Regression and Classification from Extinction

by
Joseph Alexander Brown

A Thesis
presented to
The University of Guelph

In partial fulfillment of requirements
for the degree of
Doctor of Philosophy
in
Computer Science

Guelph, Ontario, Canada

ABSTRACT

REGRESSION AND CLASSIFICATION FROM EXTICTION

Joseph Alexander Brown
University of Guelph, 2014

Advisor
Daniel Ashlock

Evolutionary Algorithms use the principles of natural selection and biological evolution to act as search and optimization tools. Two novel Spatially Structured Evolutionary Algorithms: the Multiple Worlds Model (MWM) and Multiple Agent Genetic Networks (MAGnet) are presented. These evolutionary algorithms create evolved unsupervised classifiers for data. Both have a property of subpopulation collapse, where a population/node receives little or no fitness implying the number of classes is too large. This property has the best biological analog of extinction.

MWM has a number of evolving populations of candidate solutions. The novel fitness function selects one member from each population, and fitness is divided between. Each of these populations meets with the biological definition of a separate species; each is a group of organisms which produces offspring within their type, but not outside of it. This fitness function creates an unsupervised classification by partitioning the data, based on which population is of highest fitness, and creates an evolved classifier for that partition.

MAGnet involves a number of evolving agents spread about a graph, the nodes of which contain individual data members or problem instances. The agents will in turn test their fitness on each of the neighbouring nodes in the

graph, moving to the one where they have the highest fitness. During this move they may choose to take one of these problem instances with them. The agent then undergoes evolutionary operations based on which neighbours are on the node. The locations of the problem instances over time are sorted by the evolving agents, and the agents on a node act as a classifier.

I would like to thank Dr. Daniel Ashlock who without his dedication, inteligence, and assistance this work would not have been possiable.

To Wendy, Richard and Peter Ashlock, Colin Lee, Mathew Page, Andrew McEachern, Justin Schonfeld, and Lisa Shiller for helping me retain some limited ammount of sanity.

Thank you to Katharine Dings, Natasha Kavanagh, and Allanah Kelly for their proof reading. In the end, any mistakes are soley mine.

To my parents, Ruthan and Gary, and my sister Holly. You have always kept me going in my darkest hours and kept me company in my brightest. To the memory of Jack and Betty Brown, my grandparents, who I know would be proud of me.

To my dear Ms. Elizabeth Reading — it is said the the English language is one of the most expressive, yet no words give justice to the expression of the amount of love and devotion which I have for you.

J.A.B.

Guelph, Ontario

> "Do it?" Dan, I'm not a Republic *serial* villain. Do you seriously think I'd explain my *master-stroke* if there remained the slightest chance of you affecting its *outcome*? I did it thirty-five minutes ago. — Ozymandias, Watchmen

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Outline

The starting point of this thesis is the K-means algorithm. This algorithm initially partitions data, then iteratively models each member of the partition by its mean value, and finally re-partitions the data based on these models, assigning data points to the partition elements based on the proximity to one of these mean values. An extension of this method is K-models which replaces the measure of "closest to a point" with "best fit to a model". However, both of these methods have limitations in the models which can be represented and which models can be selected. Such limitations are overcome via the use of evolutionary operators acting on discrete structures and the models selection via fitness based reproduction.

Two novel Evolutionary Algorithms (EAs): the Multiple Worlds Model (MWM) and Multiple Agent Genetic Networks (MAGnet) algorithms are high order generalizations of this method. The hypothesis of this work is that interacting simultaneous modeling and classification of data, performed by an evolutionary algorithm, can yield useful exploratory depictions of partitions of data. Two algorithms are developed and tested on a variety of types of data. These are the multiple worlds model and MAGnet. As these models

are of high order, an intuitive sense of how the data is clustered is available to the human expert; they fulfill a need not seen in current Genetic Algorithms, Genetic Programming, Evolutionary Programming, or Evolutionary Strategies; the ability to *classify via solution*. They find uses in various problems in modeling and regression, bioinformatics, and game theory. They are both spatially structured evolutionary algorithms and exhibit a behaviour which is much like an extinction event.

A number of introductory biology textbooks [73, 17] imply that the only significant cause of extinction is human intervention, such as: habitat loss, alien/introduced species, pollution, overexploitation, and climate change. However, evolution requires natural extinction events — called the *background extinction rate* [93]. Even mass extinctions can allow for evolution to flourish as they remove species from ecological niches allowing "windows of opportunity" for new species to take their place [74]. Such extinction-like events have been modeled directly in evolutionary computation but have not been seen as emergent properties of the model.

Such emergent properties can be seen in both algorithms. In Multiple Worlds this involves a population in which the fitness falls off in comparison to the other populations. In MAGnet this consists of a node in the MAGnet graph with no remaining problem instances. These are both analogs to natural extinction, which introduces *extinction* into Evolutionary Computation in a natural way. In both cases these extinction-like events represent the algorithm reaching a conclusion about the natural number of categories for a problem.

1.2 Major Contributions

- A Generalization of the K-means algorithm, known as K-models. This generalization extends the model of the process from being a point, to any other structure. Demonstrated is this process with the selected

models being lines.

- Definition of the Multiple Worlds Model of evolution. This is an evolutionary framework which has multiple evolving populations where fitness is shared between populations, however, no genetic information is traded. This model allows for a process of adaptive radiation to classify data via a set of evolved classifiers. It is an analog to species and a process similar to extinction occurs during this evolutionary process. This framework is demonstrated for problems in partitioning regressions, motif discovery in synthetic and biological DNA sequences, and a radio demographic model.

- Definition of the Multi Agent Genetic Network. This evolutionary algorithm places multiple single instances of a problem on various nodes in a graph; the evolving agents are able to move these instances about the graph. This allows for a sorting of instances in the problem as well as the creation of a set of evolved classifying agents. This algorithm is demonstrated upon the iterated prisoner's dilemma and lead to the production of a new prisoner's dilemma agent know as *Trifecta*.

- A formalism for finite strategies in matrix games — Dominator theory. This theory takes into account the issues inherent in using Evolutionary Stable Strategies in order to speculate on how an Evolutionary Algorithm will progress.

1.3 Organization of the Thesis

The body of the thesis is organized as follows:

Chapter 2 examines the history of evolutionary thought in biology which are inspirations for the algorithms used in this dissertation. Chapter 3 presents a number of evolutionary algorithms, how extinction has been represented, niche methods, and ensemble systems. Chapter 4, examines the

K-models algorithm which is an extention to K-means which allows for other simple models to be cluster centres other than "closest to a point". Chapter 5 presents the Multiple Worlds Model of evolution. Chapter 6, 7 and 8 present the applications for the Multiple Worlds Model of evolution. Chapter 9 presents the Multi Agent Genetic Network. Chapter 10 demonstates the usefulness of this algorithm for the classification and agent discovery for the well known non-zero sum simultaneous game called the iterated prisoner's dilemma. The formalism for dominators, a method for finding a finite state agent which scores optimally against a set of finite state agents, is outlined in Chapter 11. Finally, Chapter 12 gives a conclusion to the monograph, presenting the accomplishments and future directions of study into these algorithms.

Chapter 2

Biological History

The earliest ideas of both evolution and species come from the Greek philosophers. Plato's theory of forms set out, that all objects, which would necessarily include life, are reflections of a set of essences, or *edie*. These edie, when seen in an imperfect reflection, account for all variation seen, much like a funhouse mirror would distort the look of an observer. Aristotle wrote four treaties on natural history: *De anima*, *Historia animalium*, *De partibus animalium*, and *De generatiome animalium*. The last monograph, on the generation of animals, makes mention of hybridization between dogs, foxes, and wolves. Further, speculation is made as to why mules would be unable to copulate when the hybridized offspring of other species do not have this problem.

Charles Darwin posited that "The fertilized germ of one of the higher animals ... becomes a far more marvelous object, for, besides the visible changes which it undergoes, we must believe it is crowded with invisible characters, proper to both sexes, to both the right and left side of the body, and to a long line of male and female ancestors separated by hundreds or even thousands of generations from the present time: and these characters, like those written on paper with invisible ink, lie ready to be evolved whenever the organization is disturbed by certain known or unknown conditions" [31].

These invisible characters would later be found, yet Darwin's idea about the germ coming from throughout the body, would be found to not be the case. The idea of the germ cell being taken from the entire organism was most famously rebuked by August Weismann, who in an experiment removed the tails of rats only to show that: "901 young were produced by five generations of artificially mutilated parents and yet there was not a single example of a rudimentary tail or any other abnormality of the organ" [108]. This furthered the notion that genetic material is not passed to offspring via a polling of the existing somatic cells, which are the differentiated cells of the body, but via a germ cell or gamete, a cell specialized for reproduction carrying traits existant from birth.

Work by Mendel [81] on pea plants is considered to be the first evidence of what would become known as *genes* in biology. His experiments with peas took two sets of pure breed plants of different varieties, e.g. tall plants and short plants. After these plants where hybridized together a medium sized plant was not discovered. Instead the plants were larger and smaller in a ratio of about 3:1. This average 3:1 ratio was found for a number of traits: height, colour, wrinkledness, position of flowers, etc. This implied the existence of a quantization of the traits; a trait would exist or would not exist. Secondly, when the second generation was created from these hybrid plants of the ones which showed the more dominant trait, they were found to keep the persistent trait with an average ratio of 2:1. Mendel stated that in the hybridization process, 1/4 carried the dominant trait, 2/4 carried both traits and displayed the dominant trait, and 1/4 had the non-dominant or recessive trait. This was further continued in the work to show how it would allow for the prediction of the results of hybridization over a number of generations and with a number of traits. This work lead to the use of Punnett squares as a visualization of the process, see Figure 2.1.

The discovery of DNA by Watson and Crick gave final form to Darwin's invisible writing. These base pairs would give a code for forming new chem-

		Seed Bb	
		B	b
Seed Aa	A	AB	Ab
	a	aB	ab

Figure 2.1: Punnett Square (dominant trait uppercase; regressive lowercase) — note 3:1 of the offspring have the dominant trait

ical structures, leading into the central dogma of biology; a strand of DNA is changed into messenger RNA which then becomes proteins. With this, biology has given a motivating reasoning as to how strings of characters can represent solutions to problems via a decoding process into the phenotype. Further, it has shown how the manipulation of those strings create new forms and how those forms are evolved into better forms by the principles of natural selection, inheritance of traits, and efficient representation. Evolutionary Algorithms use principles of all these biological processes to inform, though it is not to say that it is not a low resolution copy of those natural processes.

However, the issue of the *species problem*, how a species arises and what is the definition, continued past Darwin's era. The works of Dobzhansky [37] used the example of fruit flies. *Drosophila pseudoobscura* was one of the first uses of genetics to bring together the ideas of genetics with population biology, showing that the variations of genetics were much greater than anticipated and that natural selection worked to aid in genetic diversity. The *Systematics and the Origin of Species* by Ernst Mayr [76] introduced the biological species concept, "species are groups of interbreeding natural populations that are reproductively isolated from other such groups" [77] which is called by Dobzhansky an *isolating mechanism* [36]. Further, his work speculated that a species would diverge given isolation by geography. These new ideas of genetics, biological species, and natural selection became a consensus view of evolution known as the *neo-Darwinian synthesis*.

The biological inspiration of the Multiple Worlds algorithm is Darwin's finches. Darwin's finches, or Geospizine, are species found in the Galápagos

archipelago, originally by Darwin, on the voyage of the H.M.S. *Beagle* (1831–1836). They are a common example of adaptive evolutionary radiation. The agents in the multiple agent genetic network are able to move through the network with instances of the problem they are meant to solve. This is a sorting behavior that both modifies the environment in which the agents reside and acts to find natural partitions in the data.

These birds were ignored by Darwin in *On the Origin of Species* [32] as they were not found to benefit the argument. Only in retrospective context did their evolutionary importance become clear. Darwin's misclassification of the finches, due to the ornithological thinking during the period, complicated matters [104]. The work of Lack [65] first gave evidence of the evolution of the Geospizine. The birds were seen to have developed numerous types of beaks in order to eat the various seeds in the islands: small beaks for cracking small seeds, big nut cracking beaks, a tool-using beak, and an interesting beak allowing for vampirism. Variations on islands where there was only one species were limited; the beaks would become general in order to eat a wide variety of seeds. When a number of species where on the same islands their beaks would specialize in order to eat different foods and thus avoid competition via niche specialization. The work of the Grants [48, 49, 50] conclusively proved this idea as they conducted multiple-year studies which involved both the tagging and monitoring of birds and the survey of the seeds available.

Biologist Thor Hanson gives another example using North American species:

> The phrase *birds of a feather flock together* has been attributed to Plato, and in nature it is generally true. You don't find coots, pigeons, or gallinules in a gaggle of geese, and a covey of quail does not contain emus ... But during the winter, Black-capped Chickadees attract a crowd. In the woods of Maine, they form the nucleus of mixed-species flocks ... The birds travel and forage together for much of the day gathering in the morning and moving

> through the woods in noisy, constantly shifting groups. This habit probably developed as a way to avoid or defend against predators like owls and hawks, but we wanted to know how these birds all managed to get along so well together. Food can be scarce in winter, and it seemed counterintuitive to invite a gang of hungry rivals to dinner when the cupboard was nearly bare. How does a mixed species flock avoid direct competition? [52]

His answer is based on behavioural changes as well as physical features as "[they] had enough data to see that nuthatches foraged mostly on the trunks, chickadees dominated the main branches, and kinglets spent their time flitting about in the side branches. It was a neat example of what ecologists call "niche partitioning", using subtle variations in behavior to divide a resource among potential competitors"[52]. Thus, behaviours is seen to have a role in the separation of species.

The biological inspiration of the Multiple Agent Genetic Networks come from the manipulation of resources by species in order to better their chances of survival. Humans are perhaps the most successful at this manipulation, e.g. agriculture. *Homo sapiens* are not the only organism which farms. Farmer ants, such as *Cyphomyrmex wheeleri*, have been known to become attached to specific species of fungi for millions of years, even where other food sources are available [80]. These species have experienced a form of 'lock-in'. When a new type of fungi is available to be cultivated a new species of ant will emerge.

Chapter 3

Evolutionary Algorithms

3.1 Introduction

Evolutionary Algorithms (EAs) are a general term for a form of biologically inspired algorithms based upon the ideas of evolution in biology, especially the ideas from genetics and molecular biology. Data structures are commonly referred to as strands of DNA, genes, alleles, or chromosomes. The manipulation of these various data structures is done via operators selected to mimic biological reproduction, or mutation. The ideas of genetic radiation, adaptation, geography impacting on species, and niche specialization all appear. Note this does not necessary only mean using the ideas of the neo-Darwinian synthesis. Incorrect or misapplied biology sometimes makes for good algorithms, e.g. Larmarkian ideas of evolution inspired hybrid algorithms incorporating local search techniques.

EAs are most commonly applied to optimization problems and prediction problems. However, examples from such as procedural content generation [71], robot control [87], and even works of art have been generated [24]. The key theme is that some manner of evolutionary theory is used in the inspiration of the algorithms.

There are many different paradigms for EAs — that is to say it is not

one algorithm, which is then modified, but a set of various algorithms all with applications and methodological differences. The three founding types perhaps are Genetic Algorithms, Evolutionary Programming, and Evolutionary Strategies. Developed independently and originally, they have merged, blended and diversified, and are now developed concurrently.

Genetic Algorithms (GA) were first introduced by Holland [55] as a method to provide approximate solutions to problems via the principles of natural selection. They are an example of a type of EA and a number of good references for this technique are available, see [46, 82, 84, 5].

In a GA a possible solution to a given problem is represented as an easily modified data structure called a *chromosome*. The chromosomes relative ability on a problem is assigned a *fitness*. This fitness is to be maximized or minimized depending on the problem. For a number of *generations* the chromosomes undergo a breeding process. This process starts with *selection*. Selection takes the fitness scores and decides which members will undergo *crossover*, *mutation*, or *survival*. In crossover, e.g. Figure 3.1, two or more selected chromosomes have areas of their structures exchanged. In mutation a single area of the structure is edited in one parent, e.g. Figure 3.2. In survival a chromosome is moved to the new population unchanged. If some part of the generational best chromosomes are passed along via survival, those are declared to be *elite*. The new population can either be of the same size or population numbers can change between generations dependent upon other factors, such as maintaining a minimal fitness value. In a *steady state* algorithm, one new structure is generated at a time. Generational algorithms will create an entire new population from the old and replace the new population over the other.

Evolutionary Programming (EP) is a form of Evolutionary Algorithm invented by Lawrence Fogel to model prediction problems [44]. Prediction problems find the next symbol most likely to occur in a sequence of symbols. The EP model manipulates finite state machines to predict the next symbol

First Parent	1010**1100**00110101
Second Parent	0010**1001**00110010
First Child	1010**1001**00110101
Second Child	0010**1100**00110010

Figure 3.1: Two Point Crossover in a GA

Before Mutation	10**10**110000110101
After Mutation	10**11**110000110101

Figure 3.2: One Point Mutation in a GA

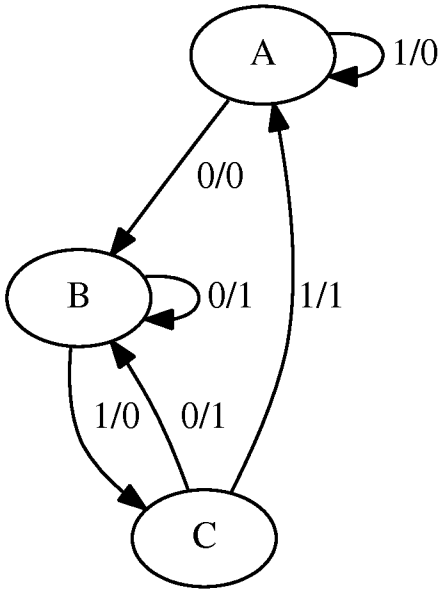


Figure 3.3: A Finite State Machine Predictor [44]

on a transition, where fitness is typically the number of correct predictions. The Finite State Machines used in EP, such as in Figure 3.3, are uniquely defined by a state diagram and an initial state. The example has an input and output alphabet of 0 and 1. Given an initial state, upon the input of a value, the machine will change its internal state and provide the output value of the transition. The finite state machine population is changed via the application of a mutation operator alone, where the parent is replaced by a child only if the number of errors made by the child is reduced.

This type of machine is used in a online process; the problem instances are ongoing during the evolution of new machines. Therefore, the concept of a generation is flexible, and is dependent upon how many children can be created and tested on the past before the next symbols arrive. The mutation operators for changing these children include: changing the connections between states, changing a transitional output, changing the initial state, adding a state, or removing a state.

Fogel [44] further speculates about the idea of a crossover operator which creates a machine, by looking at the majority logic of the population. Fogel notes that this operator is only useful when the machines show a clear majority.

Evolutionary Strategies (ES) created by Rechenbreg [94, 95] and then continued by Schwefel [99, 100] primarily relied upon the genetic operations of selection and mutation. This technique was first used in order to create shapes with minimal drag in wind tunnels. In terms of the biological inspiration they are perhaps more Larmarkian in their approach as it is meant to model the changes to a single individual over time more than a species. The best individual (sometimes individuals) is (are) kept and a number of mutated versions are created. The fitness operator is applied and the best individual is selected for mutation for the next generation.

3.2 Spatially Structured Evolutionary Algorithms

In an extension to the idea of an EA, multiple populations have been introduced in an effort to locate multiple solutions to a problem as well as to provide better results, at least for some problems. Spatially Structured Evolutionary Algorithms allow for parallel implementation, that is the topology of the genetic information can be distributed such that it matches the topology of either the single cluster's processing elements or a distributed network of elements. Each of these topologies [18, 28, 30] have various effects on how they distribute the workload, however the common elements are multiple populations and the transfer of genetic materials between populations by either: exchanging members of the population or allowing breeding between populations. Further, using a spatial algorithm preserves diversity without the need for an explicit comparison of individuals used in nicheing [84] and other diversity preservation techniques [60].

Island models [109, 110] are a common type of spatially structured EA. In this model a number of subpopulations, thought of as populations on distinct islands, are evolved. Each subpopulation is evolved, subject to its own fitness evaluation, selection, and reproduction. In each generation there is a migration of individuals between islands. This allows for the parallelization of the entire population with low transfer cost; the transfer is made of chromosomes between populations, managed by a *migration operator*. Each island maintains its own genetic type, allowing for different evolutionary trajectories, and so areas of the search space can be explored. Migration allows for the trajectories to move towards a final common solution. When problem is separable, the problem can be decomposed into a number of non-interacting subproblems. As these different evolutionary trajectories commonly model different subproblems, island techniques are particularly suitable. Subpopulations with non-trivial probability will find solutions to each of the sub-

problems [109]. Migration then allows for the hybridization of subpopulation to conjoin in a general solution. An example, is the problem of maximizing the number of ones in a bit string. Each of the positions in the string are independent, in terms of their benefits to fitness, and there are many equally fit strings, especially with low fitness. As the island models are very likely to find differing trajectories, the migration creates different strings, which have portions of the solution which will then cross into the general solution.

Graph-based evolutionary algorithms [27] use a number of graphs as connections in a steady state evolutionary algorithm. A vertex v is selected uniformly at random in the graph, and a neighbour is selected for breeding based on a local selection rule, e.g. roulette selection between the neighbours. The resultant offspring replaces the parent on v subject to a replacement rule, e.g. replace the parent if no worse than the parent. [27] looked at 26 different graphs, including random graphs, fully connected, and toruses. It then tested these graph's effectiveness against 23 problem instances from One-Max, De Jongs Functions, Griewangk Function, Self Avoiding Walks, Plus-one-recall-store, DNA Barcodes, and Differential equation solution. Each of these problems looked at very different representation of the evolving structure, e.g. One-Max is a problem on binary strings and Plus-one-recall-store is represented by trees. The findings showed that there can exist significant differences in the results depending on which graph is used dependent on the problem — there is not a singularly good graph. Further, a taxonomy of problems can be obtained via the performance on various graphs. The numerical problems were grouped together, as were the Self Avoiding Walk instances of various sizes.

Simulations of ring species [57] have also been implemented as an evolutionary algorithm. Ring species are those which members are spatially located about a ring or line and are only allowed to interbreed with nearby neighbours. Members adjacent on the ring are close in terms of their genetics, and have no issues in creating fertile offspring. Conversely, pairs of

individuals with large distances on the ring are genetically divergent to the point where their unions are infertile. This method has been used as an alternative to Island models in order to provide a diverse set of solutions to the same problem.

Applications of ring species, first introduced in [9], were applied to the Tartus problem. Later, applications of this algorithm were made to a kind of finite state autometa known as a side effect machine [10], and to the self avoiding walk problem [8]. A block of competent individuals, i.e. the 'found population', are placed on a short arc of the ring. The simulation uses the same methods of crossover, and mutation as GAs. However, the selection is made based upon the neighbors of a location. Once a breeding pair is found, their offspring looks for a nearby empty node in the ring. If no nearby empty node exists, it replaces the individual at an occupied node, if its fitness is greater than or equal to that of the current occupant. This process continues for some number of breeding events until the ring is filled, or an adequate solution is located. An empty ring accepts any solution and so encourages exploration, but as the ring fills in the full portions require improvement to survive and the algorithm smoothy transitions into exploitation.

3.3 Symbolic Regression and Genetic Programming

Regression is the approximation of a data set by choosing parameters for a model. That is, given a set of points $D = (x_1, x_2, \ldots, x_n, y) \in R^{n+1}$, find a function, $y' = f(x_1, x_2, \ldots, x_n)$ such that the error, defined normally as

$$error^2 = \Sigma_{(x_1,x_2,\ldots,x_n)\in D}(y' - y)^2,$$

is minimized.

Koza's approach of Genetic Programming (GP) [64] uses a tree structure

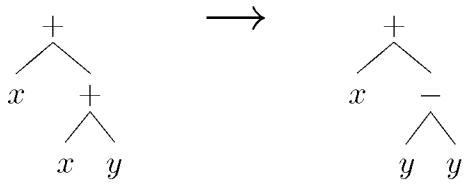


Figure 3.4: Mutation in a GP Tree

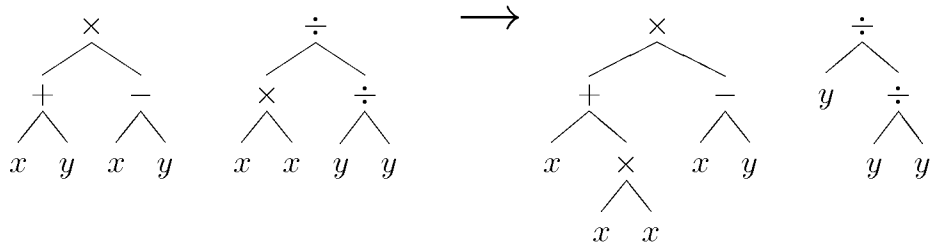


Figure 3.5: Crossover in a GP Tree

in order to represent a function. When GP is used in this fashion the process is called symbolic regression. The tree consists of function nodes drawn from a set of operations (see Figure 3.5) and terminal nodes which are values drawn from the domain of the function or constants. Each of these trees have a *fitness* — its ability to model the function measured by its error from the sample points. The trees have operators of *crossover* and *mutation* applied to them over a number of generations. The crossover is a binary operation, which mixes subtrees via a cut and paste method. The mutation is a unary operation which builds a new subtree from the mutation point.

