problem Hard for a Classifier System? Technical report, Santa Fe Working Paper, 1992.

[262] Booker D LB and, Holland J (1989) Classifier systems and genetic algorithms. Machine learning: paradigms and methods table of contents : 235–282.

[263] Holland J, Booker L, Colombetti M, Dorigo M, Goldberg D, et al. (2000) What Is a Learning Classifier System? LECTURE NOTES IN COMPUTER SCIENCE : 3–32.

[264] Wilson S (2000) State of XCS Classifier System Research. LECTURE NOTES IN COMPUTER SCIENCE : 63–82.

[265] Kovacs T (2002) Learning classifier systems resources. Soft Computing-A Fusion of Foundations, Methodologies and Applications 6: 240–243.

[266] Drugowitsch J (2008) Design and analysis of learning classifier systems: A probabilistic approach. Springer.

Chapter 2

The Application of Michigan-Style Learning Classifier Systems to Address Genetic Heterogeneity and Epistasis in Association Studies

Abstract

Genetic epidemiologists, tasked with the disentanglement of genotype-to-phenotype mappings, continue to struggle with a variety of phenomena that obscure the underlying etiologies of common complex diseases. For genetic association studies, genetic heterogeneity (GH) and epistasis (gene-gene interactions) epitomize well recognized

phenomenon that represent a difficult, but accessible challenge for computational biologists. While progress has been made addressing epistasis, methods for dealing with GH tend to "side-step" the problem, limited by a dependence on potentially arbitrary cutoffs/covariates, and a loss in power synonymous with data stratification. In the present study, we explore an alternative strategy (Learning Classifier Systems (LCSs)) as a direct approach for the characterization, and modeling of disease in the presence of both GH and epistasis. This evaluation involves (1) implementing standardized versions of existing Michigan-Style LCSs (XCS, MCS, and UCS), (2) examining major run parameters, and (3) performing quantitative and qualitative evaluations across a spectrum of simulated datasets. The results of this study highlight the strengths and weaknesses of the Michigan LCS architectures examined, providing proof of principle for the application of LCSs to the GH/epistasis problem, and laying the foundation for the development of an LCS algorithm specifically designed to address GH.

2.1 Introduction

Currently, bioinformatics progress is being driven by two trends (1) advancing biotechnology, (i.e. cheaper, more reliable tools) that supplies an overwhelming abundance of new data, and (2) an ever-increasing appreciation for the complexities inherent to biological systems at both the individual and population levels [1, 2]. As geneticists strive to identify "markers for" and "etiologies of" common complex disease, it has become increasingly clear that the statistical and analytical techniques,

well suited for Mendelian studies, are insufficient to address the demands of common complex disease [2]. Phenomena that have been recognized to complicate the epidemiological mapping of genotype to phenotype include epistasis, gene-environment interaction, phenocopy, epigenetics, phenotypic heterogeneity, trait heterogeneity, and genetic heterogeneity [3]. Epistasis, or gene-gene interaction, is a particularly challenging problem given that a gene's association with disease may only be seen in the context of at least one other gene. Genetic heterogeneity (GH), refers to the presence of different underlying genetic mechanisms resulting in the appearance of the same or similar disease phenotype [4]. While the ability to address some of these phenomena will likely depend on continued technological advancement (epigenetics) [5], improved environmental data collection (gene-environment interaction, and phenocopy) [6], or the refinement of phenotypic observation and classification (phenotypic and trait heterogeneity) (e.g. [7]), the quality and availability of genetic code information make epistasis and GH obvious targets for bioinformatic development.

This study concurrently examines epistasis and GH, modeled as they might simultaneously occur in a single nucleotide polymorphism (SNP) genetic association study. While the detection and modeling of epistasis has received a great deal of attention [8–10], methods for dealing with GH are lagging behind, clinging to a traditional epidemiological paradigm that seeks to find a single best model of disease within a given data set [3]. Evidence of GH appears in countless genetic studies via the identification of multiple candidate genes/alleles, or the failure to replicate

candidate gene/allele findings within either linkage or association studies [3, 11, 12]. Two forms of GH have been described; allelic and locus heterogeneity. Traditional genetic terminology defines an allele as being an alternate form of a gene, while a locus refers to the specific location of a gene. Allelic heterogeneity occurs when two or more alleles of a single locus are independently associated with the same trait, while locus heterogeneity occurs when two or more DNA sequence variations at distinct loci are independently associated with the same trait. In the context of SNP association studies, where individual SNPs (sometimes within the same gene) are referred to as loci, the practical difference between allelic and locus heterogeneity is lost, and is therefore ignored in the context of this study. From a computer science prospective, the problem of genetic heterogeneity is similar to a latent or "hidden" class problem. While the disease status (case or control) of each patient is already known, the individuals making up either class would be more accurately subtyped into two or more "hidden" classes, each characterized by an independent predictive model of disease. The impact of GH on the detection and modeling of epistasis has been explored using Multifactor Dimensionality Reduction (MDR) [13]. This work has demonstrated that GH dramatically hinders MDR's power to detect/model all underlying attributes involved in the underlying epistatic GH, but additional examination indicated that MDR retains significant power to identify either the dominant branch of GH or at least one of the underlying attributes [14, 15]. Existing methods for explicitly dealing with GH include sample stratification, the M test, the β test, the admixture test,

ordered subset analysis, cluster analysis, latent class analysis, and factor analysis, all reviewed in [3]. With the exception of the admixture test (that tests a more general hypothesis of GH) all of the methods rely on covariate data (i.e. genetic risk factors, demographic data, phenotypic data, or endophenotypes) in order to identify more homogeneous subsets of patients. This is in line with the standard epidemiological paradigm that seeks a single best disease model within a given homogeneous sample. The obvious drawback of these methods is that they completely rely on the availability, quality, and relevance of these covariates. Additionally, stratification represents a reduction in sample size, leading to an inevitable loss in power. In order to address these concerns and embrace the problem of GH directly, we propose the application and exploration of learning classifier systems (LCSs). This class of algorithm breaks from the traditional single model paradigm by evolving a solution comprised of multiple rules, consequently avoiding a need for covariates and data stratification. For these reasons, we hypothesize that LCS algorithms will be useful for the detection, characterization, and modeling of GH.

The goal of this study is to provide proof of principle for the application of LCSs to the GH problem and to lay the foundation for the development of an LCS algorithm specifically designed to address this problem domain. The following study involves (1) implementing standardized versions of existing Michigan-Style LCSs (XCS, MCS, and UCS), (2) simulating a spectrum of datasets of varying size and challenge that model GH and epistasis concurrently, (3) performing a sweep of the major run parameters for

each LCS implemented, and (4) performing a quantitative and qualitative evaluation of each LCS across the entire spectrum of simulated GH/epistatic data sets.

2.2 Methods

2.2.1 Learning Classifier Systems

Learning classifier systems (LCS) combine machine learning with evolutionary computing and other heuristics to produce an adaptive system that learns to solve a particular problem. LCSs are closely related to and typically assimilate the same components as the more widely utilized genetic algorithm (GA). The goal of LCS is not to identify a single best model or solution, but to create a cooperative set of rules or models that together solve the task. The solution evolved by an LCS is represented as a population of rules/models that are utilized collectively to make decisions/classifications. Therefore, it would seem that the unique ability of an LCS to evolve different solution 'niches' to represent different subspaces of a given problem would be well suited to address GH. Since their advent, the LCS concept has inspired a multitude of implementations adapted to manage the different problem domains to which it has been applied (e.g. autonomous robotics, classification, knowledge discovery, and complex systems modeling). The potential for LCSs to be applied to biological problems (e.g. epidemiological, medical, genetic) has been demonstrated for complex disease classification tasks [16–23], epidemiological surveillance/knowledge

discovery [24–26], and protein structure prediction [27, 28], to name a few. This study will be the first to utilize LCSs to explicitly address the problems of GH and epistasis in complex disease classification and modeling.

Within the LCS community of algorithms there is an impressive diversity of architectures and heuristic modifications that have been implemented with the intention of improving prediction accuracy, run time, solution compactness and solution comprehensibility to address a given problem domain [29]. The infancy of LCS research saw the emergence of two founding classes of LCSs, referred to as the Michigan and Pittsburgh styles. The Michigan-style is characterized by a population of rules with a GA operating at the level of individual rules and the evolved solution is represented by the entire rule population. Alternatively, the Pittsburgh-style is characterized by a population of variable length rule-sets (where each rule-set is a potential solution) and the GA typically operates at the level of a single rule-set. Because of the major differences in both algorithm architecture and population/solution representation, the application of these two styles to the GH problem will be explored separately. The present study focuses on Michigan-style systems, while a separate study of Pittsburgh-style systems will follow.

Michigan-style LCSs, often varying widely from version to version, generally possess four basic components; (1) a population of rules or classifiers, (2) a performance component that assesses how well the population of rules collectively explain the data, (3) a reinforcement component that distributes the rewards for correct prediction to

each of the rules in the population, and (4) a discovery component that uses different operators to discover new rules and improve existing ones. Learning progresses iteratively, relying on the performance and reinforcement components to drive the discovery of better rules. For a complete LCS introduction and review, see [29].

2.2.1.1 Implementing LCS

The three implemented Michigan-style LCSs were selected based on their prominence, relevance, simplicity, and availability. These included XCS [30], UCS [31], and MCS [32]. Each of the LCSs examined in this study were encoded in Python. This was done to (1) standardize the coding language, (2) put the different LCS algorithms in a flexible, readable language to facilitate and promote future algorithm development for biologists and other non-computer scientists, (3) allow command line control over all run parameters, (4) gain a detailed understanding of each system, and most importantly (5) to standardize the rule representation. Additionally, wrapper scripts were written to run and evaluate each LCS using Dartmouth's 888 processor Linux Cluster, "Discovery". These implementations are available on request (*ryanurbanowicz@gmail.com*) and will be posted on the LCS and GBML Central webpage.

Rule representation (a.k.a. knowledge representation) constitutes one of the most problem domain dependent components of an LCS. The manner in which rules are encoded must be amenable to the type of attribute values making up the problem

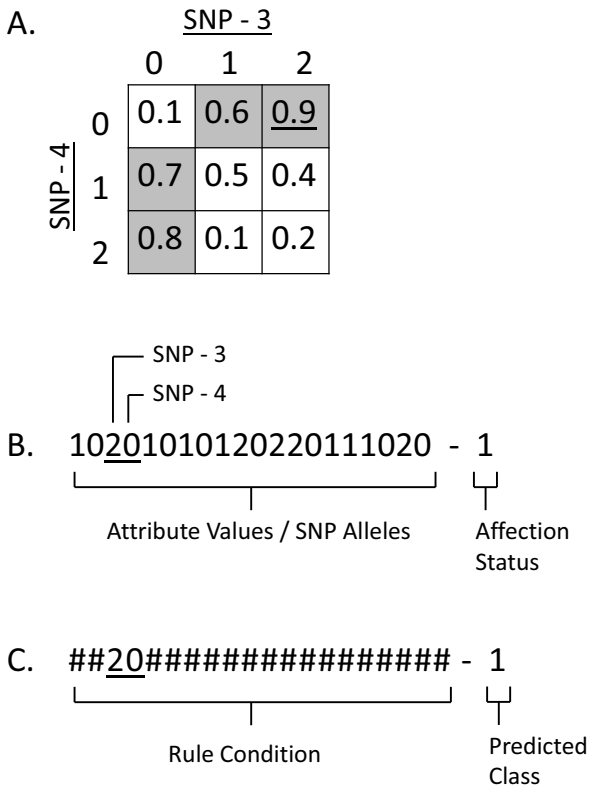

Figure 2.1: The Quaternary Rule Representation: (A.) An example penetrance table modeling an epistatic interaction between SNPs 3 and 4 (simulated data are generated using tables like this). Shaded cells indicate genotypes with high penetrance (probability of disease). (B.) An example input string from a data set generated from A, where the three SNP attribute values (encoded as 0, 1, and 2) represent three potential allele states (e.g. *AA*, *Aa*, and *aa* respectively). (C.) An example rule utilizing the quaternary representation where the condition of the rule is represented by a string of characters from the quaternary alphabet $(0, 1, 2, \#)$, where '#' acts as a wildcard such that a rule condition 10#0 would match the attribute inputs 1000, 1010, and 1020. The predicted class for a rule is represented as a binary value, where only two classes are possible (0 = control and 1 = case). Notice how the rule illustrated here has evolved to identify the "high-risk" genotype (SNP3 = 2 and SNP4 = 0) underlined in A. Ideally, an LCS would evolve rules that specify genotype combinations that best discriminate between cases and controls.

Table 2.1: LCS Algorithm Summaries

System	Year	Citation	Style	Fitness	Learning/Credit Assignment	GA
XCS	1995	[30]	Michigan	Accuracy	Q-Learning-Like Reinforcement Learning	Action Set
UCS	2003	[31]	Michigan	Accuracy	Supervised Learning	Correct Set
MCS	2004	[32]	Michigan	Strength	Q-Learning Like Widrow-Hoff	Population Set

domain (e.g. symbolic, discrete, or real-valued). For this study, rules were encoded using a quaternary rule representation, well suited for discrete, nominal, SNP attributes and a discrete affection status (case/control). Figure 2.1 illustrates this quaternary representation as well as how it may be utilized to evolve concise, interpretable rules. LCSs are relatively complex algorithms with many interacting components. In standardizing the rule representation, other algorithmic components that had relied on the original representation were adjusted accordingly (e.g. crossover and mutation mechanisms).

The major distinguishing features of the LCSs examined here include fitness basis, learning mechanism, where the GA operates, and the incorporation of accessory heuristics. Table 2.1 summarizes the features that are often used to broadly categorize different systems. For a more detailed review of features common to different LCS systems see [29]. Section 2.2.1.2 gives an overview of each system and makes note of any modifications made with respect to the original published algorithm.

2.2.1.2 Three Michigan-Style LCSs

XCS, or the eXtended Classifier System [30], and its immediate predecessor, ZCS [33], represent an architectural revolution aimed at increasing the understandability and performance of LCSs. To date, XCS is still the most popular and best understood of the many existing systems, having directly inspired the majority of modern implementations (including UCS and MCS). XCS is a re-engineering of Holland's original LCS model [34] designed to evolve a complete action map (an all-inclusive and accurate representation of the problem space) that may evolve both maximally correct (e.g. using the first four attributes from the rule in Figure 2.1C. ##20 - 1), and maximally incorrect rules (e.g.##20 - 0). This characteristic allows XCS the flexibility to be widely applicable to many problem domains, including data mining and classification [20, 35]. The implementation of XCS used in this study is a modified version of an existing open source Python script available at http://www.dia.fi.upm.es/∼jamartin/download.htm.

UCS, or the sUpervised Classifier System [31], preserves much of the XCS architecture but replaces reinforcement learning with supervised learning, encouraging the formation of best action maps (rule sets of efficient generalizations) and alters the way in which accuracy (and thus fitness) is computed. UCS was designed specifically to address single-step problem domains such as classification and data mining, where delayed reward is not a concern. The implementation of UCS used in this study is directly based on the aforementioned XCS python script, but modified to meet the

UCS algorithmic specifications described in [31].

MCS, or the Minimal Classifier System [32], is a maximally simple, strength-based, bare bones version of XCS. Having been developed strictly to examine a theoretical line of LCS study, MCS is unlikely to be directly applicable to real world problems. Its inclusion in this study is viewed as a negative control to which the other systems may be compared. This implementation of MCS is directly based on the aforementioned XCS python script, but modified to meet the MCS algorithmic specifications described in [32], with two exceptions; (1) the mutation operator described in [32] was replaced with the simpler one utilized by XCS, and (2) subsumption, a generalizing mechanism found in XCS, was added.

Each Michigan-style LCS was additionally modified by replacing respective ternary rule representations with the quaternary ones described above, and by adding a user controlled option for the GA to use tournament selection over roulette wheel selection [36, 37]. As suggested in [36], tournament selection was implemented with tournament sizes proportionate to the respective set size. The proportion ($\tau = 0.5$) was selected for this study based on preliminary testing.

2.2.2 Data Simulation

This evaluation of LCS algorithms calls for simulated data sets that concurrently model GH and epistasis as they might appear in a SNP gene association study of common complex disease. All data sets were generated using a pair of distinct, two-

locus epistatic interaction models, both utilized to generate instances (i.e. case and control individuals) within a respective subset of each final data set. Each two-locus epistatic model was simulated without Mendelian/main effects, as a penetrance table. In total, each simulated data set contains 4 predictive attributes, where only two attributes are in fact predictive within a respective subset of the data. Additionally, each data set contains 16 randomly generated, non-predictive attributes, with minor allele frequencies randomly selected from a uniform distribution ranging from 0.05 to 0.5. Generation of the data sets was achieved in three steps; (1) Two-locus epistatic penetrance tables were generated for a specified heritability and minor allele frequency, (2) balanced data sets were generated using different random seeds and these two-locus penetrance tables, (3) pairs of different data sets were randomly sampled and merged to form new, balanced data sets, now with underlying GH. Due to the computational demands of LCSs, this study limited its evaluation to 4 GH/epistasis model combinations that were selected to represent a diversity of potential etiological disease configurations. For simplicity the minor allele frequency of each predictive attribute was set to 0.2, a reasonable assumption for a common complex disease SNP. The four model combinations included the following pairs of heritabilities: (1) Combo A = 0.1 & 0.1, (2) Combo B = 0.1 & 0.4, (3) Combo C = 0.2 & 0.2, and (4) Combo D = 0.4 & 0.4. For each model combination, data sets were generated as having four different sample sizes $(200, 400, 800, 1600)$ and two GH ratios (50:50, 75:25), where 50:50 implies that each of the two underlying epistatic models are equally represented

in the final simulated data set. Additionally, we introduce a third dimension to the evaluation of each model combination that explores the computational difficulty to detect relevant interacting attributes/SNPs. Specifically, for each model combination, three pairs of epistatic penetrance tables were chosen, labeled as "E" = Easy, "M" = Moderate, and "H" = Hard. This characterization of difficulty is dependent on a penetrance table based value that summarizes the ability of SURF [38] (an attribute selection algorithm based on ReliefF [39]) to identify attribute dependencies. This was done to present the LCS algorithms with a challenging diversity of potential underlying interactions. Together, a total of 96 data set configurations (4 *Model Combos* x 4 *Sample Sizes* x 2 *Ratios* x 3 *Difficulties*), and a total of 1440 data sets (15 random seeds each) were simulated.

2.2.3 System Evaluations

It is well known that the parameter settings of an LCS (or any other evolutionary algorithm for that matter) can have a significant impact on the algorithm's ability to perform. Therefore, before evaluating each LCS across the spectrum of simulated data sets described above, a parameter sweep was conducted to roughly optimize the parameter settings for each respective algorithm. This parameter sweep was run using what was considered to be three of the easiest configurations of data sets (the three penetrance table difficulties for Combo D, with a sample size of 1600, and a ratio of 50:50) with statistical comparisons being made across all 45 data sets of

this group (3 *Difficulties* x 15 *Random Seeds*). While it is possible that selecting parameters based only on the "easiest" data configuration may bias the results towards performing better on these "easier" data sets, this was a necessary tradeoff to meet computational limitations. Additionally, even those configurations that we have deemed "easy" constitute a very challenging computational task with the same number of underlying predictive attributes as those configurations we have deemed "hard". The parameters examined for each Michigan-style LCS include population size $(800, 1600, 2400, 3200, 6400)$, selection scheme (*Tournament* or *Roulette Wheel*), subsumption (*on* or *off*), and the learning reward $(100, 500, 1000, 2000)$. LCS performance was tracked according to the following metrics that have been ordered here according to their importance to this study; (1) estimate of power to correctly detect predictive attributes, (2) classification accuracy on testing data, (3) computational time, and (4) solution generality. To determine which parameters were significant predictors of LCS performance, we employed students t-tests for selection and subsumption, and ANOVA followed by Tukey's HSD posthoc analysis for population size and reward. Once parameters were selected for each system, a complete evaluation over all simulated data sets was performed. Any algorithm parameters not evaluated in the parameter sweep were left at their respective default settings.

In both the parameter sweep and complete evaluation, a 10-fold cross validation (CV) strategy was employed to track average testing accuracy and account for overfitting. Every one of the 1440 data sets were randomly partitioned into 10 equal

parts and each algorithm is run 10 separate times during which 9/10 of the data is used to train the algorithm, and a different 1/10 is set aside for testing. In total, each LCS was run 14,400 times, learning over the course of 1,000,000 iterations, with intermediate evaluations completed after 100,000 and 500,000 iterations.

Power, or success rate, is typically estimated in these types studies by tracking the frequency with which an algorithm successfully identifies the correct underlying model or attribute(s), across some number of data set replicates. This type of estimation is not applicable for LCSs that evolve a solution made up of an entire population of rules. While the function of an LCS is to evolve a solution that can accurately classify patients, it is more important, and arguably much more difficult to evolve a solution that is meaningful and interpretable for a genetic epidemiologist. With this in mind, we hypothesize that if an LCS is truly learning the underlying model(s), attributes that are important to the underlying model(s)(predictive attributes) will tend to be specified (assigned a 0, 1, or 2) more frequently within rules of the population. Conversely, attributes that are not involved in the underlying model(s) (noise attributes) will tend to be generalized ('#'/don't-care symbol used) more frequently within rules of the population. Therefore, power is estimated by counting the number of times an attribute is specified across all rules comprising the population, as opposed to how often a '#' is used. Specifically, success is achieved if the total number of rules specifying '#' for each of the four predictive attributes is lower than for any other noise attribute. Since each data set is split and run 10 times each (as part of

CV), we further characterize success at the data set level as having found the correct 4 predictive attributes in the majority (> 0.5) of the 10 CV runs.

Statistical evaluations were completed using R. Logistic regression, modeling each data set dimension and all pair-wise combinations of dimensions, was employed for the evaluation of power. Either ANOVA, paired with Tukey's HSD posthoc analysis, or the non-parametric Kruskal-Wallis test paired with Mann-Whitney plus a Bonferroni correction, was employed for the evaluation of testing accuracy, training accuracy, solution generality, and run time. Normality was tested graphically with normal quantile-quantile plots.

2.3 Results

2.3.1 Parameter Sweep

The results of the parameter sweep led to the selection of the run parameters shown in (Table 2.2). While the power and testing accuracy of each system tended to improve significantly ($p \ll 0.001$) as the population size was increased above 800, a population size of 1600 was selected for used in each algorithm because (1) it yielded the most significant step-wise improvement ($p < 0.001$) in both XCS and UCS, (2) it offers a reasonable tradeoff between performance and computational time (where a doubling of the population size leads to a significant ($p \ll 0.001$), approximately doubled run time), and (3) the selection of an identical population size for each

Table 2.2: Parameter Settings

System	**Pop. Size**	**Sel.**	**Sub.**	**Reward**
XCS	1600	Roulette	On	100
UCS	1600	Roulette	On	NA
MCS	1600	Tournament	On	1000

algorithm allows for a more logical comparison. In addition, while previous work has demonstrated the value of utilizing proportionate tournament selection in XCS [36, 37], this sweep identified that MCS (testing accuracy only) alone benefited from this type of selection ($p \ll 0.001$) having no significant impact on power for any of the three systems. Subsumption was generally found to benefit each system, UCS and XCS saw a significant improvement in both power and generality, and a significant reduction in run-time (each $p \ll 0.001$), while MCS saw a significant improvement in testing accuracy ($p < 0.001$). Lastly, the reward value, not used by UCS, had little impact on MCS, so the default setting of 1000 was utilized, while XCS showed a significant improvement in both power and testing accuracy at a value of 100 ($p < 0.05$).

2.3.2 Michigan LCS Evaluation

Table 2.3 summarizes statistics describing the performance of the respective Michigan LCSs. It is important to note that averages are calculated over every simulated

Table 2.3: System Summary Statistics

System	Best Power	Average Power	Best Testing Accuracy	Average Testing Accuracy	Average Training Accuracy	Average Generality	Average Run Time (Min.)
XCS	1.0	0.1979	0.7232	0.5942	0.8782	0.7615	168.82
UCS	1.0	0.1819	0.7132	0.6020	0.9197	0.5910	211.14
MCS	0.0	0.0	0.51	0.4915	0.7320	0.4788	272.36

data set described. Lower overall averages of power and testing accuracy are therefore expected. "Best" power and testing accuracy refers to the highest values found for a given LCS within a particular data set configuration. "Average" values are meant to compare between LCSs, while "best" values indicate the success of a given system. The most apparent result of this study is the failure of MCS to achieve competitive power or testing accuracy for any of the simulated data sets over the course of one million learning iterations. Specifically the MCS yielded zero power to detect the correct predictive attributes, a best average testing accuracy of only 0.51 (a small but significant increase from 0.5/random ($p << 0.001$)), the lowest average solution generality, and the longest run time.

Findings for both XCS and UCS indicate definite promise. Figure 2.2 summarizes evaluations of power and testing accuracy across the four dimensions of simulated data. Logistic regression models indicate that each of the selected data set dimensions are significant predictors of power (i.e. model combo, sample size, mix ratio, and difficulty), as is the choice of LCS algorithm ($p << 0.001$). Additionally, regression model interactions, pairing algorithm choice with model combo, sample size and mix ratio respectively, were all significant ($p << 0.001$) indicating that the choice of

LCS impacts the power to detect attributes given varying values of any of these three dimensions. Conversely the interaction between algorithm choice and difficulty was not significant, which suggests that while the difficulty dimension significantly impacted both power and testing accuracy ($p << 0.001$) where data sets designated as "Easy" were found to have significantly higher power and testing accuracies than those designated as "Medium" ($p << 0.001$), which had significantly higher power and testing accuracies than those designated as "Hard" ($p << 0.001$), the algorithm choice has no impact power given a change in difficulty. While XCS and UCS both achieve significantly higher power than MCS ($p << 0.001$), there is no overall significant power difference between the two.

Evaluation of the LCS algorithm's testing accuracies indicate that UCS performs the best ($p << 0.001$) with XCS as a close second. Additionally, all data set dimensions were significant predictors of testing accuracy (each $p < 0.001$), while the only dimension that significantly impacted training accuracy was sample size. Of note, data sets with a mix of 75:25 yielded significantly higher testing accuracies than 50:50 ($p < 0.05$). This would suggest that the LCSs are more proficient at learning a model that dominates the sample population. Because our power estimates are dependent on finding "both" underlying models, it is not surprising that while average testing accuracy is higher for the 75:25 ratio, power is much lower. It should be noted that the underlying models of the simulated data are specifically designed to explain a limited portion of the individuals' disease states (disease penetrance < 1.0),

meaning that the maximum achievable testing accuracies are expected to be much lower than 1.0. Continued evaluation demonstrated that XCS evolves rule populations with a much greater generality ($p << 0.001$), and does so in significantly less time ($p << 0.001$) than either UCS or MCS. While every evaluation described above was performed after one million learning iterations, performance after only 100,000 iterations was also tracked. The effect on power and testing accuracy, made by allowing each algorithm to run 10 times longer, is summarized by the following; (1) MCS showed no improvement, (2) the power of XCS did not significantly change, while it's testing accuracy declined a small but significant amount ($p << 0.001$), (3) neither the power nor testing accuracy of UCS significantly changed. Overall, the additional run time did not seem advantageous for any of the algorithms in this study.

2.4 Discussion and Conclusions

The problems presented to Michigan LCSs in this study are far from trivial, and the bar for success was set purposefully high. It is important to consider that all underlying models are purely epistatic, which means that having only one of two interacting attributes in a model will have no impact on predictive ability. In the context of biology, the occurrence of a purely epistatic interaction is certainly a possibility. This type of interaction poses a particular dilemma for evolutionary algorithms that intuitively rely on the "breadcrumbs" of main effects to drive the evolution of an association model. Given this, one would expect that the ability of an evolutionary

algorithm (such as LCS) to detect and model a purely epistatic interaction (like those modeled in this study) would be no better than that for a random search, which could not be guaranteed to find the optimal solution. The only way to guarantee the identification of a purely epistatic interaction would likely be with an exhaustive search of all attribute interactions, which quickly becomes impractical/impossible as the number of potentially involved attributes and examined n-way interactions increases. The LCS implementations examined in this study evolve rules that specify a particular genotype combination. This may prove to be advantageous, even in the context of pure epistasis, in that if the LCS stumbles across one of the predictive genotypes involved in the interaction, this might offer a surrogate "breadcrumb" leading to the discovery of alternative rules/genotypes encompassed by that interaction. While this study models the most challenging scenario of purely epistatic interactions alongside of GH, we would expect that any main effects influencing real-life complex disease would greatly improve the success of an evolutionary algorithm such as LCS. Additionally, because this study emphasized the importance of finding both underlying models, the power to find at least one underlying model was not explored, although we would expect such power to be significantly higher (as was found in [15]), especially for simulations with a GH mix ratio of 75:25.