One of the requirements when using a GP approach for symbolic regression is a set of operations, which are sufficient for representing the data set. Further, the selected operations must obey a closure property. An evolved function should never return a value outside of the domain, or contain an operation which can have an error based on an input, e.g. division by 0. Two methods can be used to avoid this: fitness penalities and operator selection. In fitness punishment, if a function falls outside of the domain or can produce a value in error, then it is assigned an arbitrarily low fitness, or

a punishment value is removed from the fitness. This approach is normally undertaken when the domain is ill-defined, or discontinuous. The evaluation, if punishment should occur, is often expensive as the GP trees are prone to isomorphic representations, e.g. the trees representing $x - x$, $x * 0$, $x - c$, where c is a constant, can all evaluate to 0 leading to a possible division by 0. Conversely, the preferred approach is to avoid the problem by selecting operators which are bounded within the domain or have protections to avoid values which would be erroneous, e.g. a division by 0 returns 0. This limits the types of functions which can be applied to a problem.

Regression for complex systems, such as time varying systems, may require multiple differing functions. Work by Shengwu et al. [101] stated two methods can be used. The first is to add domain specific functions, which would allow large changes and discontinuity in a single function. Such operators as floor, heaviside step, boxcar, or if-then statements can be used to create an arbitrarily accurate model. They caution this will increase the size of a final function, increase search space and make for a less human interpretable solution.

The second method is to change representation of the solution. In their paper [101], representation is a group of functions that are separated by discontinuity points. The evolutionary method changes both the points as well as the functions in order to regress upon multiple functions. This approach works well on the functions for which it was applied, finding both the discontinuity points and the regressors with low error. However, it is extremely limited. First, the number of discontinuity points must be known *a priori*, and the authors comment that finding the optimal number of points "is still a difficult problem" [101]. Secondly their approach is for finding what amounts to a single function for each portion of the data, not a number of functions in the same data set.

Most importantly the idea of the introduction of known discontinuity points leads to a problem in the parsimony of the final solution and to prob-

lems in extending the method into higher dimensions. An example would be $y = max(x^2 - 5, 2)$ (see Figure 3.6) a two dimensional version of an issue presented, using their method both sides of the parabola would have to be modeled independently on each side. A more parsimonious solution would have it modeled by a single function and the line modeled by another. The most parsimonious solution being the function $y = max(x^2 - 5, 2)$. However, it is not necessarily the right model of the data if one were interested in the reasoning for the model being used. For example, if this model was that of a protein which does cell repair and we were looking at levels based on temperature, we might conclude from the two function model that there is a level of cell repair which is normal, and that there is a need for more of this protein when the temperature is in at an extreme high or low. Such requirement of parsimony is of course problematic and situationally dependent. We require enough simplicity to make an appropriate model.

Extending such an approach into multiple regression would require for a discontinuity point to be transformed into a discontinuity vector/plane, meaning that it will only work on hyperplane separable problems. This now introduces a much harder partitioning problem.

The idea, however, of changing the representation of a problem is well informed. The issue is to make this representation able to account for the discontinuity, while being expandable into high dimensional spaces where applications exist.

3.4 Extinction in EAs

Previous studies into the use of extinction in evolutionary algorithms have attempted to directly model the rate of extinction, rather than it being an emergent property of the model. Greenwood et al. [51] use a stochastic stressor, which is applied to the population killing all those with fitness values below it, before allowing them to produce offspring. This allows for both pe-

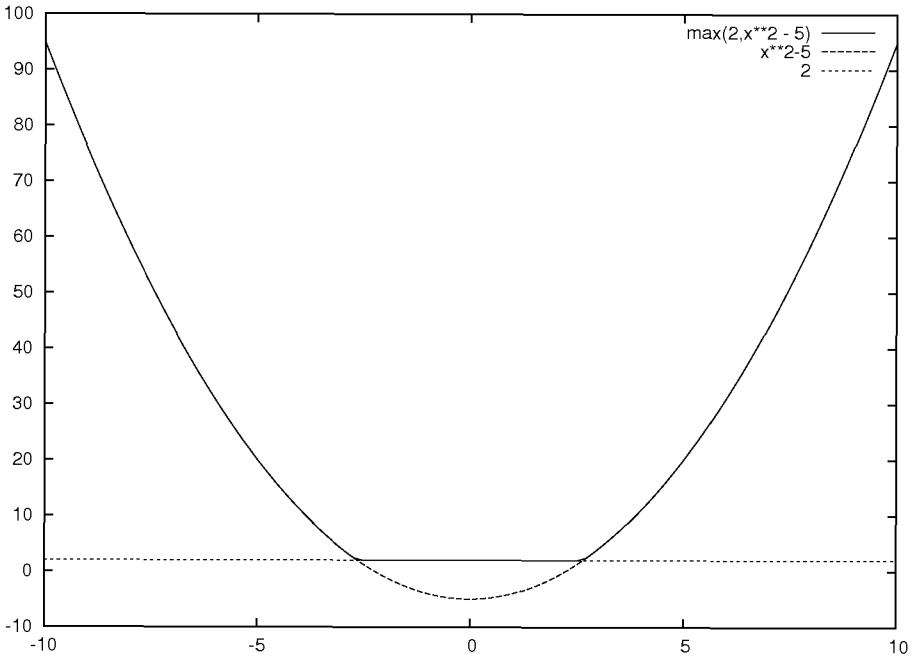


Figure 3.6: $y = max(x^2 - 5, 2)$ modeled by $y = x^2 - 5$ and $y = 2$

riods of limited interaction of the stressor and large extinction periods when the stressor is high. Further, study of this model by Fogel et al. [43] found it to be best on extreme ends of fitness landscapes, smooth and highly rugged. Extensions of this model have been made in Particle Swarm Optimizers [111] which would reinitialize velocity after a defined period. However, this model does not have much biological analog to species extinction as members of swarms have normally not been seen as different species, and simply reinitializing velocity would be biologically closer to a swarm being scared away from current locations; much like a farmer using bird poppers in a grape field to keep his crop.

The ideas of extinction in EAs also informs nicheing techniques. The goal of these techniques are to maintain diversity in the population by imposing a penalty on chromosomes which are close in terms of a metric placed on the gene space. *Fitness sharing* gives each of the chromosomes a fitness equal to the chromosome's raw fitness divided by the numner of chromosomes in the area of its niche [45]. In *crowding* techniques new chromosomes are created which replace close neighbours to current members of the population [61]. The second feature of these techiques is that they will preserve multiple solutions to a problem in the same population. It should be noted that these multiple solutions are based off the same problem space, not subsections of it which is seen in ensemble systems, hence the created chromosomes maintain diversity. This is useful for multiobjective problems especially, and a mechanism for preventing crowding is used in the common NSGA-II [34].

As nicheing techniques commonly require that a metric be imposed on the genes, this is not always a simple process depending on the complexity of the chromosome, if the geneotype must have a translation process this becomes an even more difficult distance to define properly. Secondly, this distance must be calculated for all the members of the population pairwise, and this increases the time complexity of the algorithm.

3.5 Ensemble Systems and Co-evolutions

The roots of ensemble systems are found in the ideas of partitioning the dataset in order to provide two, or more, classifiers [33]. This supervised method of classifier creation begins by breaking up the dataset with a hyperplane, and then creates classifiers for both partitions. The classification is made via finding which side of the hyperplane the data, to be classified upon, rests, and when in doubt a k-nearest neighbours rule is followed. The developed classifier is then used inside of the partition in order to fully classify the data. A downfall of this method is the requirement of a good hyperplane selection, in order to give such classification. This requirement is overcome in ensemble systems by placing multiple classifiers over the same set of data.

Ensemble systems are a supervised classification method in which multiple models have an average or vote taken as to the classification of points in a set of data [89]. This allows for a set of good classifiers to be taken in concert to provide a better accuracy than only one classifier alone. This process is an analog to the idea of bringing together a set of experts and asking their opinions on the data. For example, looking at the review process of a peer reviewed journal, more than one reviewer will be sent the same paper by an editor. These reviewers give a classification of the paper in question, such as accept or reject. The editor acts as the meta-classifier, looking at the reports provided from the reviewers to give a final classification of accept or reject. By having multiple reviewers, the goals are to remove bias of the individual reviewers, find mistakes, and prevent an unqualified reviewer from needlessly rejecting an article.

This is meta-level classification from a diverse set of classifying models, which need not necessarily be from the same family of classifiers, e.g. a decision tree, an expert system, and a support vector machine could be used in the meta-classification. The method requires classifiers, which are both good classifiers on their own and able to correct mistakes made by other members of the ensemble. That is the decisions made must be diverse.

Two main methods are used in order to change the weights of the votes for the classifiers: *boosting* and *bagging*. In boosting, three classifiers are designed from a set of training data. The first is trained on a random subset of the training examples. The second is trained on an informative subset formed by showing examples to the first classifier by taking half of them, which are classified correctly and half of which are classified incorrectly. Finally, the third classifier is created by training on the examples where the two other classifiers disagree in their classifications. For a new sample, where the first two agree, their classification is returned, otherwise, classification is made by the third method. In bagging, or bootstrap aggregating, the pre-trained classifiers make a simple majority vote on the data.

By having a family of classifiers there is also the ability to have a confidence of score associated. For example, if two of three classifications agree, we would have a confidence of two-thirds in our classifier. This does not necessarily imply that a point with high confidence is correct, no more than one with a low confidence is incorrect. However, with good selection of the underlying classifiers, this relation holds [86].

In terms of an evolutionary approach using these concepts the idea of ensemble systems using an island model has been developed [13]. This supervised classifier system uses a number of GP populations acting as voting classifiers on the set of data which are evolved. This co-evolutionary system has been found to outperform bagging and in the majority of cases outperform boosting. Similar co-evolutions have been used in order to increase the classification via neural networks [88].

Chapter 4

K-models

Areas of this chaper first appeared as Ashlock, Brown, and Corns *K-models Clustering, a Generalization* [7].

4.1 Motivations

Clustering is an exploratory technique for tentatively classifying data. Clustering algorithms are used to search for patterns in data, to reduce the size of a data set by selecting representatives of clusters, and to generate hypotheses about the character of a data set. This study introduces *K-models* clustering, a natural generalization of K-means clustering [70, 72, 19]. The similarity measure that places data items in the same K-means cluster is proximity according to a distance measure. K-models clustering modifies K-means by replacing proximity to a cluster center with minimal squared error according to K instances of a statistical model, where the instances of the model play the same roles that clusters do in K-means.

The name K-models has been used for specific techniques which replace the mean with expectation maximization and hidden Markov models [59]. However, the definition for K-models specified in the thesis does not require a particular model, rather it works on any model for which an error measure

can be computed and fitted. K-models can, in fact, mix very different models. K-models, as realized in this thesis, is a generalized framework for which the choice of model is a, possibly dynamic, choice. The common feature across all realizations is the iterative nature. That is, tentatively assign data members to a model, fit the model to the data members for which it is assigned, and reassign data members if a new model fits them better. As with the original K-means algorithm, iterations of fitting and reassignment are continued until no reassignment is required.

K-means selects a set of K initial cluster centers. It then iteratively assigns points to clusters according to proximity to a cluster centers and then recomputes cluster centers as the mean position of the cluster. This continues until points stop changing clusters and the final collection of clusters is reported. It is possible to take the point of view that the location of the cluster centers is a parameter estimation problem. Once this has been done, it is natural to ask what are the consequences of replacing the parameter estimation problem for good cluster centers with parameter estimation for some other statistical model. Here the model for least squares fit of a line in two dimensions is used to provide an arena for proof-of-concept for K-models. Linear regression is a widely known and extensively studied statistical model and provides behaviour very different from K-means.

Gaussian mixture models, selected with the expectation maximization (EM) algorithm [35, 112], are a popular model based clustering algorithm. K-models performs a very similar task — clustering points based on the quality of their fit with multiple statistical models derived from the data — but it does it in a very different way. The K-models algorithm is fast, comparable to traditional K-means for the case where the statistical model is a least-squares line. The EM-algorithm is famous for being slow. Both algorithms share the property that they can converge to local minima of their quality criteria. The speed of K-models permits this problem to be solved by sampling, as long as the statistical model being used can be computed

quickly. A comparison of K-models with mixture models is an early priority for future research.

Multiclustering [1, 62] is an ensemble clustering technique that amalgamates the results of many samples of the K-means algorithm. It is trivial to swap some instance of K-models in place of K-means in this algorithm; if the data warrant it, the multiclustering technique could amalgamate information from many different types of K-models clustering.

4.2 Definition of K-models

The K-models algorithm is given in Algorithm 1. As with K-means, the K in K-models denotes the number of clusters. A single design feature is different between K-means and K-models, the replacement of proximity to a cluster center with minimal squared error for a model. As a result K-models has an important commonality with K-means. The initial partition of the data is random but all other steps of the algorithm are deterministic. This means that the techniques for sampling initializations for the K-means algorithm can also be applied to K-models. We will call a choice of initial partition for the algorithm a *sample* of the algorithm.

Algorithm 1. K-models

Input: 1) A set S of points in $\mathbb{R}^n$.
2) A desired number k of clusters.
3) A choice of statistical model.

Output: A category function $C : S \to \{0, \ldots, k-1\}$

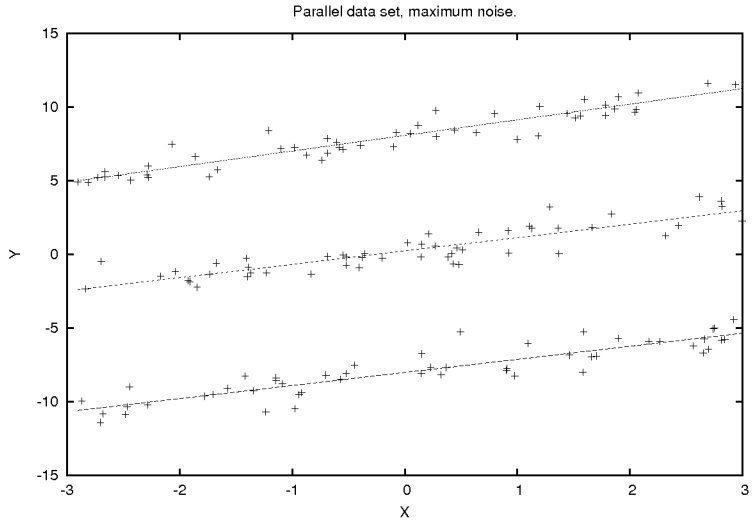
Details:

Partition the data randomly into k clusters.
Repeat
Fit a model to each cluster.
Assign points to clusters based on which
model has the smallest squared error.
Until(No points change clusters).
Report final cluster assignment as C.

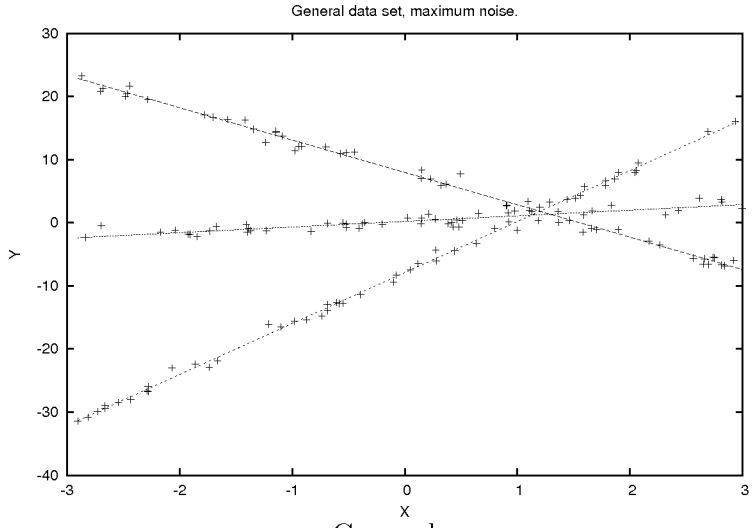
It is worth noting that if we use, as the statistical model, proximity to a point then K-means appears as a special case of K-models.

4.3 Experimental Design

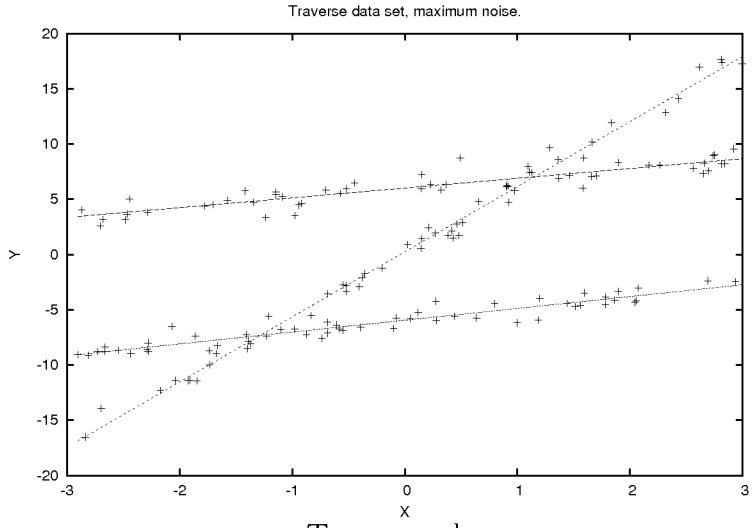
Ten sets of data were constructed for evaluating K-models on two-dimensional data using the fitting of least squares lines for the statistical model. The first nine data sets are constructed using three sets of planted linear patterns with three different levels of noise. The sets of lines used are named in Table 4.1. For each line, fifty random samples are taken in the window $-3 \leq x \leq 3$ with a Gaussian random variable added to the y coordinate. The noise levels $\sigma = 0$, $\sigma = 0.5$, and $\sigma = 1.0$ are used. A set of 1000 samples of the K-models algorithm for $k = 3$ are taken for each of these data sets.
The tenth data set is a set of 150 points sampled uniformly at random from a circle of radius five centered at the origin. A set of 100 samples of the K-models algorithm for $k = 6$ are taken for this data set. Since the data set does not fit a linear model well, the number of clusters is not critical but should be more than two to permit the lines to approximate circular arcs.



Parallel



General



Transversal

Figure 4.1: Shown are the highest noise data sets for each of the three groups of lines used in the experiments, together with the lines discovered by the K-models algorithms.

Line set	Lines
Parallel	$y = x - 8, \quad y = x, \quad y = x + 8$
General	$y = -5x + 8, \quad y = 8x + 8, \quad y = x$
Transversal	$y = x - 6, \quad y = x + 6, \quad y = 6x$

Table 4.1: Sets of lines used to create planted linear patterns.

4.4 Results

The three data sets with planted linear patterns with the highest noise level are graphed in Figure 4.1 together with the set of three lines with lowest sum-of-square-error (SSE) found by the K-models algorithm. Table 4.2 gives, for the first nine data sets, the minimum SSE located and the number of times the clustering that yielded that SSE was found in 1000 trials. This latter is called "times correct" because the lowest SSE pattern closely reproduced the coefficient of the lines used to embed the linear patterns. It is interesting to note that the number of distinct clusterings found by the algorithm was quite small with most clusterings located many times. This is probably because of the presence of the planted linear patterns.

The data sets based on the parallel lines were the most difficult while the general lines were the easiest. This suggests that the K-models technique using fitted lines will scale well to higher dimensional data sets. When more dimensions are available the freedom of lines to fail to be parallel or which skew, increases. This in turn means that collections of lines have a lower chance of being parallel or in some other non-general configuration.

Figure 4.2 shows the data set obtained by sampling a circle together six lines located by applying the K-models algorithm. These lines are from the algorithm with the lowest SSE out of 1000 samples. Notice that the lines fitted by the algorithm are spaced equally about the circle, modeling equal arcs of the circles as lines. While necessarily imperfect, this is a reasonable treatment of the data.

Noise Level:	$\sigma = 0$	$\sigma = 0.5$	$\sigma = 1$
Parallel			
Times Correct:	274	229	183
Best SSE	7.98E-9	19.1	76.6
General			
Times Correct:	689	607	653
Best SSE	6.07E-8	17.5	68.4
Traversal			
Times Correct:	541	584	488
Best SSE	1.18E-8	18.1	70.3

Table 4.2: Shown are the number of times in 1000 trials that the lowest SSE set of models were discovered and the value of the minimum SSE for each of the nine data sets with planted linear patterns.

4.5 Discussion

One of the most striking features of this first test of a K-models algorithm is the large number of times the algorithm discovered the same clustering. The effect of adding noise to the data sets with planted linear patterns was visible but modest and the correct patterns were located in spite of the noise. In the noise-free data sets the algorithm correctly reconstructed, to eight decimal places, the coefficients of the lines that were sampled.

This study is a proof-of-concept for K-models. The planted linear pattern data, even at the highest noise level, are constructed to permit the algorithm to function efficiently. We note that the algorithm does function efficiently in this context. The circular data set is one without a planted linear pattern and the algorithm gives a sensible result for these data. In all the experiments the SSE statistic was used to select the best set of linear models and hence the best clustering. The strength of this statistic is addressed by the data displayed in Table 4.3. When the K-models algorithm found the correct set

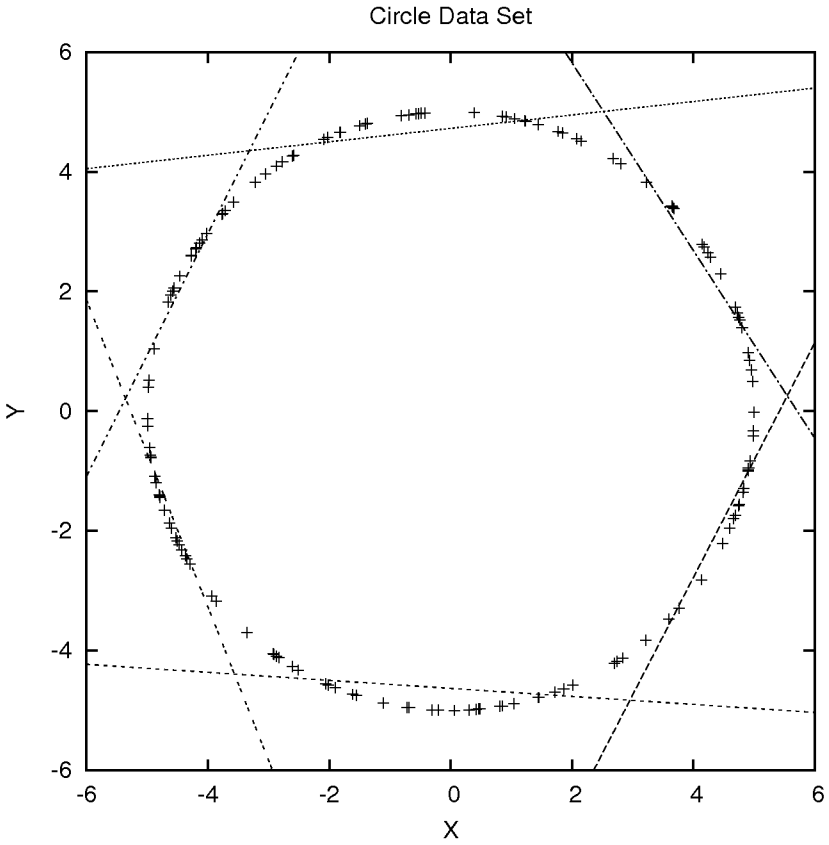


Figure 4.2: Shown are the results of applying K-models to 150 points randomly samples from a circle of radius five centered at the origin for $K = 6$

of linear models it found them with a substantial margin in SSE.

By defining the statistical model as "minimize distance to cluster centers" we can reproduce K-means as an instance of K-models. This shows that K-models is a natural generalization of K-means. This, in turn, means that the large body of research both examining and incorporating K-means is open for re-examination with the K-models technique; a plethora of potential future work.

While this study used least squares lines in two dimensions, any statistical model could have been used. It is also easy to use K-models on mixtures of

Best and second best SSE						
Pattern Type	$\sigma = 0.0$		$\sigma = 0.5$		$\sigma = 1.0$	
Parallel	7.98E-9	431	19.1	440	76.6	488
General	6.07E-8	335	17.5	344	68.4	364
Transversal	1.18E-8	230	18.1	315	70.3	320

Table 4.3: Shown are the best and second-best SSE, over 1000 samples, for the nine data sets with planted linear patterns.

different types of statistical models: lines, planes, and hyperplanes of various dimensions. In this case, evolutionary computation could be used to select the set of models and initial data partition with SSE serving as a fitness function to be minimized.

When K-models is used, it yields a collection of K statistical models. If the data were transformed from their original form to a vector of squared-error values with respect to each of the models located, then the resulting transform functions as a kernel of the sort used in support vector machines[19]. It is also likely that the technique can be used for visualization or dimension-reduction.

K-models is a simple type of classification via a regression. When a reasonable statistical model is available for a set of data, K-models should be used rather than an evolutionary method. However, when such a model is not available or this is not known, evolutionary methods provide a technique for the creation of a partitioning regression model.