The results of the parameter sweep serve not only to roughly optimize each LCS for a more thorough comparison, but demonstrate the importance of subsumption and selection strategy for this problem domain. The evaluation of each system over

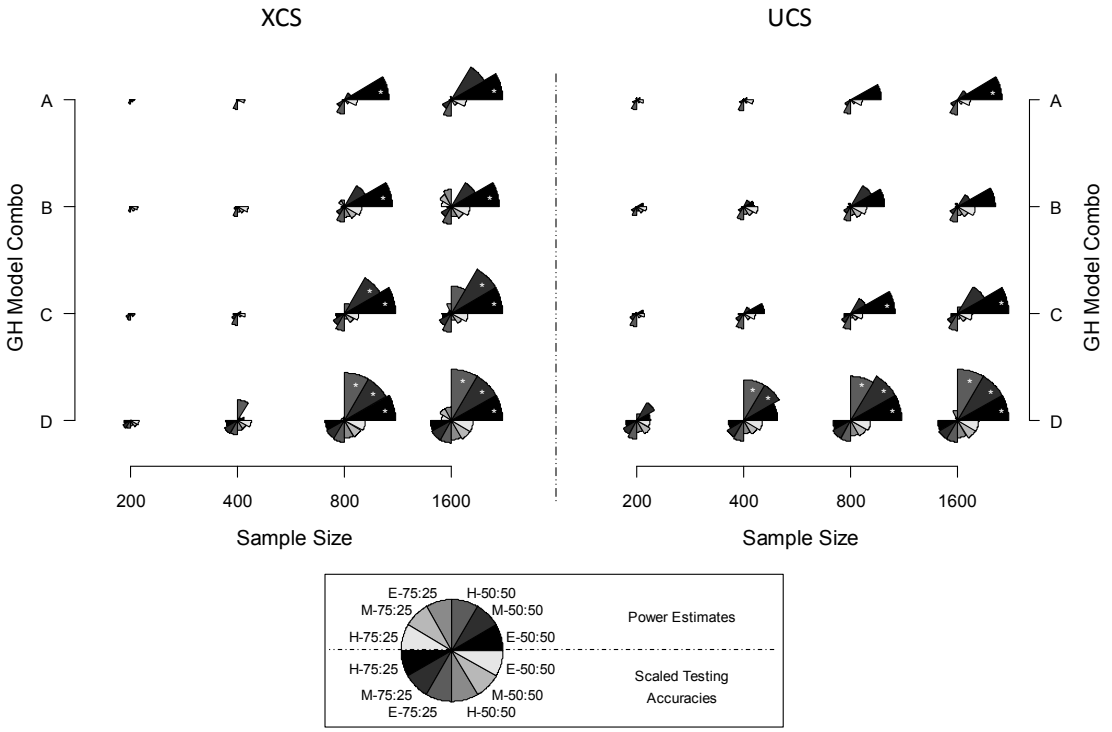

Figure 2.2: Two segment diagrams are drawn to summarize power estimates and testing accuracies for XCS (left) and UCS (right) over each dimension of the simulated data (i.e. model combination, sample size, mix ratio, and penetrance table difficulty − see section 2.2.2). The top half of each circle consists of power estimates. Any data set configuration with a power > 0.8 is denoted with a white star. The bottom half of each circle consists of "scaled" testing accuracies ((average testing accuracy $-0.5)/0.5$) that illustrates any improvement in average testing accuracy above 0.5 (random). Each circle segment represents an evaluation over 15 random seed data sets. Model combos are as follows; A = 0.1 & 0.1, B = 0.1 & 0.4, C = 0.2 & 0.2, and D = 0.4 & 0.4.

the complete spectrum of simulated data suggests the following; (1) the selected rule representation allowed each system to evolve interpretable rules that model specific genotype combinations, (2) both XCS and UCS were able to achieve significant power (> 0.8) for a number of challenging data configurations, as well as average testing accuracies significantly higher than 0.5 ($p << 0.001$), (3) both XCS and UCS struggle to make good use of additional learning iterations in this problem domain and (4) while XCS and UCS show promise in detecting all underlying attributes, they currently offer no obvious ability to characterize and distinguish the underlying GH (i.e. identify which pair of attributes model disease in a particular subset of the data).

The results of this study support the employment of Michigan LCSs to address the detection, modeling and characterization of attributes associated with common complex disease given the complicating presence of underlying GH and epistasis. However, these findings do not necessarily indicate that a given LCS will perform "better" within any other problem domain. We intend to follow this study with a similar evaluation of Pittsburgh-style LCSs applied to this same GH/epistasis problem. The collective findings of these two studies will be used to direct the development of a problem-specific LCS algorithm, aimed at (1) improving the power to detect underlying attributes, (2) being able to distinguish distinct underlying models constituting the GH, and (3) enhancing the overall interpretability of the evolved LCS rule-population solution.

Acknowledgments

This work was supported by the William H. Neukom 1964 Institute for Computational Science at Dartmouth College along with NIH grants AI59694, LM009012, and LM010098. We would like to thank Jose Martin for making an open source Python implementation of XCS freely available, and Jeff Kiralis for his efforts in developing the difficulty metric utilized by this study.

2.5 Bibliography

[1] Moore J, Ritchie M (2004) The challenges of whole-genome approaches to common diseases. Jama 291: 1642.

[2] Moore J, Asselbergs F, Williams S (2010) Bioinformatics Challenges for Genome-Wide Association Studies. Bioinformatics .

[3] Thornton-Wells T, Moore J, Haines J (2004) Genetics, statistics and human disease: analytical retooling for complexity. TRENDS in Genetics 20: 640–647.

[4] Sing C, Boerwinkle E, Moll P (1985) Strategies for elucidating the phenotypic and genetic heterogeneity of a chronic disease with a complex etiology. Progress in clinical and biological research 194: 39.

[5] Bock C, Lengauer T (2008) Computational epigenetics. Bioinformatics 24: 1.

[6] Kuehn B (2006) NIH Initiatives to Probe Contribution of Genes, Environment in Disease. JAMA 295: 1633.

[7] on DSM-IV APATF (1994) DSM-IV: diagnostic and statistical manual of mental disorders .

[8] Ritchie M, Hahn L, Roodi N, Bailey L, Dupont W, et al. (2001) Multifactor-dimensionality reduction reveals high-order interactions among estrogen-metabolism genes in sporadic breast cancer. The American Journal of Human Genetics 69: 138–147.

[9] Cordell H (2002) Epistasis: what it means, what it doesn't mean, and statistical methods to detect it in humans. Human Molecular Genetics 11: 2463.

[10] Moore J, Gilbert J, Tsai C, Chiang F, Holden T, et al. (2006) A flexible computational framework for detecting, characterizing, and interpreting statistical patterns of epistasis in genetic studies of human disease susceptibility. Journal of theoretical biology 241: 252–261.

[11] Straub R, MacLean C, O'Neill F, Burke J, Murphy B, et al. (1995) A potential vulnerability locus for schizophrenia on chromosome 6p24–22: evidence for genetic heterogeneity. Nature Genetics 11: 287–293.

[12] Buxbaum J, Silverman J, Smith C, Kilifarski M, Reichert J, et al. (2001) Ev-

idence for a susceptibility gene for autism on chromosome 2 and for genetic heterogeneity. The American Journal of Human Genetics 68: 1514–1520.

[13] Ritchie M, Hahn L, Moore J (2003) Power of multifactor dimensionality reduction for detecting gene-gene interactions in the presence of genotyping error, missing data, phenocopy, and genetic heterogeneity. Genetic epidemiology 24: 150–157.

[14] Ritchie M, Edwards T, Fanelli T, Motsinger A (2007) Genetic heterogeneity is not as threatening as you might think. Genetic Epidemiology 31: 797.

[15] Digna T, Dudek R, Ritchie M (2009) Exploring the performance of multifactor dimensionality reduction in large scale SNP studies and in the presence of genetic heterogeneity among epistatic disease models. Hum Hered 67: 183–192.

[16] Congdon C (1995) A comparison of genetic algorithms and other machine learning systems on a complex classification task from common disease research .

[17] Holmes J (1997) Discovering risk of disease with a learning classifier system. In: Proceedings of the Seventh International Conference of Genetic Algorithms (ICGA97). Citeseer, pp. 426–433.

[18] Walter D, Mohan C (2000) ClaDia: a fuzzy classifier system for disease diagnosis. In: Evolutionary Computation, 2000. Proceedings of the 2000 Congress on. volume 2.

[19] Bernado E, Llora X, Garrell J (2001) XCS and GALE: a comparative study of two learning classifier systems with six other learning algorithms on classification tasks. In: Proceedings of the 4th international workshop on learning classifier systems (IWLCS-2001). Citeseer, pp. 337–341.

[20] Bacardit J, Butz M (2007) Data mining in learning classifier systems: Comparing xcs with gassist. Lecture Notes in Computer Science 4399: 282.

[21] Kharbat F, Bull L, Odeh M (2007) Mining breast cancer data with XCS. In: Proceedings of the 9th annual conference on Genetic and evolutionary computation. ACM, p. 2073.

[22] Gao Y, Huang J, Rong H, Gu D (2007) LCSE: learning classifier system ensemble for incremental medical instances. Lecture Notes in Computer Science 4399: 93.

[23] Unold O, Tuszyński K (2008) Mining knowledge from data using Anticipatory Classifier System. Knowledge-Based Systems .

[24] Holmes J (1996) A genetics-based machine learning approach to knowledge discovery in clinical data. In: Proceedings of the AMIA Annual Fall Symposium. American Medical Informatics Association, p. 883.

[25] Holmes J (2000) Learning classifier systems applied to knowledge discovery in clinical research databases. Lecture notes in computer science : 243–262.

[26] Holmes J, Sager J (2005) Rule discovery in epidemiologic surveillance data using EpiXCS: an evolutionary computation approach. LECTURE NOTES IN COMPUTER SCIENCE 3581: 444.

[27] Stout M, Bacardit J, Hirst J, Smith R, Krasnogor N (2009) Prediction of topological contacts in proteins using learning classifier systems. Soft Computing-A Fusion of Foundations, Methodologies and Applications 13: 245–258.

[28] Bacardit J, Stout M, Hirst J, Sastry K, Llorà X, et al. (2007) Automated alphabet reduction method with evolutionary algorithms for protein structure prediction. In: Proceedings of the 9th annual conference on Genetic and evolutionary computation. ACM New York, NY, USA, pp. 346–353.

[29] Urbanowicz R, Moore J (2009) Learning Classifier Systems: A Complete Introduction, Review, and Roadmap. Journal of Artificial Evolution and Applications 2009.

[30] Wilson S (1995) Classifier fitness based on accuracy. Evolutionary computation 3: 149–175.

[31] Bernadó-Mansilla E, Garrell-Guiu J (2003) Accuracy-based learning classifier systems: models, analysis and applications to classification tasks. Evolutionary Computation 11: 209–238.

[32] Bull L (2004) A simple payoff-based learning classifier system. Lecture notes in computer science : 1032–1041.

[33] Wilson S (1994) ZCS: A zeroth level classifier system. Evolutionary Computation 2: 1–18.

[34] Holland J (1995) Escaping brittleness: The possibilities of general-purpose learning algorithms applied to parallel rule-based systems .

[35] Mansilla E, Llorà X, Guiu J (2001) XCS and GALE: a comparative study of two learning classifier systems on data mining. In: Revised Papers from the 4th International Workshop on Advances in Learning Classifier Systems. Springer-Verlag, p. 132.

[36] Butz M, Sastry K, Goldberg D (1857) Tournament selection in XCS. In: Proceedings of the Fifth Genetic and Evolutionary Computation Conference (GECCO-2003). Citeseer, volume 1869.

[37] Butz M, Sastry K, Goldberg D (2003) Tournament selection: Stable fitness pressure in XCS. Lecture Notes in Computer Science : 1857–1869.

[38] Greene C, Penrod N, Kiralis J, Moore J (2009) Spatially Uniform ReliefF(SURF) for Computationally-Efficient Filtering of Gene-Gene Interactions. BioData mining 2: 5.

[39] Kira K, Rendell L (1992) A practical approach to feature selection. In: Proceedings of the ninth international workshop on Machine learning table of contents. Morgan Kaufmann Publishers Inc. San Francisco, CA, USA, pp. 249–256.

Chapter 3

The Application of Pittsburgh-Style Learning Classifier Systems to Address Genetic Heterogeneity and Epistasis in Association Studies

Abstract

Despite the growing abundance and quality of genetic data, genetic epidemiologists continue to struggle with connecting the phenotype of common complex disease

to underlying genetic markers and etiologies. In the context of gene association studies, this process is greatly complicated by phenomena such as genetic heterogeneity (GH) and epistasis (gene-gene interactions) that constitute difficult, but accessible challenges for bioinformatisists. While previous work has demonstrated the potential of using Michigan-style Learning Classifier Systems (LCSs) as a direct approach to this problem, the present study examines Pittsburgh-style LCSs, an architecturally and functionally distinct class of algorithm, linked by the common goal of evolving a solution comprised of multiple rules as opposed to a single "best" rule. This study highlights the strengths and weaknesses of the Pittsburgh-style LCS architectures (GALE and GAssist) as they are applied to the GH/epistasis problem.

3.1 Introduction

In the modern era of complex disease research, bioinformatisists and genetic epidemiologists have teamed up in search of disease markers and etiologies. While genome-wide association studies are a current favorite strategy to search for markers and etiologies of disease, these studies tend to focus on "main effects", or associations between an individual SNP and some disease phenotype. While this may be well suited for Mendelian diseases, it has become increasingly clear that different statistical and analytical techniques are needed to address the demands of common complex disease [1]. Phenomena recognized to complicate the epidemiological mapping of genotype to phenotype include epistasis, genetic heterogeneity, phenotypic hetero-

geneity, trait heterogeneity, gene-environment interaction, phenocopy, and epigenetics [2]. Epistasis, or gene-gene interaction, is a particularly challenging problem given that a gene's association with disease may only be seen in the context of at least one other gene. Genetic heterogeneity (GH), refers to the presence of different underlying genetic mechanisms resulting in the appearance of the same or similar disease phenotype [3]. The quality and availability of SNPs and other genetic code information make epistasis and GH obvious targets for bioinformatic development. Previous work examining this pair of phenomena, evaluate the impact of GH on the detection and modeling of epistasis using Multifactor Dimensionality Reduction (MDR) [4]. This work demonstrated that GH dramatically hinders MDR's power to detect/model all underlying attributes involved in the underlying epistatic interactions, but additional examination indicated that MDR retains significant power to identify either the dominant branch of GH or at least one of the underlying attributes [5, 6]. In the present study, GH and epistasis are modeled concurrently as they might occur simultaneously in a SNP-based genetic association study. Over the last decade, the detection and modeling of epistasis has seen a considerable amount of progress [7–9]. In contrast, methods for dealing with GH continue to lag behind, having relied on a traditional epidemiological paradigm with seeks to find a single best model of disease learned from a given data set [2]. These methods rely on covariate data (i.e. phenotypic data, genetic risk factors, demographic data, or endophenotypes) in order to identify more homogeneous subsets of patients. An obvious downside to this dependency, is

that the success of these methods relies on the availability, quality, and relevance of these covariates. Additionally, stratification of a dataset represents a reduction in sample size for respective analysis, leading to an inevitable loss in power. In order to address these concerns, Urbanowicz and Moore recently proposed the application and exploration of learning classifier systems (LCSs) as an alternative approach to managing GH [10]. LCSs represent a class of algorithm that requires neither covariates nor data stratification, and breaks from the traditional single model paradigm by evolving a solution comprised of multiple rules. For these reasons, it was hypothesized that LCS algorithms would be useful for the detection, characterization, and modeling of GH. In [10], Michigan-style LCSs (one of two major veins of LCS architectures) were implemented and evaluated on the GH/epistasis problem. The results of that study provided a proof of principle for the use of LCSs, highlighted the strengths and weaknesses of the Michigan-style systems, and laid the foundation for the development of an LCS algorithm specifically designed to address GH. In the present study, we explore Pittsburgh-style systems, in an effort to obtain a well-rounded perspective on LCS's ability to address the GH/epistasis problem, and to compare their ability to handle the GH/epistasis problem so that a suitable architectural foundation may be selected upon which to develop an LCS customized to this epidemiological task. This study involves (1) implementing standardized versions of existing Pittsburgh-Style LCSs (GALE and GAssist), (2) performing a sweep of the major run parameters for each LCS implemented, and (3) performing a quantitative and qualitative evaluation

of each LCS across the entire spectrum of simulated GH/epistatic data sets.

3.2 Methods

3.2.1 Learning Classifier Systems

LCSs combine machine learning with evolutionary computing and other heuristics to produce an adaptive system that learns to solve a particular problem. LCSs are closely related to and typically assimilate the same components as the more widely utilized genetic algorithm (GA). The goal of LCS is not to identify a single best model or solution, but to create a cooperative set of rules or models that together solve the task. The solution evolved by an LCS is represented as a population of rules, or rule-sets, that are utilized collectively to make decisions/classifications.

The infancy of LCS research saw the emergence of two founding classes of LCSs, referred to as the Michigan and Pittsburgh styles. The Michigan-style is characterized by a population of rules with a GA operating at the level of individual rules and an evolved solution represented by the entire rule population. Alternatively, the Pittsburgh-style is characterized by a population of variable length rule-sets (where each rule-set is a potential solution) and the GA typically operates at the level of a single rule-set. Additionally, Michigan-style systems learn iteratively from a dataset (learning from once instance at a time) while Pittsburgh-style systems learn in a batch-wise fashion, learning from each instance in the dataset every iteration. Of

note, the Pittsburgh-style systems implemented for this study evolve "ordered" rule sets, also known as decision lists, where rule order is important to the decision making process. Because of these differences in algorithm architecture and solution size, the application of these two styles to the GH problem have been explored separately. The present study focuses on Pittsburgh-style systems, as a parallel to the recent Michigan-style LCS study [10].

Pittsburgh-style LCSs, generally possess three basic components; (1) a population of rule/classifier sets, (2) a performance component that assesses how well rule sets collectively explain the data, and (3) a discovery component that uses different operators to discover new rules and improve existing ones. For a complete LCS introduction and review, see [11].

3.2.1.1 Implementing LCSs

Each of the implemented Pittsburgh-style LCSs were selected based on their prominence, relevance, and availability. The systems implemented include GALE [12,13], and GAssist [14], each re-encoded and modified in Python. This was done to (1) standardize the coding language, (2) put the different LCS algorithms in a flexible, readable language to facilitate and promote future algorithm development for biologists and other non-computer scientists, (3) allow command line control over all run parameters, (4) gain a detailed understanding of each system, and most importantly (5) to standardize the rule representation. Additionally, wrapper scripts were

written to run and evaluate each LCS using Dartmouth's 888 processor Linux Cluster, "Discovery". These implementations are freely available on the LCS & GBML Central webpage ($http://gbml.org/$).

A quaternary rule representation, identical to what was used in [10] was implemented in the selected Pittsburgh systems. This representation is well suited for the discrete, nominal, SNP attributes and the discrete affection status (case/control) characteristic of this problem domain. In short, the condition of the rule is represented by a string of characters from the quaternary alphabet $(0, 1, 2, \#)$ where # acts as a wildcard, and the intergers represent alternative SNP alleles. In standardizing the rule representation of GALE and GAssist, other algorithmic components that had relied on the original representation were adjusted accordingly (e.g. crossover and mutation mechanisms).

3.2.1.2 Two Pittsburgh-Style LCSs

While the systems GALE and GAssist share a number of features common to Pittsburgh systems, including batch/offline learning, ordered rule-sets (decision lists), accuracy based fitness, and supervised learning, they have distinct population architectures, and possess a very different set of supporting heuristics. Also, encodings of both systems were originally implemented in Java, and allowed for multiple knowledge representations. For simplicity only the applicable quaternary representation described above was encoded in the present python implementations. Every rule-set

(a.k.a. agent) of a Pittsburgh system represents a potential solution to a classification problem. GALE, or the Genetic and Artificial Life Environment [12, 13] is described as a fine-grained parallel evolutionary algorithm. GALE uses a 2D grid to evolve a population of rule-sets spatially, where discovery mechanisms may operate only within the local neighborhood. GAssist, or (Genetic clASSIfier sySTem) [14] descends from GABIL [15], having introduced several modifications to make it one of the most competitive and flexible Pittsburgh systems to date. These include elitism, an adaptive discretization intervals (ADI) rule representation, windowing, intelligent initialization, minimum description length (MDL)-based fitness, and the incorporation of an explicit default rule, detailed in [14]. In GAssist, genetic operators function at the population level.

3.2.2 System Evaluations

Each system was evaluated over the spectrum of simulated datasets described and generated in [10]. In brief, 1440 simulated SNP datasets, representing 96 data set configurations of differing GH heritability combinations, sample sizes, mix ratios, and difficulties were utlized. LCS evaluations were performed exactly as they were in [10], adding two new power estimates, and an additional tracking parameter to the previously used metrics; (1) an estimate of power to correctly detect predictive attributes (MichiganPower(MP)), (2) testing accuracy (10-fold cross validation strategy employed), (3) computational time, and (4) solution generality. As a precursor

to evaluating each LCS across the spectrum of simulated datasets, a parameter sweep was conducted in order to roughly optimize the parameter settings for the respective algorithms. Parameters examined in GALE include: (1) the probability of wild incorporation (0.5 and 0.75), (2) pruning; a rule deletion mechanism (*on* or *off*), (3) resource allocation; what instances were made available to cells within the 2D board (uniform, and pyramidal), and (4) maximum population size (MPS); the maximum number of rule-sets $(100, 625, 2500)$ [12]. Alternatively, parameters examined in GAssist include: (1) probability of wild incorporation (0.5 and 0.75), (2) population initialization (random, smart, class-wise (cw)), (3) MDL fitness (*on* or *off*), (4) an explicit default class (*on* or *off*), (5) windowing/window size (1, 2, 4), and (6) MPS $(100, 625, 2500)$ [14].

Power, or success rate, is typically estimated in these types studies by tracking the frequency with which an algorithm successfully identifies the correct underlying model or attribute(s), across some number of data set replicates. This type of estimation is not applicable to either Michigan or Pittsburgh LCSs that both evolve solutions made up of many rules. While the function of an LCS is to evolve a solution that can accurately classify patients, it is more important, and arguably much more difficult to evolve a solution that is meaningful and interpretable for a genetic epidemiologist. With this in mind, [10] developed "MP", based on the idea that attributes unimportant to the underlying model(s) (noise attributes) would tend to be generalized ('#'/don't-care symbol used) more frequently within rules of the popula-

tion. Pittsburgh-style LCSs, which evolve solutions made up dramatically fewer rules, required the addition of two additional power estimates, (BothPower(BP) and SinglePower(SP)) that indicate whether a precise predictive rule exists for both underlying epistatic models, or for at least a single underlying model, respectively. In addition, this study also tracked the proportion of the rules that were accurate predictive rules (i.e rules that accurately specified both attributes of an epistatic pair, but generalized across all other attributes with "#").

All statistical evaluations were completed using R. Logistic regression, modeling each data set dimension and all pair-wise combinations of dimensions, was employed for the evaluations of power. ANOVA and Tukey's HSD posthoc analysis was employed for the evaluation of accuracies, generality, predictive rule proportion, and run time. Differences were considered to be significant at ($p < 0.05$).

3.3 Results

3.3.1 Parameter Sweep

Completion of a parameter sweep led to the selection of the following parameters for GALE (MPS = 2500, Wild = 0.75, Pruning = On, Rescource Allocation = Uniform) and for GAssist (MPS = 625, Wild = 0.75, Initialization = CW, MDL Fitness = On, Default Class = On, Windows = 4). Any algorithm parameters not evaluated in the parameter sweep were left at their respective default settings. While both sys-

tems showed an improvement in both MP and testing accuracy with increasing MPS, GAssist runs allotted a MPS of 2500 failed to complete within a reasonable amount of time ($<$ 30 hours), thus an MPS of 625 was selected. Early testing of the two Pittsburgh systems indicated that both algorithms were struggling to perform well on the allotted task within the maximum time frame of 30 hours. Examination of the rule sets suggested that generalization may have been contributing to the slow initial learning of both systems. An examination of "wild" as a parameter, indicated that a value of 0.75 yielded a significant improvement for all power estimates, and testing accuracy. For GALE, the "uniform" resource allocation showed significantly improved testing accuracy at the expense of about twice as much run time, and "pruning" (a built in function of GAssist) approximately halved run time while significantly improving agent genenerality. In GAssist,(1) "CW" initialization significantly improved testing accuracy while taking significantly less run time than "smart", (2) turning on MDL fitness significantly reduced run time while increasing generality, MP, and BP, (3) using an explicit default class significantly reduced run time, generality, and BP while significantly improving testing accuracy, MP, and SP, and (4) windowing, designed to reduce computational time and increase generalization [14], did just that, along with significantly improving MP and BP over the increase in window size from 1-4.

3.3.2 Pittsburgh LCS Evaluations

Table 3.1 summarizes statistics describing the performance of the respective Pittsburgh LCSs. It is important to note that averages are calculated over every simulated dataset with the intention of comparing the overall performance of systems. Low overall averages of power and testing accuracy are therefore expected. "Best" power and testing accuracies come from the configuration of datasets with the highest respective values. All values in the table are representative of the performance of agents with the highest fitness in their respective evolved populations. Figures 3.1 and 3.2 summarize evaluations of testing accuracy (TA) and each power estimate across the four dimensions of simulated data. An immediate observation of this study was that the MP estimation method developed for [10], showed little to no success for any Pittsburgh systems examined. Logistic regression models indicate that each of the selected dataset dimensions (i.e. model combo, sample size, mix ratio, and difficulty) are significant predictors of all power estimations examined, as is the choice of LCS algorithm. Additionally, the interaction between algorithm choice and sample size for SP was significant, indicating that the choice of Pittsburgh LCS impacts the power to evolve at least one precise predictive rule given varying sample sizes. While MP and BP values over all simulated datasets were generally quite low, GAssist yielded significantly higher MP and SP than GALE.

Accuracy evaluations indicate that GAssist evolves agents with significantly higher testing accuracies, and significantly lower training accuracies than GALE. All dataset

Table 3.1: LCS Algorithm Summaries

Assorted	**GALE**	**GAssist**	**Power Estimates**	**GALE**	**GAssist**
Best Testing Accuracy	0.7162	0.7192	**Best MP**	0.0	0.2
Average Testing Accuracy	0.5828	0.6004	**Average MP**	0.0	0.0083
Average Training Accuracy	0.7862	0.7189	**Best BP**	0.1333	0.1333
Average Agent Size	29.60	8.01	**Average BP**	0.0118	0.0104
Average Generality	0.7889	0.7876	**Best SP**	1.0	1.0
Average Run Time (Min.)	113.79	208.02	**Average SP**	0.4403	0.5972
Average Predictive Rule %	0.03278	0.14703			

dimensions were found to be significant predictors of both testing and training accuracies in both systems. Similar to [10], datasets with a mix of 75:25 yielded significantly higher testing accuracies than 50:50. This supports the expectation that LCSs would be more proficient at learning a model that dominates the sample population. It is therefore not surprising that while the LCSs examined obtain higher testing accuracies and SP for datasets with a 75:25 model ratio, power estimates that rely on finding both underlying models (i.e. MP, and BP) obtain significantly lower values. Further evaluation demonstrates that GAssist, while evolving agents with the fewest average number of rules, also evolves solutions that on average are less general and require a longer runtime. Table 3.2 depicts one such agent evolved by GAssist.

While every evaluation described above was performed over 100 learning iterations, performance after only 25, and 50 iterations was also tracked. The impact of learning iteration on testing accuracy and the power estimates is summarized by the following; (1) testing accuracy and SP continued to significantly improve within

Table 3.2: Evolved GAssist Agent

Condition	**Class**	**Matched**	**Correct Class**	**Accuracy**
1 1 ##################	0	101	96	0.9505
0 # 0 0 ################	0	453	266	0.5872
## 1 1 ################	0	167	125	0.7485
0 0 #### 0 ########### 0	0	143	82	0.5734
###################	1	576	425	0.7378

iterations 25-100 and 50-100 for all systems, (2) none of the Pittsburgh systems saw significant improvement in MP from 50-100 iterations, (3) from iterations 50-100, GALE and GAssist each saw a small drop in BP. Overall, additional iterations and run time would seem to benefit testing accuracy and SP while suggesting little promise of improving MP or BP in the Pittsburgh LCSs examined.

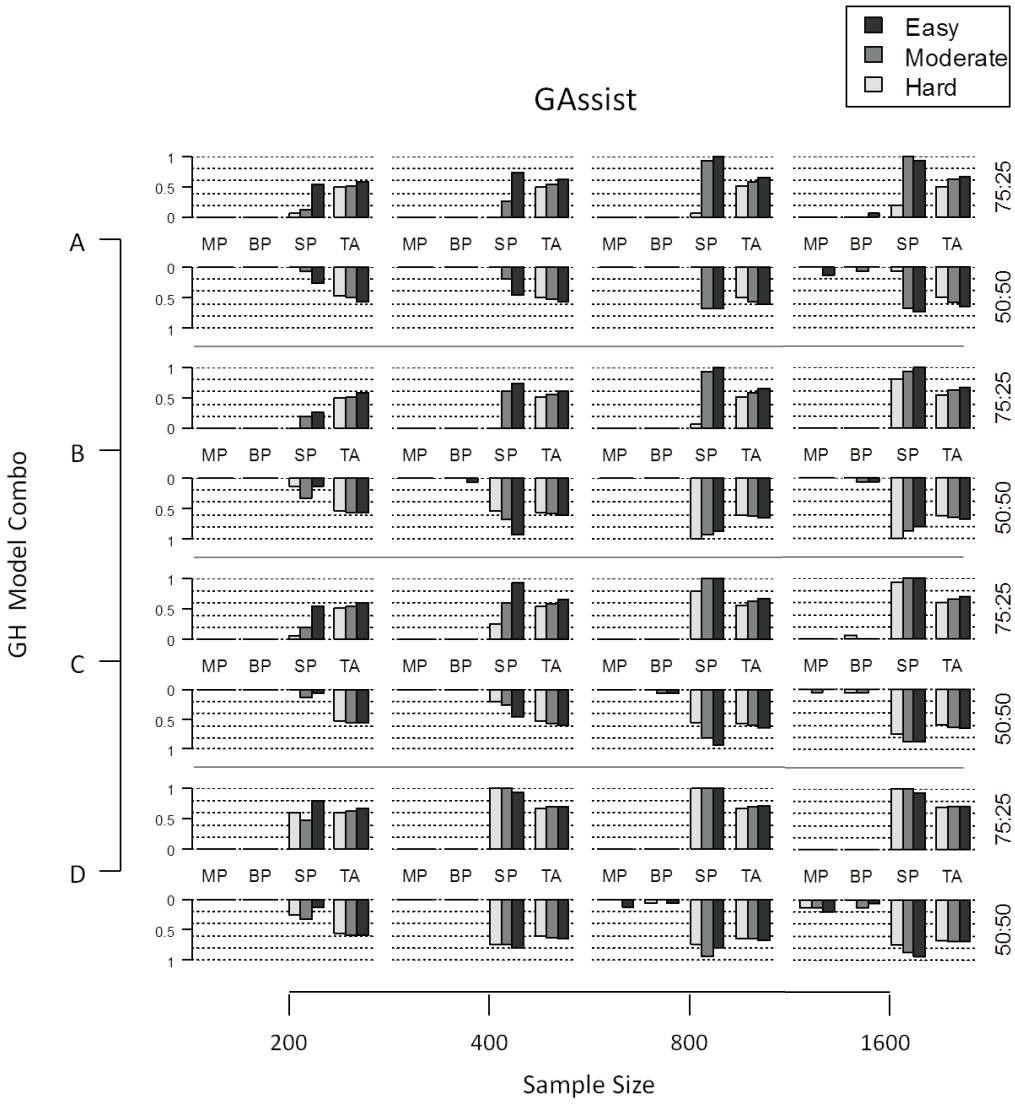

Figure 3.1: An illustration of testing accuracy and the three power estimates gathered from the GALE evaluation. The plot depicts the results over each dimension of the simulated dataset (i.e. model combination, sample size, mix ratio, and penetrance table difficulty / see section 3.2.2). The bars of each sub-plot represent an evaluation over 15 random seed datasets. Model combos include the following pairs of heritability; A = 0.1 & 0.1, B = 0.1 & 0.4, C = 0.2 & 0.2, and D = 0.4 & 0.4.

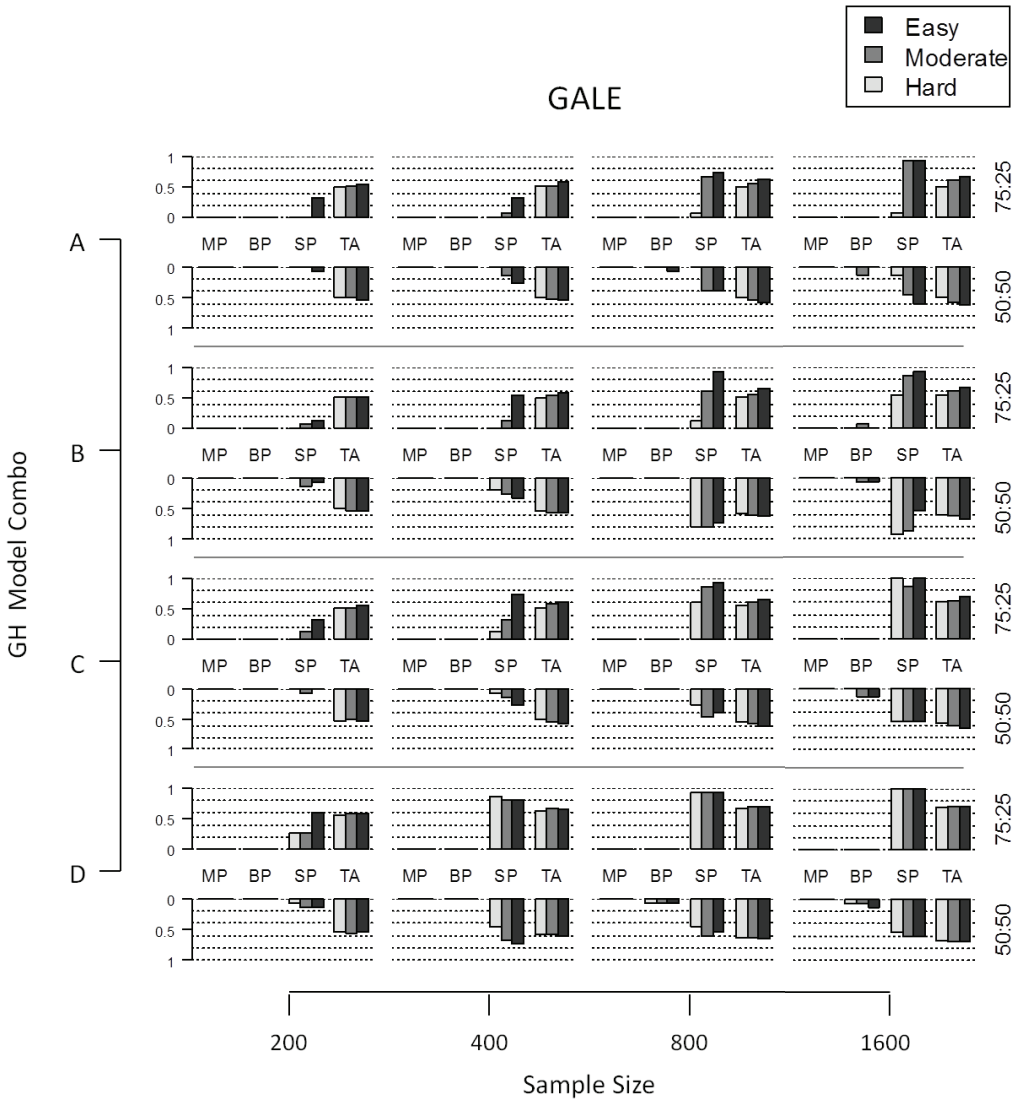

Figure 3.2: An illustration of testing accuracy and the three power estimates gathered from the GAssist evaluation. The plot depicts the results over each dimension of the simulated dataset (i.e. model combination, sample size, mix ratio, and penetrance table difficulty / see section 3.2.2). The bars of each sub-plot represent an evaluation over 15 random seed datasets. Model combos include the following pairs of heritability; A = 0.1 & 0.1, B = 0.1 & 0.4, C = 0.2 & 0.2, and D = 0.4 & 0.4.

3.4 Discussion and Conclusions

One of the most obvious advantages of the two Pittsburgh-style LCSs examined is the compact solutions that they evolve. Compared to the population-sized solutions evolved by the Michigan-Style systems [10], GALE and GAssist evolve agents that are compact enough for an epidemiologist to directly extract knowledge. Overall, GALE and GAssist both show promise addressing the GH/epistasis problem domain, evidenced by competitive testing accuracies and manageable solution sizes. While the study-wide average testing accuracy of GAssist is comparable to that of UCS in [10], the apparent failure of both Pittsburgh LCSs to evolve a rule set with a correct predictive rule characteristic of both underlying epistatic models (i.e. BP), represents a key challenge for adapting this style of LCS to the GH problem. One likely reason for this difficulty, stems from two intertwined features (i.e. the decision list and default rule), common to both Pittsburgh systems, designed to speed up and improve classification accuracy and agent compactness [14]. By ordering the rules of an agent (via the decision list), the explicit pressure on rule-sets to be accurate, intuitively drives the evolution of a default rule (a rule that is entirely general, or a "catch-all") that implicitly covers all un-matched instances, or in the case of an explicit default rule (as found in GAssist), covers the "default-class". While this is an effective method for condensing rule-set size, the default rule (either evolved, or explicitly included) essentially eliminates the specification of predictive rules for an entire class, reducing rule diversity, and likely having a direct impact on the power

estimation methods intended to evaluate LCSs.

The results of the parameter sweep serve not only to roughly optimize each LCS for a more thorough comparison, but demonstrate the importance of the accessory features available to each respective system. The evaluation of each system over the complete spectrum of simulated data suggests the following; (1) the selected rule representation allowed each system to evolve interpretable rules that model specific genotype combinations, (2) both GALE and GAssist were able to achieve significant power to identify at least one predictive rule from one of the two underlying models (> 0.8) for a number of challenging data configurations, as well as average testing/prediction accuracies significantly higher than 0.5, (3) both GALE and GAssist are computationally intensive, and would likely benefit from a greater number of learning iterations, and (4) the small rule-sets evolved by GALE and GAssist offer the potential for users to identify underlying GH. As an example of such interpretability, refer to the agent depicted in Table 3.2 where the underlying GH is correctly characterized by the first and third rules that specify two high risk epistatic genotype combinations involving two separate pairs of attributes. In this example the correct underlying models involve attributes/SNPs (1, 2) and (3, 4) respectively.

The results of this study support the employment of LCSs to address the detection, modeling and characterization of attributes associated with common complex disease given the complicating presence of underlying GH and epistasis. However, these findings do not necessarily indicate that a given LCS will perform "better" or

"worse" within any other problem domain to which they might be applied. The collective findings of [10] and the present study will be used to direct the development of a problem-specific LCS algorithm, aimed at (1) improving the power to detect underlying attributes, (2) being able to distinguish distinct underlying models constituting the GH, and (3) enhancing the overall interpretability of the evolved LCS rule-population solution.

Acknowledgments

This work was supported by the William H. Neukom 1964 Institute for Computational Science at Dartmouth College along with NIH grants AI59694, LM009012, and LM010098. We would like to thank Xavier Llora, and Jaume Bacardit for making open source Java implementation of their respective LCS algorithms freely available.

3.5 Bibliography

[1] Moore J, Asselbergs F, Williams S (2010) Bioinformatics challenges for genome-wide association studies. Bioinformatics 26: 445.

[2] Thornton-Wells T, Moore J, Haines J (2004) Genetics, statistics and human disease: analytical retooling for complexity. Trends in Genetics 20: 640–647.

[3] Sing C, Boerwinkle E, Moll P (1985) Strategies for elucidating the phenotypic

and genetic heterogeneity of a chronic disease with a complex etiology. Progress in clinical and biological research 194: 39.

[4] Ritchie M, Hahn L, Moore J (2003) Power of MDR for detecting gene-gene interactions in the presence of genotyping error, missing data, phenocopy, and genetic heterogeneity. Genetic epidemiology 24: 150–157.

[5] Ritchie M, Edwards T, Fanelli T, Motsinger A (2007) Genetic heterogeneity is not as threatening as you might think. Genetic Epidemiology 31: 797.

[6] Digna T, Dudek R, Ritchie M (2009) Exploring the performance of multifactor dimensionality reduction in large scale SNP studies and in the presence of genetic heterogeneity among epistatic disease models. Hum Hered 67: 183–192.

[7] Ritchie M, Hahn L, Roodi N, Bailey L, Dupont W, et al. (2001) MDR reveals high-order interactions among estrogen-metabolism genes in sporadic breast cancer. The American Journal of Human Genetics 69: 138–147.

[8] Cordell H (2002) Epistasis: what it means, what it doesn't mean, and statistical methods to detect it in humans. Human Molecular Genetics 11: 2463.

[9] Moore J, Gilbert J, Tsai C, Chiang F, Holden T, et al. (2006) A flexible computational framework for detecting, characterizing, and interpreting statistical patterns of epistasis in genetic studies of human disease susceptibility. Journal of theoretical biology 241: 252–261.

[10] Urbanowicz R, Moore J (2010) The Application of Michigan-Style Learning Classifier Systems to Address Genetic Heterogeneity and Epistasis in Asssociation Studies. In: Proceedings of the 12th annual conference on Genetic and evolutionary computation. ACM, p. In Press.

[11] Urbanowicz R, Moore J (2009) Learning Classifier Systems: A Complete Introduction, Review, and Roadmap. Journal of Artificial Evolution and Applications .

[12] Llora X, Garrell J (2000) Automatic Classification and Artificial Life Models. In: Proceedings of Learning Workshop IEEE and Univesidad Carlos III.

[13] Llora X, Garrell J, et al. (2001) Knowledge-independent data mining with fine-grained parallel evolutionary algorithms. In: Proceedings of the Third Genetic and Evolutionary Computation Conference, Morgan Kaufmann. Citeseer, pp. 461–468.

[14] Bacardit J (2004) Pittsburgh genetic-based machine learning in the data mining era:representations, generalization, and run-time. Ph.D. thesis, Enginyeria i Arquitectura La Salle, Ramon Llull University, Barcelona, European Union(Catalonia, Spain).

[15] De Jong K, Spears W (1991) Learning concept classification rules using genetic algorithms. In: Proceedings of the Twelfth International Joint Conference on Artificial Intelligence. Citeseer, pp. 651–656.

Chapter 4

An Analysis Pipeline with Visualization-Guided Knowledge Discovery for Michigan-Style Learning Classifier Systems: Interpreting the Black Box

Abstract

Michigan-style learning classifier systems (M-LCSs) represent an adaptive and powerful class of evolutionary strategies that have been successfully applied to many

problem domains including data mining and classification. However, M-LCSs have not been widely utilized in real-world data mining applications. This is largely due to the complexity of the algorithms and the solutions they generate. Specifically, M-LCSs distribute their learned solution over a sizable population of rules that collectively represent the solution. Such solutions are notoriously difficult and tedious to extract knowledge from. In the present study we introduce an M-LCS analysis pipeline that combines novel visualizations with objective statistical evaluation for the identification of predictive attributes, and reliable rule generalizations in noisy single-step data mining problems. Visualizations include a clustered heat-map of rule population generalizations, and a network illustrating the frequency of attribute pair co-occurrence. Using permutation testing, we demonstrate how to discriminate between significant predictive and non-predictive attributes within an evolved rule-population. This work considers an alternative paradigm for knowledge discovery in M-LCSs, shifting the focus from individual rules to a global, population-wide perspective. We apply this pipeline to the identification of epistasis (i.e. attribute interaction), and heterogeneity in noisy, simulated genetic association data.

4.1 Introduction

Learning classifier systems (LCSs) [1] are a rule-based class of algorithms that combine machine learning with evolutionary computing and other heuristics to produce an adaptive system. The goal of LCS is not to identify a single best model or rule, but

to create a cooperative set of rules that collaborate to solve the given problem. This feature makes LCSs appealing for application to complex multifactor problem domains such as function approximation [2], clustering [3], behavior policy learning [4], and classification/data mining [5]. Data mining in bioinformatics problems can be particularly challenging, involving noisy and complex problem landscapes. Recently, both Michigan and Pittsburgh-style LCSs were applied to the detection and modeling of simulated genetic disease associations with epistasis and heterogeneity [6, 7]. These evaluations identified the strengths and weaknesses of using either a Michigan or Pittsburgh-style LCS on these types of complex, noisy problems.

In summary, both styles yielded accurate predictions, but Pittsburgh-style LCSs (P-LCSs) evolved much more compact (and therefore more interpretable) solutions. However, in doing so, they relied heavily on the system's ability to evolve precise rules for expert interpretation [7, 8]. Additionally, P-LCSs often incorporate default rules and hierarchical decision making that speed up accurate batch learning but can hinder the complete characterization of heterogeneous and overlapping problem sub-domains.

Evaluation of Michigan-style LCSs (M-LCSs) expectedly yielded solutions comprised of large rule sets and a greater tendency to over-fit training data [6, 8]. Large rule-sets present obvious challenges for interpretation and knowledge extraction. However, M-LCSs learn iteratively from the dataset (one sample at a time) intrinsicly accommodating learning in problem subspaces and overlap. While the distributed

nature of an M-LCS solution is an obvious roadblock for classic rule-centric interpretation, we propose that global strategies (examining rule trends over the entire population) will offer a better approach to knowledge discovery in these algorithms.

Knowledge discovery has been described as the "non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data" [9]. Efforts to improve knowledge discovery in P-LCSs or similar genetics based machine learning (GBML) algorithms are synonymous with improving overall P-LCS performance (i.e. accuracy, generality, speed, and solution compactness), since these algorithms explicitly strive to evolve a concise rule set that can be directly interpreted [5, 10, 11]. For M-LCSs, most efforts aimed at facilitating knowledge discovery focus on reducing the size of the rule population by rule compaction [12–15] or condensation [16,17], or alternatively, by modifying the rule representation [17,18]. These strategies are all in-line with the classic paradigm of knowledge discovery wherein the goal is to identify explicit rules for interpretation that are accurate, and maximally general.

Kharbat, Odeh and Bull took a somewhat different approach to M-LCS rule compaction (rule-dependent as opposed to data-dependent) in order to extract minimal and representative rules from the original rule-set [19–21]. Rules were clustered based on similarity and these clusters were used to generate aggregate average rules and aggregate definite rules presenting the common characteristics of the cluster. These resulting aggregate rules are then interpreted by an expert seeking knowledge discov-

ery. This strategy reflects a more global perspective of rule-set evaluation looking for common patterns in a large population of rules. In the present study we seek to extend this line of global thinking.

The challenges involved in applying a M-LCS to single-step data mining problems include (1) distinguishing predictive attributes from those that are not important (noise attributes), (2) identify attribute interactions, i.e. what combinations of attributes are valuable for making predictions, and (3) identify heterogeneity, i.e. are different attributes or attribute combinations important within different subsets of the dataset. Previously in [6], we introduced a global power estimation strategy that identifies attributes, determined by LCS, to be more important for accurate classification based on their frequent specification within rules of the population.

In the present study we lay out an analysis pipeline to address the challenges described above. In summary, our proposed analysis pipeline includes the following steps: (1) run the M-LCS algorithm with 10-fold cross validation (CV) on target dataset, (2) run a permutation test with 1000 permutations (each with 10-fold CV), (3) confirm significance of testing accuracy, (4) identify significant attributes and significantly co-occurring pairs of attributes, (5) train the M-LCS algorithm on the entire dataset (for visualization), (6) generate a clustered heat-map of rule-population, (7) generate a network depicting attribute co-occurrence, and (8) combine statistical results with visualizations to interpret and generate hypothesis for further exploration and validation. We apply this pipeline to a complex simulated genetic association

dataset embedded with heterogeneity and epistasis as a demonstration of its efficacy.

"Visualization of classification models can create understanding and trust in data mining models" [22]. While visualization strategies are not entirely new to the LCS field, to date they have been applied only to track learning progress within the search space [23, 24]. The key difference here, is that visualization is applied directly to knowledge discovery for the identification of global attribute generalizations. Also, unique to gene association studies, we demonstrate that this analysis pipeline facilitates the identification and characterization of genetic heterogeneity and epistasis.

The remainder of the paper is organized as follows. A description of the M-LCS algorithm utilized in this study is given in section 4.2. A description of the target bioinformatics problem and dataset evaluated in this study is given in section 4.3. An explanation of the analysis pipeline and our results are detailed throughout section 4.4. Conclusions are drawn in section 4.5.

4.2 Learning Classifier Systems

Within the LCS community of algorithms there is an impressive diversity of architectures and heuristic modifications that have been implemented with the intention of improving prediction accuracy, run time, solution compactness and solution comprehensibility to address a given problem domain. The infancy of LCS research saw the emergence of two founding classes of LCSs, referred to as the Michigan (M) and Pittsburgh (P) styles. The M-LCS is characterized by a population of rules with a GA

operating at the level of individual rules and the evolved solution is represented by the entire rule population. Alternatively, the P-LCS is characterized by a population of variable length rule sets (where each rule-set is a potential solution) and the GA typically operates at the level of a single rule-set.

M-LCSs, often varying widely from version to version, generally possess four basic components; (1) a population of rules or classifiers, (2) a performance component that assesses how well the population of rules collectively explain the data, (3) a reinforcement component that distributes the rewards for correct prediction to each of the rules in the population, and (4) a discovery component that uses different operators to discover new rules and improve existing ones. Learning progresses iteratively, relying on the performance and reinforcement components to drive the discovery of better rules. For a complete LCS introduction and review, see [25].

UCS, or the sUpervised Classifier System [26], is based largely on the very successful XCS algorithm [27], but replaces reinforcement learning with supervised learning, encouraging the formation of best action maps (rule sets of efficient generalizations) and altering the way in which accuracy (and thus fitness) is computed. UCS was designed specifically to address single-step problem domains such as classification and data mining, where delayed reward is not a concern. The implementation of UCS applied in this study is the same as the one used in [6] adopting mostly default parameters with the exception of 200,000 learning iterations, a population size of 2000, tournament selection, uniform crossover, subsumption, and a ν of 1. ν has been

described as a "constant set by the user that determines the strength [of] pressure toward accurate classifiers" [28], and is typically set to 10 by default. A low ν was used to place less emphasis on high accuracy in this type of noisy problem domain, where 100% accuracy is only indicative of over-fitting. Also, as in [6], we employ a quaternary rule representation, where for each SNP attribute, a rule can specify genotype as (0, 1, or 2), or instead generalize with "#", a character that implies that the rule doesn't care about the state of that particular attribute.

4.3 Bioinformatics Problem

As geneticists strive to identify "markers for" and "causes of" common complex disease, it has become increasingly clear that the statistical and analytical techniques, well suited for Mendelian studies, are insufficient to address the demands of common complex disease [29, 30]. Epistasis (gene-gene interaction) and genetic heterogeneity are two of several phenomena reviewed by [31] believed to hinder the reliable identification of predictive genetic markers in association studies. Epistasis refers to the interaction between multiple genetic loci wherein the effect of a given loci is masked by one or more others. Genetic heterogeneity (GH) occurs when the same genetic disorder or phenotype can be caused by different, independent genetic mechanisms or pathways (involving one or more alleles or loci).

While the detection and modeling of epistasis has received a great deal of attention [32–34], methods for dealing with GH are lagging behind, clinging to a traditional

epidemiological paradigm that seeks to find a single best model of disease within a given data set [31]. From a computer science prospective, the problem of genetic heterogeneity is similar to a latent or "hidden" class problem. While the disease status of each patient is already known, the individuals making up either class would be more accurately sub-typed into two or more "hidden" classes, each characterized by an independent predictive model of disease. Existing methods for explicitly dealing with GH include sample stratification, the M test, the β test, the admixture test, ordered subset analysis, cluster analysis, latent class analysis, and factor analysis, all reviewed in [31]. With the exception of the admixture test (that tests a more general hypothesis of GH) all of the methods rely on covariate data (i.e. genetic risk factors, demographic data, phenotypic data, or endophenotypes) in order to identify more homogeneous subsets of patients. This is in line with the standard epidemiological paradigm with seeks a single best disease model within a given homogeneous sample. The obvious drawback of these methods is that they completely rely on the availability, quality, and relevance of these covariates. Additionally, stratification represents a reduction in sample size, leading to an inevitable loss in power.

To address the problem of GH directly, LCSs were employed and evaluated in [6] and [7]. Since LCSs break from the "best model" paradigm, and evolves a solution comprised of a cooperative set of rules, it offers an intuitive strategy for addressing the GH problem without data stratification. In both studies, datasets were generated that concurrently modeled epistasis and GH as they might simultaneously occur in

a single nucleotide polymorphism (SNP) genetic association study. All datasets were generated using a pair of distinct, two-locus epistatic interaction models, both utilized to generate instances (i.e. case and control individuals) within a respective subset of each data set. Two-locus epistatic models were simulated without Mendelian/main effects, as penetrance tables. In total, each simulated data set contained 4 predictive attributes, where only two attributes were predictive within a respective subset of the data. Additionally, each data set contained 16 randomly generated, non-predictive attributes, with minor allele frequencies randomly selected from a uniform distribution ranging from 0.05 to 0.5.

In [6], M-LCSs demonstrated the ability to evolve accurate rule populations on these data sets. Also power estimation based on the frequency of attribute generalization/specialization proved successful. However, characterization of attribute patterns within the rule population remained a considerable challenge. Specifically, when examining individual rules in the population, it was quite common to find highly fit rules that correctly specified the attributes of an underlying epistatic model but that also specified one or two other non-predictive attributes. In these situations, maximal accuracy has been achieved at the expense of optimal generality. Additionally, these M-LCS evaluations lacked statistical justification for differentiating significant predictive attributes from non-predictive (noise) attributes.

A largely ubiquitous goal for LCS algorithmic design and application is to simultaneously maximize both accuracy and rule generality. However, these goals can be at

odds with one another, especially in noisy problem domains where small, "chance" associations between non-predictive attributes and class lead to their inclusion in rules. In many popular M-LCSs, fitness measures are explicitly accuracy based, and generalization is encouraged by implicit pressures of the system [16, 27] or supplemental heuristics such as subsumption [35]. In comparing XCS with GAssist, the tendency for XCS to over-fit training data, (especially in smaller datasets) was observed [36]. Over-fitting often indicates that the system is learning structure that is idiosyncratic to the training set and therefore generalization is occurring sub-optimally. Without prior knowledge of the problem complexity or structure, achieving the ideal balance between accuracy and generalization may be impractical or even impossible. With that in mind, it seems unreasonable to expect LCSs to automatically evolve optimal individual rules for interpretation. The task then becomes: what reliable attribute patterns can we derive from the rule population?

In this study we demonstrate the utility of our analysis pipeline using a dataset from [6] and [7] that concurrently models epistasis and genetic heterogeneity as described earlier. Minor allele frequencies of all 4 predictive attributes were 0.2, heritabilities for both underlying models were 0.4, relative model architecture difficulty was "Easy", the ratio of samples representative of either model was 50:50, and the dataset sample size was 1600. Evaluation of this dataset in [6] with an implementation of the UCS algorithm [26] yielded significant power to detect the underlying predictive attributes. This dataset was chosen to provide a clear example of this

pipeline analysis.

In addition, as a negative control we repeat the pipeline analysis on a second dataset within which all attributes were non-predictive. Specifically, this dataset is a class-permuted version of the dataset described above.

4.4 Analysis Pipeline

Our proposed analysis pipeline includes the following steps: (1) run the M-LCS algorithm with 10-fold cross validation (CV) on the dataset, (2) run a permutation test with 1000 permutations, (3) confirm significance of testing accuracy, (4) identify significant attributes and significantly co-occurring pairs of attributes, (5) train the M-LCS algorithm on the entire dataset, (6) generate a clustered heat-map of the rule-population, (7) generate a network depicting attribute co-occurrence, and (8) combine statistical results with visualizations to interpret and generate hypothesis for further exploration and validation. While in this study we utilize UCS and focus on a very specific data mining problem, this pipeline could be expanded to knowledge discovery in any M-LCS applied to a single step data mining problem.

4.4.1 Run the M-LCS

This section details steps 1 and 2. First we employ a 10-fold CV strategy in order to determine average testing accuracy and account for over-fitting. The dataset is

randomly partitioned into 10 equal parts and UCS is run 10 separate times during which 9/10 of the data is used to train the algorithm, and a different 1/10 is set aside for testing. We average training and testing accuracies over these 10 runs. Next we set up our permutation test. We generate 1000 permuted versions of the original dataset by randomly permuting the affection status (class) of all samples, while preserving the number of cases and controls. For each permuted dataset we run UCS using 10-fold CV. In total, permutation testing requires 10,000 runs of UCS. We perform this analysis using "Discovery", a 1372 processor Linux cluster.

4.4.2 Significance Testing of M-LCS Statistics

This section details steps 3 and 4. First, and most importantly, we confirm that our average testing accuracy from Step 1 is significantly higher than by random chance. We utilize a typical one-tailed permutation test with a significance threshold of $p < 0.05$. To determine p for a test statistic (in this case average testing accuracy) calculate the test statistic for each of the 1000 permuted, CV analyses. If the true test statistic from Step 1 is greater than 95% of the 1000 permuted runs, you can reject the null hypothesis at $p < 0.05$. If the true test statistic is greater than all of the 1000 permuted runs, a minimum p of 0.001 is achieved. Our analysis on the aforementioned dataset yielded a significant testing accuracy of 0.701 ($p = 0.001$).

If average testing accuracy is not significantly high, this suggests that M-LCS was unable to learn any useful generalizations from the data, and there is little reason to

progress with the rest of this analysis pathway. Failure to obtain a significant testing accuracy suggests either the absence of a useful generalization within the data, or that the run settings for the M-LCS were inappropriate for the detection of an existing generalization (e.g. a larger population size or greater number of learning iterations may be required).