Chapter 5

Multiple Worlds Model (MWM) of Evolution

5.1 Algorithm Definition

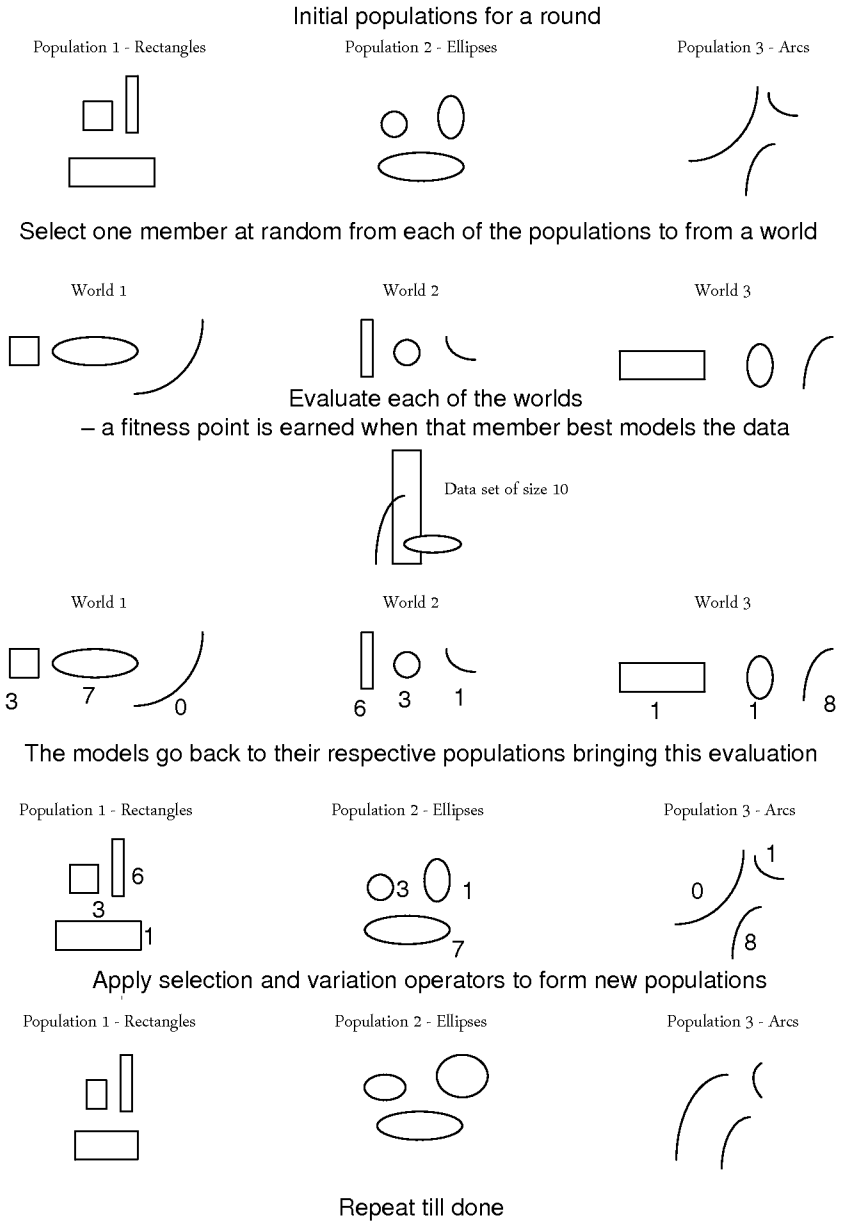
The *Multiple Worlds Model* (MWM) of evolution is applied in situations where a number of distinct agents with interacting roles must be evolved, see Figure 5.1. The model begins with a number of populations, believed to be at least the correct number of interacting roles, with the goal that each population will model one such role. This is similar to how Darwin's finches will evolve to exploit differing food sources. If a partitioning of roles or data exists, then the created agents should begin to meet those needs via specialization. If no such role partitioning occurs, or the number of populations is greater than the number of potential roles, then a subset of the populations should receive a low fitness or the populations should fight by creating similar agents. Specialization into roles is enabled by the fitness function.

As the populations are infertile by the interactions between species, there is no transfer of genetic material via a migration, such as that in the Island models. Instead each of the populations acts independently in terms of ge-

netic operators and creation of new agent members of the population. The only interactions between species are made during the evaluation of the fitness. This novel approach to the fitness evaluation is what sets the multiple worlds model apart from other spatially structured EAs. In other spatially structured EAs the evaluation is made in each population by its own population members upon a problem. In multiple worlds, all of the populations are brought together for the evaluation.

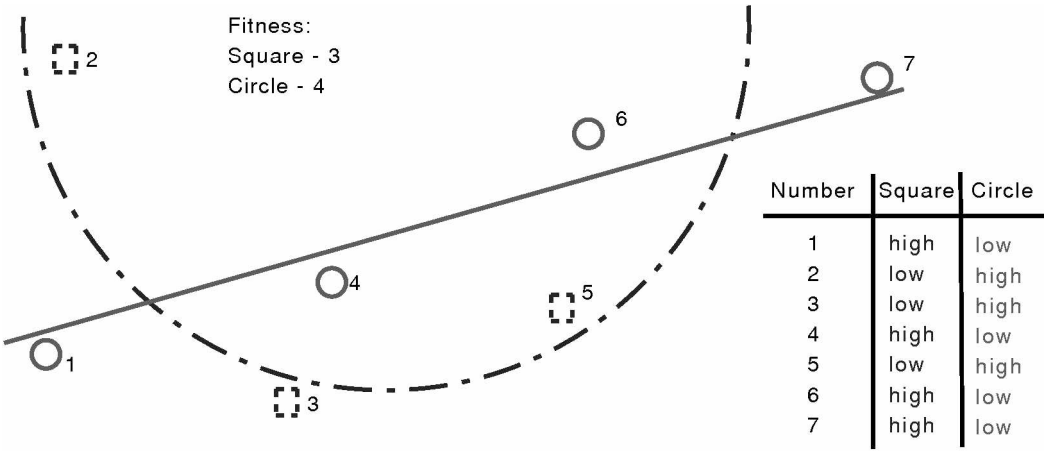
In fitness evaluation, the populations are shuffled. The corresponding (first, second, ..., penultimate, last) members of each population are then grouped with one member from each population, formally a *world*. The score of each agent in a world is then computed where the points of fitness are only rewarded to the population member with the lowest error or best score on each of the objects being classified. The population in effect wins the fitness from the other populations. As a number of worlds are created, a fit agent is not only one which does well against the members of its own breeding group, but does well against other species in claiming a large share of finite amount of resources. This inter and intra-species fitness evaluation and non-migration between species of genetic material is what causes specialization. It models both interspecies competition for resources and intraspecies selection pressure via natural selection.

In this model the number of requisite classes is not necessarily needed *a priori*. Competition between species is often enough to force those species without a specialization out of existence. We call this property *subpopulation collapse*, a direct analog to an extinction event upon a species, where a population in this model receives a sufficiently low fitness. This is a signal that the selected starting number of populations was too large, and that a smaller number of populations is enough to specialize in order to fulfil all the roles in the data set.




```
for some number of generations do
  randomize the worlds
  for all worlds do
    for all datapoints do
      award the point to the model with the best fitness
    end for
  end for
  for all populations do
    Select breeding pairs based on fitness
    Apply Crossover/Mutation
  end for
end for
```

Figure 5.1: Demonstration and Pseudocode of the Multiple Worlds system



Number	Square	Circle
1	high	low
2	low	high
3	low	high
4	high	low
5	low	high
6	high	low
7	high	low

Figure 5.2: Sample fitness evaluation of the one world drawn from the square and circle populations of regressive lines

5.2 Example Fitness Evaluation - Regression

In this section, we look at a sample fitness evaluation of a world. Taking two populations, which will be referenced as circle and square, the ordering of these populations is randomized. One regressive line is selected from the circle and square populations as seen in Figure 5.2; the circle straight line and the square curved line. Each point in the dataset is then measured to both of the lines; this distance will be referred to as either low or high. The points are then scored based on the lines. A line will score the points closest to it, e.g. point 3 is closer to the square line than the circle line; square wins the point. The number of won points is then given as the fitness of that line, i.e. circle scores 4 and square scores 3 in this example.

The square and circle populations now breed as per a normal genetic algorithm/programming, applying selection based of these scored points, undergoing genetic operators.

5.3 Parallelism

The Multiple Worlds Model, like the majority of SSEA, is well suited for parallel implementation. If we assume a number of processing elements equal to the number of populations in the model; holding all other parameters about the population the same; and having the dataset, which is available for each processing element in local memory, and all fitness evaluations assumed to have the same computational time, we can come to the following guidelines about its parallel implementation. The genetics of the populations are independent in terms of data and communication, hence each population's selection and breeding can progress in a naturally parallel manner consistent with island models. The time taken in the selection and breeding steps, *caeteris paribus*, takes the same amount of processing steps for each population. Hence, the novel fitness evaluation is the only step where the issues of parallelism are required to be considered.

The fitness function can be broken down into two phases in a population: 1) evaluation of the chromosome on each datapoint and 2) the finding maximum or minimum evaluation score of the chromosomes across a world. The first step, again, has no dependencies in data or processing; *caeteris paribus*, it should take the same time for each population. Hence, the only communication and processing dependencies are caused when finding the maximum or minimum evaluation score of the chromosomes across worlds. This is finding a maximum/minimum value over a number of processing elements equal to the number of populations. Finding this requires $O(N)$ comparisons, were N is the number of populations, and O(N) communications, to evaluate and inform a population that it has won the point of fitness. The total number of communications for a single generation is therefore in the order of $O(cN)$ were c is a constant defined as the number of datapoints. The size of these communications would be the population number and the evaluation of the model. There is no need to transfer the chromosome at any step.

5.4 Experimental Overview

To examine the abilities of the multiple worlds model of evolution, there were three major experiments conducted in differing application domains. These selections show the evolutionary framework to have a diverse range of applications, real or discrete representations. The commonality in the problem is a set of multiple interacting roles are present in the data and there is a requirement to classify the data into which model is responsible. When such interacting roles are not present, i.e. the data fits under a single model, then MWM obviously should not be applied. It would be prudent to use a single population to optimize. However, MWM has a implicit tendency to find the number of roles which are present in the data.

The question as to if the applications presented are appropriate for evolutionary methods is not within the scope of this research. The experiments are to show the utility of the framework for when a specific closed form or more exact modeling method is not known. There is the problem of "no-free lunch"; when a good problem specific model is available then a generalized framework, is of course, unsuited and undesirable. The method is unsupervised in its classifications and exploratory in nature. The results are compared to a known partitioning in the first two experiments via the Rand index. However, the ability to determine if the final classification is suitable is dependent on the application domain expert.

The first experiment, Chapter 6, examines the utility of the MWM for the creation of partitioning regressions. This is a real valued dataset, which the MWM model is applied to simultaneously partition and model. In addition it discovers the correct number of classes in the data. This initial proof of concept was applied to sets of points with planted data, lines are used as the representation. Two of the data sets are used as positive examples of the technique. The first models a set of lines using lines, much like the K-models examples, the second applies the MWM to modeling planes with lines. The final data set is an intentionally bad choice in the model, representing spheres

with lines, it shows a negative result.

A normal GA, with a number of genes in the chromosome set equal to the number of worlds in MWM, is examined for a simplified version of the problem. The GA has a number of benefits over MWM in terms of the settings: more fitness evaluations during the run, settings to encourage exploration of the space, etc. This first is to show that MWM meets a need beyond the basic methods, the ability to evolve multiple interacting agents. It demonstrates the issues with overburdening a single evolved chromosome with multiple models as it allows for a poor model to or positively be selected for due to a set of good models being in the same chromosome, i.e. genetic draft. Finally, this test shows the importance of having error represented in the fitness function when it is available. In this instance the distance of the evolved regression lines to the data which it represents is an explicit error measure. Further, the MWM model is able to discover the number of classes in the data, the GA does not allow for a collapse and will use the extra model to obtain low error when possible.

The biological motifs, the second experimental domain in Chapter 7, demonstrates MWM applied to a discrete problem. A number of various data creation methods with their known classifications are presented for the test set. The discrete example does not demonstrate the requirement for an explicit error term in order to provide a solution. No error term is apparent in the motifs. By having a number of populations which exceeds the number of classes, the system has an implicit error term. The evolutionary pressure of the competition between populations for resources forces the populations to specialize in a manner similar to as if a error term was present. When the number of populations is set equal to the number of known classes, this degraded the performance, holding all other parameters equal. The extra populations are, therefore, the cause of increased performance in modeling the natural partitions.

Examined is also the issues of representation in the modeling. A human

readable model was selected for the MWM, partially in order to make the analysis of the discovered models understandable. This representation does cause some interesting effects to the ability for classification. First, the created motifs for reverse complement data set the are themselves self reverse complements. In some instances where a population scores high on the Rand index but does not witness a collapse event there is an issue of a motif being a subset of another motif in the set of classifiers. The representation makes such issues appear. However, in the later event, as the representation is human readable, a domain expert with little effort could see such a subsetting and apply a correction, such as counting both motifs as being the same class.

Finally, the third experiment, Chapter 8, is a modeling of a group of radio stations competing for market share as they broadcast to the same listener base. This was selected for the pedagogical effect of having a social simulation example which demonstrates the training of agents for multiple roles without a need for having to specify the agent roles in the design of the simulation. Radio stations are modeled via a playlist, a string which contains the order of songs and advertisements. While all radio stations have the same representation, obviously, creating a single radio station type and broadcasting this on every station is not appropriate. This is not meeting market demand, each location on the dial would play the same music, and meeting with customer preferences would lead to the creation of a station averaged over the preferences of all listeners. This does not meet with our experience of having different stations playing different music. Further, radio listeners never have more than one channel on at time, as that would confuse the listener. The stations are in competition. Hence, multiple worlds is an appropriate model as this is a situation which has interacting distinct agents. An initial test is made as a benchmark to demonstrate that two listeners with violently opposed tastes will generate stations playing two completely different playlists. Without an explicit error term, much like the motifs, a number of populations greater than the number of actual types of listeners

leads to the radio stations competing over the consumer base more than if there was the correct number of classes. The simplicity of the model also acts as a verification for the MWM — when listener profiles are dis/similar in enjoyment the resulting stations are dis/similar in terms of the playlists.

Chapter 6

Partitioning Regression

6.1 Lines Tests

The Multiple Worlds model of evolution was first applied to the problem of partitioning regression by Brown and Ashlock [25].

6.1.1 Function Stacks

Function Stacks [6] are a form of Genetic Programming based on *Cartesian Genetic Programming* [83]. Instead of the normal parse tree structure, a directed acyclic graph is used. Each of the nodes takes arguments of either a value from the dataset or the output of a node with a higher index. The backward links to the calculations allows the ability to avoid having to re-perform a calculation that has already been made. This can result in huge space (and hence execution time) savings. The motivation of using this structure is that the data structure is linear and of a fixed size removing the issue of program bloat. The lack of a need for repeated subtrees that recalculate values needed in multiple locations yields a space savings that makes data structures with a modest fixed size practical. In addition, standard methods of crossover and mutation for linear structures can be used rather than subtree crossover and

tree mutations, avoiding the disruptive character of these operators. We use the operators in 6.1. These operations are good at the creation of a linear regressor. These operations obey the closure property due to the use of a protected division, therefore the regressor created by the function stacks will always be a well defined function, $f : \mathbb{R}^n \to \mathbb{R}$.

Table 6.1: Operations Used in the Nodes of the Function Stacks

Name	Arity	Definition
neg	1	negates X
sel	1	scales X by the ephemeral constant
sqt	1	square root of X
sqr	1	squares X
sin	1	sine of X
cos	1	cosine of X
add	2	adds X and Y
sub	2	subtracts Y from X
mul	2	multiplies X with Y
dup	2	divides X by Y, if $Y = 0$ then 0 is the result
max	2	argument maximum of X and Y
min	2	argument minimum of X and Y
wav	2	weighted average of X and Y with the weight given by the ephemeral constant

6.1.2 Rand Index

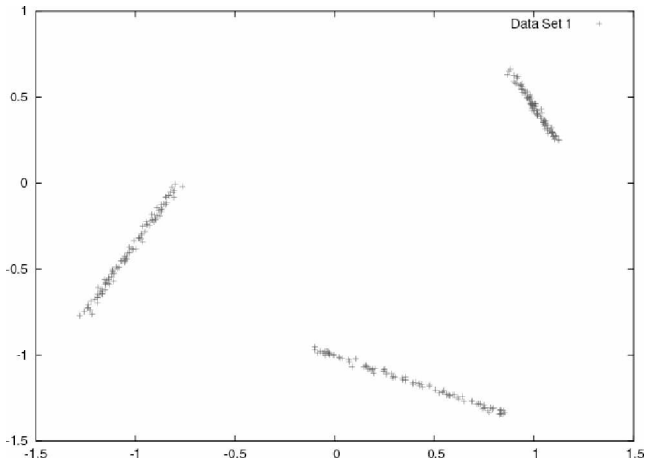
The Rand index [92] is a measure of similarity of partitions. When a correct partitioning is known, this calculation can be used as a quality measure. The Rand index produces a real value from the interval $[0, 1]$, where an exact match of the induced and actual partitions would score 1. The Rand index is calculated as follows: Let D be a set of data and let P and Q be two partitions of this data into classes. The Rand index comparing P and Q is the fraction of pairs from D that are either in the same class in both P and Q or are in distinct classes in both P and Q.

6.1.3 Experimental Settings

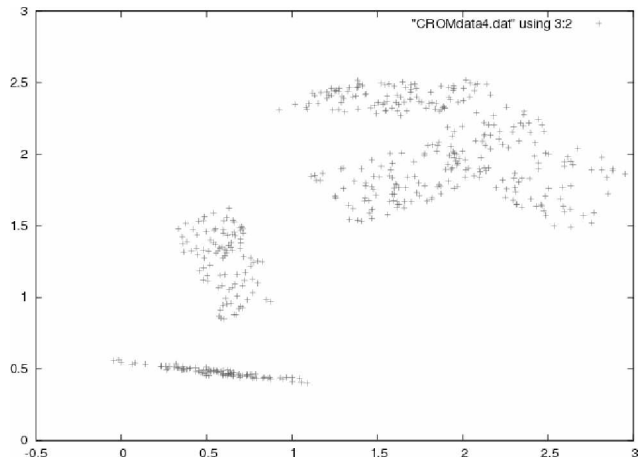
The data sets used sampled 100 points of each of the three classes for a total of 300 data points. The first dataset, DATAONE, are points sampled from three skew lines in six dimensions (Fig. 6.1(a)). DATATWO are points sampled from six skew planes in six dimensions (Fig. 6.1(b)). DATATHREE are points sampled from three concentric spheres in three dimensions (Fig. 6.1(c)). DATATHREE is a negative test case for the method; it is a set of data which has an insufficient set of operators to be modeled successfully. These figures show an estimate of the mean performance based on a large number of samples. For n samples these error bars would scale as $\frac{1}{\sqrt{n}}$. It is therefore not surprising the best sample is outside of the estimate of the mean. This is one of the results of being able to run an arbitrarily large number of experiments, something not seen in biology.

The number of nodes/worlds was set to be 6 whereas the data sets provided 3 classes of data. This is to provide a basis to see if the system will provide a sub-population collapse or if it will be prone to a fight between populations to overfit the data.

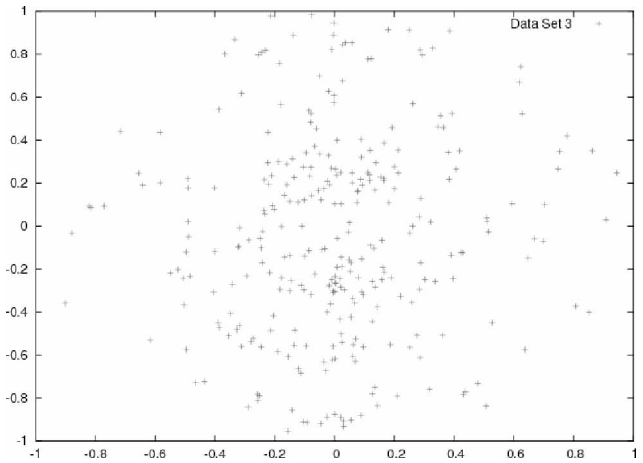
The model of evolution used is tournament selection, which takes four members of the population, orders them by fitness, and replaces the bottom two with replicates of the top two that are subjected to crossover and mutation. The crossover is a two-point crossover, which exchanges nodes. The mutation was chosen from three types probabilistically. The mutations are to change the operator on a node, change a single link between nodes, or change the ephemeral constant. The rate of change to the ephemeral constant was held steady at 40%, while the mutation to an operator and links where set to 20%/40%, 30%/30%, and 40%/20%. The process of evolution lasts for 1000 generations which has been found to provide more than enough for the population to converge to a steady state. The population size was varied in 20 member intervals between 20 and 100. To allow for use of the normal distribution in statistical tests the number of replicates was set to 30.



(a) DATAONE



(b) DATATWO



(c) DATATHREE

Figure 6.1: Two dimensional projections of the data sets.

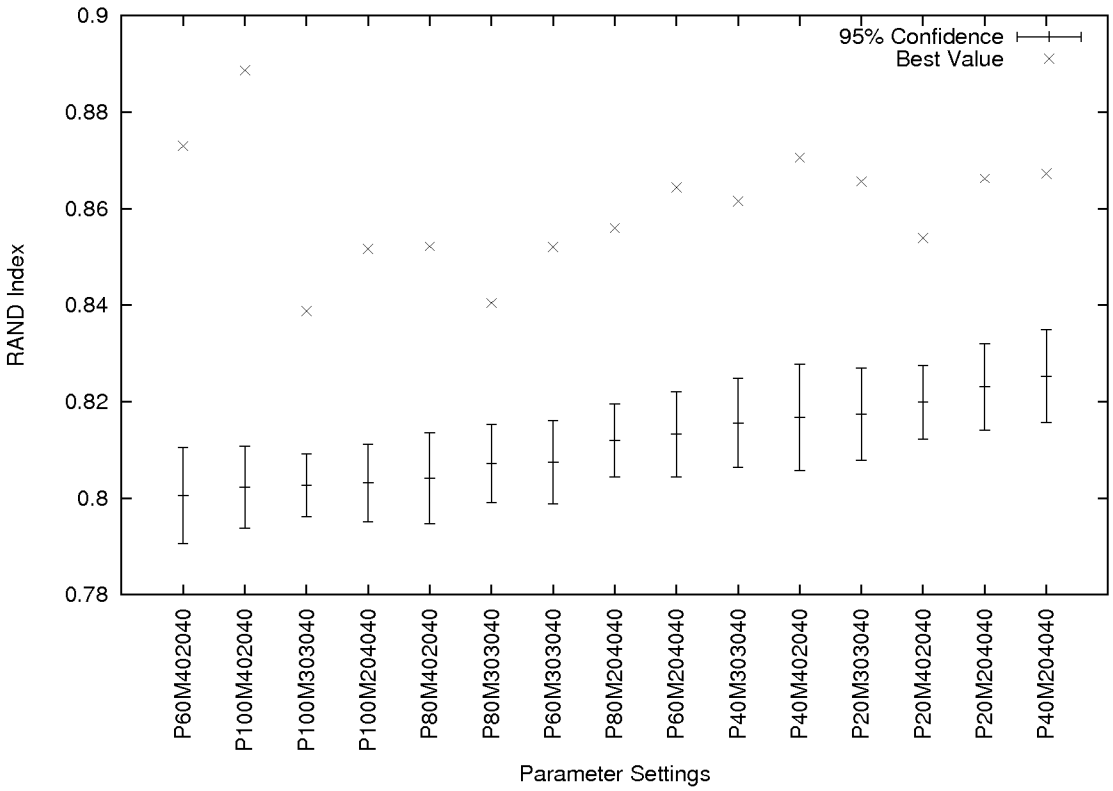


Figure 6.2: DATAONE results with a 95% confidence interval about the mean and best value. Sorted by mean from smallest to largest. The data labels are read as P number in the population, M mutation levels of operator, link, and constant change rates.

6.1.4 Results

The regressors located in all cases are highly resistant to change in the population and mutation types. Visible sub-population collapse events happened in the experiment using DATAONE and DATATWO. The Rand index scores were in the 0.8 range with the best replicate scoring 0.9 for DATAONE and 0.85 for DATATWO, as shown in Figures 6.2 and 6.3. Some of the produced regressors were modeling less than 10 of the data points, showing that the number of populations selected to model the data was too large as expected.

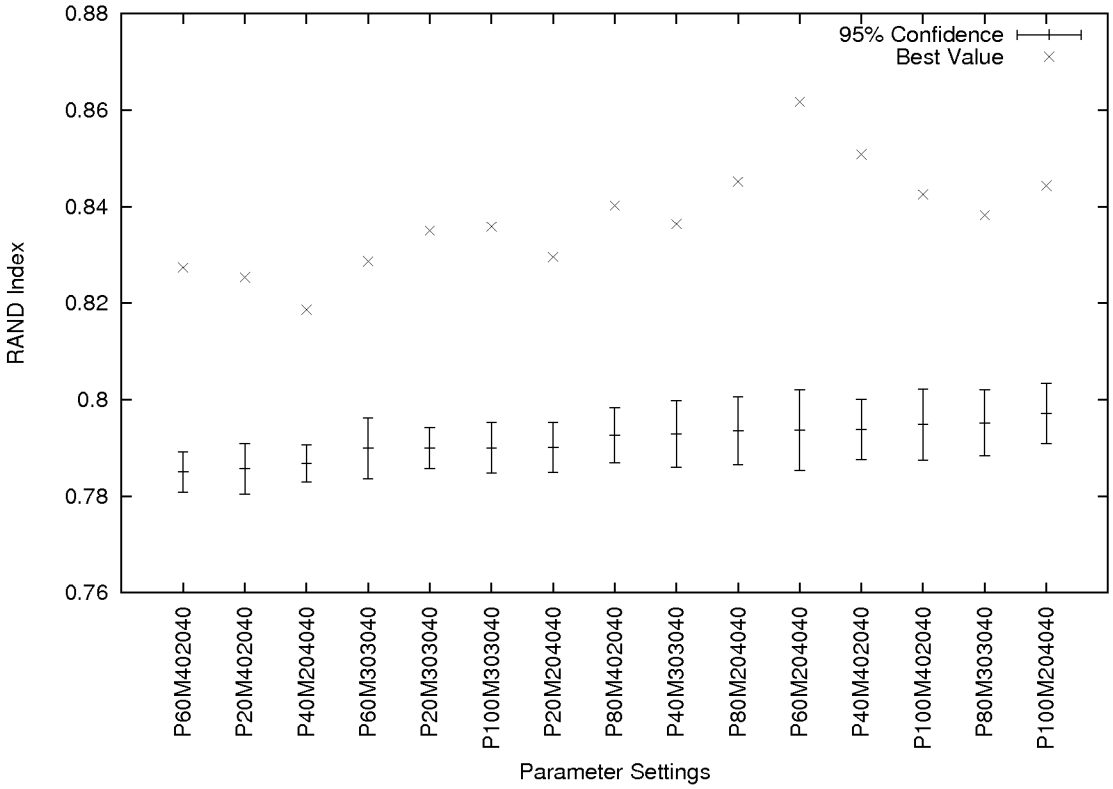


Figure 6.3: DATATWO results with a 95% confidence interval about the mean and best value. Sorted by mean from smallest to largest. The data labels are read as P number in the population, M mutation levels of operator, link, and constant change rates.

Even where the collapse did not happen within these data sets, the regressors created still kept like classes together, and each class of the three was usually modeled by two of the six regressors. While this is not the optimal situation, the removal of these like regressors, in terms of the created equation, as a post processing step would improve the Rand score.

DATATHREE did not have a prevalence of such events reaching a steady optimum just above a Rand index of 0.6, which is only slightly better than a completely random partition, see Figure 6.4. The concentric circles were given as a difficult set to model as the operations provided are insufficient to

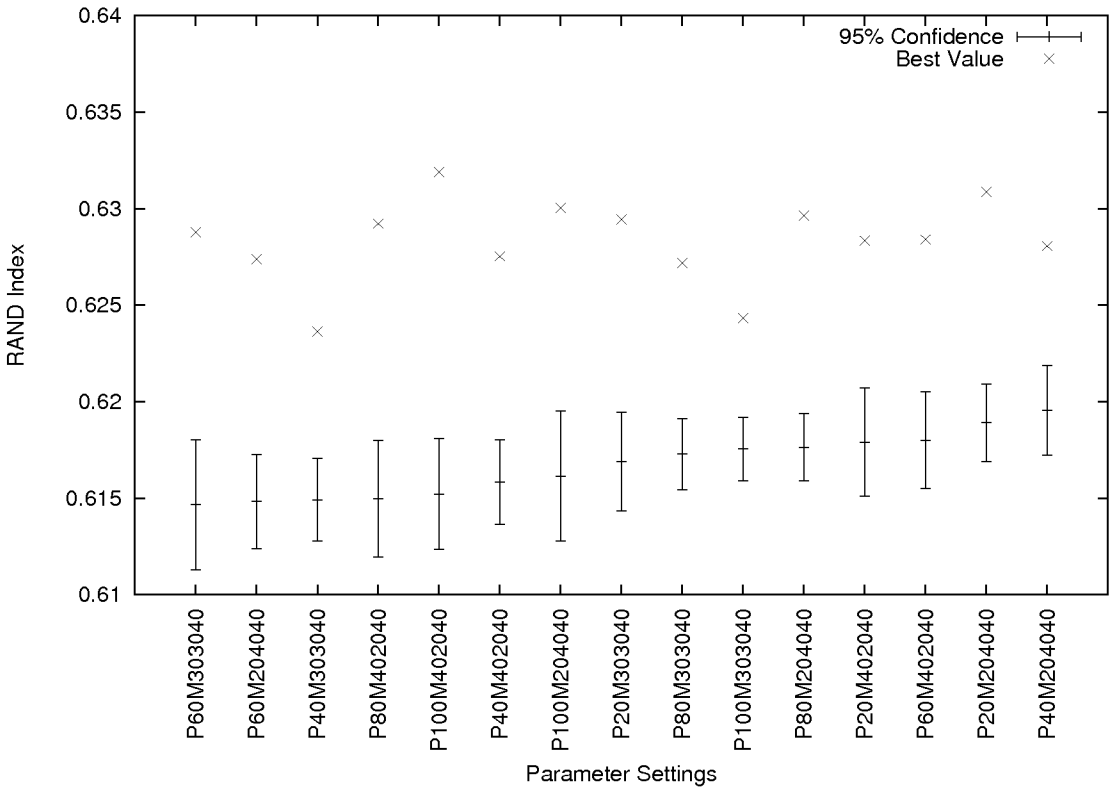


Figure 6.4: DATATHREE results with a 95% confidence interval about the mean and best value. Sorted by mean from smallest to largest. The data labels are read as P number in the population, M mutation levels of operator, link, and constant change rates.

fully model the data; they are meant for producing lines. In this instance the Rand index also suffers as the lines wish to model the circles by producing an approximation by multiple lines. As the Rand index tests the partitioning even a set of lines which is a fair approximation of the circle would gain a low score as the lines produced for the modeling of a circle would not share the same class. The addition of operations more suited to circles, such as distance to a given point, would allow for a sufficient modeling for this data set.

The starting number of populations was intentionally set to be larger

than the number of known classifications in order to explore if the properties of sub-population collapse would be evidenced. The evolved regressors not only were able to partition the data correctly into their classes but displayed sub-population collapse yielding a reasonable number of partitions. The final scored quite high on the Rand Index — which would be impossible without collapse.

The property of sub-population collapse is an answer to Shengwu et al.'s "difficult problem" of finding the number of classes or discontinuities in the data [101]. Rather than having a human decide it *a priori*, evolution naturally wants to find the appropriate number of partitions where the system will become stable and converge.

The creation of these regressors can allow for the classification of harder data sets. Once regressors are chosen, they can be used to perform an error transform. This transform maps a data point to the vector of the modeling error of multiple regressors on the data point. The transformed data should (i) be often far more separable than the original data, (ii) be coerced to have a higher or lower dimension, depending on the application, (iii) permit a simple classification of new data points that did not participate in selection of the regressors.

6.2 Comparison to a GA

For the remainder of the chapter we examine a test between a Genetic Algorithm and the Multiple Worlds Model for regression on a set of crossing lines.

6.2.1 Experimental Settings

The GA and the MWM had the following settings. The settings for the GA bias it towards diversity (larger population/weaker tournament settings) and a longer evolutionary search (more fitness evaluations). The representation

of a GA is a chromosome consisting of n lines of a slope and intercept. The MWM had n populations of a chromosome of a single line's slope and intercept. Both used the same operations for crossover, two-point. The mutation for the GA allows for up to n changes in order to be fair as MWM is allowed one per population. Fitness for the GA for a classification is defined as the minimum error loci in the chromosome. The fitness for a chromosome is the sum of the error on each of these points. Fitness for the MWM model is explained in the next section. The selection operation for both was a tournament selection which selected four chromosomes for MWM and seven for GA, and replaced the top two with the bottom two. Population size was set to 40 for MWM and 1000 for the GA. The Number of generations was set to 100 for both, this gives the GA 2.5 times the number of fitness evaluations than MWM is allowed. Experiments were run for thirty replicates.

Both the MWM model and the GA were tested on a dataset produced by adding a Gaussian error term from points drawn from three crossing lines, with 50 data points drawn from each line. Chromosome/population sizes, hence the number of assumed classifications, was ranged from one less to one more, i.e. from two to four, than the number of actual known classes in the dataset.

6.2.2 Importance of an Error Term in Fitness Evaluation

Initial tests using these settings for MWM produced findings with low Rand index scores and high error. This development was deemed to be caused by the fitness function only taking into account that a model only needs to beat the other models in the world. It does not need to provide a good model, only a better one than those about it. The fitness function was changed from being the number of points which a classifier in a world has the lowest error score, to being the number of points on which the model was closest divided by an error term for those points. The error term is the sum of the error

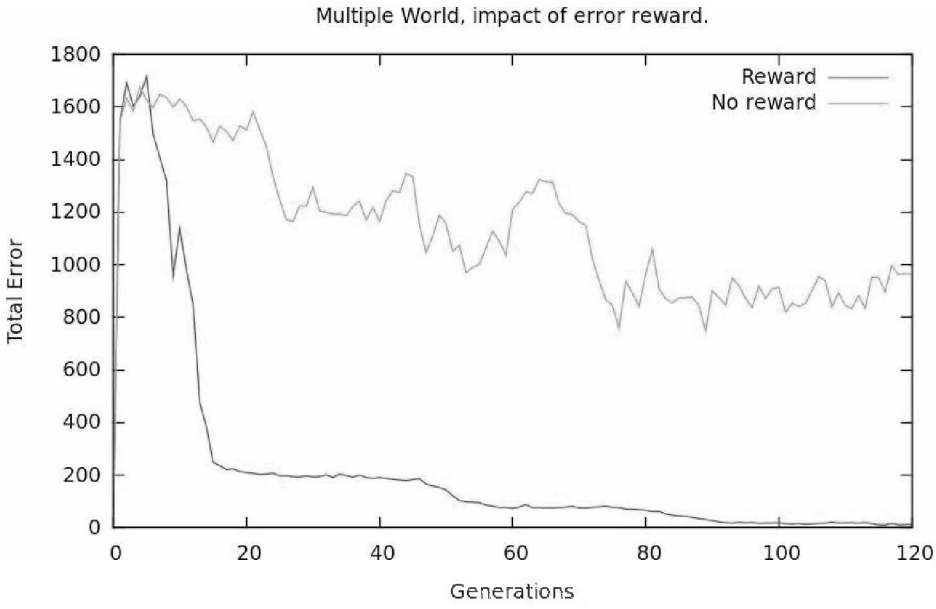


Figure 6.5: Error of the best MWM model per generation. Two fitness functions are used, one which takes into account a reward for reducing error and another that does not take error into account.

N	BEST RAND MWM	BEST RAND GA	#SOLUTIONS MWM	#SOLUTIONS GA
2	0.712	0.704	10	1
3	0.991	0.982	30	1
4	0.982	0.973	30	8

Table 6.2: Comparison between MWM and a GA for the three lines. Given is the number of Models assumed, the best Rand index value, and the number of unique solutions found.

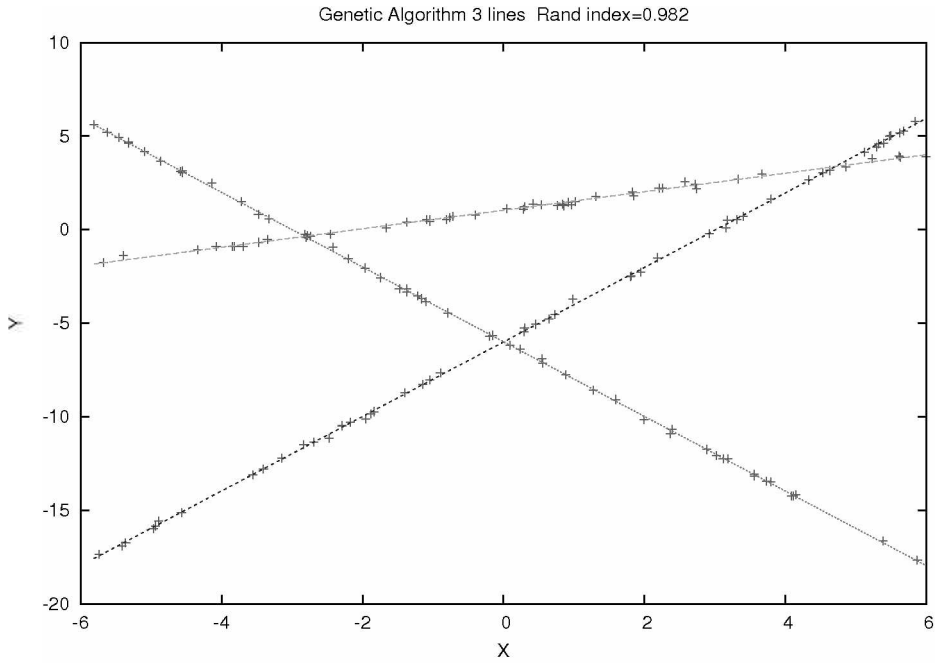
on the points plus the number of points. Hence, a point with no error is scored as a full point of fitness, as the error increases the amount scored for a point is reduced; there is a reward for reduction in error. There is now an incentive to not only have the best model for a point, but to also model the point well. As seen in Figure 6.5, the fitness function taking into account the error, provides a significant improvement in the MWM ability to provide a model with low error to the points.

6.2.3 Results

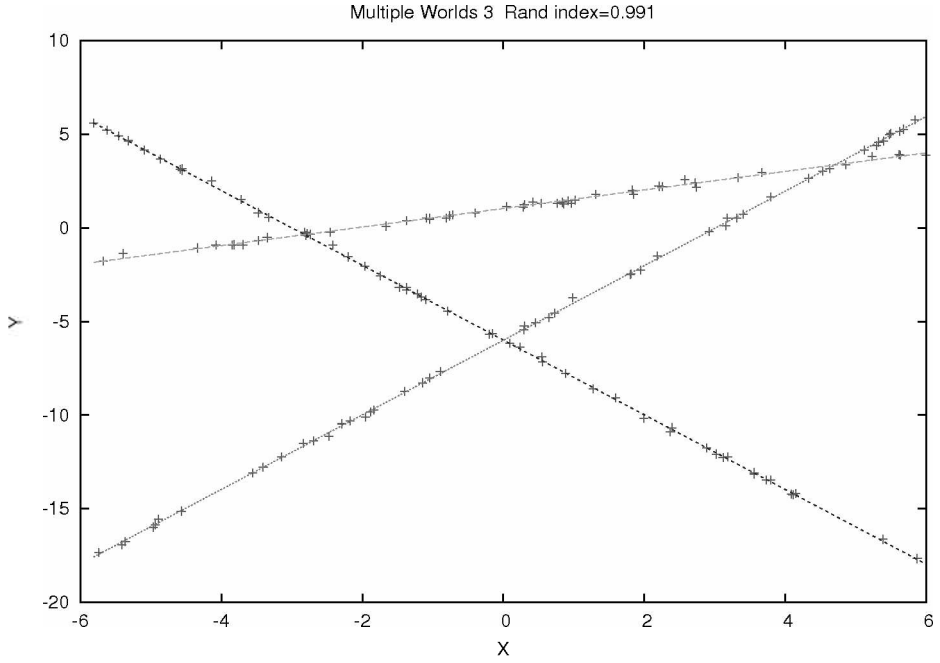
When we take into account the error term in the fitness evaluation, MWM outperforms a normal GA on this simple problem. It produces both better Rand index scores for the best solution as well as increase to the diversity in the set of possible solutions, see Table 6.2. The lines found by MWM assuming four classes found a classification with the same Rand index score as the GA given the correct number of classes.

Presented in Figure 6.6 are the best classifications of the GA and MWM for the three lines crossing, assuming correctly that 3 lines exist in the data. Both produce a good classification of the set and the majority of the difference comes from the ambiguous data points at the crossing, which also explains why the classification does not score perfectly on the Rand index. When considering the assumption of four sets of data, the MWM algorithm shows how the property of sub-population collapse allows for a better classification.

Looking at Figure 6.7(a) the extra classification set takes three datapoints from the proper classification and places a line across them. The MWM, Figure 6.7(b), has a collapsed population which does not capture a single point in the dataset. It is evident that one of the classifying lines has taken on slightly more error in order to protect a greater number of points from being stolen from another line. This protection causes the extra line to gain none of the points and have a zero fitness score. The fitness function balances the desires to classify the most points as well as the reduction of error. The GA by only wanting to reduce error will place this extra line in order to simply take a few of the points in a line with no error.

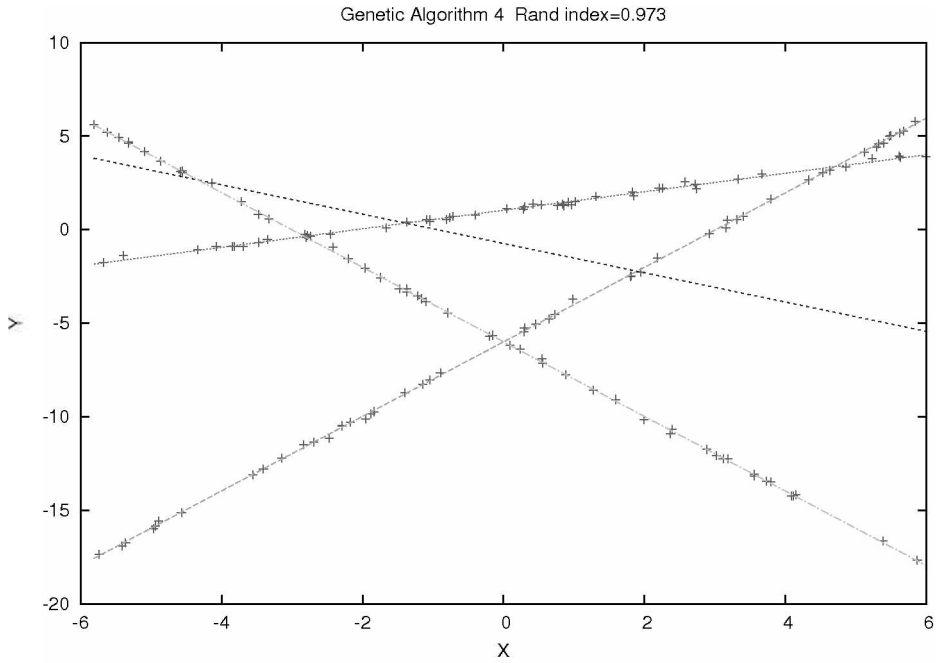


(a) GA

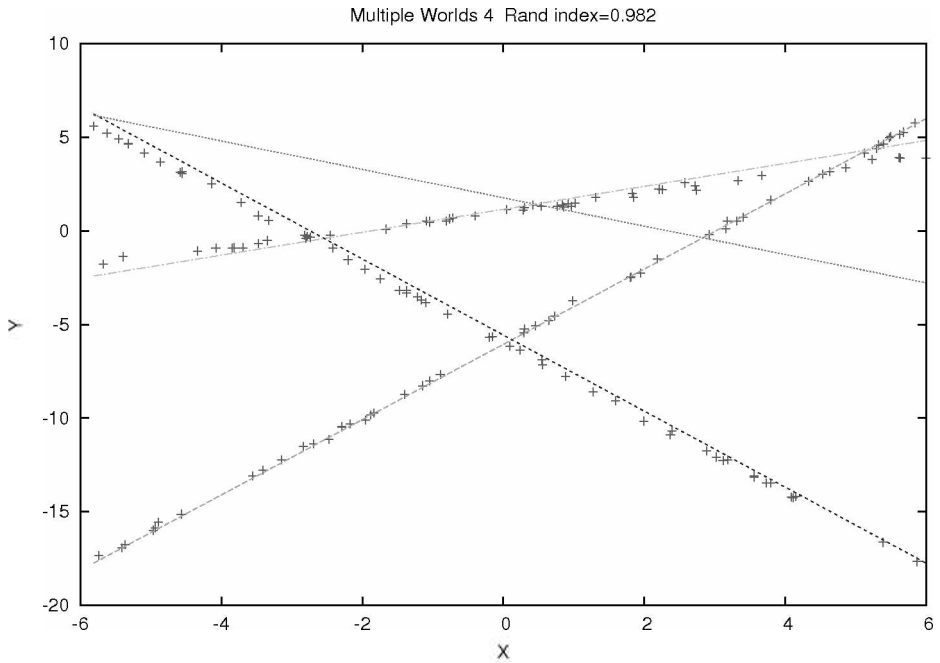


(b) MWM

Figure 6.6: Comparison of the MWM v. GA on the three lines set assuming correctly there are three models.



(a) GA



(b) MWM

Figure 6.7: Comparison of the MWM v. GA on the three lines set assuming incorrectly there are four models.

Chapter 7

Motif Discovery

This section was drawn from the two publications *Multiple Worlds Model* [22] and *More Multiple Worlds Evolution for Motif Discovery* [23].

7.1 Motifs

The goal both to find motifs known to be present, and to apply the motif to unseen examples for the discovery of the resulting motifs which may have interesting biological contexts. For example, HIV-1 differs from the human genome in the NF-κB binding region, this allows for a targeted RNA interference pathway which can suppress the virus [105]. The problem of motif finding has been examined via numerous methods such as Greedy algorithms [54], Expectation Maximization [15], Gibbs Sampling [68][97], Hidden Markov Models [66], Evolutionary Computation [67][103], and others [69]. Sequence motifs allow a biologist to classify different organisms or species, find similar functions in differing organisms, or to find sequence alignments.

There are three major biological categories for which *sequence motifs* are used: deoxyribonuclec acid (DNA), ribonuclec acid (RNA), and proteins. DNA is the information storage molecule for most forms of life on earth, other forms are RNA viruses and protein based prions. Both DNA and RNA

are long polymer backbones of nucleotides consisting of a phosphate group stripped of one oxygen atom, a sugar known as ribose and one base. It is sufficient to name each nucleotide by the base as it is the only differing location. The amino acids in a base of DNA are cytosine (C), adeine (A), guane (G), and thymine (T). RNA replaces thymine with uracil (U). The pairs interlink in many cases to form a double strand — C bonds only with G and A bonds only with T or U. As a result, it is sufficient to only look at one strand in order to determine the structure. Proteins are defined by a sequence built from 20 amino acids.

Sequence motifs, in terms of their mathematical representation, are a set of strings which consist of symbols from a set of nucleotides or amino acids. To a biologist, a sequence motif is useful if it has a biological significance to a specific organism. Motifs can be expressed in a variety of methods, such as IUPAC, PROSITE, and TRANSFAC. We employ a simple degenerate expression. A motif in this expression is a string of bases and wild cards, which substitute for any one of a set of bases. For example, the expression

$$C - [AT] - G - [GAT]$$

would decode to accept the set of strings

$$\{CAGG, CAGA, CAGT, CTGG, CTGA, CTGT\}.$$

This degenerate representation is used as a backbone for IUPAC-code, which maps each of the 15 base combinations to a single letter. Therefore, the IUPAC-code is just a compression; the IUPAC-code is not used in this study as the degenerate expression allows for an easy comparison of the classifiers for which bases appear in wild cards. The PROSITE notation uses the IUPAC-codes with an extra operator that allows for repetitions of the same symbol to be represented. For example, the set of {AA,AAA,AAAA} would be represented by A(2,4).

All these expressions are restrictive in terms of representational ability. Some biologists have stated that degenerate motifs are the same as regular expressions, which is false. Looking at the *STOP* codon provides one of the most simple expressions of the limitations of degenerate motifs. The set of {TAG,TAA,TGA}, also referred to as amber, ochre, and opal respectively, cannot be represented as a degenerate motif. The smallest consensus motif for *STOP* is T-[GA]-[GA], which includes TGG, or the codon of Tryptophan. The *STOP* codon cannot be represented by degenerate motifs, IUPAC, or PROSITE. Degenerate motifs, IUPAC, and PROSITE do not have the representational ability of regular expressions; instead they assume an *independence* between symbols. To avoid this error, the representation could be extended to take a full codon as an atomic element. However, such a change to a codon-based representation would just push the problem to an assumption of independence between codons and would require a set of 4^3 atomic symbols.

One representation which provides a visualization of the relation between bases in a motif is TRANSFAC. The TRANSFAC matrix representation uses a degenerate expression, and will give a frequency of occurrence of each base within sequences to produce the motif. The frequencies give more information to the user about which strings are included or not included in the motif. TRANSFAC attempts to avoid the issue of the independence assumption. The user can extract upper and lower bounds on occurance of symbols in relation to each other. This extraction method would allow for a range of dependence levels to be discovered. Futhermore, as TRANSFAC provides a degenerate motif to match against, the information in the relational matrix is likely not used in an application. Not using this relation matrix in application would cause the same problems of an independance assumption as the other methods.

Given these flaws in the representation, the questions arise as to why these representations should be in use, and why tools should be made using

such representations. First, there is the practical difficulty of changing a large number of databases which use one of these existing representations. Second, the current methods are human-readable; a biologist with little training in formal grammar can understand the use of these systems. Third, the assumption might not be a flaw even if the data comes from a restricted set of regular languages with independence, or a very weak dependence between symbols.

7.2 Experimental Settings

The fitness measure for a motif must cause the maximal number of matches to occur, but must prevent motifs from only winning if they are more general. As the source of a motif from a population must be evaluated in terms of a world, a simple number of matches is not appropriate as a measure. In this case, the motifs would become just [CGAT] repeated. There must be some penalty for overgeneralization. To provide this penalty, the following method is used: the motif is tested against each string to be classified; the number of times the motif matches this string is found as well as the number of possible string which the motif would match. We will call this number the *latitude* of the motif, and the score of the motif upon the string is the number of matches over latitude. The motif with the highest score in a world wins that data point. This is to both encourage a motif to fit a number of points, as well as preventing a motif from gaining all the points by just placing a sufficiently general motif over the set of strings; [CGAT] as the most general motif has a high matching ability but a maximal latitude.

For example, if the motif to be matched was [CGAT]-C-[GA] and we had the string CCACG, then we would calculate the score as follows: first, the motif has a latitude of 8, as it matches the set of strings {CCG, CCA, GCG, GCA, ACG, ACA, TCG, TCA}. Second, it matches the string twice, with CCA from index one and ACG from index three. Hence, the final score for

this motif on this string is $\frac{2}{8}$. A more exact motif would be [CA]-C-[AG], which would score $\frac{2}{4}$. More general would be [CGAT]-[CGAT]-[CGAT], which would score $\frac{3}{16}$.