Once a significant testing accuracy is confirmed we next use the permutation test analysis to identify attributes in the dataset that show significant importance in making accurate classification. Here we describe distinct test statistics for making such an inference from the rule population: (1) Specificity Sum (SpS) and (2) Accuracy Weighted Specificity Sum (AWSpS). The SpS is identical in principle to our previously described power estimation strategy [6]. The premise for this statistic states that if an M-LCS is learning useful generalizations while training on the data, attributes that are important for making correct classifications will tend to be specified more frequently within rules of the population. Alternatively, attributes that are not useful for making correct classifications will tend to be generalized (#/don't care symbol used). SpS is calculated separately for each attribute, and is simply the number of rules in the population that specifies a value for that attribute (as opposed to having a '#'). This calculation takes rule numerosity into account, since numerosity represents the number of copies of that rule presently in the population. As an alternative to SpS, we also consider AWSpS, which simply weights the SpS by the respective rule accuracies. Like SpS, AWSpS is calculated separately for each attribute, and takes

numerosity into account. Table 4.1 gives a simple example in which both SpS and AWSpS are calculated from a population of 4 rules. Notice how attributes 'X1' and 'X4' stand out from the rest.

The intuition behind using AWSpS vs. SpS, is based on the idea that: at any given learning iteration, it is possible for a non-predictive attribute to be specified frequently in rules of the population just by chance. Consider the hypothetical situation in which a predictive attribute and non-predictive attribute happen to be specified with equal frequency. We would expect that, overall, rules specifying the predictive attribute would tend to have higher accuracies than rules with the non-predictive attribute. Therefore, by weighting specification frequency by accuracy, we would be more likely to correctly favor the predictive attribute as important to classification and avoid false positives. Conversely, by chance, non-predictive attributes could be specified along with predictive attributes in a highly accurate rule. In this case, the non-predictive attributes might receive a parasitic boost from the overall high accuracy of the rule, potentially encouraging a false positive. Because of this dilemma we consider both statistics.

Again, we use permutation testing to determine the significance of attributes using the SpS and AWSpS test statistics. For each attribute we separately calculate SpS and AWSpS over the 10 CV rule populations trained in Step 1. For each attribute we compare this sum to the 1000 respective sums from permutation testing to determine whether the true SpS or AWSpS is significantly higher than we would expect by

Table 4.1: Example calculation of SpS and AWSpS

Attribute	X1	X2	X3	X4	Class	Numerosity	Accuracy
R1	2	#	#	1	0	5	0.73
R2	#	0	#	2	1	1	0.51
R3	2	0	#	1	0	2	0.88
R4	#	0	1	#	1	1	0.62
SpS	7	4	1	8			
AWSpS	5.41	2.89	0.62	5.92			

chance. Significance values are calculated as previously described for each attribute. This step provides us with a statistically justified strategy for discriminating between predictive and non-predictive attributes in a M-LCS rule population.

Table 4.2 organizes the SpS and AWSpS statistic results for our target dataset. Half the samples in the dataset were generated with a predictive epistatic interaction between attributes 'X0' and 'X1', while the other half were generated with a different epistatic interaction between attributes 'X2' and 'X3'. All other attributes were randomly simulated as non-predictive. The SpS and AWSpS for our four predictive attributes (i.e. X0, X1, X2, X3) are significantly higher than would be expected by chance with sums dramatically larger than non-predictive attributes. Notice that with the AWSpS statistic, two non-predictive attributes barely make the significance cutoff. However their AWSpS values are dramatically lower than the predictive attributes. Investigators could also approach this analysis with a two-tailed permutation test

(with a significance cutoff of $p < 0.025$), and ask the questions: which attributes are specified more frequently, and which attributes are specified less frequently than we would expect by chance? Here, such an analysis would have precisely identify the correct 4 predictive attributes significantly high with no false positives, and in addition would identify 6 of the non-predictive attributes as having been significantly under-specified, suggesting that the M-LCS learned to avoid specifying these attributes.

The last test statistic we describe considers pair-wise attribute co-occurrence within rules. We consider this statistic in order evaluate attribute interactions as well as to help discriminate between interaction and heterogeneity. We calculate co-occurrence for every non-redundant pair-wise combination of attributes in the dataset. For a dataset with 20 attributes (such as the one we examine here), we calculate 190 co-occurrence values. We calculate co-occurrence as follows: for every pair of attributes we go through each of the 10 CV rule populations and sum the number of times that both attributes are concurrently specified in a given rule. In the example from Table 4.1, the Co-occurrence Sum (CoS) for the attribute pair (X1,X2) would be 2 since the pair only co-occurs in rule 3, and the rule has a numerosity of 2. The significance of co-occurrence scores are calculated in the same manner as before using the permutation test analysis for each pair of attributes.

Table 4.3 organizes the top CoS statistic results for our target dataset (only significant CoSs with sums greater than 3000 are displayed). 43 out of the 190 CoSs were identified as significantly higher than by chance. Each of these significant pairs

Table 4.2: SpS and AWSpS Results

Attribute	SpS	p-value	AWSpS	p-value
X0	10885	0.001*	7589.49	0.001*
X1	11359	0.001*	7936.43	0.001*
X2	10569	0.001*	7369.84	0.001*
X3	10150	0.001*	7114.25	0.001*
X4	3863	0.999	2482.56	0.888
X5	3240	1	2090.05	1
X6	5217	0.737	3446.47	0.18
X7	5484	0.915	3647.67	0.336
X8	4429	0.95	2927.85	0.482
X9	5334	0.985	3569.25	0.484
X10	5907	0.414	3948.81	0.04*
X11	5725	0.414	3933.61	0.037*
X12	5273	1	3518.87	0.761
X13	4443	1	2854.43	0.996
X14	3709	1	2391.91	0.978
X15	5108	0.916	3425.64	0.355
X16	3613	1	2299.56	1
X17	3639	1	2302.01	0.992
X18	5933	0.629	4004.96	0.085
X19	4591	1	2940.28	0.999

had at least one predictive attribute represented. Seeing that we found the four predictive attributes to be significantly over-represented, it follows that co-occurrence pairs including at least one of these attributes would turn up more frequently than by chance. Of particular note, we found that our two modeled, epistatic pairs of predictive attributes yielded the two highest CoSs. Below these two pairs, there is an immediate drop-off in the magnitude of CoS (almost halved). Also note that the magnitude of these two highest CoSs are well above the SpSs for all non-predictive attributes given in Table 4.2.

While additional analysis could examine higher order co-occurrence between all 3-way combinations of attributes (or beyond), we can use pair-wise analysis to infer higher order attribute interactions. For example, if a predictive, 3-way interaction existed in the data (between X0, X1, and X3) we would expect similar pair-wise CoSs between attribute pairs (X0,X1), (X1,X2), and (X0,X2). Using this logic we could hypothesize, given the results in Table 4.3, that attribute pairs (X0,X1), and (X2,X3) each represent an interacting pair, but since the four other pair-wise combinations of these attributes have about half the co-occurrence we might suspect heterogeneity as opposed to a 3 or 4-way interaction. Section 4.4.4 will offer a visualization of these co-occurrence results.

Table 4.3: Top Co-occurrence Sum Results

Attribute Pairs		CoS	p-value
X0	X1	8060	0.001*
X2	X3	7373	0.001*
X1	X2	4223	0.001*
X0	X2	4079	0.001*
X1	X3	3974	0.001*
X0	X3	3829	0.001*
X1	X11	3621	0.001*
X1	X7	3574	0.001*
X0	X11	3540	0.001*
X2	X10	3485	0.001*
X0	X7	3462	0.001*
X3	X10	3392	0.001*
X1	X18	3379	0.001*
X0	X18	3264	0.001*
X1	X15	3255	0.001*
X0	X15	3035	0.001*
X2	X12	3020	0.005*
X2	X18	3016	0.001*

4.4.3 Visualization - Heat-Map

This section details steps 5 and 6. A heat-map population visualization requires a single rule population. All M-LCS runs up to this point have been dedicated to obtaining test statistics and making statistical inferences. Our first step towards visualization is to train the M-LCS on the entire dataset. M-LCS algorithms are typically very adaptive with a tendency to maintain some level of diversity in the population as it attempts to search for better and better rules. As a result we would expect that some proportion of the rules will be poor classifiers with useless generalizations. As previously mentioned, rule compaction [12–15] and condensation [16, 17] algorithms offer a method of eliminating useless rules and compacting the size of the rule population. While beyond the scope of the present study, such an algorithm could be used at this point in the analysis pipeline in an attempt to remove some of these useless, "noisy" rules. However, based on preliminary observations exploring rule compaction algorithms we would caution the reader that a dramatic reduction in the size of the rule population may be counterproductive to successfully identifying global patterns.

Our next step includes re-encoding the rule population. The objective of the heat-map visualization is to discriminate predictive attributes from non-predictive attributes and to look for patterns of attribute interaction and heterogeneity. Therefore we re-encode each rule such that any specified attribute is coded as a 1 while a '#' is coded as a 0. Additionally, we expand our rule population such that there are N copies of each rule reflecting respective numerosities. The last processing step before

visualization is to apply a clustering algorithm to the coded and expanded rule population. Clustering a population of M-LCS rules for rule compaction was pioneered in [20, 21]. In the present study we employ agglomerative hierarchical clustering using hamming distance as the distance metric. Clustering is performed on both axes (i.e. across rules and attributes). Both clustering and 2D heat-map visualization was performed in R using the *hclust* and *gplots* packages, respectively.

Finally, we apply 2D heat-maps to highlight global attribute patterns for knowledge discovery in M-LCSs. Figure 4.1 gives our 2D heat-maps visualizing the rule population. In Figure 4.1A, the only clearly discernable pattern is the relative high specification of the four modeled attributes (X0, X1, X2, and X3) on the left. These four columns stand out as having more yellow color than the others. After clustering the utility of this visualization become much more apparent. While portions of Figure 4.1B are relatively noisy, we see two dominant rule patterns emerge, one involving attributes X0 and X1, and the other involving attributes X2 and X3. Note how hierarchical clustering separates these independent epistatic models, suggesting the presence of a heterogeneity vs. higher order interactions. If a four way interaction was modeled, we would instead expect these attributes to cluster together.

Apart from the 2D heat-map, we repurposed a 3D heat-map visualization tool that can accommodate up to 6 dimensions of information and allows users to interactively explore the 3D visualization space. This software was adapted to bioinformatic visualization in [37] using Unity3D (http://unity3d.com). Beyond obtaining a global

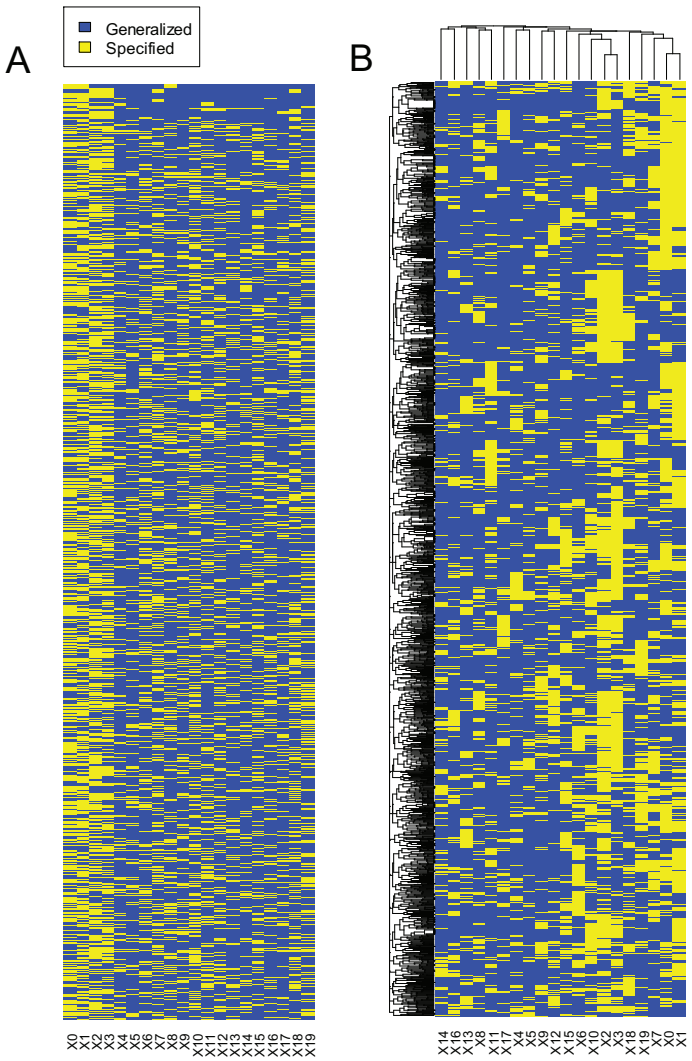

Figure 4.1: Heat-map visualizations of the evolved M-LCS rule population. (A) illustrates the raw rule population after encoding and expansion. Each row in the heat-map is 1 of 2000 rules comprising the population. Each column is one of the 20 attributes. Four of these (X0, X1, X2, and X3) were modeled as predictive attributes. (B) illustrates the same population after hierarchical clustering on both axes. According to the attribute dendrogram, each pair of interacting attributes (X0,X1) and (X2,X3), modeled in this data, cluster together best of all attributes. Note that all four predictive attributes do not cluster together, but instead a heterogeneous pattern emerges within the rule population.

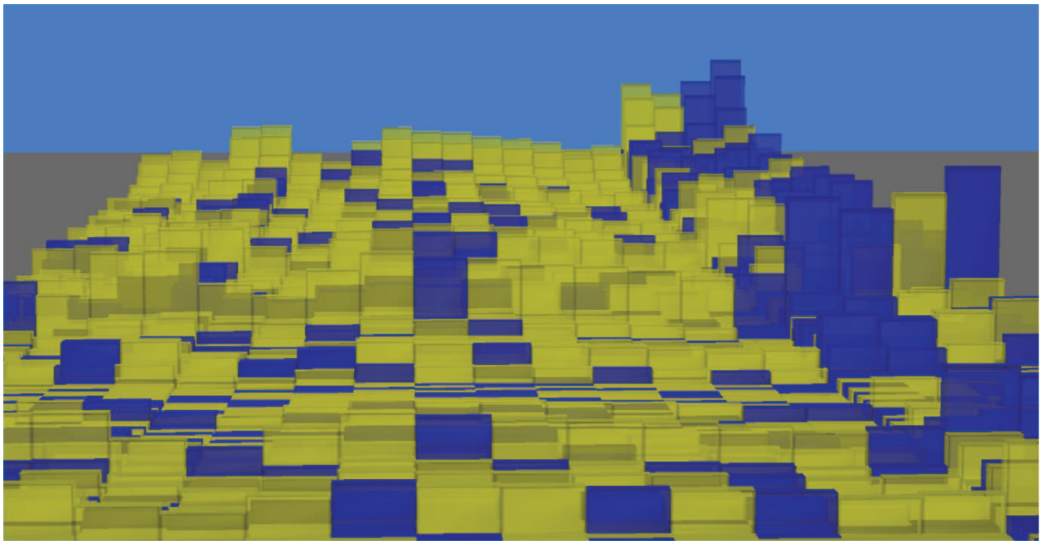

Figure 4.2: 3D visualization for the identification of interesting rules. In this figure, specified attributes are blue, while '#' attributes are yellow. The height of attributes within each row/rule is the product of the SpS for that attribute and the numerosity of the rule.

perspective of the rule population, this software could also be used to facilitate the identification of particularly interesting rules from the rule population by adding rule parameters such as class, accuracy, fitness, numerosity, and/or action set size as potential dimensions of the visualization. Users can assign 3 spatial dimensions along with bins and color (top and sides of bars). Figure 4.2 illustrates this 3D visualization, and offers a simple example of how these extra dimensions may be utilized.

4.4.4 Visualization - Co-Occurrence Network

This section details step 7. The network visualization may be generated using either the M-LCS run from step 5, or from the 10 CV runs from step 1. We use the 10 CV runs, since it includes statistical analysis and summation over 10 CV runs. To generate our co-occurrence network we use Gephi (http://gephi.org/) an open source graph visualization software. Using the 190 CoSs calculated above, we generate an adjacency matrix in a format consistent with Gephi requirements. Using Gephi, we generate a fully connected, undirected network, where nodes represent individual attributes, the diameter of a node is the SpS for that attribute, edges represent co-occurrence, and the thickness of an edge is the respective CoS. Additionally, Gephi offers a built-in function to filter edges from the network based on edge weight. Users can progressively filter out edges representing smaller CoSs in order to identify/highlight dominant patterns.

Figure 4.3 gives the co-occurrence network illustrating the 190 CoSs calculated in section 4.4.2. Filtering the network is a largely subjective process in which the investigator can isolate dominant patterns. In Figure 4.3, images A through C show the network with a progressively more stringent edge weight filter. In each, the strong co-occurrence between (X0,X1) as well as between (X2,X3) can be observed, illustrating the interaction between these respective attribute pairs. In addition, the diameter of the nodes draws attention to the same four predictive attribute. Figure 4.3C accurately captures the underlying relationships modeled in this simulated

dataset. Specifically we observe interaction between pairs (X0,X1) and (X2,X3) but independence between the pairs, as would be expected given the embedded heterogeneity.

4.4.5 Guided Knowledge Discovery

This section details step 8. The final step in our analysis pipeline involves the merging of all empirical and subjective observations made thus far in order to characterize useful rule generalizations, and generate hypotheses for further investigation. Thus far, we have (1) successfully identified our four predictive attributes, significantly differentiated them from "noise" attributes, (2) identified strong, significant co-occurrence between both epistatic attribute pairs, but observed a dramatic drop in co-occurrence for other pair-wise combinations of the four predictive attributes (evidence against a higher order interaction), (3) observed separate clustering of the significant attribute pairs in the 2D heat-map (evidence of heterogeneity), and (4) observed two dominant attribute co-occurrence pairs (evidence of interaction within each pair, and heterogeneity between them). Together these observations correctly reflect the predictive attributes, interactions, and heterogeneity embedded in this complex simulated data.

This global rule population analysis may be supplemented by, or used to direct traditional rule interpretation wherein the relationship between class and specific attribute states is sought. Alternatively this pipeline can guide hypothesis generation

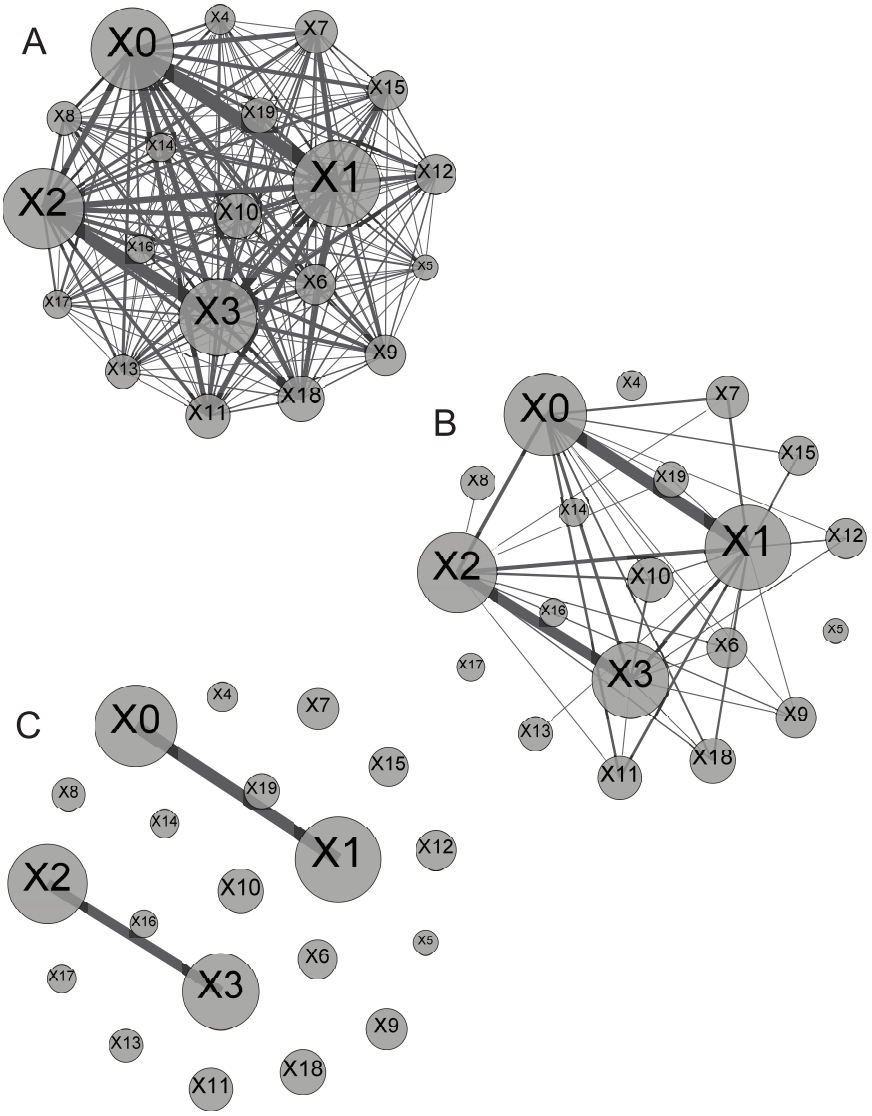

Figure 4.3: Co-occurrence networks. (A) Illustrates the fully connected network before any filtering is applied. The diameter of a node is the SpS for that attribute, edges represent co-occurrence, and the thickness of an edge is the respective CoS. (B) The network after filtering out all CoSs that did not meet the significance cutoff. (C) The network after filtering out all but to the two highest CoSs.

and further investigation, such as the confirmation of heterogeneity. Also, this pipeline could be expanded, using the first pass as a filter to identify significant attributes. A subsequent analysis would consider only those attributes found to be significant.

4.4.6 Negative Control Analysis

As mentioned in section 4.3 we repeated the analysis described above on a dataset with no predictive attributes for comparison. As expected, this analysis confirmed that the average testing accuracy of 0.502 was not significant ($p = 0.497$). Without a significant testing accuracy, subsequent analysis of SpS, AWSpS and CoSs is meaningless. However for the purposes of this study we report on these analyses for reference. SpS and AWSpS analyses each identified a single attribute (X13) as significant (p-values of 0.034 and 0.048 respectively) using a one-tailed test. However, with a two-tailed test, none of the attributes make the significance cutoff. CoS analysis identified 16 out of 190 attribute pairs as significant. However, the magnitude of these CoSs are well below even the smallest of SpSs in this secondary analysis (approximately 3x smaller). This represents a successful negative control for this analysis pipeline.

4.5 Conclusions

While M-LCS algorithms are inherently powerful, flexible, and adaptive, the complexity and ambiguity of the solutions they evolve are obvious deterrents for application to data mining tasks. Often the population of rules evolved by an M-LCS has been compared to a "black box", alluding to the incomprehensible internal workings that separate data input from accurate classification. This study introduces a complete analysis pipeline for global knowledge discovery from M-LCS rule populations. We combine significance testing, with novel test statistics, and visualization strategies to overcome the black box nature of M-LCS solutions. Using a simulation study embedded with a complex genetic association problem (epistasis and heterogeneity), we discriminate between predictive and non-predictive attributes, characterize attribute interaction, and make observations suggesting underlying heterogeneity in the dataset. To the best of our knowledge this is the first investigation of M-LCS in which statistical significance has directly guided knowledge extraction in the rule population.

While this pipeline offers a promising pathway for interpretation, this type of analysis is limited by substantial computational expense in running the M-LCS algorithm along with the necessary permutations and cross validation. Given current computational technology, this makes analysis of large-scale genetic datasets impractical. M-LCSs and the analysis pipeline introduced here would be better suited to candidate studies with a refined set of potentially predictive attributes. In future studies

we plan to apply this pipeline to real world genetic association analysis in search of novel gene-disease relationships and evidence of heterogeneity in study samples. Beyond UCS and the bioinformatic problem considered in this work, this pipeline has the potential for application to any M-LCS tasked with a single step data mining problem.

Acknowledgments

We thank Douglas Hill for his expertise in managing the 3D heat-map software and Christian Darabos for his expertise in network visualizations using Gephi. This work was supported by NIH grants AI59694, LM009012, and LM010098.

4.6 Bibliography

[1] Urbanowicz R, Moore J (2009) Learning classifier systems: a complete introduction, review, and roadmap. Journal of Artificial Evolution and Applications 2009: 1.

[2] Butz M, Lanzi P, Wilson S (2008) Function approximation with xcs: Hyperellipsoidal conditions, recursive least squares, and compaction. Evolutionary Computation, IEEE Transactions on 12: 355–376.

[3] Shi L, Shi Y, Gao Y (2009) Clustering with xcs and agglomerative rule merg-

ing. In: Proceedings of the 10th international conference on Intelligent data engineering and automated learning. Springer-Verlag, pp. 242–250.

[4] Howard G, Bull L, Lanzi P (2008) Self-adaptive constructivism in neural xcs and xcsf. In: Proceedings of the 10th annual conference on Genetic and evolutionary computation. ACM, pp. 1389–1396.

[5] Bacardit J, Burke E, Krasnogor N (2009) Improving the scalability of rule-based evolutionary learning. Memetic Computing 1: 55–67.

[6] Urbanowicz R, Moore J (2010) The application of michigan-style learning classifier systems to address genetic heterogeneity and epistasis in association studies. In: Proceedings of the 12th annual conference on Genetic and evolutionary computation. ACM, pp. 195–202.

[7] Urbanowicz R, Moore J (2011) The application of pittsburgh-style learning classifier systems to address genetic heterogeneity and epistasis in association studies. Parallel Problem Solving from Nature–PPSN XI : 404–413.

[8] Bacardit J, Butz M (2007) Data mining in learning classifier systems: comparing xcs with gassist. Learning Classifier Systems : 282–290.

[9] Fayyad U, Piatetsky-Shapiro G, Smyth P, Uthurusamy R (1996) Advances in knowledge discovery and data mining .

[10] Llorà X, Reddy R, Matesic B, Bhargava R (2007) Towards better than human capability in diagnosing prostate cancer using infrared spectroscopic imaging. In: Proceedings of the 9th annual conference on Genetic and evolutionary computation. ACM, pp. 2098–2105.

[11] Bacardit J, Garrell J (2007) Bloat control and generalization pressure using the minimum description length principle for a pittsburgh approach learning classifier system. Learning Classifier Systems : 59–79.

[12] Wilson S (2002) Compact rulesets from xcsi. Advances in Learning Classifier Systems : 65–92.

[13] Fu C, Davis L (2002) A modified classifier system compaction algorithm. In: Proceedings of the Genetic and Evolutionary Computation Conference. Morgan Kaufmann Publishers Inc., pp. 920–925.

[14] Dixon P, Corne D, Oates M (2003) A ruleset reduction algorithm for the xcs learning classifier system. Learning Classifier Systems : 20–29.

[15] Gao Y, Huang J, Wu L (2007) Learning classifier system ensemble and compact rule set. Connection Science 19.

[16] Kovacs T (1997) Xcs classifier system reliably evolves accurate, complete and minimal representations for boolean functions. Cognitive Science Research Papers - University of Birmingham CSRP .

[17] Lanzi P (2001) Mining interesting knowledge from data with the xcs classifier system. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001). pp. 7–11.

[18] Butz M, Lanzi P, Llorà X, Goldberg D (2004) Knowledge extraction and problem structure identification in xcs. In: Parallel Problem Solving from Nature-PPSN VIII. Springer, pp. 1051–1060.

[19] Kharbat F, Bull L, Odeh M (2007) Mining breast cancer data with xcs. In: Proceedings of the 9th annual conference on Genetic and evolutionary computation. ACM, pp. 2066–2073.

[20] Kharbat F, Odeh M, Bull L (2007) New approach for extracting knowledge from the xcs learning classifier system. International Journal of Hybrid Intelligent Systems 4: 49–62.

[21] Kharbat F, Odeh M, Bull L (2008) Knowledge discovery from medical data: an empirical study with xcs. Learning Classifier Systems in Data Mining : 93–121.

[22] Seifert C, Lex E (2009) A novel visualization approach for data-mining-related classification. In: Information Visualisation, 2009 13th International Conference. IEEE, pp. 490–495.

[23] Butz M (2007) Documentation of xcsfjava 1.1 plus visualization. MEDAL Report 2007008.

[24] Smith R, Jiang M (2009) Milcs in protein structure prediction with default hierarchies. In: Proceedings of the first ACM/SIGEVO Summit on Genetic and Evolutionary Computation. ACM, pp. 953–956.

[25] Urbanowicz R, Moore J (2009) Learning Classifier Systems: A Complete Introduction, Review, and Roadmap. Journal of Artificial Evolution and Applications 2009.

[26] Bernadó-Mansilla E, Garrell-Guiu J (2003) Accuracy-based learning classifier systems: models, analysis and applications to classification tasks. Evolutionary Computation 11: 209–238.

[27] Wilson S (1995) Classifier fitness based on accuracy. Evolutionary computation 3: 149–175.

[28] Orriols-Puig A, Bernadó-Mansilla E (2008) Revisiting ucs: Description, fitness sharing, and comparison with xcs. Learning Classifier Systems : 96–116.

[29] Shriner D, Vaughan L, Padilla M, et al. (2007) Problems with genome-wide association studies. Science 316: 1840c.

[30] Eichler E, Flint J, Gibson G, Kong A, Leal S, et al. (2010) Missing heritability and strategies for finding the underlying causes of complex disease. Nature Reviews Genetics 11: 446–450.

[31] Thornton-Wells T, Moore J, Haines J (2004) Genetics, statistics and human disease: analytical retooling for complexity. TRENDS in Genetics 20: 640–647.

[32] Cordell H (2002) Epistasis: what it means, what it doesn't mean, and statistical methods to detect it in humans. Human Molecular Genetics 11: 2463.