The populations each have 100 members consisting of degenerate motifs of size 3, 5, and 7, which where initialized randomly, with each symbol occurring in equal likelihood. For the self-driving finite state machines and the HLA sets, described below, also examined was a motif length of 9. The selection for breeding is a tournament which takes four members of the population, orders them based on their fitness, and replaces the bottom two with replicates of the top two. The copies then undergo mutation and crossover. The crossover operator is a two point crossover. It probabilistically selects two points in the motif and swaps the symbols between the selected positions. The mutation operator randomly assigns a new wildcard to each symbol within a motif with 10% probability. To allow for statistical analysis of the results, 30 replicates with differing initial random generator seeds were run for each testing data set.

In all cases, one more than the actual number of classes of data is used. This is in order to examine the property of sub-population collapse and as will be explained is benefical to the algorithm's solutions. If the score of a population is less than 10% of the total number of points we say that the sub-population has collapsed.

7.3 Data sets

This section looks at the various data sets used in order to test the MWM model for the discovery of motifs.

7.3.1 GC Content

This synthetic data set is modeled on random DNA strings, created in two classes, each of size 1000. The first set has approximately 40% GC content,

the other has approximately 60% GC content. Sequences of length 250 with these exact percentages were created, randomly shuffled, and a prefix of length 150-250 was selected uniformly at random from the sequence. This prefix selection means that the GC content has a binomial distribution and the two classes overlap at the boundary.

7.3.2 Reverse Complement Motifs

The reverse complement data set consists of 1000 uniform random samples of DNA strings, each with an embedded motif. This data set represents DNA with a meaningful motif whose DNA strand is not known. The two classes of embedded motifs are the reverse complement of each other.

DNA is written from the 5' end to the 3' end. Two strands bond from end to end to form a double helix so the reverse complement of a stand is the strand which would bind in the double helix.

As an example:
(5') CAT (3')
(3') GTA (5') after complement
(5') ATG (3') after reverse

7.3.3 Self-Driving Markov Automata

A *self-driving* Markov automata is a finite state transducer with the same input and output sets, an example is shown in Figure 7.1. The automata has a probability associated with each state. This probability distribution determines which symbol is emitted as input/output. The machine begins at state zero and emits a symbol. This symbol is then used in a feedback loop to transition the machine into its next state and a new symbol is emitted. This process continues until a DNA state of the desired length is generated. If the probability distributions on the states were uniform, then the data generated would be indistinguishable from that generated by a single uniformly

distributed random generator of DNA. Controlling the Entropy of the distribution on each state acts as an effective method of creating a tuneable level of structure in the data.

The *Shannon entropy*[78] of a discrete distribution with probabilities $p_1, p_2, \ldots, p_m$ is given in Equation 7.1.

$$E = -\sum_{i=1}^{m} p_i log_2(p_i) \tag{7.1}$$

If the discrete distribution is sampled a large number of times, then the Shannon Entropy is the average number of bits required in order to describe one of the samples. As the DNA alphabet has 4 symbols, a uniformly random sample would require 2 bits to represent a base. Distributions were created by generating four integers in the range 1-100 and then dividing by the sum of the integers to give a final probability of each of the four DNA bases. In order to get bounded entropy distributions, this process is repeated until one, which satisfies the bound, is discovered.

The datasets created had 3 and 4 classes respectively with 500 examples. A dataset of each number of classes was created with a maximum entropy level in the finite state machine generation of 1.4, 1.6, 1.8, and 2.0.

7.3.4 Self-Driving Finite State Machines

Self-Driving Finite State Machines are an extension of finite state machines. A Mealy FSM [79] is a transducer that operates over a finite alphabet of input symbols and responds from a finite alphabet of output symbols, as it traverses a finite number of states. It is defined as a five-tuple, $\langle Q, I, Z, \delta, \omega \rangle$, where Q is the set of states, I is the set of inputs, Z is the set of output symbols, δ is the state transition function, such that $\delta : I \times Q \rightarrow Q$, and ω is the output values, such that $\omega : I \times Q \rightarrow Z$.

In order to make a machine self-driving, the output of a machine is fed

	Emission Probabilities				Next State if			
State	$P(C)$	$P(G)$	$P(A)$	$P(T)$	C	G	A	T
0	0.439	0.023	0.374	0.164	1	1	6	1
1	0.255	0.014	0.284	0.447	4	7	5	7
2	0.275	0.067	0.101	0.556	4	2	2	4
3	0.176	0.536	0.242	0.046	5	2	0	0
4	0.075	0.434	0.305	0.186	6	5	3	6
5	0.363	0.004	0.386	0.247	4	4	7	2
6	0.469	0.097	0.159	0.276	0	3	5	1
7	0.034	0.363	0.346	0.257	3	1	5	3

Example Output:

TACTCTTAAGCGAATTAACAAGACC-
GAACAGACTTCCGTTGTTTGACTGT-
ACAGTGTGCAGCCCCCAGTGCAGCG-
AACGAATTTCAACAGGTGAGATTTT

Figure 7.1: An example self-driving automata used to create strings with entropy 1.8. (top) and an example DNA string generated via the finite state machine (bottom).

back into the input of the machine. Ergo, the output drives the input, creating a progressively longer output string. The sets are created by machines using C as the initial symbol to start the machine, taking {C,G,A,T} with an output alphabet of $\lambda \cup \{C,G,A,T\} \cup \{C,G,A,T\}^2$, i.e. the machine can output between zero and two bases per output. This makes the output outside of the space of regular expressions, as it can create languages which cannot be recognized by a deterministic FSM (see proof in Appendix A). The first 50 bases of output of a machine are removed. This allows for the machine to produce a more settled output, similar to burning in a Markov chain.

A set of 50 different machines were created in order to allow tests on this system and others. Figure 7.2 shows the various data sets placed in a neighbour joining taxonomy based on the 3-mer spectrum string kernel. This is calculated by taking the normalized occurrences for each possible 3-mer as a 64-element descriptor. We selected a few examples for testing. The first are DS0 and DS1, selected primarily as they were just labelled as the first two and hence made an initial testing bed for the software and analysis methods. The second are DS25 and DS37, which have a large distance between the 3-mers. The third are DS25 and DS9, beside each other on the joining tree. The closer on the tree two sets are, the closer they are in terms of 3-mer distance, and therefore the more difficult the classifications should become.

7.3.5 Human Leukocyte Antigen

A major histocompatibility complex in humans is the human leukocyte antigen system (HLA). A large number of genes related to the immune systems rests in this locus. The HLA class I antigens (A, B, C, D, E, E, F, and G) are peptides from inside the cell, which might contain viral peptides if present. These peptides are created from digested proteins which have been broken down by the proteasomes. Each of the peptides is a small polymer of about 9 base in length. T-cells, also called CD8 positive cells, are hunter-killers of cells containing foreign antigens. The HLA class II antigens (DRA, DRB,

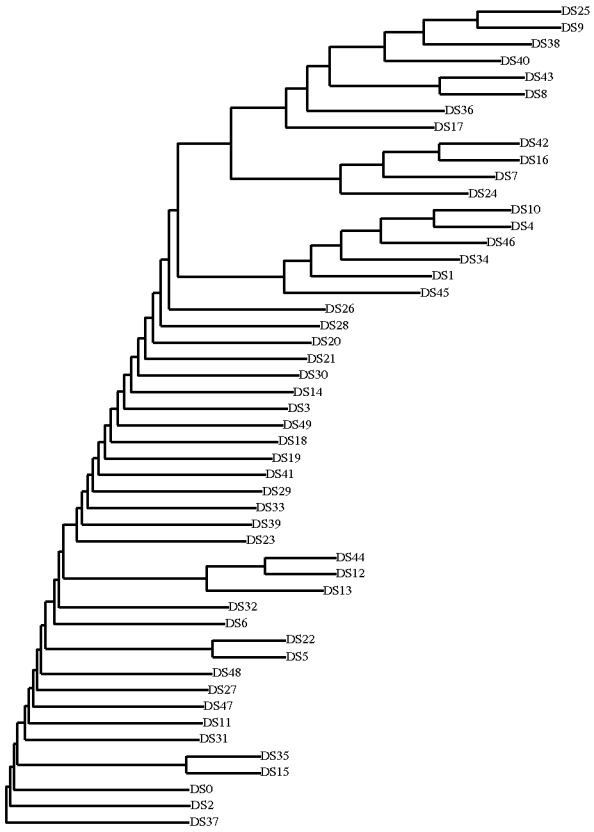


Figure 7.2: Neighbour joining taxonomy based on the 3-mer spectrum string kernel distance on the generated self-driving FSM. Note the used data sets locations: DS25 and DS37 have the largest distance, DS25 and DS9 are beside each other in the tree.

DQA1, DQB1, DPA1, DPB1, DMA, DMB, DOA, and DOB) are antigens which are outside the cell to the T-lymphocytes. These antigens act as a signal to stimulate T-helper cells to undergo mitosis; this stimulates antibody production by B-cells, and the production of antibodies to that specific antigen. The major histocompatibility complex gene products are involved in the pathogenesis of many diseases, including autoimmune problems.

The data set contains 500 examples from the HLA class I antigens and 500 examples from the HLA class II antigens. This set was produced from data in the IMGT/HLA Database. This database is part of the international ImMunoGeneTics project and provides a specialist database for sequences of the human major histocompatibility complex. In addition to sequences, the database contains both information on how each sequence was derived and data on the validation of the sequences (http://www.ebi.ac.uk/imgt/hla/).

7.4 Results

7.4.1 GC Content

sub-population collapse is a rare event to be observed with the GC data set. Only two replicates in the size three motif, showed a reduction in the classes down to the correct number of two. As expected, those with a collapse event had much higher Rand index scores. The best motif set, as shown in Table 7.1 clearly shows that the middle motif of [CGAT]-[CGAT]-[GA] has collapsed. The [CGAT] being the universal set is in many ways the motif making no guess. The 40% class is represented by [CAT]-[GT]-[AT] which has an AT richness compared to the motif of G-[CA]-[CG] which classifies the 60% GC content class.

The smaller sized motifs found better classifiers for the data, most likely as they have to account for less dependency between symbols. As we are looking for the number of hits over latitude, smaller variations on a smaller search space allow for cleaner classifications between the sets.

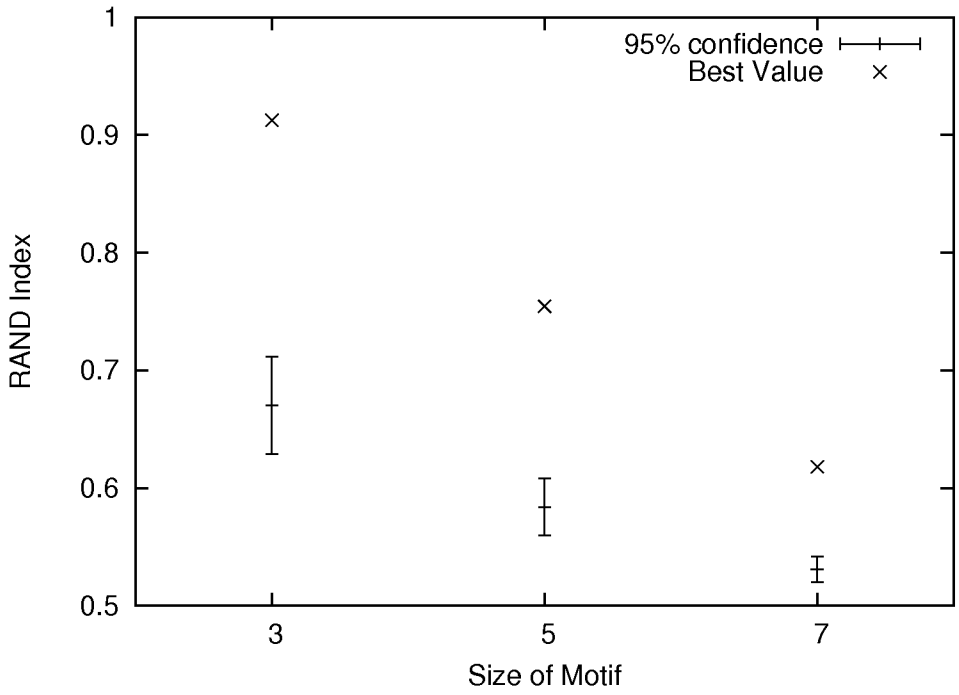


Figure 7.3: 95% Confidence intervals and best value of Rand index for the GC content data set

The lower results in Rand index scores did not necessarily come from missclassification. In many of the cases it was evident that two populations were splitting a class as the third took all of the other class, which is a lack of a sub-population collapse, see Table 7.2. The species in this situation have found an equilibrium where they both have found an extremely small niche. The difference can be as small as a single base in the motif.

7.4.2 Reverse Complement Motifs

In general the motifs hover about 0.5 on the Rand index, see Figure 7.4. This basically means that the two classes are completely split between the created motifs. The embedded motifs are reverse complements and the remainder of the string is completely uniform, therefore, the only information for which

Table 7.1: Best motif set found of size 3 with scores for the two classes of 40% and 60% GC content.

Class	Degenerate Motif and Fitness Score		
	[CAT]-[GT]-[AT]	[CGAT]-[CGAT]-[GA]	G-[CA]-[CG]
1	9	76	920
2	916	79	5
Total	925	155	920

Table 7.2: Examination of the levels of sub-population collapse in the GC dataset by presenting the number of populations with greater than 10% of the dataset out of 30 replicates.

	Populations		
Motif Length	1	2	3
3	0	2	28
5	0	0	30
7	0	4	26

Table 7.3: Best motif set found of size 5 with scores for the two classes for the Reverse Compelement Motif

Class	Degenerate Motif and Fitness Score		
	[GA]-[GAT]-[CGA]-G-G	[CA]-G-[AT]-T-T	C-[GT]-[CA]-[AT]-C
1	140	735	125
2	337	383	280
Total	477	1118	405

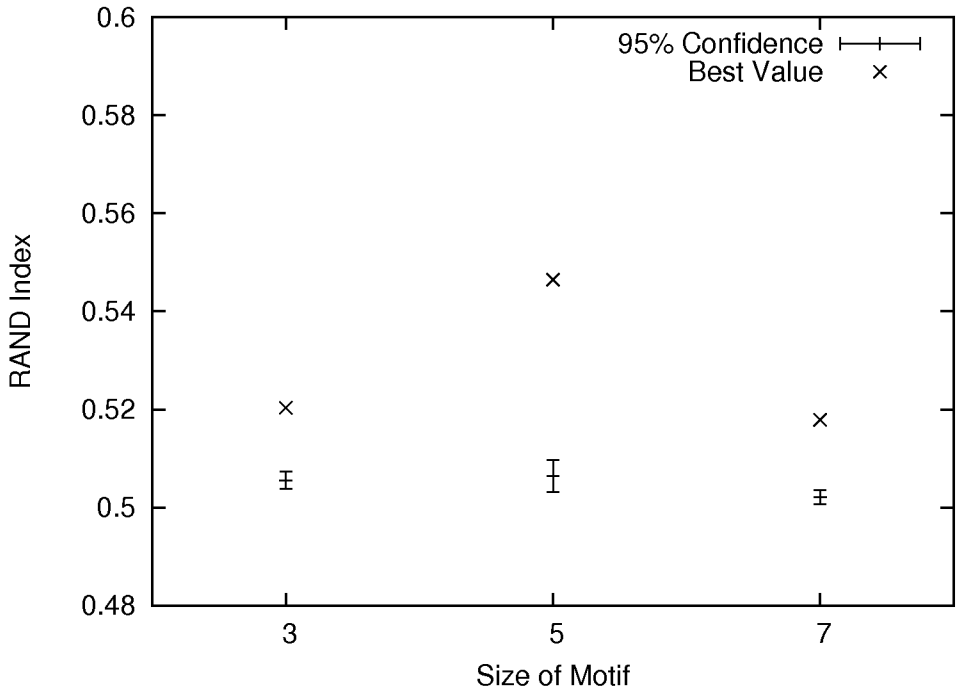


Figure 7.4: 95% Confidence intervals and best value of Rand index for the Reverse Complement Motif dataset

the generated motifs can classify with comes from the reverse complement. In general these generated motifs are close to the reverse complements of themselves, self complements. This explains why the classes are near equally split for the majority of replicates. The fitness difference between the motifs is therefore generated based upon latitude and random noise in the dataset.

Looking at the best motif found, see Table 7.10, this result is able to place the majority of the first embedded motif together, but has problems classifying the other correctly. The high scoring motif does have a comparatively low latitude when compared to the others in the set. No sub-population collapse is evident, again as the random noise and latitude does this partition.

7.4.3 Self-Driving Markov Automata

As evidenced in the GC data, there is a large divergence between the best populations which have had some form of sub-population collapse to those which have not. The Self-Driving Markov data continues this trend of a collapse happening within the best scoring motif classifiers. In the three class Self-Driving Markov data, see Figure 7.5, there is again a large difference between the mean and best valued classifiers.

Looking at the best results in Table 7.4, each of the entropy levels, excluding 2.0, exhibit sub-population collapse. However, notice how the motifs in the maximum entropy 2.0 results have divided the classes correctly, excepting that the first and third motifs have shared the third class. The first motif, CTG, is a proper subset of the third motif, [CT]-[CGAT]-G. In this case we have two populations which are fighting over the same resources and have been able to reach an equilibrium on a limited resource. This leads to one species to be a specialist type of the other species, a *nested degeneracy*. The first motif with a small latitude is able to resist collapse as it is extremely specialized. CTG must be a very good signifier of this class as it is able to support an entire population. [CT]-[CGAT]-G on the other hand is resistant to noise and it is able to pick up on the general content of CTG

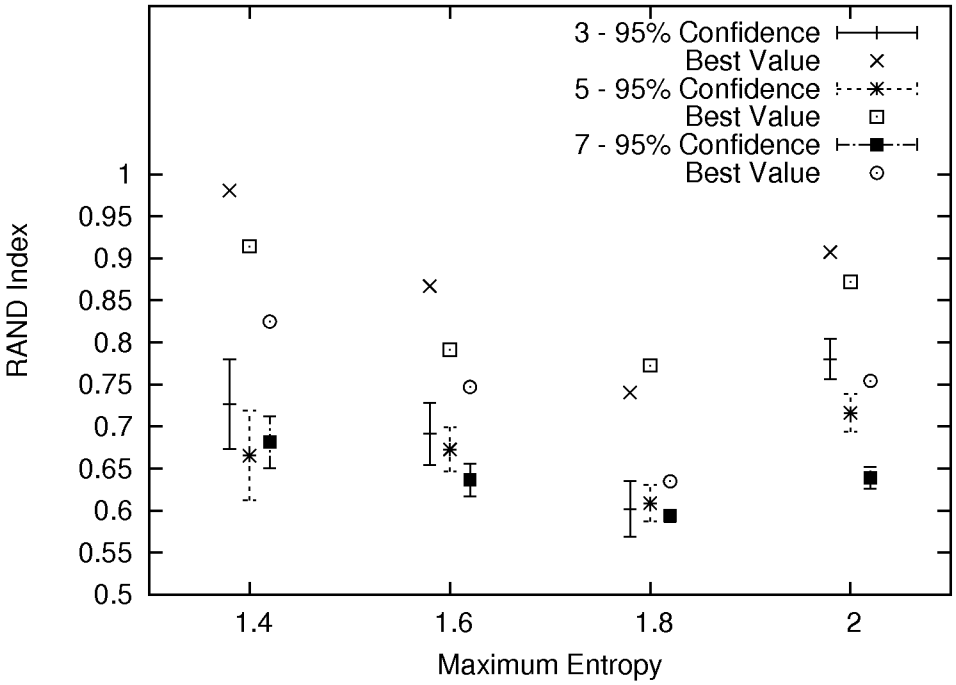


Figure 7.5: 95% Confidence intervals and best value of Rand index for the Self-Driving Markov datasets with three classes

Table 7.4: Best motifs for the three class Self-Driving Markov data sets

Class	Degenerate Motif and Fitness Score			
Best motif set for 1.4 maximum entropy				
	[AT]-[GAT]-[CA]	[AT]-[CA]-[CGA]	[GA]-[CG]-[GT]	T-[CGA]-[CAT]
1	1	2	497	0
2	3	497	0	0
3	0	18	18	482
Total	4	499	515	482
Best motif set for 1.6 maximum entropy				
	[CT]-GT	GAG	G-[CG]-[CA]	[CAT]-AT
1	357	56	87	0
2	0	471	16	13
3	58	0	0	442
Total	415	527	103	455
Best motif set for 1.8 maximum entropy				
	A-[GT]-AG-[CAT]	[GT]-[CG]-G-[GAT]-[CAT]	[CA]-[GAT]-C-[CGAT]-[GA]	[CA]-[CGAT]-AGT
1	8	15	133	344
2	374	71	1	54
3	73	382	9	36
Total	455	468	143	434
Best motif set for 2.0 maximum entropy				
	CTG	[CGA]-C-[CAT]	[CT]-[CGAT]-G	[CGAT]-[GAT]-[AT]
1	5	487	4	4
2	44	0	0	456
3	213	2	283	2
Total	262	489	287	462

Table 7.5: Examination of the levels of sub-population collapse in the three classed Self-Driving Markov dataset by presenting the number of populations with greater than 10% of the dataset out of 30 replicates.

		Populations			
Motif Length	Entropy	1	2	3	4
3	1.4	1	18	9	2
	1.6	1	9	16	4
	1.8	3	8	11	8
	2.0	0	2	11	17
5	1.4	4	9	13	4
	1.6	1	5	14	10
	1.8	0	7	19	4
	2.0	0	0	5	15
7	1.4	1	6	14	9
	1.6	0	0	9	21
	1.8	0	1	14	15
	2.0	0	0	3	27

richness as well as perturbations in the class. The first motif by having such a low latitude is also able to gain members of the other classes. If this set of motifs was reduced by removing CTG, those points in class three are given [CT]-[CGAT]-G. Looking at the other motifs in the set, [CGA]-C-[CAT] and [CGAT]-[GAT]-[AT], while both are very similar in both the first and third symbols, the second symbols are inverses of each other.

The collapses become less predominate in the best motif classifiers as the entropy increases, from a total of 4 in 1.4 to that of 14 in 1.8. Increases in the size of motifs also lead to lower Rand scores, though this does not become statistically significant until a maximum entropy of 2.0. The number of collapses across the replicates, see Table 7.5, shows that smaller motifs are more likely to result in underestimating the number of classes. This table further confirms the previous result showing that less collapses happen as the entropy increases.

The four class data, Figure 7.6, again shows that the smaller motifs in

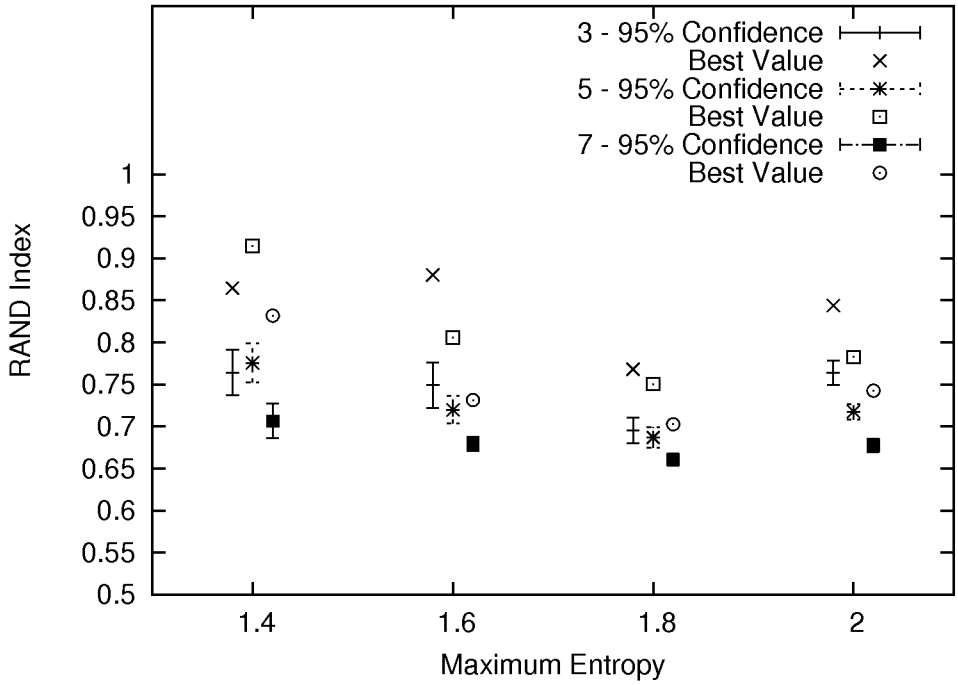


Figure 7.6: 95% Confidence intervals and best value of Rand index for the Self-Driving Markov datasets with four classes

Table 7.6: Best motifs for the four class Self-Driving Markov data sets

Class	Degenerate Motif and Fitness Score				
Best motif set for 1.4 maximum entropy					
	[CGAT]-[CGA]-CA-[CGA]	[GA]-G-[CA]-[CG]-[GAT]	G-[CT]-[GA]-[AT]	[GAT]-T-[CAT]-[GA]-[AT]	TGG-[AT]-A
1	77	319	23	60	21
2	497	0	0	0	3
3	0	0	3	480	17
4	0	8	484	0	8
Total	574	327	510	540	49
Best motif set for 1.6 maximum entropy					
	[CA]-[GAT]-A	[CA]-[CGAT]-[GA]	[CG]-[AT]-[CG]	AGC	[CGT]-[CA]-[CT]
1	0	0	486	13	1
2	93	0	4	397	6
3	89	0	0	4	407
4	440	6	36	0	18
Total	622	6	526	414	432
Best motif set for 1.8 maximum entropy					
	[AT]-G-[CGAT]	A-[CGT]-T	[CGT]-[AT]-[CT]	[GT]-[GAT]-A	A-[CGAT]-[GAT]
1	38	357	16	27	62
2	69	160	80	187	4
3	0	119	278	96	7
4	483	1	6	10	0
Total	590	637	380	320	73
Best motif set for 2.0 maximum entropy					
	A-[CGA]-[GA]	[GAT]-[CAT]-[CT]	[AT]-[CG]-[CGA]	[GT]-[CG]-[CT]	[CGT]-[CAT]-[CAT]
1	24	29	54	22	371
2	500	0	0	0	0
3	0	47	86	367	0
4	51	285	0	69	95
Total	575	361	140	458	466

Table 7.7: Examination of the levels of sub-population collapse in the four classed Self-Driving Markov dataset by presenting the number of populations with greater than 10% of the dataset out of 30 replicates.