[33] Ritchie M, Hahn L, Moore J (2003) Power of multifactor dimensionality reduction for detecting gene-gene interactions in the presence of genotyping error, missing data, phenocopy, and genetic heterogeneity. Genetic epidemiology 24: 150–157.

[34] Moore J, Gilbert J, Tsai C, Chiang F, Holden T, et al. (2006) A flexible computational framework for detecting, characterizing, and interpreting statistical patterns of epistasis in genetic studies of human disease susceptibility. Journal of theoretical biology 241: 252–261.

[35] Wilson S (1998) Generalization in the XCS classifier system .

[36] Bacardit J, Butz M (2004) Data mining in learning classifier systems: comparing xcs with gassist. Urbana 51: 61801.

[37] Moore J, LARI R, HILL D, Hibberd P, Madan J (2011) Human microbiome visualization using 3d technology. In: Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing. p. 154.

Chapter 5

Instance-Linked Attribute Tracking and Feedback for Michigan-Style Supervised Learning Classifier Systems

Abstract

The application of learning classifier systems (LCSs) to classification and data mining in genetic association studies has been the target of previous work. Recent efforts have focused on: (1) correctly discriminating between predictive and non-predictive attributes, and (2) detecting and characterizing epistasis (attribute interaction) and

heterogeneity. While the solutions evolved by Michigan-style LCSs (M-LCSs) are conceptually well suited to address these phenomena, the explicit characterization of heterogeneity remains a particular challenge. In this study we introduce attribute tracking, a mechanism akin to memory, for supervised learning in M-LCSs. Given a finite training set, a vector of accuracy scores is maintained for each instance in the data. Post-training, we apply these scores to characterize patterns of association in the dataset. Additionally we introduce attribute feedback to the mutation and crossover mechanisms, probabilistically directing rule generalization based on an instance's tracking scores. We find that attribute tracking combined with clustering and visualization facilitates the characterization of epistasis and heterogeneity while uniquely linking individual instances in the dataset to etiologically heterogeneous subgroups. Moreover, these analyses demonstrate that attribute feedback significantly improves test accuracy, efficient generalization, run time, and the power to discriminate between predictive and non-predictive attributes in the presence of heterogeneity.

5.1 Introduction

Despite the rising quality and abundance of genetic data, epidemiologists continue to struggle with the connection of complex disease phenotypes to reliable genetic and environmental markers. While strategies seeking single locus associations (i.e. main effects) are often sufficient to address diseases that follow Mendelian patterns of inheritance, their application to diseases characterized as complex has yielded limited

success [1, 2]. Epistasis and heterogeneity have been recognized as phenomena that complicate the epidemiological mapping of genotype to phenotype [3]. The quality and availability of genetic data makes these phenomena accessible and attractive targets for bioinformatic development.

The term *epistasis* was coined to describe a genetic "masking" effect viewed as a multi-locus extension of the dominance phenomenon, where a variant at one locus prevents the variant at another locus from manifesting its effect [4]. In the present context, epistasis simply refers to attribute interaction. *Heterogeneity*, referring to either *genetic* heterogeneity (locus and allelic) or *environmental* heterogeneity, occurs when individual (or sets of) attributes are independently predictive of the same phenotype (i.e. class). While the detection and modeling of epistasis has received a great deal of attention [5–7], methods for dealing with heterogeneity are lagging behind, clinging to a traditional epidemiological paradigm that seeks to find a single best model of disease within a given data set [3].

Learning classifier systems (LCSs) [8] are a rule-based class of algorithms that combine machine learning with evolutionary computing and other heuristics to produce an adaptive system. LCSs represent solutions as sets of rules affording them the ability to learn iteratively, form niches, and adapt. These characteristics make the application of LCSs to the problem of heterogeneity, in particular, intrinsically appealing.

As proof of principle, Michigan and Pittsburgh-style LCSs were applied to the

detection and modeling of simulated epistatic and heterogeneous genetic disease associations [9, 10]. These evaluations identified the strengths and weaknesses of using either a Michigan or Pittsburgh-style LCS on these types of complex, noisy problems. Ultimately these analyses suggested that Michigan-style LCSs (M-LCSs), may be best suited to these particular problems. However, knowledge discovery in M-LCSs remained problematic. In [11] an analysis pipeline with visualization-guided knowledge discovery was introduced to address this obstacle. In addition to significance testing, the pipeline presented subjective strategies for inferring patterns of attribute interaction and heterogeneity from the rule-set. Still lacking, was a strategy for explicitly identifying heterogeneity and linking instances in the dataset to respective heterogeneous subgroups. The present study addresses this dilemma with the introduction of *attribute tracking*.

The concept of adding memory to LCSs has been previously explored in [12–14]. However, this form of memory was temporary and designed to improve performance on non-Markov, multi-step problems by preserving limited information regarding a previous state. Alternatively, attribute tracking is a collective form of memory designed to be applied to single-step supervised learning problems. Essentially this mechanism learns which attributes are most important to the accurate classification of each individual instance, storing and accumulating this knowledge independent of the rule population. This mechanism is driven by incremental learning and therefore is specific to M-LCSs.

In addition, this study introduces *attribute feedback* a heuristic that draws upon the knowledge learned in attribute tracking to promote efficient generalization and improve learning in the presence of noisy, complex, and heterogeneous data. Attribute feedback probabilistically directs generalization pressures in the genetic algorithm (GA) based on relative attribute tracking scores. Specifically, we implement attribute feedback into both mutation and uniform crossover mechanisms. This is in line with efforts in LCS research to target attributes most likely to be relevant to a given process. For instance [15] sped up XCS by ordering the matching process by attribute specificity. [16] later extended this idea to an attribute list knowledge representation in a Pittsburgh-style LCS (P-LCS).

The present study implements both attribute tracking and attribute feedback evaluating them over a variety of simulated datasets embedded with patterns of attribute interaction and heterogeneity. The remainder of the paper is organized as follows. Methods for the implementation and evaluation of attribute tracking and feedback are given in section 5.2. The results with discussion is given in section 5.3. Conclusions are drawn and future work is discussed in section 5.4.

5.2 Methods

In this section we describe (1) the LCS algorithm and parameters used in this investigation, (2) attribute tracking, (3) attribute feedback, and (4) our experimental evaluation of the proposed mechanisms.

5.2.1 M-LCS

The M-LCS is characterized by a population of rules with a GA operating at the level of individual rules. The evolved solution is represented by the entire rule population. M-LCSs, often varying widely from version to version, generally possess four basic components; (1) a population of rules or classifiers, (2) a performance component that assesses how well the population of rules collectively explain the data, (3) a reinforcement component that distributes the rewards for correct prediction to each of the rules in the population, and (4) a discovery component that uses different operators to discover new rules and improve existing ones. Learning progresses iteratively, relying on the performance and reinforcement components to drive the discovery of better rules. For a complete LCS introduction and review, see [8].

UCS, or the sUpervised Classifier System [17], is a M-LCS based largely on the very successful XCS algorithm [18], but replaces reinforcement learning with supervised learning, encouraging the formation of best action maps (rule sets of efficient generalizations) and altering the way in which accuracy (and thus fitness) is computed. UCS was designed specifically to address single-step problem domains such as classification and data mining, where delayed reward is not a concern, and showed particular promise when applied to attribute interaction and heterogeneity in [9].

For evaluation purposes we implement attribute tracking and feedback into a python encoding of the UCS algorithm [9]. We refer to UCS with attribute tracking as (AT-UCS) and when attribute feedback is added we refer to it as (AF-UCS). We

utilize mostly default run parameters with the exception of 200,000 learning iterations, a population size of 1600, tournament selection, uniform crossover, subsumption, attribute mutation probability = 0.04, crossover probability = 0.8, and $\nu = 1$. ν has been described as a "constant set by the user that determines the strength [of] pressure toward accurate classifiers" [19], and is typically set to 10 by default. A low ν was used to place less emphasis on high accuracy in this type of noisy problem domain, where 100% accuracy is only indicative of over-fitting. While we run each algorithm for a maximum of 200,000 iterations, we stop and evaluate the systems after 10,000, 50,000, and 100,000 as well. Also, as in [9], we employ a quaternary rule representation, where for each SNP attribute, a rule can specify genotype as (0, 1, or 2), or instead generalize with "#", a character that implies that the rule doesn't care about the state of that particular attribute. The implementation described above is available on request (*ryanurbanowicz@gmail.com*) and will be posted on the LCS and GBML Central webpage.

5.2.2 Attribute Tracking

Attribute tracking is stored as a vector of summed accuracy scores that accumulate over learning iterations. Each instance in the dataset has it's own unique vector with length equal to the number of attributes. Therefore AT-UCS will maintain a matrix of attribute tracking scores with the same dimensions as the training data (*attTrack*). Tracking scores for a given instance are only updated during learning

iterations in which that instance are called. Following the typical formation of the match and correct sets (described in [17]), the rules of the correct set are passed to the environment where the attribute tracking vectors are stored. For each rule in the correct set we do the following: (1) check each attribute in the rule to see if it has been specified (i.e. not '#'), (2) if so, update the attribute tracking score for that attribute by adding the accuracy of the rule. Rule accuracy is simply the number of correct classifications made by the rule, divided by it's experience (i.e. the number of matches it's made). Pseudo-code for attribute tracking is given in Algorithm 1.

After learning is complete, normalization, clustering, and visualization can be applied to attribute tracking scores. In noisy data, some instances will be easier to classify than others. These instances will tend to accumulate higher attribute tracking scores making comparisons between instances more challenging. To overcome this, we normalize attribute tracking scores within each instance such that they lie between 0 and 1. Normalization is achieved here by dividing each tracking score by the sum of all tracking scores for that instance.

Our visualization strategy begins with agglomerative hierarchical clustering of the attribute tracking matrix in R. Clustering is performed on both axes (i.e. across instances and attributes). Next, we visualize the clustered scores in a heat-map. Section 5.3.1 will present examples of such visualizations also completed in R.

Algorithm 1 Attribute Tracking

Require: $trainData, maxIter, rulePop, attTrack$

for $iteration \in maxIter$ **do**

$instance \leftarrow trainData[iteration\%\ \text{len}(trainData)]$

$[M] \leftarrow rulePop.matches$

$[C] \leftarrow [M].correctClass$

for all $rule \in [C]$ **do**

for all $att \in rule$ **do**

if att**not** $=$ "#" **then**

$attTrack[instance][att] += rule.accuracy$

end if

end for

end for

run GA on [C]

end for

return $rulePop, attTrack$

5.2.3 Attribute Feedback

Attribute tracking, described above, can be run either on it's own or in concert with attribute feedback. Attribute feedback converts the attribute tracking scores into probabilities that in turn are used to direct the GA mechanisms. The premise behind attribute feedback, is to apply collective experience, specific to a given instance, to pressure the GA into focusing on attributes likely to be valuable for accurate classification, and avoid specifying attributes that are not. Ideally this will promote efficient generalization, and reduce over-fitting. Additionally, since this pressure is specific to each instance it should promote niching and maintain relevant diversity in the rule population.

Another benefit to attribute feedback involves the persistence of the accumulated experience. Typically, knowledge learned by an LCS is contained exclusively in the rule population. Since it is possible for good rules to be deleted by chance, there is the potential for knowledge loss that in turn can slow down learning. By including a form of persistent memory we can, in some part, preserve experiential knowledge of the rule population that was lost in deletion.

Preliminary analysis indicated that employment of attribute feedback early in the learning process yielded significantly less success. This makes sense since the knowledge accumulated by attribute tracking early on is likely to be poor. Therefore we implement attribute feedback such that at a given iteration it is utilized with probability equal to the proportion of completed algorithm iterations ($pComplete$).

Thus early in learning, it is employed sparingly, while later it is employed frequently.

We implement attribute feedback into both mutation and uniform crossover. Once either mechanism is called we first decide whether to use the original mechanism, or the respective mechanism with attribute feedback. As mentioned, we utilized attribute feedback with probability = $pComplete$. If attribute feedback is to be used we begin by calculating the specification probability (P_spec) for each attribute in the instance. We obtain these probabilities by transforming the current vector of tracking scores for that instance. First, we subtract the minimum tracking score from all scores in the vector. Next, we find the maximum score ($maxT$) in this new vector and divide each score by $maxT + maxT * 0.01$ giving us the desired probabilities. We add 1% of $maxT$ to the denominator to ensure that the attribute with the highest track score has less than a 100% P_spec. However, the attribute with the lowest track score will inevitably have a 0% P_spec. The choice of 0.01 here is arbitrary and could potentially be varied by the user as a run parameter. If the instance has not yet been involved in a learning iteration ($maxT = 0$) the P_spec for each attribute is set at 50%.

5.2.3.1 Mutation

The original mutation mechanism permutes attributes in a rule condition ($cond$) to either specify the instance's state, or generalizes it (i.e. add a '#'/don't care symbol). The probability of mutation is equal for each attribute in the condition ($pMut$).

Alternatively with attribute feedback, if the attribute is currently generalized, then we only flip to the specified state with a probability of P_spec. Here, attributes learned by the attribute tracking to be less important to classification are encouraged to remain generalized, and vise-versa. If the attribute is currently specified, then we only flip to '#' with a probability of $1 - P\_spec$. Here, attributes learned by attribute tracking to be more important to classification are encouraged to remain specified, and vise-versa. Pseudo-code for mutation attribute feedback is given in Algorithm 2.

5.2.3.2 Uniform Crossover

The goal of uniform crossover with attribute feedback is to produce an offspring that takes the best from both parents (Schwarz), and put what's left in the other offspring (DeVito) [20]. Again, crossover can only swap the specification of an attribute with generalization. Therefore, for a given attribute, parent rules either agree to specify, agree to generalize, or differ. When they differ, the Schwarz offspring receives the specified attribute with a probability of P_spec. Here, attributes learned by the attribute tracking to be important to classification tend to be specified in Schwarz and, by default, generalized in DeVito. The opposite is true for attributes with respectively lower attribute tracking scores. Pseudo-code for uniform crossover with attribute feedback is given in Algorithm 3.

Algorithm 2 Mutation Attribute Feedback

Require: $cond, instance, attTr, pMut, pComplete$

if $random < pComplete$ **then**
  $minT \leftarrow \min(attTr)$
  for all $T\_Val \in attTr$ **do**
    $T\_Val \leftarrow T\_Val - minT$
  end for
  $maxT \leftarrow \max(attTr)$
  for all $att \in cond$ **do**
    $P\_spec \leftarrow 0$
    if $random < pMut$ **then**
      if $maxT = 0$ **then**
        $P\_spec \leftarrow 0.5$
      else
        $P\_spec \leftarrow attTr[att]/maxT + maxT * 0.01$
      end if
      if $cond[att] =$ "#"**and**$random < P\_spec$ **then**
        $cond[att] \leftarrow instance[att]$
      else if $cond[att]$**not** = "#"**and**$random > P\_spec$ **then**
        $cond[att] \leftarrow$ "#"
      else
        no mutation
      end if
    end if
  end for
else
  run mutation without attribute feedback
end if
return *cond*

Algorithm 3 Uniform Crossover Attribute Feedback

Require: $p1Cond, p2Cond, instance, attTr, pComplete$

if $random < pComplete$ **then**
  $minT \leftarrow \min(attTr)$
  for all $T\_ValinattTr$ **do**
    $T\_Val \leftarrow T\_Val - minT$
  end for
  $maxT \leftarrow \max(attTr)$
  $Schwarz \leftarrow p1Cond$
  $DeVito \leftarrow p2Cond$
  for all $att \in \text{len}(instance)$ **do**
    if $maxT = 0$ **then**
      $P\_spec \leftarrow 0.5$
    else
      $P\_spec \leftarrow attTr[att]/maxT + maxT * 0.01$
    end if
    if $Schwarz[att]$**not** $= DeVito[att]$ **then**
      if $Schwarz[att]$**not** $=$ "#"**and**$random > P\_spec$ **then**
        swap $Schwarz[att]$ with $DeVito[att]$
      end if
      if $Schwarz[att] =$ "#"**and**$random < P\_spec$ **then**
        swap $Schwarz[att]$ with $DeVito[att]$
      end if
    end if
  end for
else
  run uniform crossover without attribute feedback
end if
return $Schwarz, Devito$

5.2.4 Evaluation

We evaluate attribute feedback using simulated datasets that concurrently model heterogeneity and epistasis as they might appear in a SNP gene association study of common complex disease [9,10]. All data sets were generated using a pair of distinct, two-locus epistatic interaction models, both utilized to generate instances (i.e. case and control individuals) within a respective subset of each final data set. Each two-locus epistatic model was simulated without Mendelian/main effects, as a penetrance table using [21,22]. Due to the computational demands of LCSs, this study limited its evaluation to 3 heterogeneity/epistasis model combinations. For simplicity the minor allele frequency of each predictive attribute was set to 0.2, a reasonable assumption for a common complex disease SNP. The three model combinations included a pair of models with a heritability of either (0.1, 0.2, or 0.4). We considered model architectural "difficulties" of both "easy" and "hard" [22]. Balanced datasets simulated from these models were generated as having four different sample sizes (200, 400, 800, or 1600) and a heterogeneous mix ratio of either (50:50 or 75:25) (e.g. 75% of instances were generated from one epistatic model, and 25% were generated from a different one). Notably, our dataset simulation strategy allows for chance model overlap in instances of the simulated dataset (i.e. by chance a given instance may correctly represent both underlying models. 20 replicates of each dataset were analyzed and 10-fold cross validation (CV) was employed to measure average testing accuracy and account for over-fitting. Together, a total of 48 data set configurations (*3 Model*

Combos x *4 Sample Sizes* x *2 Ratios* x *2 Difficulties*), and a total of 960 data sets (20 random seeds each) were simulated. With 10-fold CV, 9600 runs of UCS, AT-UCS, and AF-UCS were completed.

For each run we track the following statistics; training accuracy, testing accuracy, generality, macro population size, the power to find both underlying models, the power to find at least one underlying model, the power to correctly rank attribute co-occurrence [11], and run time. Power is a reflection of our ability to reliably mine knowledge from the evolved rule population. Each of these values represent an average over the 10 CV runs.

Statistical comparisons between UCS and AF-UCS were made using the Wilcoxon signed-rank tests due to a lack of normality in the value distributions. All statistical evaluations were completed using R. Comparisons were considered to be significant at $p \leq 0.05$.

5.3 Results and Discussion

5.3.1 Attribute Tracking

As an example of attribute tracking we train AT-UCS on two of the aforementioned datasets. Both have underlying epistatic models with heritability = 0.4 and architecture = "easy". The first model includes attributes X0 and X1, and the second includes attributes X2 and X3. Each dataset has sample size = 1600. However, one

dataset has a heterogeneous mix ratio of 50:50, and the other 75:25. First we focus on the 50:50 dataset in which both underlying models are equally represented in the dataset. As described in [11], for the purposes of visualization we train AT-UCS on the entire dataset (no CV).

Figure 5.1 illustrates the raw attribute tracking scores accumulated for each instance over the course of 200,000 learning iterations. Even before normalization, the first branch of the y-axis heirarchy divides the instances into two major subgroups, each characterized by a different pair of high scoring attributes, i.e (X0, X1) and (X2,X3). Specifically the top half of instances generally correspond with strong tracking scores for both X0 and X1, while the bottom half generally correspond with X2 and X3. Through the accumulated experience of attribute tracking scores, we can see the instance heterogeneity that has been embedded in this dataset. Also note that in-line with the characteristics of this simulated dataset, there appears to be a roughly equal representation of the two heterogeneous groups.

Figure 5.2 illustrates the same attribute tracking scores given in Figure 5.1 after normalization. With normalization, an attribute's tracking score is given with respect to the other scores in a instance as opposed to all scores in the dataset. Normalization removes overall magnitude bias from clustering, allows us to better compare attribute patterns between instances. We would expect some instances to obtain higher overall attribute tracking magnitudes during training, since certain niches of the underlying problem may be solved sooner than others (by chance or because they were easier

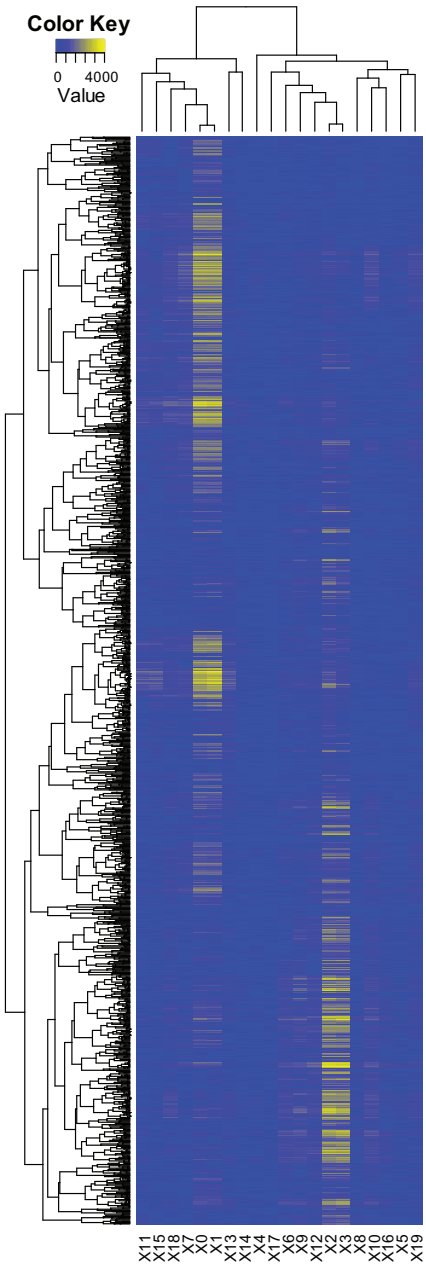

Figure 5.1: Heat-map of raw AT-UCS attribute tracking scores for a dataset with a 50:50 ratio of heterogeneity. Each row in the heat-map is 1 of 1600 instances comprising the training data. Each column is one of 20 attributes in that dataset (X0, X1, X2, and X3 are predictive). Yellow indicates higher tracking scores, while blue indicates lower ones.

to find). When this occurs, instances representative of that niche will match rules with higher accuracy for a greater number of epochs (i.e. cycles through the dataset), resulting in a larger accumulation of attribute tracking scores. Note how patterns of instance heterogeneity and attribute interaction observed in Figure 5.1 become more apparent after normalization in Figure 5.2. [11] employed visualizations of evolved M-LCS rule populations to facilitate knowledge discovery. Given that the patterns recognized in Figure 5.2 quite accurately reflect the heterogeneity and interactions modeled in the given dataset, we propose that similar analysis and visualization of attribute tracking scores can also serve to guide knowledge discovery in M-LCS data mining.

As a secondary example, Figure 5.3 illustrates normalized attribute tracking scores for the dataset with an underlying heterogeneous instance ratio of 75:25. In this analysis, we see that roughly 3/4 of the top instances show a pattern of strong tracking scores for X0 and X1 (corresponding to the 75%). Differently, we see that roughly the bottom 1/4 of instances cluster together with moderately strong tracking scores for X2 and X3 (corresponding to the 25%). It is worth noting that because all our models have a heritability < 1, we expect a certain number of instances to be unreliably classified based on the given set of attributes. Therefore we expect a certain amount of noise in our attribute tracking scores.

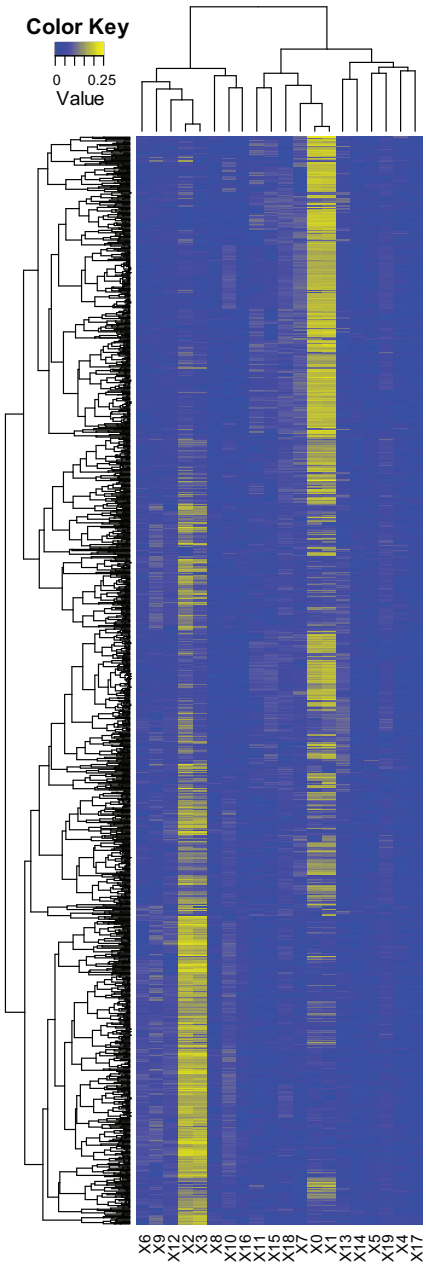

Figure 5.2: Heat-map of normalized AT-UCS attribute tracking scores for a dataset with a 50:50 ratio of heterogeneity. Each row in the heat-map is 1 of 1600 instances comprising the training data. Each column is one of 20 attributes in that dataset (X0, X1, X2, and X3 are predictive). Yellow indicates higher normalized tracking scores, while blue indicates lower ones.

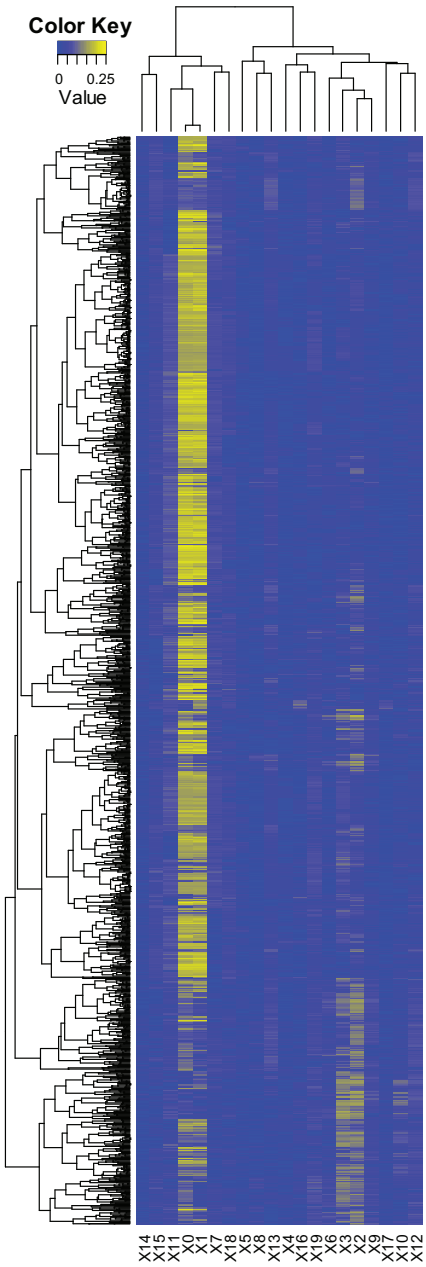

Figure 5.3: Heat-map of normalized AT-UCS attribute tracking scores for a dataset with a 75:25 ratio of heterogeneity. Each row in the heat-map is 1 of 1600 instances comprising the training data. Each column is one of 20 attributes in that dataset (X0, X1, X2, and X3 are predictive). Yellow indicates higher normalized tracking scores, while blue indicates lower ones.

Table 5.1: Comparing AF-UCS with UCS statistics after 200,000 learning iterations over all simulated datasets.

Statistics	UCS	AF-UCS	p-value
Train Accuracy	0.8544	0.8446	**
Test Accuracy	0.6061	0.6104	**
Both Power	0.3094	0.3750	**
Single Power	0.7167	0.7010	*
Co-Occurrence Power	0.2448	0.2833	**
Generality	0.7137	0.7672	**
Macro Population	1317.00	1070.83	**
Run Time (min)	36.17	33.53	**

* $p < 0.05$

** $p << 0.001$

5.3.2 Attribute Feedback

Table 5.1 summarizes each of the evaluation statistics tracked for either AF-UCS or UCS (as the control) across all simulated datasets considered in this study. Generally speaking the addition of attribute feedback to UCS led to a small but significant decrease in training accuracy, and a similarly small but significant increase in testing accuracy. Taken together this implies that AF-UCS is, to a small extent, resisting over-fitting, better than UCS alone.

As described in [11] we employ three power estimation statistics. For each, an Accuracy Weighted Specificity Sum (AWSpS) is used to determine successful identification of the modeled attributes. Both Power (BP) refers to the proportion of dataset replicates, within which we successfully identified the predictive attributes from both heterogeneously simulated epistatic models. Single Power (SP) refers to the proportion of dataset replicates, within which we idendified the predictive attributes from at least on of the two epistatic models. Co-occurrence Power (CP), detailed in [11], considers the pair-wise attribute co-occurrence within rules of the population as a means of characterizing attribute interactions. Here CP refers to the proportion of dataset replicates, within which our two modeled attribute pairs have the top two co-occurrence scores within rules of the population. These power estimates reflect our ability to extract knowledge from a given analysis. Generally speaking, AF-UCS demonstrates a fairly dramatic and significant increase in both BP and CP at the expense of a small, less significant decrease in the overall SP.

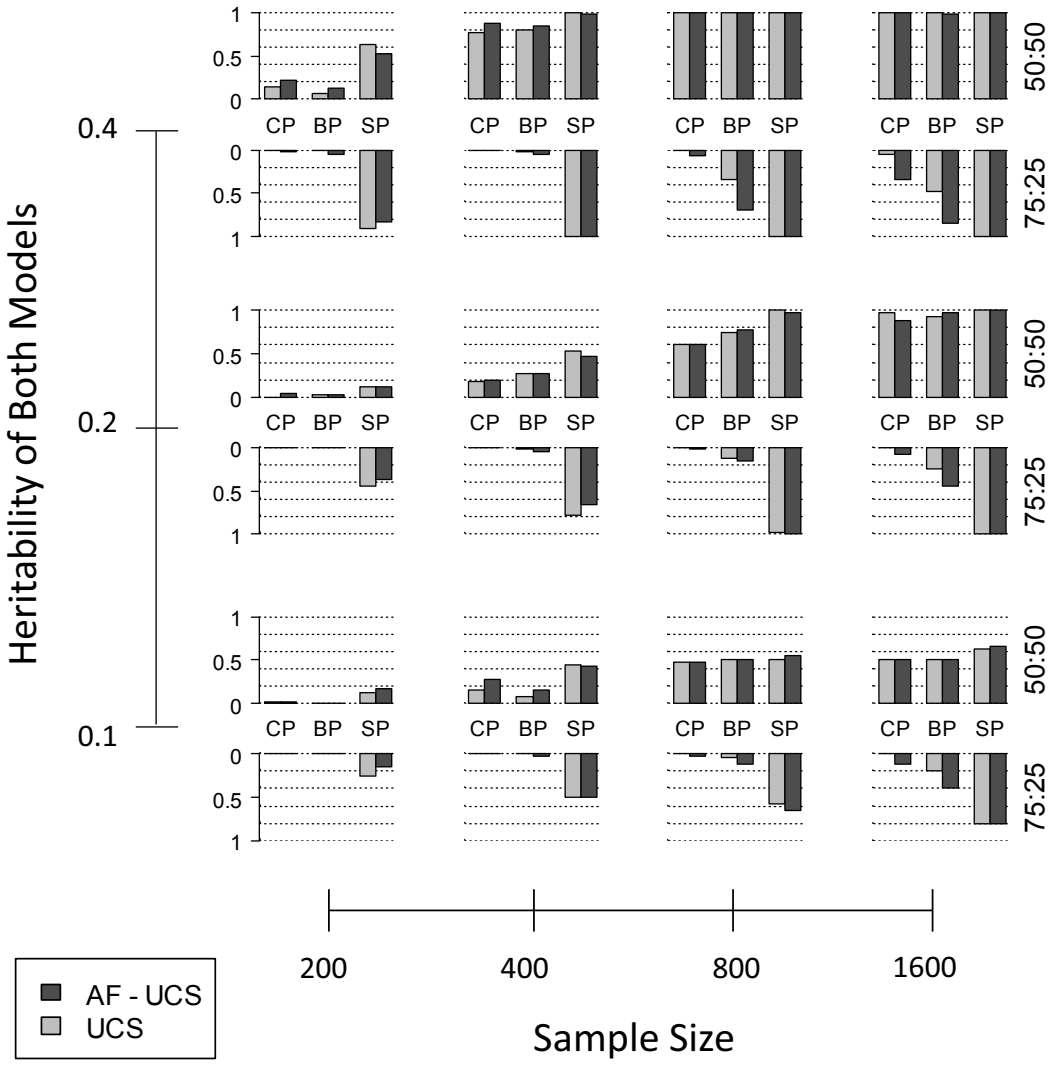

Figure 5.4: Summary of power analyses after 200,000 learning iterations. Each bar represents a power statistic estimate averaged between 'easy" and "hard" model architectures out of 20 dataset replicates. Three power statistics are presented: CP = Co-occurrence Power, BP = Both Power, and SP = Single Power. Each sub-plot corresponds to a specific combination of underlying model heritabilities and sample size. Each of six rows is labeled on the right side with the corresponding instance heterogeneity ratio.

Figure 5.4 gives a more detailed look at how attribute feedback impacts the different power estimates. From this figure we can see how attribute tracking performs with relation to UCS alone on datasets with different heritabilities, sample sizes and heterogeneous mix ratios. Of particular note, this figure suggests that AF-UCS shows the greatest BP and CP improvement over UCS when there is a disproportionate mix ratio (i.e. 75:25). AF-UCS is losing some SP when sample sizes are smaller.

Lastly, Table 5.1 indicates that AF-UCS tends to largely increase the generality of rules in the population while simultaneously reducing the macro population size in comparison to UCS. Evaluations after only 10,000 iterations reveals that AF-UCS has a similarly significant effect on generality and macro population size even when attribute feedback is only active 5% of the time (i.e. 10,000 / 200,000). Additionally, a welcome side-effect of increased rule generality, we observed a small but significant reduction in run time.

AF-UCS also impacts the visualizations described in section 5.3.1. Specifically, attribute feedback promotes higher rule generality that, in turn, yields a visualization with less noise, making the true attribute patterns that much clearer to detect (not shown due to space constraints). This reduction in noise benefits the knowledge discovery process making the integration of attribute tracking and feedback a beneficial combination.

5.4 Conclusions and Future Work

The present study introduces attribute tracking as a mechanism for explicitly characterizing heterogeneity and interaction in M-LCSs applied to supervised learning problems. Attribute tracking may be used to guide knowledge discovery, or cluster instances into heterogeneous subsets for further analysis. This study also introduces attribute feedback, a secondary mechanism that naturally pairs with attribute tracking. Exploiting attribute tracking as a form of persistent experiential memory, attribute feedback adds an efficient generalization pressure to mutation and uniform crossover. Evaluation over a spectrum of simulated datasets indicates that attribute feedback can significantly increase testing accuracy, generality and the power to find all predictive attributes in the data, while decreasing macro population size, run time, and the power to find at least one interacting pair of predictive attributes.

Attribute tracking and feedback are likely to be valuable in complex, noisy, data mining problems such as the one described in this study, especially when the goal of analysis is the thorough characterization of underlying patterns of interaction and heterogeneity. Unfortunately, M-LCS algorithms do not typically scale well, and the analyses presented in this study are computationally expensive. This makes large-scale analyses (examining large numbers of attributes) impractical given our current implementation. P-LCS strategies such as GAssist [23] and BioHel [24] offer an alternative to M-LCS data mining with the advantage of better scalability.

However, novel strategies for M-LCS knowledge discovery, such as those intro-

duced in [25], [11] and the present study, generate new opportunity for data mining in M-LCSs. In future works we hope to optimize the implementation of AF-UCS, considering a number of possible alternatives including (1) an adaptive frequency of attribute feedback utilization, (2) the incorporation of punishment into attribute tracking for rules that match but do not make it into the correct set, and (3) the integration of this work with other advancements in the LCS field, including efficient matching, parallelization, and flexible rule representation. Ultimately we will combine the AF-UCS algorithm with the analysis pipeline described in [11] for the analysis of complex genetic and environmental association in human disease.

Acknowledgments

This work was supported by the William H. Neukom 1964 Institute for Computational Science at Dartmouth College along with NIH grants AI59694, LM009012, and LM010098.

5.5 Bibliography

[1] Shriner D, Vaughan L, Padilla M, et al. (2007) Problems with genome-wide association studies. Science 316: 1840c.

[2] Eichler E, Flint J, Gibson G, Kong A, Leal S, et al. (2010) Missing heritability and

strategies for finding the underlying causes of complex disease. Nature Reviews Genetics 11: 446–450.

[3] Thornton-Wells T, Moore J, Haines J (2004) Genetics, statistics and human disease: analytical retooling for complexity. TRENDS in Genetics 20: 640–647.

[4] Bateson W (1909) Mendel's Principles of Heredity. Cambridge Univ .

[5] Cordell H (2002) Epistasis: what it means, what it doesn't mean, and statistical methods to detect it in humans. Human Molecular Genetics 11: 2463.

[6] Ritchie M, Hahn L, Moore J (2003) Power of mdr for detecting gene-gene interactions in the presence of genotyping error, missing data, phenocopy, and genetic heterogeneity. Genetic epidemiology 24: 150–157.

[7] Moore J, Gilbert J, Tsai C, Chiang F, Holden T, et al. (2006) A flexible computational framework for detecting, characterizing, and interpreting statistical patterns of epistasis in genetic studies of human disease susceptibility. Journal of theoretical biology 241: 252–261.

[8] Urbanowicz R, Moore J (2009) Learning Classifier Systems: A Complete Introduction, Review, and Roadmap. Journal of Artificial Evolution and Applications .

[9] Urbanowicz R, Moore J (2010) The application of michigan-style learning classifier systems to address genetic heterogeneity and epistasis in association studies.

In: Proceedings of the 12th annual conference on Genetic and evolutionary computation. ACM, pp. 195–202.

[10] Urbanowicz R, Moore J (2011) The application of pittsburgh-style learning classifier systems to address genetic heterogeneity and epistasis in association studies. Parallel Problem Solving from Nature–PPSN XI : 404–413.

[11] Urbanowicz R, Granizo-Mackenzie A, Moore J (Submitted) An Analysis Pipeline with Visualization-Guided Knowledge Discovery for Michigan-Style Learning Classifier Systems. Evolutionary Intelligence SI: Advancements in Learning Classifier Systems .

[12] Cliff D, Ross S (1994) Adding temporary memory to zcs. Adaptive Behavior 3: 101–150.

[13] Lanzi P (1998) Adding memory to xcs. In: Evolutionary Computation Proceedings, IEEE World Congress on Computational Intelligence. pp. 609–614.

[14] Lanzi P, Wilson S (2000) Toward optimal classifier system performance in non-markov environments. Evolutionary Computation 8: 393–418.

[15] Butz B, Lanzi P, Llora X, Loiacono D (2008) An analysis of matching in lcss. In: Proceedings of the 10th annual conference on Genetic and evolutionary computation. ACM, pp. 1349–1356.

[16] Bacardit J, Burke E, Krasnogor N (2009) Improving the scalability of rule-based evolutionary learning. Memetic Computing 1: 55–67.

[17] Bernadó-Mansilla E, Garrell-Guiu J (2003) Accuracy-based learning classifier systems: models, analysis and applications to classification tasks. Evolutionary Computation 11: 209–238.

[18] Wilson S (1995) Classifier fitness based on accuracy. Evolutionary computation 3: 149–175.

[19] Orriols-Puig A, Bernadó-Mansilla E (2008) Revisiting ucs: Description, fitness sharing, and comparison with xcs. Learning Classifier Systems : 96–116.

[20] Twins. Dir. Ivan Ritman. Perfs. Arnold Schwarzenegger and Danny DeVito 1988.

[21] Urbanowicz R, Kiralis J, Sinnott-Armstrong N, Heberling T, Fisher J, et al. (Submitted) GAMETES: A Fast, Direct Algorithm for Generating Pure, Strict, Epistatic Models with Random Architectures. BMC Bioinformatics .

[22] Urbanowicz R, Kiralis J, Fisher J, Moore J (Submitted) Predicting Difficulty in Simulated Genetic Models: Metrics for Model Architecture Selection. BMC Bioinformatics .

[23] Bacardit J (2004) Pittsburgh genetics-based machine learning in the data mining era: Representations, generalization, and run-time. PhD disertation .

[24] Bacardit J, Stout M, Hirst J, Sastry K, Llorà X, et al. (2007) Automated alphabet reduction method with evolutionary algorithms for protein structure prediction. In: Proceedings of the 9th annual conference on Genetic and evolutionary computation. ACM, pp. 346–353.

[25] Kharbat F, Odeh M, Bull L (2008) Knowledge discovery from medical data: an empirical study with xcs. LCSs in Data Mining : 93–121.

Chapter 6

Characterizing Heterogeneity and Epistasis in Bladder Cancer Susceptibility: A Michigan-Style Learning Classifier Approach

Abstract

Detecting complex patterns of association between genetic and environmental risk factors and disease risk has become an important target for bioinformatic and epidemiological research. In particular, strategies that accommodate multifactor interactions or heterogeneous patterns of association can offer new insights in association studies

wherein traditional analytic tools have proven unsuccessful. In an effort to concurrently address these phenomena, previous work has successfully considered the application of learning classifier systems (LCSs), a flexible class of evolutionary algorithms that distributes learned associations over a population of rules. Subsequent work addressed the inherent problems of knowledge discovery and interpretation within these algorithms by introducing a pipeline that combined statistical and subjective strategies for global rule analysis. Additionally, the characterization of heterogeneity within instances trained by the LCS has been explicitly addressed with the incorporation of an attribute tracking mechanism. While these previous advancements were evaluated in the context of complex simulation studies, the present study applies these collective works to a real world investigation of bladder cancer susceptibility. Using this LCS approach, we replicate the identification of previously characterized factors that modify bladder cancer risk: i.e. SNPs from a DNA repair gene (XPD), and smoking. Furthermore, we apply attribute tracking to identify potentially heterogeneous groups of subjects from the dataset. Statistical analysis comparing clinical outcome variables between these groups identified a significant, meaningful difference in survivorship between the cases comprising the two largest groups, even after correcting for age of diagnosis. A marginally significant difference in time to recurrence was also noted. These results support the hypothesis that an LCS approach can offer greater insight into complex patterns of association, allowing for the detection, characterization, and modeling of both epistasis and heterogeneity.

6.1 Introduction

A major goal for epidemiologists is the identification of genetic and environmental factors, predictive of common complex diseases (e.g. cancer). Traditional Mendelian (single-gene) approaches, such as those typically adopted in genome-wide association studies (GWAS), have yielded limited success when applied to most common diseases, at best identifying common variants that contribute only modestly to a given phenotype [1, 2]. Perceived problems, common to these analyses, such as the lack of reproducibility [3, 4], the observation of "missing heritability" [2, 5, 6], and the sheer number of candidate factors identified, are all suggestive of a complex pattern of association. Complexity references the number of factors involved, the influence of interaction (e.g. additive or epistatic), and the confounding of heterogeneity. From an evolutionary perspective, these types of complex disease associations would be expected as the logical byproduct of canalization and the accumulation of cryptic genetic variation [7].

The term *epistasis* was coined to describe a genetic "masking" effect viewed as a multi-locus extension of the dominance phenomenon, where a variant at one locus prevents the variant at another locus from manifesting its effect [8]. This type of interaction effect could plausibly occur between sets of genetic or environmental factors. To date, the detection and modeling of epistasis and general interaction effects, has received a great deal of attention. Random forests [9], multi-factor dimensionality reduction (MDR) [10], DICE [11], combinatorial partitioning method, [12], logistic

regression [13], patterning and recursive partitioning [14] and Bayesian pathway modeling [15] represent just a handful of available strategies.

By comparison, strategies that accommodate heterogeneity are in the minority. The meaning of the term *heterogeneity* is context dependent. In the context of admixture, heterogeneity simply refers to genetic differences in population structure [16]. In genetic modeling, a heterogeneous model describes the independent effect of some number of factors [17]. Similarly, in the context of association studies, heterogeneity references an independence effect observed in three different phenomena: *allelic heterogeneity*, *locus heterogeneity*, and *phenocopy* [18]. Allelic heterogeneity occurs when two or more alleles of a single locus are independently associated with the same trait, while locus heterogeneity occurs when two or more DNA sequence variations at distinct loci are independently associated with the same trait. Heterogeneity, referring to either *genetic heterogeneity* (locus and allelic) or *environmental heterogeneity* (i.e. phenocopy), occurs when individual (or sets of) factors are independently predictive of the same phenotype. Additionally, *trait heterogeneity* occurs when a trait or disease has been defined with insufficient specificity such that it is actually two or more distinct underlying traits [18]. Ultimately in the context of data mining genetic and environmental factors within an association study, there is no practical distinction between genetic heterogeneity, trait heterogeneity, and phenocopy, since these phenomena manifest the same type of independent associations. From a computer science prospective, the problem of heterogeneity is similar to a latent or "hidden"

class problem. While the disease status (case or control) of each patient is already known, the individuals making up either class would be more accurately sub-grouped into two or more "hidden" classes, each characterized by an independent predictive model of disease.

As mentioned, there are far fewer successful strategies for dealing with the heterogeneity problem. Most strategies that address epistasis, neglect to consider the impact of heterogeneity. An exception to this is seen in an evaluation of MDR that demonstrated that simulated heterogeneity dramatically hinders MDR's power to detect all underlying modeled factors [19]. Statistical approaches such as the admixture test [20], M test [21], and β test [22] are specific to family-based data and can only identify the existence of heterogeneity rather than characterize it. The most common approach to heterogeneity is to try and remove its confounding effect via data stratification. This has been addressed using strategies such as ordered subset analysis [23], latent class analysis [24, 25], tree-based recursive partitioning [26], and clustering [27, 28]. These methods pre-process the dataset based on genetic risk factors, demographic data, phenotypic data, or endophenotypes in order to form more homogeneous subsets of subjects. This is in line with the standard epidemiological paradigm that seeks to find a single best disease model within a given homogeneous sample. The obvious drawback of these methods is that their success completely relies on the availability, quality, and relevance of these covariates. Additionally, stratification represents a relative reduction in sample size, leading to an inevitable loss in

power to detect associations within these homogeneous subsets.

Only a few strategies have been considered that concurrently examine the problems of epistasis and heterogeneity without resorting to some form of stratification. These include MDR [19, 29], random forests [30], association rule discovery [31], and CHAMBER [32]. While some algorithms have been successful in accommodating the problem of heterogeneity, explicit characterization has remained a major challenge. CHAMBER, an algorithm that uses graph-building is uniquely the first to jointly consider the characterization of both interaction and heterogeneity. Specifically, heterogeneity is characterized through the identification of groups of individuals, within which, different predictive attributes are correlated with disease risk.

Given the apparent complexity of common disease, and the likelihood that multilocus interactions and heterogeneity are not only present but likely to be ubiquitous components of disease risk [33, 34], it is critical to develop powerful new strategies that concurrently address these phenomena. Strategies that make assumptions about the nature, number, or source of underlying factors will inevitably be susceptible to complicating phenomena.

Learning classifier systems (LCSs) [35] are a rule-based class of algorithms that combine machine learning with evolutionary computing and other heuristics to produce an adaptive system. LCSs represent solutions as sets of rules, affording them the ability to learn iteratively, form niches, and adapt. This class of algorithm breaks from the traditional single model paradigm by evolving a solution comprised of mul-

tiple rules, consequently avoiding a need for covariates and data stratification. These characteristics make the application of LCSs to the problem of heterogeneity, in particular, intrinsically appealing.

Previously, we have explored the application of different LCS algorithms to the detection and modeling of simulated epistatic and heterogeneous genetic disease associations and demonstrated their ability to successfully detect predictive factors in the presence of heterogeneity, and an extreme, precisely defined form of epistasis referred to as *pure epistasis* [36, 37]. A purely epistatic interaction occurs between n loci that do not display any main effects [38, 39]. These proof of principle analyses identified Michigan-style LCSs (M-LCSs) to be the most promising implementation for our particularly complex, noisy problem domain. However, knowledge discovery in M-LCSs remained problematic. In [40] we developed an analysis pipeline with visualization-guided knowledge discovery to address this obstacle. In addition to significance testing, the pipeline presented subjective visualization strategies for inferring patterns of attribute interaction and heterogeneity from the rule-set. Still lacking, was a strategy for explicitly identifying heterogeneity and linking instances in the dataset to respective heterogeneous subgroups. In [41] we introduced attribute tracking and feedback as mechanisms to address this problem, and improve learning and generalization in the M-LCS algorithm. While the aforementioned efforts to apply LCSs to the characterization of both epistatic and heterogeneous patterns of association were evaluated over a diverse spectrum of simulated datasets we have

yet to apply them to a real-world investigation of common complex disease. In the present study we apply our extended M-LCS approach to the investigation of bladder cancer susceptibility.

Like other common complex diseases, cancer is recognized as a multifactorial disease that results from complex interactions between many genetic and environmental factors. In an effort to validate the utility of our M-LCS approach we investigate the relationship between DNA repair gene SNPs, smoking, and bladder cancer susceptibility in 355 cases and 559 controls enrolled in a population-based study of bladder cancer. This dataset has been previously analyzed in [42] using a multifaceted statistical approach that included the application of logistic regression, MDR, and information theory. That study identified XPD codon 751 and 312 SNPs along with smoking as the best predictors of bladder cancer.

The present study applies AF-UCS [41], our extended M-LCS algorithm, to the same dataset in an attempt to replicate previous findings and characterize any patterns of interaction or heterogeneity that may be associated with bladder cancer risk. We demonstrate how the clustering of attribute tracking scores within instances of the dataset may be used to identify sample subsets, post-training. These subsets aim to capture heterogeneous patterns of disease associations. Additionally, we attempt to validate these subgroups with statistical comparisons examining clinical outcome variables. Specifically, we examine age of diagnosis, survivorship, age of recurrence, and tumor stage/grade between reliable patient clusters. The remainder of the paper

is organized as follows. Materials and methods are described in section 6.2. Results and discussion are given in section 6.3. Conclusions are drawn in section 6.4.

6.2 Methods and Materials

In this section we describe (1) the bladder cancer data examined in this study, (2) the M-LCS algorithm and parameters used in training the algorithm, and (3) the analytical pipeline and statistical analysis utilized in this study.

6.2.1 Dataset

6.2.1.1 Study Group

The bladder cancer dataset considered in this study was previously collected and examined in [42]. The final dataset comprised of 355 cases and 559 controls including 7 SNPs, 2 covariate factors, and smoking. Cases were collected among New Hampshire residents, ages 25-74 years, from July 1, 1994 to June 30, 1998 from the State Cancer Registry. Within 15 days of diagnosis, the state mandated rapid reporting system requires submission of an initial report of cancer, and a definitive report within 120 days. To be eligible for the study subjects were required to have a listed telephone number and speak English. Physician consent was obtained before contacting eligible bladder cancer patients. A total $n = 459$ bladder cancer cases were interviewed, which was 85% of the cases confirmed to be eligible for the study. Non-participants

included ($n = 10$) whose physician denied patient contact, ($n = 63$) were reported as deceased by a household member or physician, ($n = 3$) no answer after 40 attempts distributed over day, evenings and weekends, ($n = 75$) declined participation, and ($n = 8$) were too ill to take part. A standardized histopathology review was conducted by the study pathologist, and from this review, eleven subjects were excluded who were initially reported to the cancer registry as bladder cancer.

All controls were selected from population lists. Controls < 65 years of age were selected using population lists obtained from the New Hampshire Department of Transportation. The file contains the names and addresses of those holding a valid driver's license for the state of New Hampshire Department of Transportation. Controls 65 years of age and older were chosen from data files provided by the Centers for Medicare and Medicaid Services (CMS) of New Hampshire. The method of control selection used in the study was successfully employed in other case-control studies conducted in the region (e.g. [43]). For efficiency, this control group was shared with a study of non-melanoma skin cancer conducted covering an overlapping diagnostic period of July 1, 1993 to June 30, 1995 [43]. Additional controls for selected for bladder cancer cases diagnosed from July 1, 1995 to June 30, 1997 frequency matched to these cases on age (25-34, 35-44, 45-54, 55-64, 65-69, 70-74 years) and gender. Controls were randomly assigned a reference date from among the diagnosis dates of the case group to whom they were matched. A total $n = 665$ controls (the total shared control group and additional controls, which was 70% of the controls confirmed to be eligible

for the study. Of the potential participants, ($n = 18$) were reported as deceased by a member of the household, ($n = 17$) no answer after 40 attempts distributed over day, evenings, and weekends, ($n = 261$) declined, ($n = 29$) were mentally incompetent or too ill to take part.

6.2.1.2 Personal Interview

Informed consent was obtained from each participant and all procedures and study materials were approved by the Committee for the Protection of Human Subjects at Dartmouth College. Consenting participants underwent a detailed in-person interview, usually at their home. Questions covered sociodemographic information(including level of education), lifestyle factors such as use of tobacco (including frequency, duration and intensity of smoking), family history of cancer and medical history prior to the diagnosis date of the bladder cancer cases or reference date assigned to controls. Recruitment procedures for both the shared controls from the non-melanoma skin cancer and additional controls were identical and ongoing concomitantly with the case interviews. Case-control status and the main objectives of the study were not disclosed to the interviewers. To ensure consistent quality of the study interviewer, interviews were tape recorded with the consent of the participants and routinely monitored by the interviewer supervisor. To assess comparability of cases and controls, we asked subjects if they currently held a driver's license or a Medicare enrollment card.

6.2.1.3 Genotyping

DNA was isolated from peripheral circulating blood lymphocyte specimens harvested at the time of interview using Qiagen genomic DNA extraction kits (QIAGEN, Valencia, CA). DNA repair genes were examined that had previously been examined in relation to bladder cancer (XRCC1, XRCC3, XPD, XPC) as well as other pathway members that physically interact with these genes (APE1), Genotyping for non-synonymous SNPs XRCC3 C/T at position 241, APE1 T/G at position 148, XPD G/A at position 312 and A/C at position 751, XRCC1 C/T at position 194 was performed by Qiagen Genomics using their SNP mass-tagging system. For XRCC1 G/A at position 399 and XPC PAT -/+, genotyping was performed by PCR-RFLP as described in [44]. Of the 1113 participating cases and controls, genotyping was performed on DNA isolated from blood on 914 (82%). For quality control purposes, laboratory personnel were blinded to case-control status. These assays achieved $> 95\%$ accuracy as assessed using negative and positive quality controls (including every 10th sample as a masked duplicate). Data were missing on 103 individuals for XRCC1 194, 70 for XRCC1 399, 2 for XRCC3, 111 for XPD 312, 53 for XPD 752, 131 for XPC and 3 for APE1.

6.2.1.4 Covariates and Clinical Variables

For the purpose of analysis, age and gender were both encoded as dichotomous variables within the dataset. Age was stratified as < 50 and ≥ 50. Smoking was

categorized into three levels (i.e. never, < 35 pack-years, ≥ 35 pack-years). The pack-year variable is calculated by multiplying the number of packs of cigarettes smoked per day by the number of years the person has smoked. The pack-year cut point was chosen based on the median number of pack-years overall. Along with the variables considered in [42], additional clinical outcome variables were recorded for the cases in this study group that are utilized in the present study. These include (1) tumor stage and grade, (2) age at diagnosis (years), (3) survival time in years (i.e. survivorship), and (4) time to first recurrence in years. Censoring occurs when we do not know the time of death or recurrence for all subjects. Censoring indicators are also included in the dataset corresponding to both survivorship and recurrence. Tumor stage and grade is included as a categorical variable with the following categories (ordered here by relative increasing severity): (1) noninvasive low grade, (2) noninvasive missing grade, (3) noninvasive high grade, (4) *in situ* tumor (TIS), (5) stage II, (6) stage III, and (7) stage IV. Noninvasive, low grade tumors are the least aggressive and have the best prognosis. Stage II tumors and higher characterize tumors that are "invading" through the bladder wall, which can then metastasize to other organs.

6.2.1.5 Summary of Previous Findings

The results of previous analyses performed in [42] are summarized here. Overall, no significant associations were identified between any single DNA repair gene SNP and bladder cancer risk. A marginally significant elevated risk was observed

in the XPD codon 751 homozygote variant among subjects who never smoked. The XRCC1 191 variant allele was associated with reduced bladder cancer risk among heavy smokers. MDR analysis identified pack-years smoking as the strongest single-factor for predicting bladder cancer risk (average testing accuracy = 0.63). The best two-factor model included XPD 751 and XPD 312 (average testing accuracy = 0.65). The best three-factor model (also the best overall model) added pack-years smoking to XPD 751 and XPD 312 (average testing accuracy = 0.66). The application of information theory suggested that the relationship between the two XPD SNPs and bladder cancer is mostly non-additive (i.e. epistatic), while the effect of smoking is mostly additive. It was noted that the two XPD SNPs were in significant linkage disequilibrium. Further analysis indicated that the combination of these SNPs into a haplotype indicated that a variant XPD haplotype was more susceptible to bladder cancer, an effect that was magnified with the inclusion of smoking. XPD stands for *xeroderma pigmentosum group D*, the gene encoding an enzyme in the nucleotide excision repair (NER) pathway. This enzyme is known to remove certain DNA crosslinks, UV photolesions, and bulky chemical adducts [45]. Interactions between XPD 312 and 751 have also been observed in relation to lung cancer risk, and several studies found that the risk of lung cancer associated with the variant allele was higher among non-smokers than among smokers [46, 47]. In the present study, analysis of the described dataset was completed without access to unique patient identifiers in accordance with IRB Human Subject Protection. Any identifiers added to track pa-

tients during learning, clustering, and comparison of clinical variables, were arbitrarily assigned for the purposes of this study.

6.2.2 M-LCS

A Michigan-style LCS (M-LCS) is characterized by a population of rules with a GA operating at the level of individual rules. The evolved solution is represented by the entire rule population. M-LCSs, often varying widely from version to version, generally possess four basic components; (1) a population of rules or classifiers, (2) a performance component that assesses how well the population of rules collectively explain the data, (3) a reinforcement component that distributes the rewards for correct prediction to each of the rules in the population, and (4) a discovery component that uses different operators to discover new rules and improve existing ones. Learning progresses iteratively, relying on the performance and reinforcement components to drive the discovery of better rules. For a complete LCS introduction and review, see [35].