		Populations				
Motif Length	Entropy	1	2	3	4	5
3	1.4	0	5	22	3	0
	1.6	0	3	12	13	2
	1.8	0	4	12	12	2
	2.0	0	0	6	18	6
5	1.4	0	2	11	16	1
	1.6	0	2	4	20	4
	1.8	0	1	7	15	7
	2.0	0	0	2	11	17
7	1.4	0	2	15	11	2
	1.6	0	1	8	13	8
	1.8	0	0	2	19	9
	2.0	0	0	2	9	19

general give better classifications; lengths 3 and 5 are both statistically significant compared to length 7 motifs. The length 3 motifs are statistically significant against length 5 when the entropy level increases to 2.0. Length 5 motifs find the best solution for maximum entropy of 1.4, length 3 motifs find the best otherwise. In general the size 4 classifier sets have less variance than the 3 class sets, as well as higher mean scores. The best classifiers, shown in Table 7.6, all have a sub-population collapse. The collapse is especially evident in the results for the 1.4 and 1.6 maximum entropy classes; the 1.6 class having a collapsed motif classifier with a total score of 6 points. Examining the levels of collapse for all replicates, see Table 7.7, we have the same findings as the three class data. That being that larger entropy sets have lower levels of collapse, and longer classifiers have less likelyhood of collapse.

In general, there is a decrease in the Rand index score of the classifiers as the size of the motif increases. This could be in part due to the fitness function. Looking at the number of matches over the latitude, a shorter

motif, holding all else equal, will find more matches on a string. Secondly, as the length of a motif increases, it is more likely two different motifs will have the same fitness score on a string. This does not allow an evolutionary algorithm, which bases decisions on relative fitness of members, a clear path to a solution. Third, there is a reduction in the search space as the motif becomes shorter. The larger motifs can find the same results, with the end of the string being the wildcard [CGAT], however, this symbol is very unlikely for an evolved motif to contain as it causes a large increase to latitude. Thus, larger length motifs will be likely to avoid the same solution as smaller motifs. Subcollapsed motifs generally conform to either becoming extremely specific, giving them a low latitude, or become extremely general, giving them a high number of matching sequences.

Finally, a test was undertaken in order to see the difference in the gain between the best fitness score made by a motif in a set, and the second best fitness scoring motif. This fitness score again being the number of times a motif matches over the latitude of the motif. Each of the final sets of motifs for the 30 replicates was rerun over the data, saving the fitness score of each motif on each DNA string. For each DNA string the difference between the best fitness score and the second best fitness score was taken and then averaged over all the strings. This gives the average gain in fitness between the classifiers, which acts as a measure of the separation between the classifiers. Looking at Table 7.8 the mean fitness gain decreases as the entropy increases. This result is statistically significant between 1.4, 1.6–1.8, and 2.0 for both number of classes and length. There is not a significant difference between 1.6 and 1.8. As the motif length increases the mean fitness gain decreases, this finding being statistically significant. Finally, between the three and four class data, as the number of classes increases their is a decrease in the mean fitness again. This last result is to be expected as there is more motifs to compare fitness against.

Table 7.8: Mean fitness gain between the best and second best classifications motifs using 30 replicates with 95% confidence intervals about the mean for the Self-Driving Markov automata datasets

Number of Classes — 3		
Motif Length	Entropy	Mean with 95% CI
3	1.4	2.430980 ± 1.04582
	1.6	0.967062 ± 0.361151
	1.8	0.801203 ± 0.253527
	2.0	0.274499 ± 0.111344
5	1.4	0.3409630 ± 0.288443
	1.6	0.0947763 ± 0.0210429
	1.8	0.0936480 ± 0.0412531
	2.0	0.0255115 ± 0.00703034
7	1.4	$0.01508660 \pm 0.00589601$
	1.6	$0.00585015 \pm 0.00210379$
	1.8	0.00523477 ± 0.0018068
	2.0	$0.00257909 \pm 0.00101677$
Number of Classes — 4		
Motif Length	Entropy	Mean with 95% CI
3	1.4	1.878950 ± 0.68789
	1.6	0.542339 ± 0.220126
	1.8	0.643860 ± 0.160495
	2.0	0.275987 ± 0.0927815
5	1.4	0.1017630 ± 0.0354012
	1.6	0.0589268 ± 0.0148564
	1.8	0.0430679 ± 0.0124636
	2.0	0.0213955 ± 0.00659435
7	1.4	$0.01526320 \pm 0.00600413$
	1.6	$0.00480588 \pm 0.00132744$
	1.8	$0.00353933 \pm 0.00129723$
	2.0	$0.00220744 \pm 0.000917727$

Table 7.9: Examination of the levels of sub-population collapse in the three classed Self-Driving Markov dataset by presenting the number of populations with greater than 10% of the dataset out of 30 replicates. Further listed is the number of perfect classifications — defined as a Rand score of 1.

DS0 and DS1				
Motif Length	1	2	3	Prefect
3	5	14	11	1
5	6	14	5	4
7	8	20	2	6
9	18	11	1	1
DS25 and DS37				
Motif Length	1	2	3	Prefect
3	12	17	1	1
5	9	20	1	2
7	13	15	2	0
9	16	12	2	1
DS9 and DS25				
Motif Length	1	2	3	Prefect
3	11	13	6	0
5	10	16	4	0
7	10	19	1	2
9	19	10	1	1

7.4.4 Self-Driving Finite State Machines

The tests for self-driving FSM gave excellent classification. In each of the test cases, each of the examples found at least one perfect classification for the 80 data points. This is surprising, given the representational ability of degenerate motifs as well as the method of generation allowing for languages outside of the bounds of regular languages. Examining the levels of sub-population collapse in Table 7.9, it is clear that the correct number of classes is discovered, or a full collapse is more prone to occur. Specific findings for each of the data sets are examined below.

DS0 and DS1

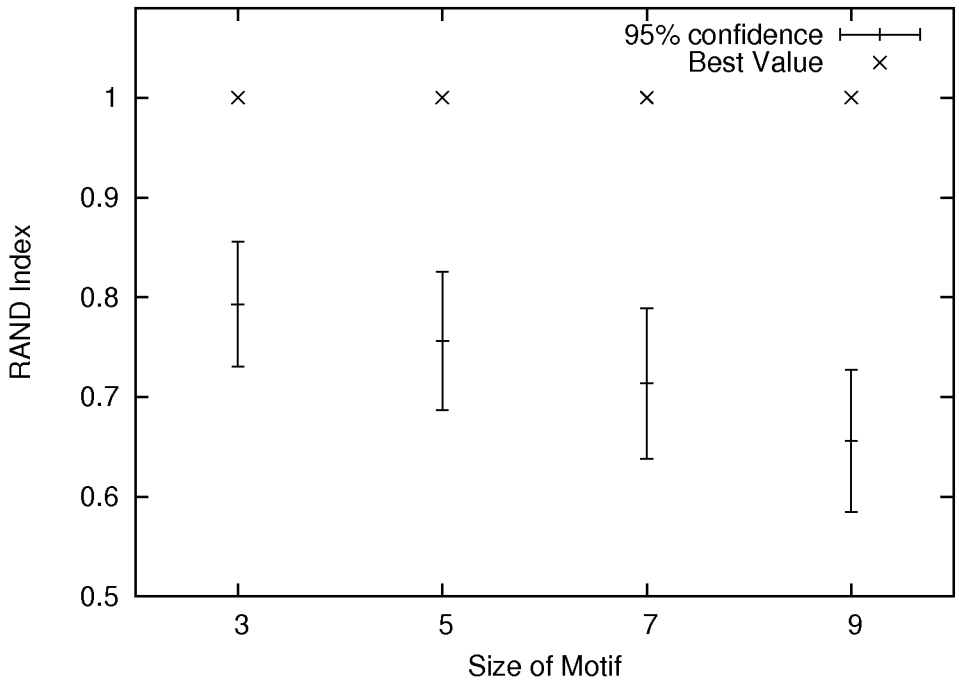


Figure 7.7: 95% Confidence intervals and best value of Rand index for the DS0 v. DS1 data set with 3 populations.

These two data sets show an interesting relation to the length of the motif. In Table 7.9 the level of complete collapse increases monotonically with the number of symbols in the string. This may provide an explanation for the slow decrease in the confidence intervals about the mean as shown in Figure 7.7. However, the number of correct size 2 is maximized for a length 7 motif; this is also the motif length with the most perfect classifications. The found motif sets are quite diverse, showing that for this motif length, there is a number of good motifs which divide the set into a natural partitioning. As these motifs are relatively distant in terms of their 3-mer string kernel, this natural split can be found.

DS25 and DS37

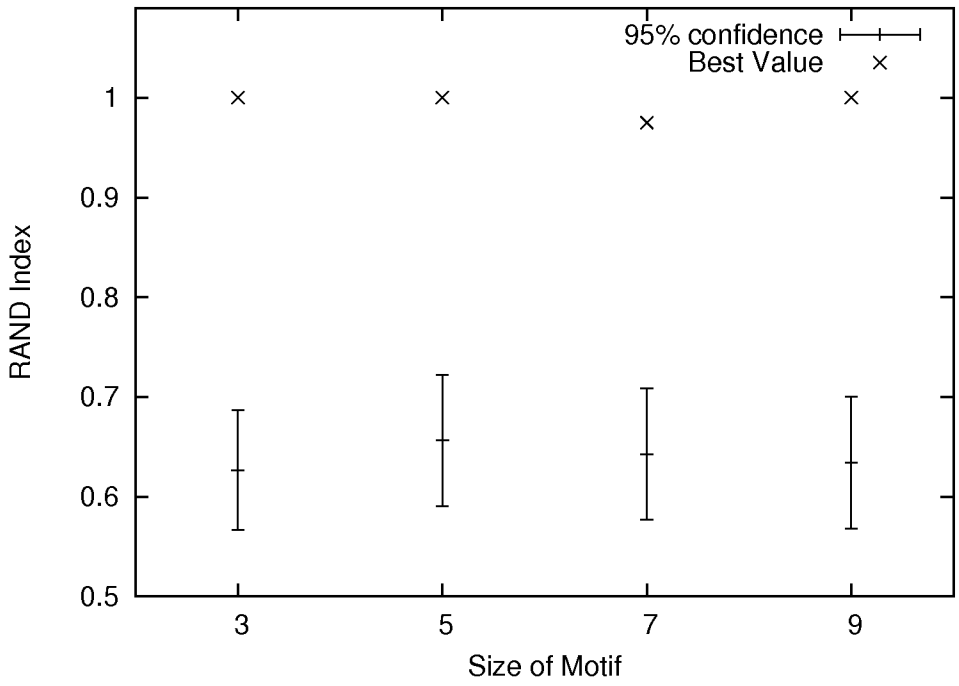


Figure 7.8: 95% Confidence intervals and best value of Rand index for the DS25 v. DS37 data set with 3 populations.

Examining DS25 and DS37 which are distant in terms of their 3-mer spectral kernel distance, this natural split becomes surprisingly harder to find. It is missing a perfect classification at a motif length of 7, though there are 3 classifications which only have a single point misclassified. The mean is robust to changes in motif length, as shown in Figure 7.8.

DS9 and DS25

As DS9 and DS25 are beside each other in terms of 3-mer kernel distance, this should be the hardest example to perfectly classify, especially when a short motif length is chosen. This is confirmed in Table 7.9 as it is not until

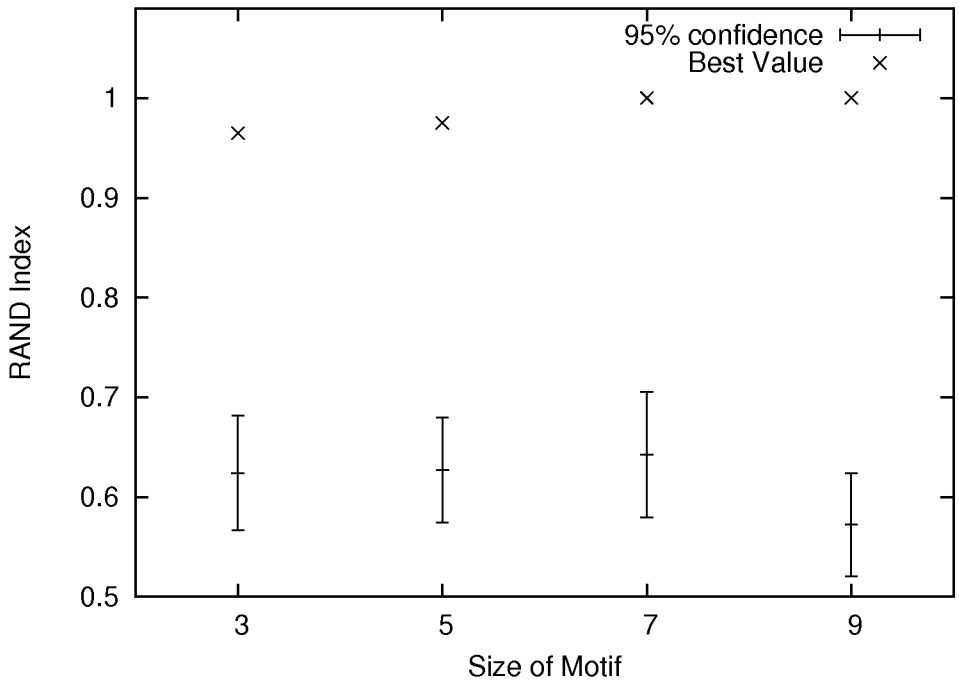


Figure 7.9: 95% Confidence intervals and best value of Rand index for the DS9 v. DS25 data set with 3 populations.

a motif length of 7 that there is perfect classification. However, it should be noted that a classification found by length 3 was able to correctly split the set over 3 classifiers, twice. The length 5 motifs split a correct classification over 3 classifiers once, and one other classifier was only off by 1 misclassified point. As seen in Figure 7.9, the mean value is resilient to changes in the length of the motif, however, there is a slight decrease in length 9, the one with the most fully collapsed classifiers.

7.4.5 Human Leukocyte Antigen

A very interesting function of the Multiple Worlds Model is explored by using more worlds than the number of classes in the data set. This is found to be actually helpful in terms of the evolutionary process. As more populations

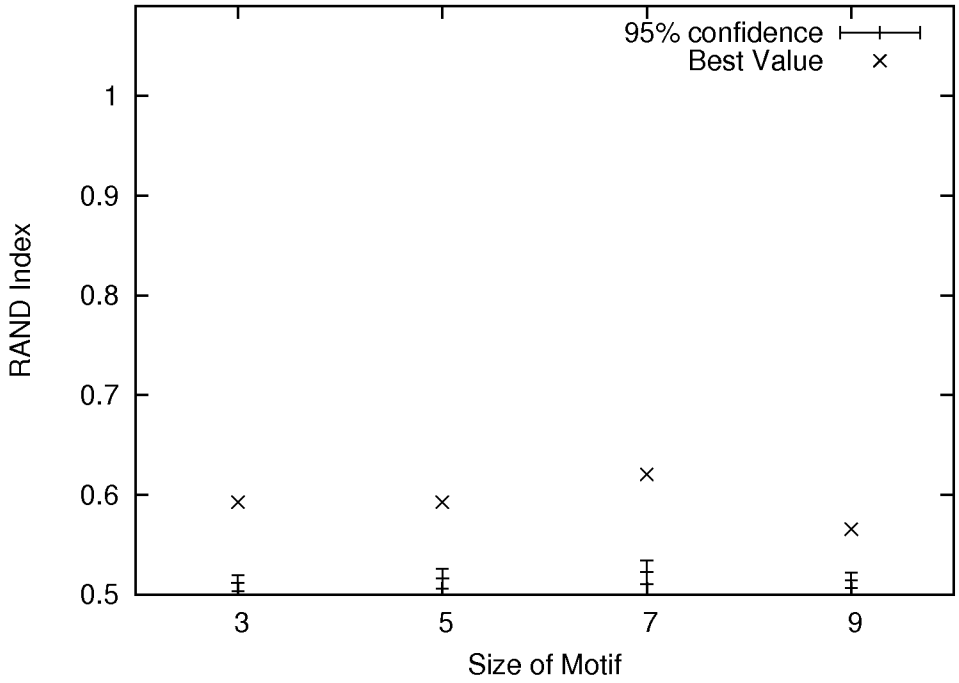


Figure 7.10: 95% Confidence intervals and best value of Rand index for the HLA data set with 2 populations.

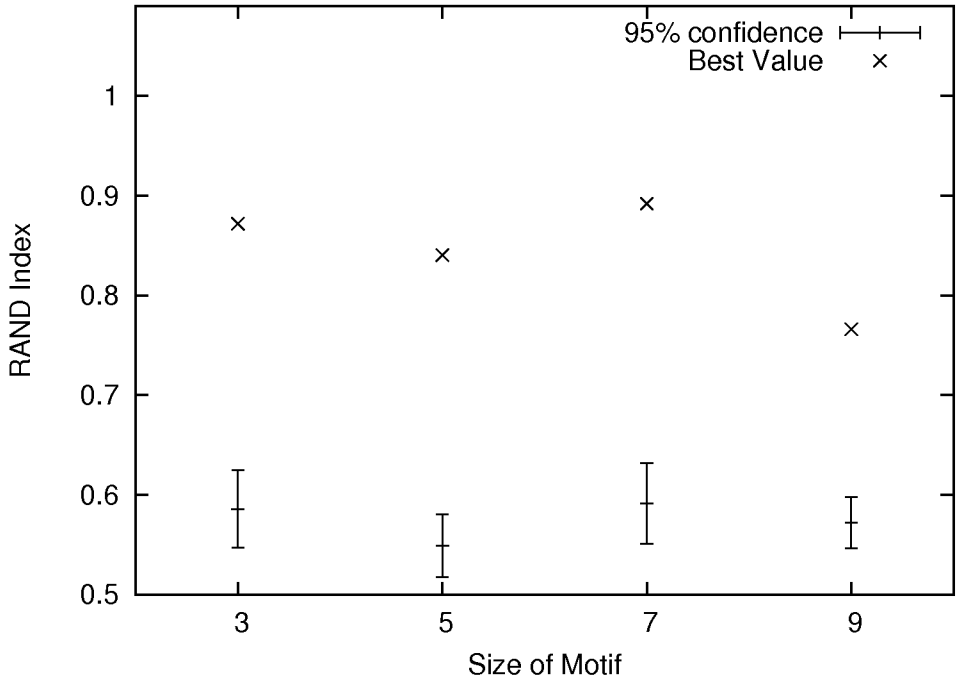


Figure 7.11: 95% Confidence intervals and best value of Rand index for the HLA data set with 3 populations.

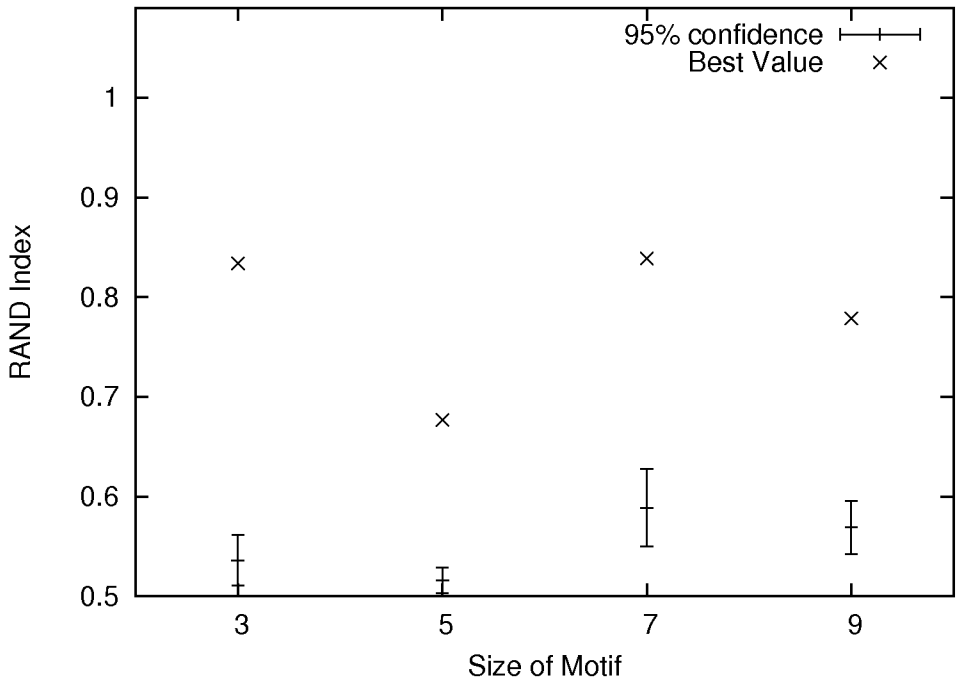


Figure 7.12: 95% Confidence intervals and best value of Rand index for the HLA data set with 3 populations with the fitness function taking the square root of the latitude.

Table 7.10: Best motif set found for motifs of size 3, 5, 7, and 9 with classification scores for the two classes for the HLA data set with 3 populations.

Class	Degenerate Motif and Fitness Score		
Best motif set for length 3			
	[CG]-[CG]-C	[GA]-[GT]-[CGAT]	[CT]-[CAT]-[CGA]
I	500	0	0
II	61	423	16
Fitness	561	423	16
Best motif set for length 5			
	T-[GT]-[AT]-[CGT]-[CGT]	[GT]-[CAT]-[GA]-T-C	A-[CGA]-[CAT]-[CA]-T
I	13	24	463
II	403	66	31
Fitness	416	90	494
Best motif set for length 7			
	[CGAT]-[CGT]-[GAT]-[CAT]-[CG]-[AT]-G	[GAT]-[CT]-[CA]-[GT]-[CG]-G-[CAT]	C-[CG]-[CGT]-[CGT]-[CGAT]-[GAT]-C
I	1	19	480
II	397	101	2
Fitness	398	120	482
Best motif set for length 9			
	[CGT]-[CGA]-[CAT]-[CG]-[CAT]-[GAT]-[GA]-[AT]-[CGAT]	[AT]-[AT]-[CGAT]-[GAT]-T [CGT]-[CT]-[GA]-G	[CGA]-[CA]-[CAT]-[CT]-C-[GT]-[GT]-[CGA]-[AT]
I	467	12	21
II	88	42	370
Fitness	555	54	391

Table 7.11: Level of sub-population collapse in the two classed HLA data set with three populations. The number of populations with greater than 10% of the data set out of 30 replicates.

Motif Length	1	2	3
3	10	17	3
5	13	14	3
7	7	17	6
9	6	15	9

exist, we can see a statistically significant improvement in the mean results for motifs of length 3, 7, and 9 (see Figures 7.10 and 7.11). Having 3 populations increases the variance of the output. There is an improvement in the best classification for all motif sizes as the population increases. A motif size of 3 or 7 gives the best classifiers in both cases. The change to the fitness function, see Figure 7.12, is found to have no positive effect upon the classifiers, and in all cases finds a worst classification, especially for the length 5 motif.

Looking at the sub-population collapse events for the 3 population data, it shows a propensity to favour the correct number of classes. In this situation, two. Note that as the mean takes into account instances where there is a collapse to a single population, i.e. only one motif is dominating the population, leading to no classification, the mean score greatly suffers compared to the best value. Looking at length 5 and 9, we can see that they are more prone to getting the number of classes incorrect. Length 5 especially is prone to a full collapse; 9 is the inverse suggesting there are more classes available. For motif length of 9, this is relatively unsurprising. Using more symbols permits more ways to divide the set of data points, and increases the probability there are more local optima.

A classifier coming to the conclusion that the set has 3 classes is not necessarily incorrect. It could be the case that it is a good classifier. The Rand index will be penalized in such an instance, but not as much as if there was a misclassification which places members of each class in the same

machine generated class. Looking at the best results for the HLA data as shown in Figure 7.10, we can see an instance, for the best classifier of size 7, where a classifier which states there is 3 classes has the best result. This is even more impressive when it is noted this classifier is the best classifier in general for all sizes. The two motifs which split class II are in many aspects similar; the fifth symbols are both [CG], the final symbol is a reciprocal, and the second string for symbols one and two is missing a base in its wildcard. This is a case where instead of a species pushing out another, the other was forced to decrease its latitude in order to be more specialized.

Looking at the best results of motif length 5, we can see another instance where the collapsed population has specialized in order to gain a small foothold. We could remove this collapsed population, share its points between the two other classifiers and perhaps come to a better result. The results for length 3 show a very interesting collapsed population, which looks for a high content of Cs in the II data. Notice how the first classifier, which scores all of I, also looks for a high C content, whereas the motif which classifies a majority of II has almost no Cs in the wildcards. It would be a safe assumption that without this third classifier, these points would be obtained by the classifier for I and not II. Hence, even though this population is collapsed, the resources it consumes are of value.