UCS, or the sUpervised Classifier System [48], is a M-LCS based largely on the very successful XCS algorithm [49], but replaces reinforcement learning with supervised learning, encouraging the formation of best action maps (rule sets of efficient generalizations) and altering the way in which accuracy (and thus fitness) is computed. UCS was designed specifically to address single-step problem domains such as classification and data mining, where delayed reward is not a concern, and showed

particular promise when applied to attribute interaction and heterogeneity in [36]. Here, we use the word attribute to mean a variable such as a SNP or a demographic variable such as gender that is used to make a prediction. Attribute is commonly used this way in data mining and machine learning.

AF-UCS [41], our expansion of UCS, incorporates attribute tracking and feedback into the basic supervised learning algorithm. Attribute tracking is a collective form of memory designed to be applied to single-step supervised learning problems. Essentially this mechanism learns which attributes are most important to the accurate classification of each individual instance, storing and accumulating this knowledge independent of the rule population. This mechanism is driven by incremental learning and therefore is specific to M-LCSs. Attribute feedback is a heuristic that draws upon the knowledge learned in attribute tracking to promote efficient generalization and improve learning in the presence of noisy, complex, and heterogeneous data. Attribute feedback probabilistically directs generalization pressures in the genetic algorithm (GA) based on relative attribute tracking scores. Specifically, attribute feedback has been implemented into both mutation and uniform crossover mechanisms.

Previously AF-UCS was evaluated using only simulated datasets. In order to accommodate the nature of real-world datasets, two additional minor modifications were made to the algorithm. The first address the calculation of training and testing accuracy when the algorithm is applied to unbalanced datasets. We replace the traditional accuracy calculation with balanced accuracy as described in [50]. Also,

while the dataset examined in this study removed instances with missing data, we have implemented a simple mechanism into AF-UCS that intuitively accommodates the presence of any missing data. Essentially this mechanism treats missing data as a wild card ('#') similar to that used in an LCS rule to indicate that we "don't care" what the state of that factor is. This allows the LCS to ignore missing data during the learning process, removing the need for imputation or removal of the subject from the dataset.

We have previously implemented attribute tracking and feedback into AF-UCS, a python encoding of the UCS algorithm [36, 41]. We adopt mostly default M-LCS run parameters. Parameters unique to this study include: 200,000 learning iterations, a rule population size of 1000, tournament selection, uniform crossover, subsumption, attribute mutation probability = 0.04, crossover probability = 0.8, and a ν of 1. ν has been described as a "constant set by the user that determines the strength [of] pressure toward accurate classifiers" [51], and is typically set to 10 by default. A low ν was used to place less emphasis on high accuracy in this type of noisy problem domain, where 100% accuracy is only indicative of over-fitting. Also, as in [36], we employ a quaternary rule representation, where for each SNP attribute, a rule can specify genotype or covariates as (0, 1, or 2), or instead generalize with "#", a character that implies that the rule doesn't care about the state of that particular attribute.

Generally speaking the utilization of larger population sizes and a greater number of learning iterations can improve the resulting performance of the evolved M-LCS

rule population. The present study adopts modest values for these parameters. It is likely that optimization of these and other run parameters would further improve the performance of the AF-UCS algorithm.

6.2.3 Statistical Analyses

We adopt the analysis pipeline introduced in [40] for the identification of significant predictive attributes identified by the AF-UCS algorithm. We extend this pipeline for the analysis of real data. In summary, our analysis includes the following steps: (1) run the AF-UCS algorithm on the dataset using 10-fold cross validation (CV), (2) run a permutation test with 1000 permutations, (3) confirm significance of testing accuracy, (4) identify significant attributes and significantly co-occurring pairs of attributes, (5) train the AF-UCS algorithm on the whole dataset (no CV), (6) generate visualization for pattern characterization, (7) repeat steps 1-3 as a second pass on the dataset that only includes attributes identified as significant from the first pass, (8) identify significant, stable clusters of subjects using attribute tracking scores, and (9) compare clinical variables between identified subject clusters. While we do not claim that this analysis pipeline is necessarily optimal, we have attempted to assemble a logical series of analytical steps that rely primarily on statistically significant empirical observations. Indeed it is likely that there are other reasonable ways to approach knowledge discovery and hypothesis generation in the present context.

6.2.3.1 First Pass

Our first pass analysis of the bladder cancer data (summarized by steps 1-6) begins by running AF-UCS on the dataset using 10-fold CV strategy in order to determine average testing accuracy and account for over-fitting. The dataset is randomly partitioned into 10 equal parts and AF-UCS is run 10 separate times during which 9/10 of the data is used to train the algorithm, and a different 1/10 is set aside for testing. We average training and testing accuracies over these 10 runs. Next we set up our permutation test. We generate 1000 permuted versions of the original dataset by randomly permuting the affection status (class) of all samples, while preserving the number of cases and controls. For each permuted dataset we run UCS using 10-fold CV. In total, permutation testing requires $10,000$ runs of AF-UCS. We perform this analysis using "Discovery", a 1372 processor Linux cluster.

Next it is critical to confirm that the average testing accuracy is significantly higher than by random chance. If average testing accuracy is not significantly high, this suggests that AF-UCS was unable to learn any useful generalizations from the data, and there is little reason to progress with the rest of this analysis pathway. We utilize a typical one-tailed permutation test with a significance threshold of $p < 0.05$. Once a significant testing accuracy is confirmed we use the permutation test to identify attributes in the dataset that show significant importance in making accurate classification. As detailed in [40] we utilize the following statistics for making such an inference from the rule population: (1) Specificity Sum (SpS) and (2) Accuracy

Weighted Specificity Sum (AWSpS). Additionally we use the permutation test to evaluate attribute interactions as well as to help discriminate between interaction and heterogeneity. This is achieved by evaluating a co-occurrence statistic (CoS) that examines all pair-wise attribute co-occurrence within rules of the population [40]. We calculate CoS for every non-redundant pair-wise combination of attributes in the dataset. In this 10 attribute dataset we calculate 45 CoSs. These CoSs are also used to generate the co-occurrence network visualization in section 6.3. To generate the co-occurrence network we use Gephi (http://gephi.org/) an open source graph visualization software. Using the 45 CoSs calculated above, we generate an adjacency matrix in a format consistent with Gephi requirements. Using Gephi, we generate a fully connected, undirected network, where nodes represent individual attributes, the diameter of a node is the SpS for that attribute, edges represent co-occurrence, and the thickness of an edge is the respective CoS. Gephi offers a built-in function to filter edges from the network based on edge weight. We utilize this feature to focus on significant co-occurrence attribute pairs.

Next we train AF-UCS on the entire dataset with no CV. This is for rule population visualization purposes (i.e. the generation of a rule population heat-map) also given in section 6.3. [40] details the re-encoding of the rule-population in preparation for visualization. In the present study we employ agglomerative hierarchical clustering using hamming distance as the distance metric. Clustering is performed on both axes (i.e. across rules and attributes). Both clustering and 2D heat-map visualization

was performed in R using the *hclust* and *gplots* packages, respectively.

6.2.3.2 Second Pass

We extend this analysis to a second pass over the data that focuses exclusively on those attributes identified as significant in the first pass. We repeat steps 1-3 including CV, permutation testing and statistical evaluation of average testing accuracy. Also, we run AF-UCS on the dataset that includes only significant attributes without CV in order to obtain a matrix of attribute tracking scores for further investigation.

Attribute tracking keeps a record of which attributes were most important for accurate classification for each subject in the dataset. Therefore, by clustering subjects in the dataset by these attribute tracking scores we aim to identify homogeneous groups of subjects defined by similar patterns of attributes determined to be important to accurate classification. As discussed in [41] before clustering, we normalize attribute tracking scores within each instance such that they lie between 0 and 1. Normalization is achieved here by dividing each tracking score by the sum of all tracking scores for that instance. This allows us to compare relative attribute patterns between instances in the dataset.

Our strategy for the selection of significant stable subject clusters involves hierarchical clustering with an assessment of uncertainly. We apply *pvclust* an R package that calculates *p*-values for hierarchical clustering via multi-scale bootstrap resampling [52]. Significant clusters are identified using approximately unbiased (AU)

p-values, where AU values $> 95\%$ are deemed significant. We apply the default "average" method of agglomerative clustering, a the default distance measure of "correlation". 1000 bootstrap replications were performed in determining AU values. Once p-values calculation was complete we assigned alphabetical group IDs to unique, significant clusters. Note that clusters include both case and control subjects.

Lastly, we compare the clinical variables between cases found in respective significant subject clusters. For obvious reasons, we have clinical variable data only for cases in our study group (and not for controls). All of the following statistical evaluation were also performed in R.

First, we discuss the evaluation of the continuous clinical variables (i.e. age at diagnosis, survivorship, and time to recurrence). The non-parametric Kruskal-Wallis one-way analysis of variance is use to determine whether these clinical variables varied by cluster. Pair-wise comparisons between subject clusters were performed using the non-parametric Mann-Whitney test.

In addition, we perform "time to failure" analyses (i.e. survival analysis) for age at diagnosis, survivorship, and time to recurrence including failure curves (i.e. Kaplan-Meier curves) for each. Since both survivorship and time to recurrence data include censoring, we adopt a Cox proportional hazard regression model (*coxph*), most widely used in medical studies. Since age at diagnosis could logically impact survivorship or time to recurrence we include it as a covariate in the covariance analysis model.

We begin with a Cox proportional hazard model including the interaction between

cluster ID and age at diagnosis. Next we use the *step* function to choose the best factors to keep in the model by Akaike's Information Criterion (AIC). AIC takes into account the balance between goodness of fit, and the number of parameters included in a model. Follow up analysis of variance between the best model identified by AIC, and a similar model excluding group specification indicates whether there is a significant difference in our clinical variable based on group designation. Comparisons were considered to be significant at $p \leq 0.05$.

Since tumor stage and grade is a categorical variable, we adopted a Chi Square test of homogeneity in order to look for differences between the cases of different clusters. Noting that certain cells of the table had very few counts, we followed up with a Fisher's exact test.

6.3 Results and Discussion

6.3.1 First Pass

Following a first pass analysis of the bladder cancer dataset with AF-UCS we observe an average training accuracy of 0.6995 and a significant average testing accuracy of 0.6042 averaged over 10-fold CV ($p = 0.001$). Table 6.1 summarizes the SpS and AWSpS statistics for each factor in the dataset. Note that significantly larger SpS and AWSpS statistics were observed for XPD 751, XPD 312, and pack-years than would be expected by chance. This indicates that these attributes are predictive of

Table 6.1: SpS and AWSpS Results

Attribute	SpS	p-value	AWSpS	p-value
XPD.751	5686	0.001*	4001.86	0.001*
XPD.312	5077	0.007*	3550.64	0.001*
pack.yr	4827	0.037*	3383.68	0.006*
XRCC1.194	4206	0.461	2818.81	0.179
age.50	4151	0.721	2800.63	0.308
male	4048	0.745	2752.61	0.358
XRCC1.399	3797	0.729	2595.78	0.427
APE1	3524	0.891	2418.03	0.667
XRCC3	3172	0.989	2166.09	0.91
XPC.PAT	2975	1.0	1994.75	0.98

disease risk. This finding corresponds with the results observed using MDR in [42]. While not statistically significant, the next largest SpS or AWSpS was observed for XRCC1 194. Recall that in [42], a smoking-conditional decreased risk was observed for this SNP.

Table 6.2 summarizes significant results for the CoS statistic. 7 of the possible 45 attribute combinations were found to occur more frequently than would be expected by chance. In particular, notice that the largest CoS value was observed for the attribute combination (XPD 751 & XPD 312). This is in line with the best two-attribute model identified by MDR in [42]. Pair-wise combinations including the

Table 6.2: Significant CoS Results

Attribute Pairs		CoS & p-value	
XPD.751	XPD.312	3367	0.001*
XPD.751	pack.yr	2757	0.001*
XPD.751	XRCC1.194	2530	0.001*
XPD.312	pack.yr	2375	0.006*
XPD.751	age.50	2345	0.016*
XRCC1.194	pack.yr	2120	0.04*
XPD.312	XRCC1.194	2105	0.046*

three attributes identified as significant in the above analysis were also found to be significantly over-represented here. These include (XPD 751 & pack-year) and (XPD 312 & pack-year). However the CoS values for these pairs vary somewhat dramatically. For example the CoS for XPD 751 & XPD 312 is about 1.5 times larger than that for XPD 312 & pack-year. The relative dissimilarity of the CoSs for these pair-wise combinations does not support the existence of a clear three way interaction [40]. We also note the XPD 751 & pack-year CoS value was the second most strongly observed pair. This seems to correspond with the observation in [42] that the XPD 751 homozygote variant indicated a marginally significant increase in risk among subjects who never smoked.

Figure 6.1 offers a visualization of the rule population evolved by AF-UCS when trained on the entire bladder cancer dataset. While somewhat noisy, a couple trends

can be seen within the rule population. First, XPD 751 and XPD 312 cluster together as columns indicating a tendency for both SNPs to be specified in rules concurrently. Second, specification of pack-years does not cluster together with XPD 751 and XPD 312 suggesting an independent, and potentially heterogeneous relationship.

Figure 6.2 gives the co-occurrence network constructed using the CoS values from this analysis. Figure 6.2B essentially is a direct visualization of the results presented in Tables 6.1 and 6.2. Again we observe evidence of interaction between XPD 751 and XPD 312, and somewhat less evidence for other attribute pairs.

For comparison to the clustering performed in the next section, Figure 6.3 offers a visualization of these normalized attribute tracking scores, clustered with all 10 attributes included. A total of 82 significant, stable clusters were identified by *pvclust* in this analysis. The largest cluster included only 106 subjects, with subject counts in subsequent clusters dropping off quickly. The large number of clusters is the result of including attribute patterns that are not necessarily useful in characterizing the significant underlying associations with disease risk. In the interest of exploring an analysis pathway that focuses only on significant attributes, we perform a second pass analysis of the data that includes only attributes XPD 751, XPD 312, and pack-years. We expect that this will also help to identify clusters that include a larger number of subjects for comparison.

While we do not evaluate the clusters identified in Figure 6.3, we do notice some interesting patterns emerge. In particular, notice how the relationship between XPD

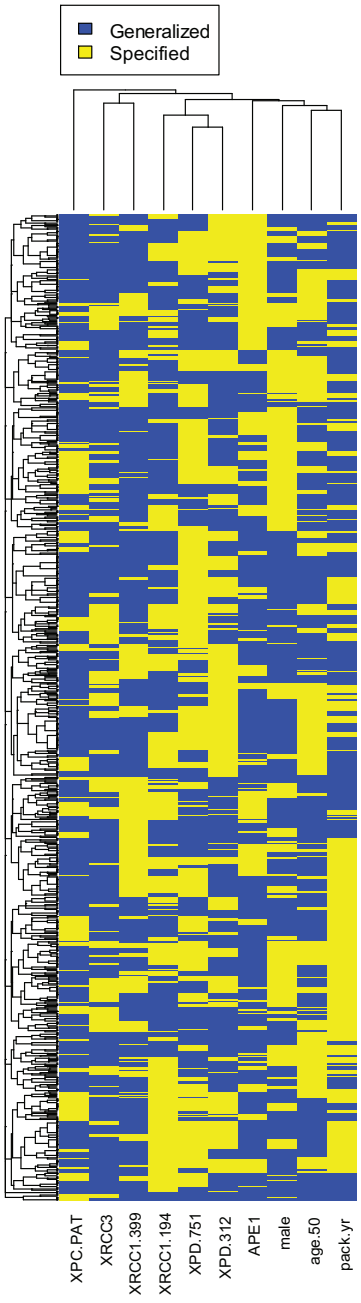

Figure 6.1: Heat-map visualization of the evolved AF-UCS rule population. Each row in the heat-map is 1 of 1000 rules comprising the population. Each column is one of the 10 attributes. Yellow indicates specification of a respective attribute within a rule, while blue indicates generalization (i.e. '#'/'don't care'. The attribute 'male' refers to gender.

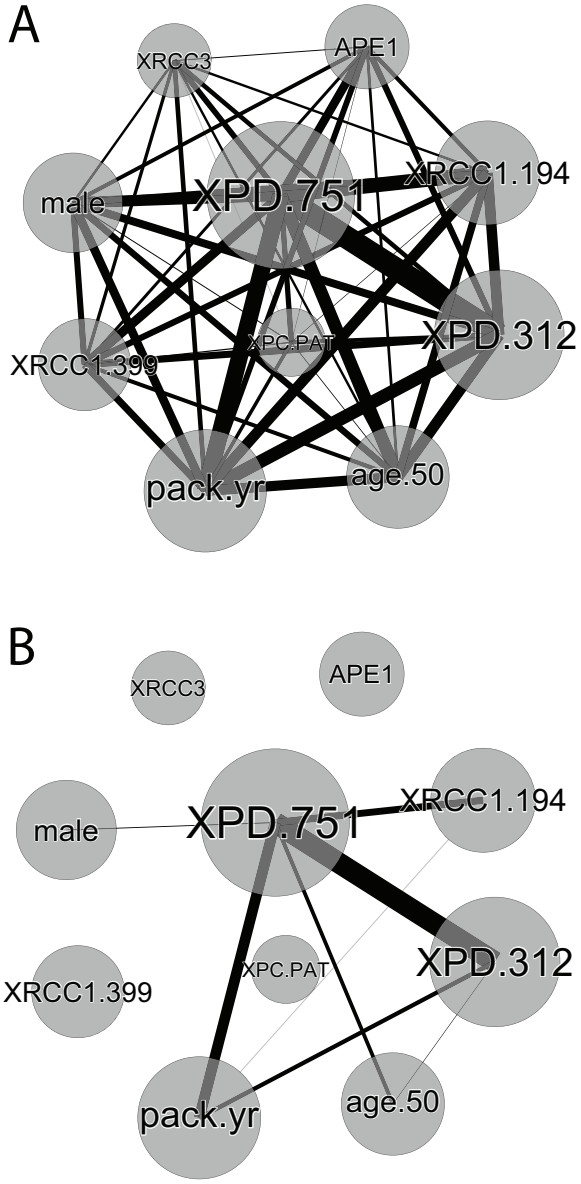

Figure 6.2: Co-occurrence network. (A) Illustrates the fully connected network before any filtering is applied. The diameter of a node is the SpS for that attribute, edges represent co-occurrence, and the thickness of an edge is the respective CoS. (B) The network after filtering out all CoSs that did not meet the significance cutoff.

751, XPD 312, and pack-years differs between different subsets of subjects. For instance subjects clustering at the top tend to have higher normalized accuracy sums for the pack-year factor, those in the middle tend to emphasize all three factors, and those at the bottom yield higher scores for the pair of significant SNPs. Some other potentially interesting patterns appear involving XRCC1 194, gender, and age. However, due to a lack of significance we will not consider these patterns further.

6.3.2 Second Pass

Following a secondary analysis of the bladder cancer dataset including only attributes (XPD 751, XPD 312, and pack-years), we observe an average training accuracy of 0.6989 and a significant average testing accuracy of 0.6968 averaged over 10-fold CV ($p = 0.001$). Recall that the MDR model including these attributes reported a testing accuracy of 0.66. We train AF-UCS on the entire 3-attribute dataset and utilize *pvclust* to identify significant, stable clusters of samples as previously described. A visualization of these clustered, normalized attribute tracking scores is given in Figure 6.4. By far, the largest two clusters are B and D. Cluster B indicates a strong pattern of high attribute tracking scores in XPD 751 and XPD 312, while Cluster D indicates a strong pattern of high attribute tracking scores for pack-years.

Now that clusters have been identified we look for differences in clinical outcome variables for the cases in these groups. Figure 6.5 summarizes differences in age at diagnosis, survival time, and time to first recurrence. Cluster G included no cases, and

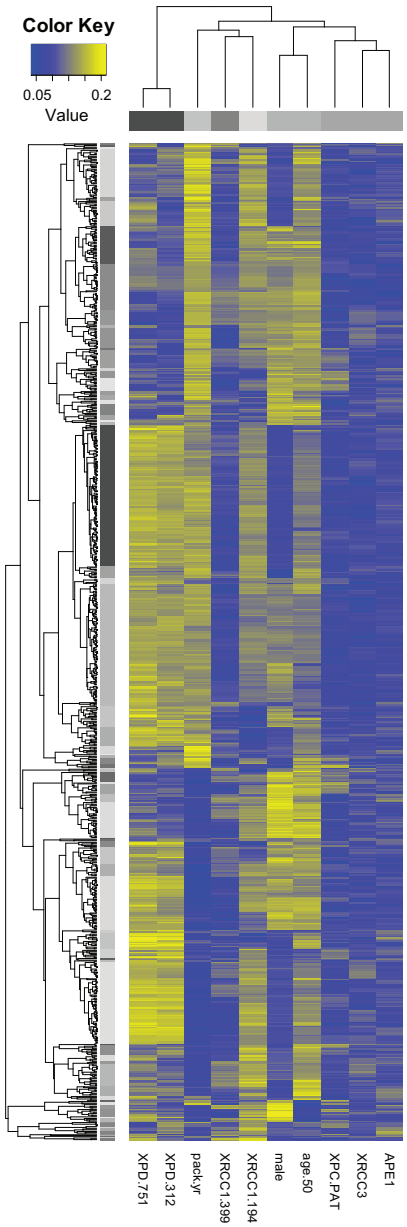

Figure 6.3: Heat-map of normalized AF-UCS attribute tracking scores for entire bladder cancer dataset (10 attributes). Each row in the heat-map is 1 of 914 instances comprising the dataset. Each column is one of 10 attributes. Yellow indicates higher normalized tracking scores, while blue indicates lower ones. The attribute 'male' refers to gender.

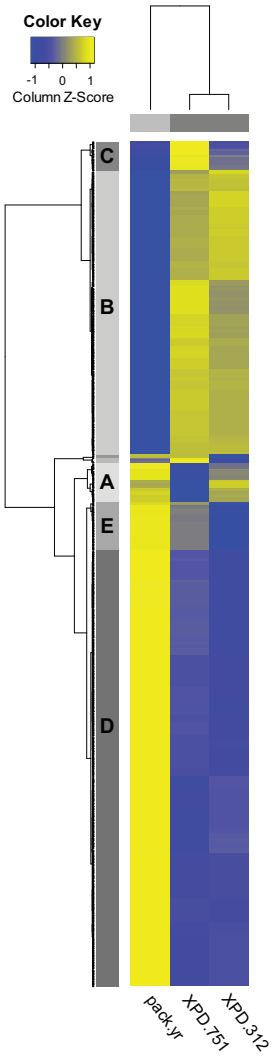

Figure 6.4: Heat-map of normalized AF-UCS attribute tracking scores for entire bladder cancer dataset (3 significant attributes). Each row in the heat-map is 1 of 914 instances comprising the dataset. Each column is one of 3 attributes. Yellow indicates higher normalized tracking scores, while blue indicates lower ones. Significant subject clusters are delineated by the blocks on the y-axis labeled alphabetically. Due to their small size, clusters F and G are not labeled, but can be seen between clusters A and B. Cluster G is adjacent to B, while cluster F is adjacent to A. In order to better highlight the attribute patterns underlying these clusters, the normalized attribute tracking scores are further scaled by instance using the *scale* feature in *pvclust*.

therefore was not included in our subsequent analysis of clinical variables. Kruskal-Wallis analysis between the cases of clusters A-F yielded a marginally significant difference for age at diagnosis ($p = 0.076$), a significant difference for survival time ($p < 0.05$), and a marginally significant difference for time to first recurrence ($p = 0.094$). For the remainder of this analysis we focus on Clusters B and D, as they are by far the largest of the subject clusters. Mann-Whitney tests comparing Clusters B and D yield a significant difference for age at diagnosis, survival time, and recurrence ($p < 0.05$).

Figures 6.6, 6.7, and 6.8 include Kaplan-Meier plots illustrating "time to failure" differences between clusters B and D. We supplement these curves by performing failure analysis for each as previously described. Examination of age of diagnosis yielded no significant difference between curves B and D as illustrated in Figure 6.6. Our examination of survivorship included age of diagnosis as a covariate. Step-wise regression analysis indicated no significant interaction between diagnosis and cluster ID. However AIC suggested that the best model included both factors as main effects. The final survival model indicated that both age of diagnosis and subject group were significant ($p < 0.05$). Follow-up analysis of variance comparing the model with and without cluster ID indicated that there was indeed a significant survivorship difference between clusters B and D ($p < 0.05$) even after correcting for age of diagnosis as a covariant. A similar analysis of recurrence indicated no significant interaction between recurrence and cluster ID. AIC again suggested that the best model included both

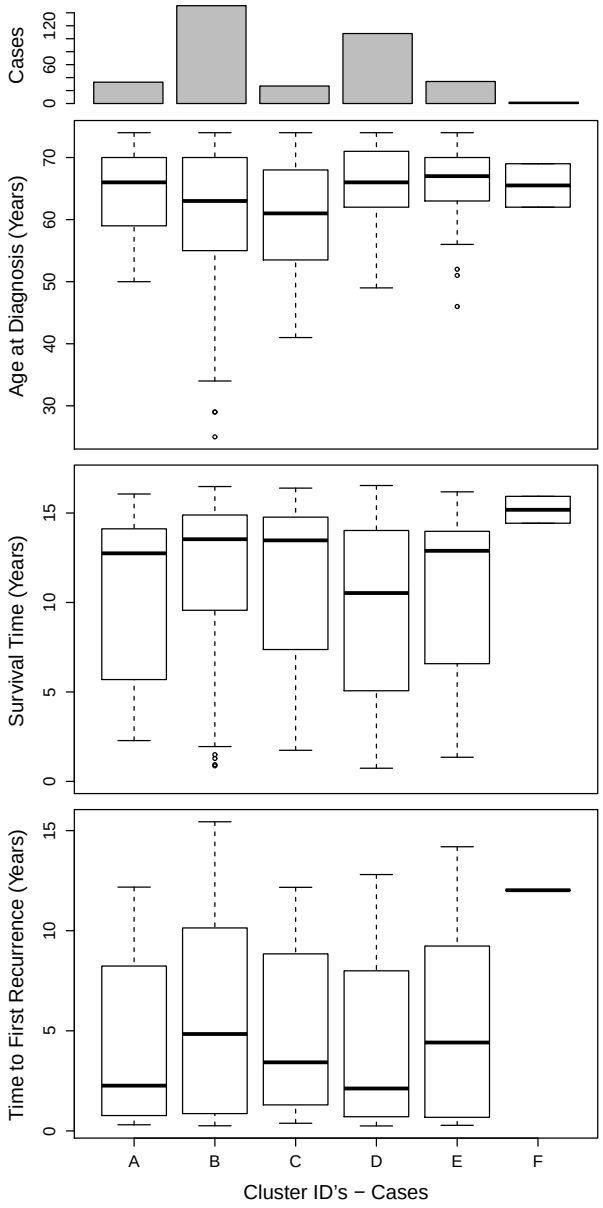

Figure 6.5: Box plots comparing age at diagnosis, survival time, and time to first recurrence between all clusters that include cases. The bars at the top give the number of cases included in each cluster.

factors as main effects. However, subsequent analysis of the final model indicated that while age of diagnosis was significant ($p < 0.05$), cluster ID was only marginally significant ($p = 0.051$). Follow-up analysis of variance comparing the model with and without cluster ID confirmed that the difference in recurrence between clusters B and D was only marginally significant after taking age of diagnosis into account ($p = 0.052$).

Lastly, we found that the data was too sparse to successfully complete either a Chi Square test or a Fisher's Exact test between all clusters and all stage/grade categories. Examination of stage/grade counts identified that the vast majority of cases included in the study were characterized as noninvasive low grade (the least severe of the categories). Comparisons of stage/grade between clusters B and D alone yielded no significant findings.

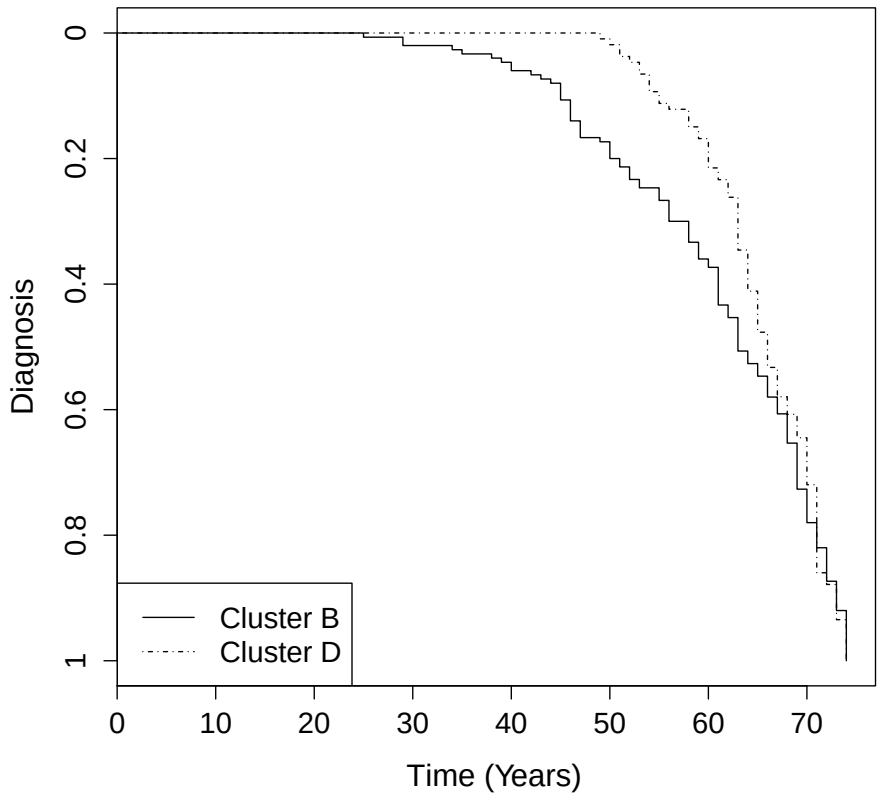

Figure 6.6: A Kaplan-Meier plot comparing age of diagnosis for clusters B and D. Lines illustrate the proportion of cases in either group that have been diagnosed with bladder cancer.

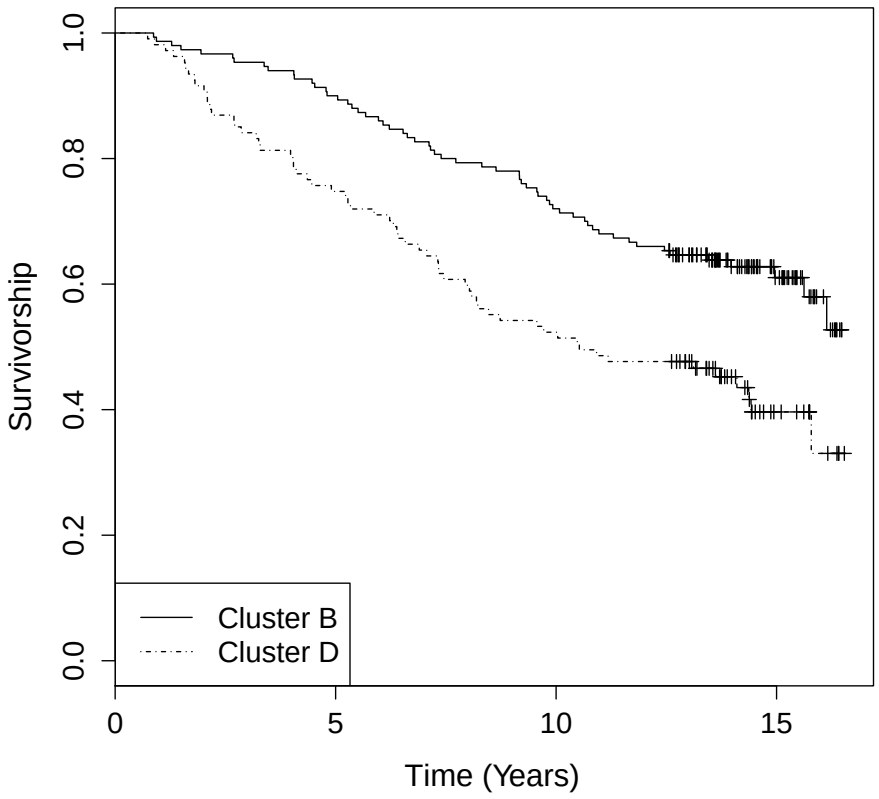

Figure 6.7: A Kaplan-Meier plot comparing survivorship for clusters B and D. Lines illustrate the proportion of surviving cases in either group. Plus signs in the curve indicate censoring.

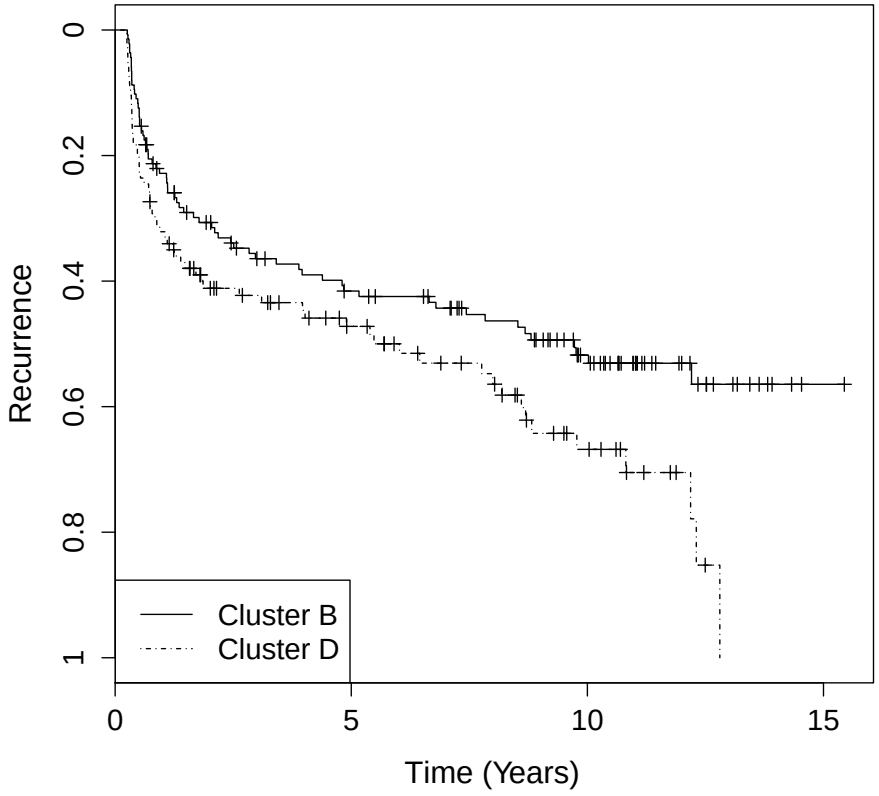

Figure 6.8: A Kaplan-Meier plot comparing age of first recurrence for clusters B and D. Lines illustrate the proportion of cases in which a recurrence of bladder cancer was observed Plus signs in the curve indicate censoring.

6.4 Conclusions

The current study explores the application of an M-LCS analysis pathway to the investigation of bladder cancer susceptibility. Our approach employs an adaptive stochastic search algorithm that makes no assumptions about the underlying patterns of association. We extend a previously described analysis pipeline for knowledge discovery and an attribute tracking strategy for the characterization of heterogeneity. This extension affords us the potential to identify etiologically heterogeneous patient subsets. This investigation successfully replicated previous findings that implicate XPD 751, XPD 312, and pack-years of smoking as being significant predictors of bladder cancer susceptibility [42]. Additionally, we are able to extend the characterization of these predictive factors, identifying evidence of interaction between XPD 751 and XPD 312, and evidence of heterogeneity between smoking and these genetic factors. We identify two large subgroups of subjects, with unique underlying patterns of attributes important to accurate classification. Statistical analyses comparing clinical phenotypes between these groups yielded a significant and dramatic difference in patient survivability, along with a marginally significant difference in recurrence time, (each after correcting for age of diagnosis). Closer inspection reveals that patients within cluster B (within which the two XPD SNPs were the most important factors for accurate classification) tended to be diagnosed earlier, displayed a significantly increased survivorship, and a marginally significant increase in time to recurrence. Alternatively, patients in cluster D (within which pack-years smoking was

the most important factor for accurate classification) tended to be diagnosed later with a significantly shorter survivorship and a marginally significant decrease in time to recurrence. While it is certainly no revelation that smoking can negatively influence patient health, we have uncovered evidence of a potentially heterogeneous smoking effect (i.e. phenocopy), not previously characterized. These findings support our claim that this proposed M-LCS analytic pathway can not only accommodate underlying heterogeneity, but can be used to characterize it. While the results presented in this study illustrate the promise of our proposed methodology, we do not claim that it has been optimized. Alternative strategies for determining patient subsets from attribute tracking scores should be considered. Also, the hypothesis generated by this work must be followed up with laboratory validation. Perceived heterogeneous patterns may be indicative of even higher order interaction, or the absence of other critically predictive factors. Simpler algorithms that can handle epistatic interactions (such as MDR) may subsequently be run independently on subject subsets to better characterize the predictive attributes involved in these ideally more homogeneous groups. This proposed strategy should benefit the generation of more targeted hypothesis and help to identify patient subgroups based directly on patterns of association as opposed to potentially inappropriate or incomplete covariate-based data stratification.

6.5 Bibliography

[1] Donnelly P (2008) Progress and challenges in genome-wide association studies in humans. Nature 456: 728–731.

[2] Manolio T, Collins F, Cox N, Goldstein D, Hindorff L, et al. (2009) Finding the missing heritability of complex diseases. Nature 461: 747–753.

[3] Kraft P, Zeggini E, Ioannidis J (2009) Replication in genome-wide association studies. Statistical science: a review journal of the Institute of Mathematical Statistics 24: 561.

[4] Greene C, Penrod N, Williams S, Moore J (2009) Failure to replicate a genetic association may provide important clues about genetic architecture. PLoS One 4: e5639.

[5] Maher B, et al. (2008) Personal genomes: The case of the missing heritability. Nature 456: 18.

[6] Eichler E, Flint J, Gibson G, Kong A, Leal S, et al. (2010) Missing heritability and strategies for finding the underlying causes of complex disease. Nature Reviews Genetics 11: 446–450.

[7] Flatt T, et al. (2005) The evolutionary genetics of canalization. The quarterly review of biology 80: 287.

[8] Bateson W (1909) Mendel's Principles of Heredity. Cambridge Univ .

[9] Pavlov Y (1997) Random forests. Probabilistic methods in discrete mathematics (Petrozavodsk, 1996) : 11–18.

[10] Ritchie M, Hahn L, Roodi N, Bailey L, Dupont W, et al. (2001) Multifactor-dimensionality reduction reveals high-order interactions among estrogen-metabolism genes in sporadic breast cancer. The American Journal of Human Genetics 69: 138–147.

[11] Tahri-Daizadeh N, Tregouet D, Nicaud V, Manuel N, Cambien F, et al. (2003) Automated detection of informative combined effects in genetic association studies of complex traits. Genome research 13: 1952–1960.

[12] Nelson M, Kardia S, Ferrell R, Sing C (2001) A combinatorial partitioning method to identify multilocus genotypic partitions that predict quantitative trait variation. Genome Research 11: 458–470.

[13] Ruczinski I, Kooperberg C, LeBlanc M (2003) Logic regression. Journal of Computational and Graphical Statistics 12: 475–511.

[14] Foulkes A, De Gruttola V, Hertogs K (2004) Combining genotype groups and recursive partitioning: an application to human immunodeficiency virus type 1 genetics data. Journal of the Royal Statistical Society: Series C (Applied Statistics) 53: 311–323.

[15] Cortessis V, Thomas D (2004) Toxicokinetic genetics: an approach to gene-environment and gene-gene interactions in complex metabolic pathways. IARC scientific publications : 127.

[16] Long J (1991) The genetic structure of admixed populations. Genetics 127: 417.

[17] Cordell H (2002) Epistasis: what it means, what it doesn't mean, and statistical methods to detect it in humans. Human Molecular Genetics 11: 2463.

[18] Thornton-Wells T, Moore J, Haines J (2004) Genetics, statistics and human disease: analytical retooling for complexity. TRENDS in Genetics 20: 640–647.

[19] Ritchie M, Hahn L, Moore J (2003) Power of multifactor dimensionality reduction for detecting gene-gene interactions in the presence of genotyping error, missing data, phenocopy, and genetic heterogeneity. Genetic epidemiology 24: 150–157.

[20] SMITH C (1963) Testing for heterogeneity of recombination fraction values in human genetics. Annals of Human Genetics 27: 175–182.

[21] Morton N (1955) Sequential tests for the detection of linkage. American Journal of Human Genetics 7: 277.

[22] Risch N (1988) A new statistical test for linkage heterogeneity. American journal of human genetics 42: 353.

[23] Schmidt S, Scott W, Postel E, Agarwal A, Hauser E, et al. (2004) Ordered

subset linkage analysis supports a susceptibility locus for age-related macular degeneration on chromosome 16p12. BMC genetics 5: 18.

[24] Shao Y, Cuccaro M, Hauser E, Raiford K, Menold M, et al. (2003) Fine mapping of autistic disorder to chromosome 15q11-q13 by use of phenotypic subtypes. The American Journal of Human Genetics 72: 539–548.

[25] Fenger M, Linneberg A, Werge T, Jørgensen T (2008) Analysis of heterogeneity and epistasis in physiological mixed populations by combined structural equation modelling and latent class analysis. BMC genetics 9: 43.

[26] Shannon W, Province M, Rao D (2001) Tree-based recursive partitioning methods for subdividing sibpairs into relatively more homogeneous subgroups. Genetic epidemiology 20: 293–306.

[27] Thornton-Wells T, Moore J, Haines J (2006) Dissecting trait heterogeneity: a comparison of three clustering methods applied to genotypic data. BMC bioinformatics 7: 204.

[28] Thornton-Wells T, Moore J, Martin E, Pericak-Vance M, Haines J (2008) Confronting complexity in late-onset alzheimer disease: application of two-stage analysis approach addressing heterogeneity and epistasis. Genetic epidemiology 32: 187–203.

[29] Digna T, Dudek R, Ritchie M (2009) Exploring the performance of multifactor

dimensionality reduction in large scale snp studies and in the presence of genetic heterogeneity among epistatic disease models. Hum Hered 67: 183–192.

[30] Lunetta K, Hayward L, Segal J, Van Eerdewegh P (2004) Screening large-scale association study data: exploiting interactions using random forests. BMC genetics 5: 32.

[31] Bush W, Thornton-Wells T, Ritchie M (2007) Association rule discovery has the ability to model complex genetic effects. In: Computational Intelligence and Data Mining, 2007. CIDM 2007. IEEE Symposium on. IEEE, pp. 624–629.

[32] Mushlin R, Gallagher S, Kershenbaum A, Rebbeck T (2009) Clique-finding for heterogeneity and multidimensionality in biomarker epidemiology research: The chamber algorithm. PloS one 4: e4862.

[33] Moore J (2003) The ubiquitous nature of epistasis in determining susceptibility to common human diseases. Human heredity 56: 73–82.

[34] Moore J, Asselbergs F, Williams S (2010) Bioinformatics challenges for genome-wide association studies. Bioinformatics 26: 445–455.

[35] Urbanowicz R, Moore J (2009) Learning Classifier Systems: A Complete Introduction, Review, and Roadmap. Journal of Artificial Evolution and Applications .

[36] Urbanowicz R, Moore J (2010) The application of michigan-style learning classifier systems to address genetic heterogeneity and epistasis in association studies. In: Proceedings of the 12th annual conference on Genetic and evolutionary computation. ACM, pp. 195–202.

[37] Urbanowicz R, Moore J (2011) The application of pittsburgh-style learning classifier systems to address genetic heterogeneity and epistasis in association studies. Parallel Problem Solving from Nature–PPSN XI : 404–413.

[38] Culverhouse R, Suarez B, Lin J, Reich T (2002) A perspective on epistasis: limits of models displaying no main effect. The American Journal of Human Genetics 70: 461–471.

[39] Urbanowicz R, Kiralis J, Sinnott-Armstrong N, Heberling T, Fisher J, et al. (Submitted) GAMETES: A Fast, Direct Algorithm for Generating Pure, Strict, Epistatic Models with Random Architectures. BMC Bioinformatics .

[40] Urbanowicz R, Granizo-Mackenzie A, Moore J (Submitted) An Analysis Pipeline with Visualization-Guided Knowledge Discovery for Michigan-Style Learning Classifier Systems. Evolutionary Intelligence SI: Advancements in Learning Classifier Systems .

[41] Urbanowicz R, Granizo-Mackenzie A, Moore J (2012) Instance-Linked Attribute Tracking and Feedback for Michigan-Style Supervised Learning Classifier Systems. GECCO .

[42] Andrew A, Nelson H, Kelsey K, Moore J, Meng A, et al. (2006) Concordance of multiple analytical approaches demonstrates a complex relationship between dna repair gene snps, smoking and bladder cancer susceptibility. Carcinogenesis 27: 1030.

[43] Karagas M, Tosteson T, Blum J, Morris J, Baron J, et al. (1998) Design of an epidemiologic study of drinking water arsenic exposure and skin and bladder cancer risk in a us population. Environmental Health Perspectives 106: 1047.

[44] Kelsey K, Park S, Nelson H, Karagas M (2004) A population-based case-control study of the xrcc1 arg399gln polymorphism and susceptibility to bladder cancer. Cancer Epidemiology Biomarkers & Prevention 13: 1337–1341.

[45] Cleaver J (2000) Common pathways for ultraviolet skin carcinogenesis in the repair and replication defective groups of xeroderma pigmentosum. Journal of dermatological science 23: 1–11.

[46] Goode E, Ulrich C, Potter J (2002) Polymorphisms in dna repair genes and associations with cancer risk. Cancer Epidemiology Biomarkers & Prevention 11: 1513–1530.

[47] Butkiewicz D, Rusin M, Enewold L, Shields P, Chorazy M, et al. (2001) Genetic polymorphisms in dna repair genes and risk of lung cancer. Carcinogenesis 22: 593.

[48] Bernadó-Mansilla E, Garrell-Guiu J (2003) Accuracy-based learning classifier systems: models, analysis and applications to classification tasks. Evolutionary Computation 11: 209–238.

[49] Wilson S (1995) Classifier fitness based on accuracy. Evolutionary computation 3: 149–175.

[50] Velez D, White B, Motsinger A, Bush W, Ritchie M, et al. (2007) A balanced accuracy function for epistasis modeling in imbalanced datasets using multifactor dimensionality reduction. Genetic Epidemiology 31: 306–315.

[51] Orriols-Puig A, Bernadó-Mansilla E (2008) Revisiting ucs: Description, fitness sharing, and comparison with xcs. Learning Classifier Systems : 96–116.

[52] Suzuki R, Shimodaira H (2006) Pvclust: an r package for assessing the uncertainty in hierarchical clustering. Bioinformatics 22: 1540–1542.

Chapter 7

Conclusion

In this work we set out to examine an alternate paradigm of epidemiological modeling, one that makes minimal assumptions about underlying associations and one that evolves a solution comprised of many models instead of a single best classifier. The flexible, adaptive nature of LCS algorithms led us to the hypothesis that they would be well suited to discovering complex patterns of disease association, and uniquely well suited to discover patterns of heterogeneity. In order to examine this hypothesis we have been inclined to address a number of challenges specific to the LCS research community as well as to machine learning in general.

In approaching chapters 2 and 3 we found that few LCS implementations were available for download, and those that are, target problems largely outside of bioinformatics. In general existing implementations lacked accessibility for researchers outside of computer science. To address this, we implemented standardized versions

of five different LCS algorithms in *python*, a coding language typically more accessible to those outside of computer science, and made them publically available for download [1, 2]. These chapters also required the utilization of simulated complex SNP association models and data. Unsatisfied with existing methodologies, we developed our own model generation strategies with the ability to precisely generate purely epistatic models with random architectures, having a specific heritability, minor allele frequencies, and population prevalence [3, 4]. This new simulation strategy allows for algorithm evaluation over "worst case scenario" interaction models as a gold standard for comparing epistasis detection algorithms. Lastly, in chapters 2 and 3 we were faced with the inherent problem of knowledge discovery in the LCS algorithm. Specifically, how do we quantify the success rate of an algorithm in detecting an underlying model, when the algorithm evolves an entire population of rules as it's solution. Prior to this work, the only published strategy for interpreting an LCS rule-set was for an expert in the respective field to explicitly examine individual rules in the evolved rule-set. We overcome this dilemma, introducing a simple global strategy for the identification of predictive attributes within the rule population. Chapters 2 and 3 apply this strategy to estimate power in respective LCS algorithm evaluations. Ultimately these evaluations demonstrate that LCS algorithms, particularly supervised Michigan-Style LCSs (M-LCSs), could successfully discriminate between predictive attributes and noise in the context of both pure epistasis and heterogeneity. This work also highlighted the shortcomings of existing LCS algorithms for application to our problem of interest. In

particular, efficient algorithm generalization, methodologies for knowledge discovery and the explicit characterization of heterogeneity remained as immediate obstacles.

Chapter 4 tackles the problem of knowledge discovery in a M-LCS. For the first time in the LCS literature we describe a strategy for identifying predictive attributes within the M-LCS rule population based on statistical significance [5]. Additionally we explore novel visualization strategies to facilitate the characterization of interaction and heterogeneity within the LCS rule population. These efforts aim to promote greater interest and confidence in M-LCS data mining.

Chapter 5 describes our efforts to improve the LCS algorithm identified as the most promising in chapters 2 and 3, namely UCS, an LCS for supervised, single step problems. We introduce attribute tracking and attribute feedback as novel heuristics that seek to improve the functionality of UCS on the problems of epistasis and heterogeneity [6]. Attribute tracking addresses the problem of heterogeneity by tracking the attributes that are most important to accurate classification for each instance in the dataset. Post-learning, these tracking scores are then used to characterize subsets of subjects with heterogeneous patterns of association. Presently this is one of only two strategies that can boast this functionality, the other being the CHAMBER algorithm [7]. Evaluations over simulated data demonstrate that the attribute feedback mechanism dramatically improves efficient generalization within the UCS framework, along with improving test accuracy, most power estimates, and run time.

Lastly, chapter 6 applies the complete M-LCS framework established in the pre-

ceding chapters to an investigation of bladder cancer susceptibility. We successfully replicate the identification of a previously described set of predictive factors (XPD 751, XPD 312, and pack-years of smoking) through the application of our M-LCS analysis pipeline. The success of the simulation studies described in Chapters 4 and 5 combined with this replication of previously identified predictive factors validates the efficacy of our proposed M-LCS analytical strategy. Additionally we characterize potentially heterogeneous subsets of patients between which we observe significant differences in survival time, and marginally significant difference in recurrence time (after adjusting for age of diagnosis). These novel findings suggest that attribute tracking is a promising mechanism for characterizing heterogeneous patterns of association.

However, we do not claim that the works described above are sufficient alone to address the problem. Ultimately, this strategy should be applied to generate hypotheses that focus on candidate factors identified as significant. Follow up work validating patterns of interaction or heterogeneity must be performed in the laboratory. Additionally the successful application of this or any other data mining algorithm is dependent on careful study design at the level of population sampling and data collection. We do claim that the application of this strategy may better guide researchers towards promising genetic and environmental candidate factors involved in complex patterns of association.

Further characterization of heterogeneous subject subsets is required to elucidate

the underlying markers of, or cause(s) of, disease. After all, the observation of heterogeneity may reflect a truly heterogeneous relationship between given factors and disease, or it may be indicative of some higher order interaction involving factors not considered or available within a given dataset. This is particularly probable when one considers the intractable problem of collecting data for most environmental exposures. Additionally a given characterized heterogeneous subgroup may itself be further characterized as heterogeneous with patterns too subtle for the present strategy to detect. For instance, we give the example of rare genetic variants. While the M-LCS approach has the potential to identify classification rules capturing rare variants, the statistical analysis pathway proposed in chapter 4 would be unlikely to identify these rare variants as having contributed significantly to accurate classification.

Other drawbacks to the algorithmic strategy described in this work include: (1) Computational time: LCS algorithms are relatively time consuming and require a great deal of computation time, especially when permutation testing and cross validation is included in analysis. (2) Reliability: LCSs are stochastic search algorithms and therefore are not guaranteed to find the best solution. (3) Scalability: due to the computational demands of the algorithm, and the fact that it makes no assumptions about the order of underlying relationships the current implementation is unlikely to scale up to large scale association studies (making candidate studies a better fit). Recent efforts in Pittsburgh-Style LCS have show particular promise in addressing this issue [8]. (4) Complexity: LCSs have seen minimal theoretical work describ-

ing their functionality, and require a number of user defined parameters to operate and potentially optimize. (5) Accessibility: due to their complexity LCSs have been largely avoided in data mining. As a result there are few implementations publically available, and even fewer that are clearly annotated or intended for general use. (6) Interpretability: while we have made some significant progress addressing this problem here, interpretation of LCS rule populations should be examined further. Many of these shortcomings may be addressed in continuing efforts.

The LCS strategy considered here has shown particular potential on problems where most other algorithms have fallen short. Future work, exploring the improvement of this algorithm and the associated analysis pathway would be recommended. First and foremost, the problem of scalability has been a major target for improvement in the LCS literature [9, 10]. Implementation of new heuristics, parallelization, and a version of the algorithm in a more efficient programming language would be likely to improve computational time and scalability. Also, the utilization of expert knowledge has shown promise in scaling down the problem size, by focusing on factors most likely to be of importance [11, 12]. Some of the complexity in running LCS could be addressed via the examination of adaptive parameters [13]. We can begin to address the accessibility of LCSs in bioinformatic data mining through the development of user-friendly LCS software including a graphical user interface (GUI). One avenue for improving the interpretability of our LCS algorithm would incorporate rule compaction for the removal of rules from the population that contribute nothing

to accurate classification [14, 15].

More specific to the work presented in chapters 5 and 6 we plan to consider alternate implementations of attribute tracking that may more reliably identify the attributes important for classification in different subjects. Specifically we can explore the incorporation of a punishment term, for those attributes specified in rules that match a given instance, but specify the incorrect class. Considering the application of attribute tracking scores to the identification of heterogeneous subsets we propose further investigation of strategies for defining patient subsets. While the implementation proposed in this work has yielded interesting results, it is not necessarily optimal.

Lastly, in order to expand the applicability of this algorithm we propose the implementation of an alternate rule representation. The quaternary rule representation adopted in this work is presently only applicable to SNPs and variables encoded into three states. Rule representations that accommodate continuous attributes and quantitative phenotypes have already been explored in the LCS literature, and could easily be applied here [16]. This would broaden the scope of this strategy to accommodate data types such as gene expression and continuous environmental factors.

These are just a few of the many possible modifications that could enhance the utility of this LCS approach to address disease complexity. This strategy has the potential to be applied to any number of complex disease analyses, which will be the primary objective of future work.

7.1 Bibliography

[1] Urbanowicz R, Moore J (2010) The application of michigan-style learning classifier systems to address genetic heterogeneity and epistasis in association studies. In: Proceedings of the 12th annual conference on Genetic and evolutionary computation. ACM, pp. 195–202.

[2] Urbanowicz R, Moore J (2011) The application of pittsburgh-style learning classifier systems to address genetic heterogeneity and epistasis in association studies. Parallel Problem Solving from Nature–PPSN XI : 404–413.

[3] Urbanowicz R, Kiralis J, Sinnott-Armstrong N, Heberling T, Fisher J, et al. (Submitted) GAMETES: A Fast, Direct Algorithm for Generating Pure, Strict, Epistatic Models with Random Architectures. BMC Bioinformatics .

[4] Urbanowicz R, Kiralis J, Fisher J, Moore J (Submitted) Predicting Difficulty in Simulated Genetic Models: Metrics for Model Architecture Selection. BMC Bioinformatics .

[5] Urbanowicz R, Granizo-Mackenzie A, Moore J (Submitted) An Analysis Pipeline with Visualization-Guided Knowledge Discovery for Michigan-Style Learning Classifier Systems. Evolutionary Intelligence SI: Advancements in Learning Classifier Systems .

[6] Urbanowicz R, Granizo-Mackenzie A, Moore J (2012) Instance-Linked Attribute

Tracking and Feedback for Michigan-Style Supervised Learning Classifier Systems. GECCO .

[7] Mushlin R, Gallagher S, Kershenbaum A, Rebbeck T (2009) Clique-finding for heterogeneity and multidimensionality in biomarker epidemiology research: The chamber algorithm. PloS one 4: e4862.

[8] Bacardit J, Stout M, Hirst J, Sastry K, Llorà X, et al. (2007) Automated alphabet reduction method with evolutionary algorithms for protein structure prediction. In: Proceedings of the 9th annual conference on Genetic and evolutionary computation. ACM, pp. 346–353.

[9] Llorà X, Sastry K (2006) Fast rule matching for learning classifier systems via vector instructions. In: Proceedings of the 8th annual conference on Genetic and evolutionary computation. ACM, pp. 1513–1520.

[10] Bacardit J, Burke E, Krasnogor N (2009) Improving the scalability of rule-based evolutionary learning. Memetic Computing 1: 55–67.

[11] Greene C, White B, Moore J (2007) An expert knowledge-guided mutation operator for genome-wide genetic analysis using genetic programming. Pattern Recognition in Bioinformatics : 30–40.

[12] Greene C, Gilmore J, Kiralis J, Andrews P, Moore J (2009) Optimal use of expert knowledge in ant colony optimization for the analysis of epistasis in hu-

man disease. Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics : 92–103.

[13] Hurst J, Bull L (2001) A self-adaptive classifier system. Advances in Learning Classifier Systems : 70–79.

[14] Wilson S (2002) Compact rulesets from xcsi. Advances in Learning Classifier Systems : 65–92.

[15] Gao Y, Huang J, Wu L (2007) Learning classifier system ensemble and compact rule set. Connection Science 19.

[16] Wilson S (2000) Get real! xcs with continuous-valued inputs. Learning Classifier Systems : 209–219.