Chapter 8

Radio Demographics

This chapter examines a pedagogical example of MWM for its application to a simple model, suitable for an undergraduate evolutionary algorithms class. The model chosen is a set of radio stations attempting to provide a playset of songs to a group of listener profiles.

8.1 Demographic Modeling

The use of demographic models for broadcasts has been examined by their industry organizations, most prominently the Nielson ratings in the United States. These ratings are provided by the viewers, who are issued diaries to list the shows they have seen during a monitoring period. The diaries are issued based on demographic sections of the population. Such modeling has a number of flaws (see [63, 16, 91]); there is an issue with the choice of different demographic groups to how to classify an individual. There is also a response bias (the listeners may not report shows in diaries, or forget to fill one out, or their responses may vary based on memory), and new methods of transmission (such as recording of live TV for later viewing, downloading shows, cell phone viewings, and tablet devices) are not accounted for in these statistics, even if a broadcast is made live.

Question 1: What are your feelings on Country Music?

-3	-2	-1	0	1	2	3
strong dislike	dislike	weak dislike	neutral	weak like	like	strong like

Table 8.1: Example survey question with response mapped on a Likert Scale

Listener Type	Advert	Top40s	Country	Rock	Talk
Rocker	-1	-3	-2	3	2
Pop-ularist	-1	3	-1	1	0
Country	-1	0	3	1	-2
Talk Caller	-1	-1	-1	-1	3

Table 8.2: Listener profiles of a Rocker, Pop-ularist, Country, and Talk Caller

The model used in this approach is based off a profile of likes and dislikes for various content types. Advertisements are given a strict dislike value which is the same against all profiles. These profiles could easily be drawn from a seven point Likert scale [96], mapping each with a score in the range $[-3, 3]$ (e.g. Table 8.1 shows an example question for country music).

8.1.1 Representation

There are two groups which will be modeled by this study: radio listeners and radio stations. The listeners are represented as a specification of preferences in the form of numerical value of the enjoyment/dislike of each particular type of music, and a value for the dislike of advertisements. We assume that all listeners start as indifferent to the various radio stations. The radio stations are represented as a list of types of music they will play for a preset interval. An interval in this case could be a programming block, a time period, or the length of a song. Each programming block is deemed to be of a standardized length for the sake of simplicity.

8.1.2 Fitness Evaluation

The fitness of a station is defined as the number of advertisements listened to by a data set of listeners. The content provided to a listener must be pleasing based upon their unique listener profile. The listeners to a radio station will refuse to listen to a station which only broadcasts advertisements. If they do not enjoy the content, then they are prone to change stations; the change will cost the station advertising revenue, and thus evolutionary fitness. The station must strike a balance between content and advertisement to be most fit.

The listener is defined by a happiness level and a listener profile as shown in Table 8.2. The happiness level is how pleased the listener currently is with the broadcast. The profile is a set of likes and dislikes of content types. A classic rocker, for example, might express a large benefit, from listening to classic rock, a small gain from talk (i.e. Shock Jocks), and a sharp decline from Top 40s music. All profiles dislike advertisements; listeners would prefer content. This happiness level is used to determine if a listener will change to another radio station subject to the distribution $C(x) = 1 - \frac{1}{1+e^{-x}}$, as shown in Fig. 8.1. When happiness is at a value of zero the listener is indifferent with half a chance to change stations. As the happiness increases from negative to positive six, the listener will saturate in terms of like or dislike, and will be certain to stay or change the channel. Because the listeners do not necessarily have the radio on at any given interval, there is always a null station choice with a constant happiness value of zero. This null station represents the choice of leaving the radio off.

In fitness evaluation, each listener starts on a random part of the dial. There is a number of time steps equal to the programming period of the stations. At the beginning of each time step, each listener generates a random number uniformly distributed in the range $0 \leq x \leq 1$. If the number is less than the value of their current happiness, based on the station they have tuned into, they stay on that station. Otherwise, they change which station

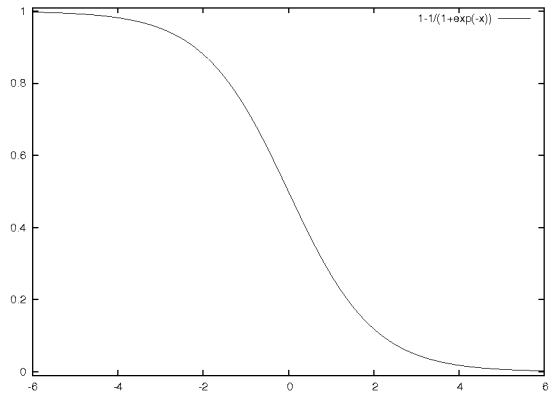


Figure 8.1: Graph of the happiness function — $C(x) = 1 - \frac{1}{1+e^{-x}}$ — probability for changing stations lowers as the 'like' a listener has for a station increases.

they are on and continue the process of checking their happiness on a station until they stop on a station. Once they have decided which station they are going to listen to for this programming period, their like for the station is adjusted based on the programming choice of the station for the current time step. If it is an advertisement, then the station receives one point of advertising revenue (fitness).

The fitness of a radio station is the number of ad-revenue scores multiplied by the fraction of advertisements in their total programming block. This model feature represents the tendency of agents to simply not listen to the radio, or perhaps switch to an MP3 device, if they are offended by too many advertisements. This normalization of fitness removes the implausible Nash equilibrium in which all stations go to a constant advertisement format.

8.2 Experimental Settings

Each population used contains 100 members. These members consist of a stations playlist of size 12, selected as it is the number of five minute intervals in an hour. Five minutes is long enough to introduce and play song of average

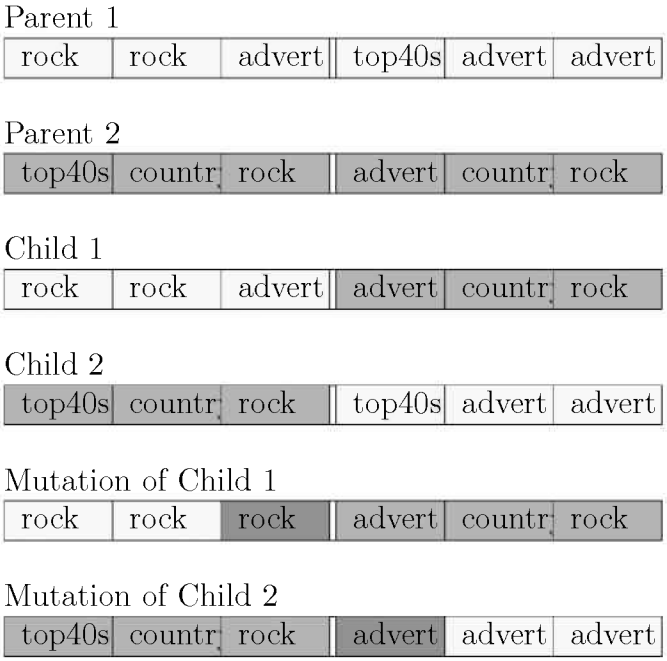


Figure 8.2: Example of breeding (crossover and mutation) between the representations of playlists. Parent one is the first radio station playlist of size six in light blue. Parent two is the second radio station playlist of size six in the darker red. A one-point crossover then occurs between the two parents at the third loci creating child one and two. The mutations of the children then happen in child one at the third position and the second child at position four, labelled in dark gray.

length [1]. Each slot in a playlists is initialized to one of the music types or advertisement with equal likelihood. The selection operation for breeding is a tournament which takes four members of the population, orders them based on their fitness, and replaces the bottom two with replicates of the top two. The copies then undergo mutation and crossover, e.g. Fig. 8.2. The crossover operator is a two point crossover; it probabilistically selects two points in the playlist and swaps the broadcast types between the selected positions. The mutation operator randomly assigns a new broadcast type to each time step in the playlist with 10% probability. This occurs for 5000 generations.

8.3 Results

In order to allow for a comparison between outputs with multiple stations, there needs to be a way to cluster similar stations together. As we know that certain choices would be more likely — such as talk shows for a population with talk callers — the station with higher frequency of the first type of show is used as a standard of placing the stations into classes. Number of adverts in this case is not a good method of classification as the model will often produce stations with the same number of advertisements, and we are interested in seeing if there is behavioural differences.

8.3.1 Simple Test — αs and βs

In order to show the partitioning power of multiple worlds model a simple example was constructed. In this case we limit the content types to three: Advert, Song A, and Song B. Two profiles were created which are the dual in terms of their enjoyment of the stations, call them listener α and listener

[1] A distribution of song lengths created from over 70,000 American songs found a relatively symmetric distribution in lengths with a mean of 242 seconds, i.e. about 4 minutes [107].

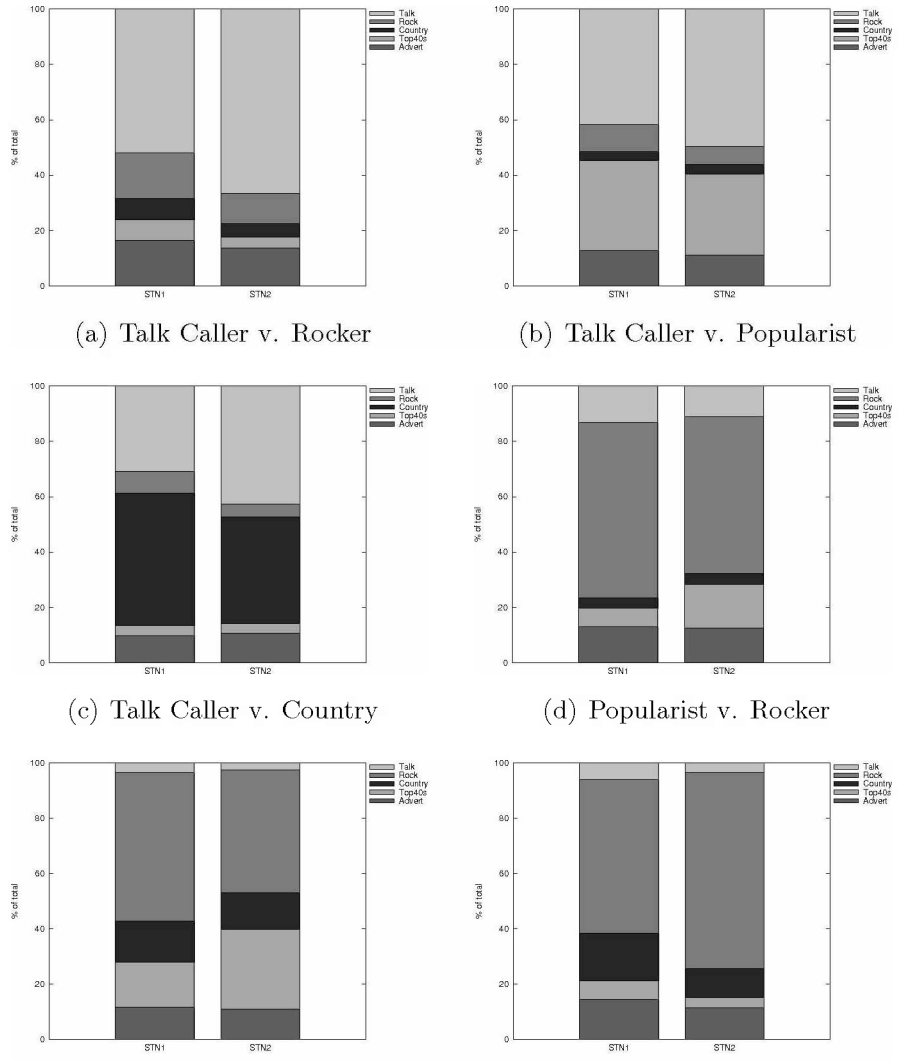
β. The mean frequencies of each type are compared:

	Advert	Song A	Song B
STATION α	1.26667	10.3333	0.4
STATION β	1.2	0.366667	10.4333

It is plainly visible from the means, the MWM creates two radio stations. The first appealing to the αs by playing Song A exclusively, and the second appealing to the βs by playing Song B. This simple experiment serves as a certificate that the system is functioning nominally.

8.3.2 Two Stations — Even Populations

In all cases the station profiles created show some similar trends. First, there is a length of content which is associated with positive feedback to one of the demographic groups which lasts for at least a quarter of the time. Often, there is then a quick switch into a content type of the other profile right before the appearance of an advertisement; the stations are attempting to get as many listeners as possible before the payoff. After all advertisements of the string have appeared, the string reverts to using any of the possiable values. Fitness has already been made or lost at this point and all selections for these locations will produce a string of equal fitness. The string is therefore epistatic; earlier changes are worth more than later changes. Further, there is an issue in the model that a listener can saturate their happiness by hearing a number of good songs in a row, they "stay on the dial" even for a set of bad content. The mean percentage of each of the play types is presented and commented upon. However, the strings of the playlists are far more informative. Figure 8.3 provides a graphic visualization of the play profiles.



(a) Talk Caller v. Rocker

(b) Talk Caller v. Popularist

(c) Talk Caller v. Country

(d) Popularist v. Rocker

(e) Popularist v. Country

(f) Rocker v. Country

Figure 8.3: Radio Station Time Allocations — Two Stations

Talk Caller v. Rocker

Shock jocks take over when rockers and talk callers are the population of listeners. The power of talk radio swiftly gains an advantage early on in the strings, allowing for ads to be played. After advertisements have been played, the epistatic nature of the string and the saturation of happiness makes for stations which have a number of unexpected plays of country and top 40s, however in the station with most talk these are reduced. The stations in this model converge in their playlists, as both talkers and rockers both like talk radio. There is no instances of a sub-population collapse, and the stations are able to coexist.

Talk Caller v. Popularist

Popularists don't mind talk shows, whereas talk callers are offended by anything. Again we observe a large movement towards talk. The popularists bounce over the dial as the stations fight for dominance in the talk market — leading to strings of talk right at the start followed by a string of talk and top 40s. The top 40s are played in the time step right before an advertisement in order to keep the audience tuned in. There are no instances of a sub-population collapse; two stations are perfectly happy to share the air.

Talk Caller v. Country

Country lovers have a strong dislike for talk, and talk callers dislike everything else. The advertisement levels between the two diverse groups is reduced compared to the talker v. popularist, and even more than the talker v. rocker. Listeners are moving to a single station and holding position, making it far more profitable to be in a specialized market. There is a single collapse event in the runs, producing a final station with no advertisement revenue. The single focus of the station holds a listener to a station for advertisements without offending.

Popularist v. Rocker

The stations above once again exibit a strong like for the same type of music in Rock. Hence, the like for rock presses out the top 40s music which would offend the audience. The station, unable to play more rock than the other resorts to differentiating itself by the choice of top 40s music. A divergence in the station profiles as the first station aims for more diversity. Station two becomes a rock station, playing only one third of the pop songs, with more allocations of shock jock talkers, who don't offend the popularists. Both profiles dislike of Country music has suppressed this type of content to the point of nonexistence in the final population. Country only appears in the strings due to genetic drift in the population and due to the epistatic nature of the play string. The final populations show no evidence of collapse, and both stations are relatively profitable.

Popularist v. Country

An interesting result happens with the popularist and Country in that neither really gets their preferred music. A popularist dislikes country, a Country is not offended by top 40s but doesn't enjoy it. This explains the higher appearance of Top 40s music in the first station. Both Country and popularists enjoyment of rock leads to a creation of rock stations. The final populations show two stations which have collapsed, as the rock profile is able to dominate the space.

Rocker v. Country

Station one targets rockers and plays half the number of top 40s selections and talk. Station two is able to play more ads by taking more country selections to make up for the offending shock jocks. The shock jocks allow for less rock to be played. The stations do not collapse and both are profitable.

8.3.3 Three Stations — Even Populations

Increasing the number of stations in many cases has a settling effect on the strings into a spectrum between the two demographic groups. Instead of harsh divisions, the middle station is prone to trying to split the difference between the two extreme ends. In cases where the demographics have dislike of the others likes, the selections and breaking into different playlists becomes more pronounced than the two station examples. Figure 8.4 provides a graphic visualization of the play profiles.

Talk Caller v. Rocker

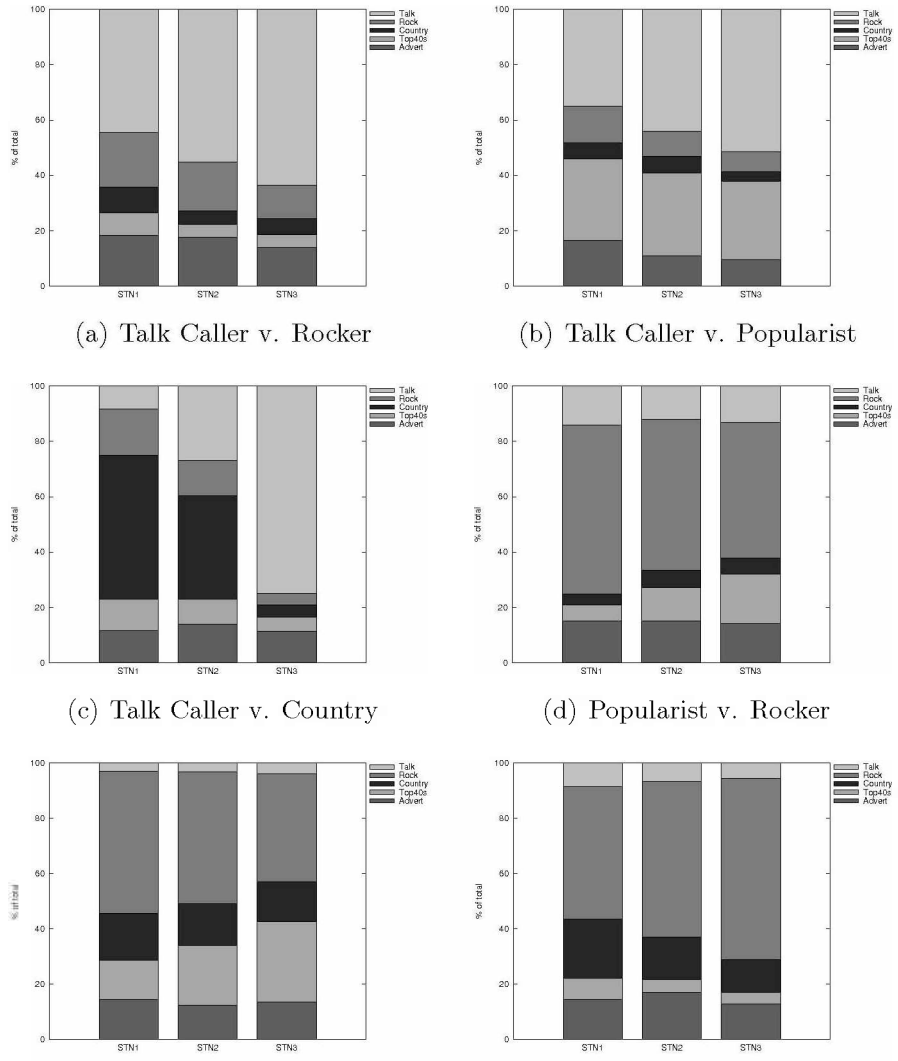
For these stations shock jocks again rule the airwaves. The three stations fighting for the profitable talk and rock markets. Playing rock music is inversely correlated with talk shows, an attempt is being made to specialize for the rockers, to pull them out of the talk only stations. The percentages for stations one and two for Rock and Talk are close to the two station model, Station three using a lower level of talk. There are no instances of collapse similar to the two station model; the demographic is able to support the three stations.

Talk Caller v. Popularist

As talk decreases, the stations move deeper into country and rock. Station one focuses on a strong mix of rock and talk, moving beyond the bounds set by the two-station model. The three-station model further diverges from the two-station model, as the number of collapses goes up to one. The support for more stations is weakening.

Talk Caller v. Country

The talk caller and Country listener are most dissimilar in terms of their likes, and this is evident in the modeling. Station one has progressed to an



(a) Talk Caller v. Rocker

(b) Talk Caller v. Popularist

(c) Talk Caller v. Country

(d) Popularist v. Rocker

(e) Popularist v. Country

(f) Rocker v. Country

Figure 8.4: Radio Station Time Allocations — Three Stations

"all talk — all the time" format. Station three is now a country/rock station. Station two attempts to take the middle ground. This model has no collapse events by pressing to the extremes to capture the listeners.

Popularist v. Rocker

The popularists and rockers both end up creating a talk station with sight differences in the levels of rock and top 40s. Rock dominates the playlist, as both popularists and rockers gain enjoyment — talk is not seen to the same extent as in all cases if a talk choice was made, it would have been better to air a rock song. The amount of top 40s is what primarily gives a differentiation from between the stations. No stations are closed due to this listener demographic.

Popularist v. Country

The three radio station model also presses into becoming a rock station as both the popularist and Countryist enjoy rock. The stations press apart in terms of their playing of country and top 40s. Talk shows are pressed off the dial. No sub-population collapse events occur demonstrating that the demographic is able to support the number of stations.

Rocker v. Country

The Country station continues to play talk radio as an alternative to rock selections, with country maintaining a low presence. The number of advertisements increases in the middling station to take on more revenue. The targeted rock station is able to play a smaller number of ads, showing the power which can be had from a single demographic. The three stations have a single sub-population collapse event showing a convergence in the profiles.

8.4 Discussion

MWM can model just as easily other broadcast media, such as television and streaming internet programming. Another application can be seen in products where a large number of alternatives or imperfect substitute goods exist; these can be modeled in much the same fashion, with minor changes based on the application desired. For example, a set of restaurants could make investments such as location, price, food style, and atmosphere. A set of consumers would be positioned based on their preferences.

This study shows a very simplistic model of the radio stations and listeners, and a number of changes can be made in order to better model the profiles. First, listener enjoyment is based on their entire history with the stations. It would be more realistic to have a memory window. That is a listener will remember perhaps only the last 3 songs from each station. This would prevent the saturation of listener to either the positive or negative side. Secondly, there is currently no decay in like or dislike over time. The model would be best served to have a decay in the values. Finally, the programming is made in fixed length programming blocks — this should be allowed to change in a more dynamic method, quite often flipping through channels. This would permit an additional level of strategy based on where content block beginnings and endings are placed. The current string approach, however, allows for a simple examination of the results produced in order to show the method to be valid.

The introduction of additions to the fitness function allows for a multitude of different studies to be preformed. Payola/Plugola, the illegal inducements provided by record companies to stations for playing specific song titles [29], by companies for products to be 'plugged' outside of an advertisment block, and political opinions being espoused [85], can easily be introduced. The model could provide a multi-objective fitness to the station for payment from normal ads as well as side payments. In the case of political opinion, another parameter would be assigned to the listener for political affiliation.

Countering this would be a probabilistic penalty; fines are received if the Payola/Plugola is discovered by the regulator.

Further, other restrictions, such as the Canadian content regulations [38], can be modeled through the addition of changes to the fitness model and the available contents. The final models would have to contain a set amount of content or a fitness penalty — a fine, would be applied. The selected content would provide less of an increase in the like of a listener as it appears more often.

The MWM has shown another application of the novel fitness determination taking into account the evaluation of fitness between populations where there is not an exchange in order to partition a space. It is interesting to note that these radio stations are not subject to the levels of sub-population collapse seen in other studies. This study is the first to use a fitness function which is not winner-take-all in terms of a point. In this case, a listener may provide fitness to both stations. Additionally, as the initial starting location on the dial is random, there is a propensity for the stations to be able to keep the listeners that they start with. In previous winner-take-all fitness functions a larger number of collapses are seen. These previous works were also determinisitic models. This implies that the model perhaps is too forgiving to bad content, or that the number of listeners was large enough to support a number of radio stations. In order to apply this in the field to real radio broadcasts, case histories and human testing would be required in order to refine the parameters.

As a cartoon, this model system of a group of radio stations is suited for presentation to a classroom. The class themselves could be asked to come up with other listener types, could be polled to create a population, or could extend the model to include other factors as an assignment or final term project.

Chapter 9

Multi Agent Genetic Network (MAGnet)

9.1 Algorithm Definition

Multiple Agent Genetic Networks (MAGnet) is a spatially structured evolutionary algorithm that sorts a collection of related problem instances into subsets through the use of evolving agents capable of moving problem instances from one node of a network to another, see Figure 9.1. The spatial structure is given in the form of a network. Each node of the network contains a collection of problem instances.

The way that agents move about the network depends on their ability to solve the problem instances present at each node. Each agent in turn finds its fitness on the problem instance currently present on the node where it is situated. The agent then has a chance to move to a location where it would have a higher fitness score based on a *movement rate*.

Each agent also has a *problem briefcase* which allows for the movement of the problem instances. It may, depending on the *briefcase rate*, pick up a problem instance on its current node and move that problem with it to the agent's new location. The agent chooses the problem instance on which it

obtains the best score. The transport of problem instances using briefcases creates a sorting depending on the agent's score. This sorting is a form of unsupervised learning of problem classes.

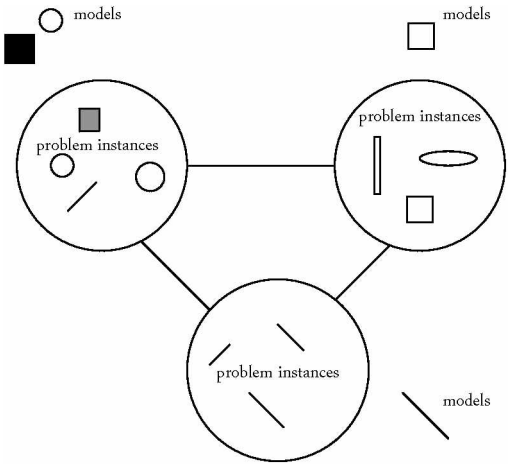
The fitness of an agent on a node, with no problems upon it, is defined to be the worst possible score. This is done to allow nodes to lose all their problem instances if the natural number of categories of problem instance is smaller than the number of nodes. The name *subpopulation collapse* in this model is used to describe the emptying of the collection of problem instances on a node.

After an agent moves, it will breed with the other agents on its new node. The agents on the node are re-evaluated, the incoming agent could have brought a new problem instance which changes the fitness, and a partner is selected. This breeding is done using normal genetic operators of crossover and mutation.

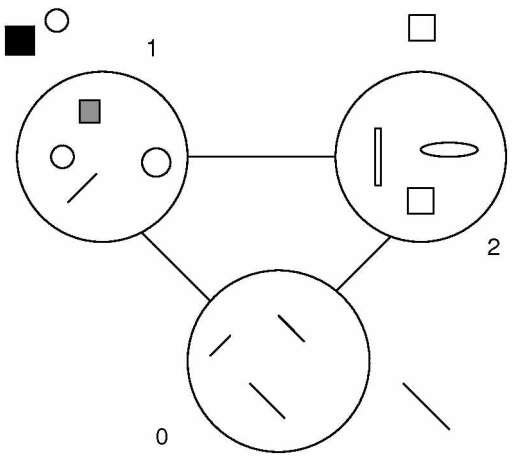
9.2 Example Agent Evaluation - Iterated Prisoner's Dilemma Agents

In this section we look at a sample agent playing Iterated Prisoner's Dilemma, explained subsequently. In this MAGnet there are only two nodes. Problem instances are other Iterated Prisoner's Dilemma strategies. The agents are also Iterated Prisoner's Dilemma playing strategies. As seen in Figure 9.2(a) there is an agent about to undergo an evaluation which is playing a strategy of Always Defect (ALLD). The problem instances on the first node are 2 Tit-For-Tat(TFT) and 1 ALLD. The second has 3 ALLD.

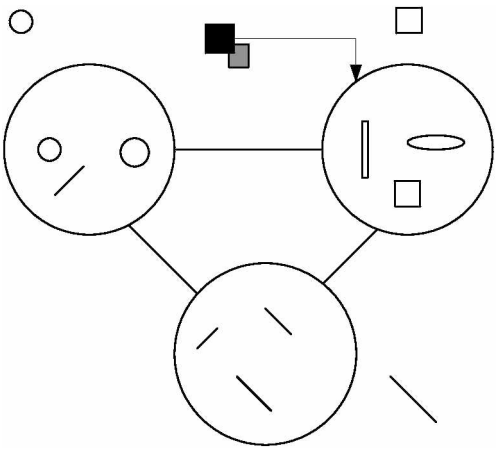
In Figure 9.2(b) we can see the evaluation of fitness on both nodes: where the agent exists and the neighbouring connected node. On its own node the ALLD agent scores 1 point each v. the two TFT agents, and 1 point from the other ALLD. Note that the best scores are known to be 3 v. a TFT and 1 v. an ALLD. Therefore, 7 points of fitness were available, of which



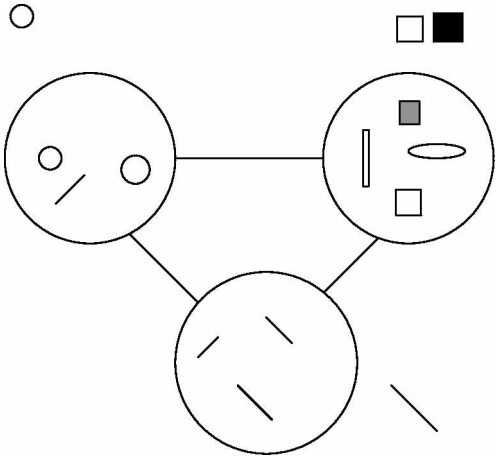
The black square is our selected model for this round



It evaluates its fitness on each of the connected nodes – it finds the problem instance it scores highest on the current node (the gray square)



The black square migrates to the node with highest fitness and brings with it the problem instance it is best, the gray square



The black square looks for other models on the node and on finding the white square undergoes genetic variation operators

```
for some number of generations do
  for all agents do
    find the adjacent node which the agent has the best fitness
    if we probabilistically select to move a problem then
      find the problem on the current node which the agent has its highest
      fitness
    end if
    move the agent and the problem to the node where the agent has the
    highest fitness
    if there are others upon that node then
      select a breeding partner and apply crossover/mutation
    end if
  end for
end for
```

Figure 9.1: Demonstration and Pseudocode of the MAGnet system

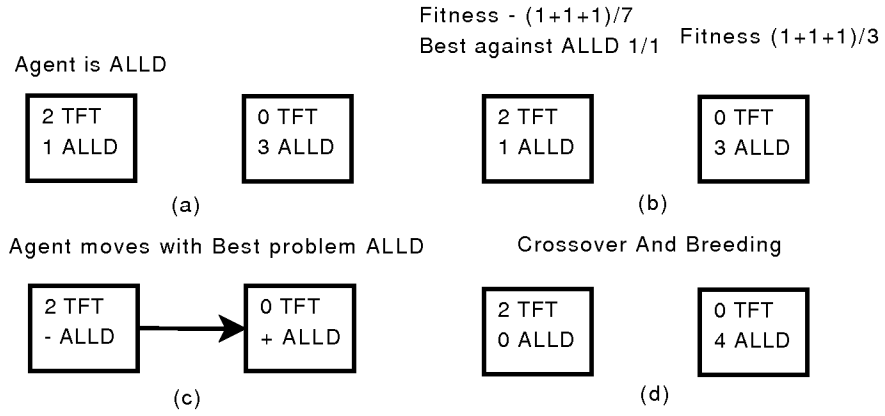


Figure 9.2: Example MAGnet agent move

the agent only got 3. On the other node, it scores maximum fitness when evaluated against an ALLD, it scores 3/3 fitness points. Hence, the new node is better for the agent. Remember, if this neighbouring node had no problem instances, it would be counted as having the lowest possible fitness score by fiat.

This agent has probalistically been selected to move a problem instance, so it finds the problem instance for which it has the highest fitness upon on its current node. That is ALLD.

In Figure 9.2(c) the selected agent now moves, taking the problem instance of ALLD with itself. The number of ALLD instances on the first node is reduced by one on the node it is leaving, and increments on the new node.

Finally, we have finished our agent's move in Figure 9.2(d). The agent is now on the new node and we know its fitness score is high for the problem instances on this node. If there is any other agents on the node it will select one at random, when these other agents came to this node on their turns they had high fitness for this set of problems, and undergo a crossover. If there is no other agents then it will mutate slightly.

9.3 Parallelism

MAGnet in many ways has its roots in the idea of breaking both data and processing over a network. In term of communications, if we assume the number of processing elements equals the number of nodes and the worst case of a fully connected graph of N nodes, we come to the following evaluation of the communications. For each agent there is a broadcast of the chromosome to each of the other nodes requiring $O(N)$ communications, followed by an evaluation of the best node to move onto, requiring $O(N)$ comparisons and $O(N)$ communications. Following this, the best node now knows to keep the chromosome which is has just been sent. A problem instance will then be passed from the originating node to the new node, in $O(1)$ communications. This entire process is on the order of $O(N)$ communications of a data structure no larger than a chromosome.

As for the algorithm progresses there is an issue of data starvation as the nodes remove problem instances, especially in a subpopulation collapse event. In the worst case it will become a sequential process - though this degenerate case implies that there is only a single class for all points in the model, which is a useful result that informs us that a single population method is more suitable. There is a few interesting efficiencies to be gained depending on the topology of the algorithm's network, especially when the number of processing elements is less than the number of nodes.

The first is that nodes of three or more edges away are independent in terms of both data and processing, as nodes only share with neighbours. Two agents with this distance between them could be evaluated in parallel. Thus, a well chosen mapping of the internal algorithmic network topology to that of the arrangements of processing elements could yield large performance gains. Though, how such mapping could be done is beyond the scope of this present work.

9.4 Experimental Overview

MAGnet aims to provide a framework for exploratory discovery of links between problem instances, giving a model which represents the instances on a node. The well-known mathematical game of iterated prisoners dilemma is selected as an application area for the experiments. This selection was made as previous trails using evolutionary algorithms have shown wildly varied results in the levels of cooperation and the agents generated. These differences occur based on the representation, used for an agent, the selected payoffs, which variation operations where used, how the players were selected to be played with each other in the population. This wild variance is not see in analytical approaches to the game, such as evolutionary stable strategies, which demonstrate that playing the move your opponent made last round, tit-for-tat, is how a rational agent will play. This is based on the assumptions of an infinite number of rounds in a game between players, all players face every other player, that players can only change a deterministic strategy with a single mutation to currently existing and infinitely growing population. No practical evolutionary algorithm can meet with these requirements, hence the apparent disconnect of a multitude of agent types flourishing in the evolved populations.

No longer is tit-for-tat the best strategy for all situations. The best play is dependent on who you are playing against. This matches with a common sense understanding that a player who always uses the same sequence of moves, will play poor, relative to a player who exploits weaknesses in opponents plays.

The problem MAGnet attempts to solve is how the opponents are related in terms of their behaviors and what style of play scores well against these opponent groups. Fingerprinting is another technique which allows for measuring behaviors of an agent. However, there are instances where a fingerprint is difficult or impossible to compute, and it provides no specification of a good play method which will defeat the fingerprinted player(s).

In order to demonstrate the utility of MAGnet on this problem two series of experiments were conducted. First there was a demonstration of the system on a small fully connected graph. This was to demonstrate that MAGnet will give a similar placement of agents in the fingerprint space. A set of commonly used agents are first examined and a new agent is discovered. This new agent is then examined for other properties, such as how it too can be exploited. A larger set of agents was then examined to demonstrate the behaviour based links.

A second series of experiments on choice of the graph is then examined. Differing the graph implies a change of what behaviours are seen in the final agents, and what conclusions can be made about behavioural links in the problem instances. MAGnet, by declaring a node without problem instances to be no longer part of the graph can remove pathways between nodes, making for disconnected regions if the initial graph is sparse. Further, it has implications for distribution of a MAGnet over an actual computer network, where each node is a different processing element.

Chapter 10

Iterated Prisoner's Dilemma

The results from this chapter were first presented in *Multiple Agent Genetic Networks for Iterated Prisoner's Dilemma* [21], and *Examination of Graphs in Multiple Agent Genetic Networks for Iterated Prisoner's Dilemma* [20].

10.1 The Prisoner's Dilemma

The prisoner's dilemma is a classic two player simultaneous game. It was developed by Merrill Flood and Melvin Dresher in the 1950s working at the RAND cooperation. The game has been studied extensively for its diverse uses in modeling problems in economics [53], biology [102], psychology [98], and political science [14][90].

A motivating story is a police investigation of two suspects in a robbery case. The police have enough evidence to convict both of the suspects on a lesser charge of breaking and entering, and require at least one of the suspects to become a witness in order to convict the other in the robbery charge. Both are placed in separate interrogation rooms and are given the choice to either squeal on the other, which is a defection, or to keep silent, which is a cooperation. Both of the suspects are given this choice simultaneously and the jail time given to one will not cause an equal reduction in the jail time

		Player B	
		Cooperate	Defect
Player A	Cooperate	C, C	S, T
	Defect	T, S	D, D

where $S < D < C < T$ and $T + S < 2C$.

Figure 10.1: Prisoner's Dilemma

of the other, i.e. the game is non-zero sum. There are four outcomes: both squeal, the first suspect squeals on the second, the second squeals on the first, and neither squeal. These are associated with jail terms, or payoffs. This payoff matrix is found in Figure 10.1. The temptation payoff (T) is received for defecting when there is a sucker (S). A cooperation payoff (C) is received when both cooperate, and a defection payout (D) when both defect. Being a sucker is the worst outcome and suckering someone is the best result. For the iterated version, the iterated prisoner's dilemma (IPD), it is further required that alternatively being the sucker and suckering someone provides a payout of less than cooperating twice. A commonly used set of numerical utility scores, when maximizing, obeying these properties are $T = 5, C = 3, D = 1$, and $S = 0$.

Assuming two rational players the Nash equilibrium is that they both squeal to the police and receive the second to worst payoff. The dilemma comes from the obvious Pareto optimal solution of both staying quiet. The iterated game provides interesting behaviours, as agents are able to form cooperative networks. There becomes an incentive to work with another player so long as the game length is unknown[1].

Axelrod[14] claims that strategies for IPD that work well against different agents have four properties:

[1]In the case that the game length is known, it can be found that both will defect for the entire game. The reasoning is that in the last move of the game, there is the Nash equilibrium to defect. Knowing that the last move will be defect from both players means that the game can be seen as having one less round. This line of reasoning then propagates back to the first round. Both players defect for the entire game.

1. Don't be envious — trying to score higher than another player means that you are aiming to destroy them and players will respond in kind.

2. Don't be the first to defect — also known as being *nice*. If both players refuse to defect first then they will both have the highest mutual score.

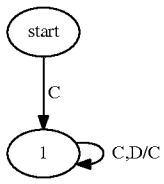
3. Reciprocate both cooperation and defection — if there is a defection then it should be responded to quickly in order to show that advantage cannot be taken, however one should then be quick to forgive a defection when receiving cooperation in order to prevent a long string of mutual defection.

4. Don't be too clever — the moves should be signals to the player as to how to act. Complicated systems lead to confusion.

10.1.1 Agents

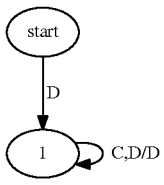
The agents used in this study are shown below as FSM:

Always Cooperate (ALLC):



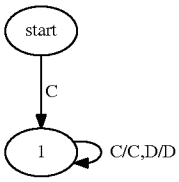
Initial:	C	
State	If C	If D
1	C→ 1	C→ 1

Always Defect (ALLD):



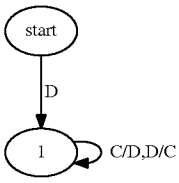
Initial:	D	
State	If C	If D
1	D→ 1	D→ 1

Tit-for-Tat (TFT):



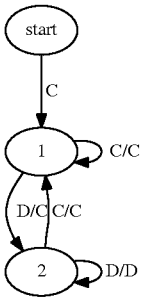
Initial:	C	
State	If C	If D
1	C→ 1	D→ 1

Psycho (*PSYCHO*):



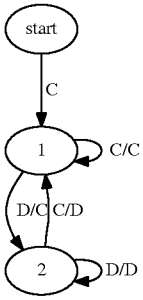
Initial:	D	
State	If C	If D
1	D→ 1	C→ 1

Tit-for-Two-Tats (*TF2T*):



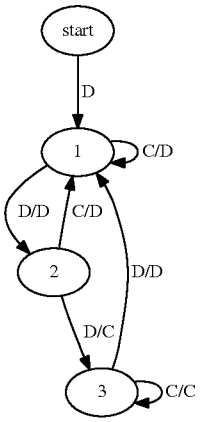
Initial:	C	
State	If C	If D
1	C→ 1	C→ 2
2	C→ 1	D→ 2

Two-Tits-for-Tat (*2TFT*):



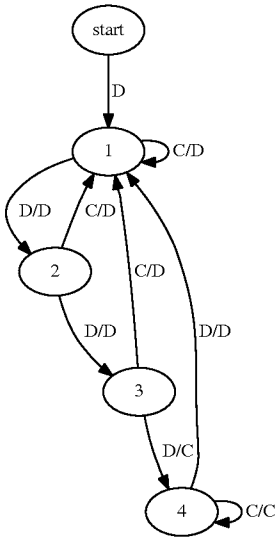
Initial:	C	
State	If C	If D
1	C→ 1	D→ 2
2	D→ 1	D→ 2

Fortress-3 (*FORT3*):



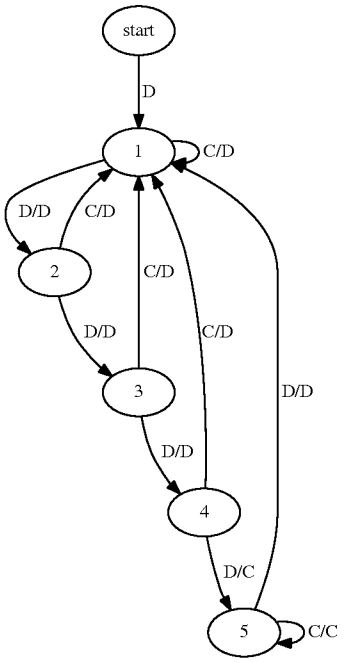
Initial:	D	
State	If C	If D
1	D→ 1	D→ 2
2	D→ 1	C→ 3
3	C→ 3	D→ 1

Fortress-4 (*FORT4*):



Initial:	D	
State	If C	If D
1	D→ 1	D→ 2
2	D→ 1	D→ 3
3	D→ 1	C→ 4
4	C→ 4	D→ 1

Fortress-5 (*FORT5*):



Initial:	D	
State	If C	If D
1	D→ 1	D→ 2
2	D→ 1	D→ 3
3	D→ 1	D→ 4
4	D→ 1	C→ 5
5	C→ 4	D→ 1

It has been recently brought to the author's attention that the naming convention used for the Fortress types differs from that of the originator [12]. The fortress number is based on the number of defections before the hand-shake, rather than the other convention of the number of states required to implement the minimal representation of the FSM. The conversion between the two conventions is simply to add one to the number of defections to get the number of states required for a minimal representation; one state to remember each of the defections and a cooperation state.

10.2 Evolution of IPD Agents

10.2.1 Representation

Representation of the agents which play iterated prisoner's dilemmas in an evolutionary algorithm has been shown to also affect the types of agents which present themselves [6][11][58]. Finite state machines (FSM) have been

found to be more cooperative than many other representations. Further, a number of previous studies, e.g. [40][41][12][106], have used evolution of FSM as a basis for the creation of agent types.

The representation used for the IPD players in this study is a Mealy FSM [79]. Mealy FSMs are transducers that operate over a finite alphabet of input symbols and responds from a finite alphabet of output symbols, as it traverses a finite number of states. It is defined as a five-tuple, $\langle Q, I, Z, \delta, \omega \rangle$, where Q is the set of states, I is the set of inputs, Z is the set of output symbols, δ is the state transition function, such that $\delta : I \times Q \rightarrow Q$, and ω is the output values, such that $\omega : I \times Q \rightarrow Z$. As the sizes of agents are fixed, the data structure used to encode this five-tuple is a state transition table. This table stores the action and transition for C and D inputs for each state, and an inital action. The machine begins in state one after the inital action is presented.

10.2.2 Evolutionary Operators

As the method with which we are constructing the agents is evolutionary in nature, the operators must be defined on the population. The evolution of FSM is controlled by a crossover operation and a set of mutation operators. First is a crossover operation, which is two-point. The crossover exchanges entire states with their transitions and outputs intact between the two machines. There are three defined mutation operations: change a initial state (10%), change a transition (40%), and change an action (50%). The first experiment will only look at the transition exchanges, which means that state actions were held constant and could only be shifted by a crossover action between different agents. The second type of mutations allows for a better search of the available space and is more in line with the Evolutionary Programming method of manipulation of FMS [44].

Table 10.1: Table of the best and worst possible scores for a machine playing a 100 move IPD.

Type	Best	Worst
ALLC	500	300
ALLD	100	0
TFT	300	104
2TFT	300	50
TF2T	400	108
PSYCHO	500	0
FORT3	392	0
FORT4	390	0
FORT5	388	0

10.2.3 Fitness Evaluation

Using the raw fitness score is misleading in this application. For example, the best score when facing an ALLD is less than the worst score when facing an ALLC. The raw numbers would bias the sub-populations to want to contain ALLC, against which any agent can score well. Axelrod has a similar problem with using the 'raw' score as a fitness measure, as he states that a "better standard of comparison is how well you are doing relative to how well someone else could be doing in your shoes ... This is the proper test of successful performance" [14]. In order to remove this bias, we normalize the score on each of the types of machines.

Equation (10.1) provides the normalization function for a single type. It will produce a value in $[0, 1]$ where 0 is the worst score and 1 is the best.

$$fitness_{type} = \frac{score_{type} - worst_{type}}{best_{type} - worst_{type}} \tag{10.1}$$

Table 10.1 shows the best and worst possible scores over 100 runs of each of the testing agents which is used to normalize the agents. These scores do not assume that the agents playing have knowledge of the length of the game. If they did they might defect in the last round. The agents

in this study do not posses enough states to count to 100, the number of rounds of play used in fitness evaluation. The normalization values for this study were created by exact methods (i.e. it is known that ALLD scores optimally on ALLC, etc.) on the well known-agent types as described in Section 10.1.1. The problem of finding these values for an arbitrary problem instance might impede generalization of the results. The required numbers can be estimated in cases where an analytical method is not feasible by using a heuristic approach, e.g. by an evolutionary algorithm. The *total fitness* of an agent is the average of the fitness it obtains against all the agents on a node. This will produce a score from $[0, 1]$.

10.3 Agents Tested

Two sets of IPD agents form the basis for the studies. The first set of agents is: ALLC, ALLD, TFT. These are the most commonly used in studies of IPD and will be referred to as the *canonical agents*. The second experimental set, the *expanded agent set*, includes: ALLC, ALLD, TFT, 2TFT, TF2T, PSYCHO, FORT3, FORT4, FORT5. This is a more comprehensive test of the MAGnet's ability to sort agent types. FORT 3-5, are also all coprime making the implementation of a dominator without at least their product of states, see proof in Section ??, in the machine impossible. These machines should split well based on their behaviours.

10.4 Association Diagrams

In these results we look at the generality of solutions to problem instances, and therefore must have a method to describe over the course of multiple runs, if two problems have similar solutions. In each run of a MAGnet we call two problem instances A and B *associated at the xth level* if there exists a sub-population node where the number of both problem types is greater or equal

to x, i.e. $|A| \geq x$ and $|B| \geq x$. This can also be presented as a percentile of the number of problem instances which are in the populations. When association value is examined over multiple runs it forms a diagram which can imply the existence of general solutions to a subset of problems. These diagrams, Figures 10.5, 10.6, 10.8, 10.9 and 10.10, use white to represent an association found between two problem instances in one-hundred runs of the MAGnet system and black means there is no association found. Note that these results are not corrected for sub-population collapse. This means that the colour of all squares will lighten when the sub-populations collapse to one node. When a general solution exists it typically yields an image with an overall lighter shade.

10.5 Inital Trials

10.5.1 Experimental Settings

A population of twenty-five IPD agents represented by finite state machines, stored as their translation tables, were used for the population of evolving agents. Ten problem instances of each IPD strategy were distributed randomly among the nodes when the network is initialized. We use a fully connected network in this inital trial. The number of nodes in the network was varied between 2 and 4. The number of states in the evolving agent machines was set to 2, 4, and 6.

10.5.2 Canonical Agents

The evolution of the population of agents in K-4/6 yielded a new agent which has been named *Trifecta*, after the horse racing bet which the first three horses must be selected in correct order. Trifecta, Fig 10.2, can very quickly make a decision of which of the three canonical agents it is matched against and play accordingly. It makes a probing defection on the first round. In the

second round, if receiving a cooperation, it knows it must be facing a ALLC and continues to defect. Otherwise, it will respond with a cooperation in round two. If it receives a cooperation it knows it is facing a TFT, otherwise it is an ALLD.

The discovery of this agent, an optimal solution for all three problem cases, prevented the MAGnet from sorting the problem cases. In each replicate of the algorithm, sub-population collapse to a single node occurred. sub-population collapse thus suggests a general solution able to play well against all problem instances has evolved. Trifecta is such a general solution for the problem instances ALLC, TFT, and ALLD. It uses the smallest number of suboptimal moves possible to sort out the problem cases. Trifecta is, however, a special purpose solution for this set of problem cases and would probably not do well in a general tournament. In particular it is not *nice*. It uses an initial defection to determine if the opponent is ALLC or not.

Upon seeing that there was a optimal solution located via the MAGnet approach yielding a sub-population collapse we now can make a further evolutionary study in order to verify the results. Tests on the set {ALLC, ALLD, TFT} where made using a GA. The GA is steady state with a population of 36 machines with the 12 least fit being replaced with the offspring created using the fit 24 parents. The selection is fitness proportional to the score gained in a round robin tournament with the set to be dominated. Each game lasted 150 rounds for 30 replicates. These one again yielded Trifecta (Fig.10.2(a)) in 18/30 runs with the number of states set to 3 and 24/30 when the number of states was increased to 4. One interesting discovery is that there is another version of Trifecta which has state 2 returning to itself and not back to state 1 (Fig. 10.2(b)). While both types will solve the problem as optimals solution to the given set, what is interesting is the final structure. In the second Trifecta state 2 is a sink and plays the strategy of TFT. It should therefore be more robust when meeting a new opponent not within this set; it defaults to playing TFT.

In order to have a test on the robustness of Trifecta and its mutant (Fig. 10.2), once again we can turn to evolution. Looking at the results of the GA we can find the optimal player against the two types to look for functional differences via the best scores. The given 3 states first yields a best average value of 3.333 meaning it is being exploited (an average score greater than 3 implies that a defection was successful) and the second gives a best average value of 2.993 showing it is able to resist exploitation. Looking at the machines created gives the reasoning behind this exploitation. The first version of Trifecta the evolved machines will play $(CDD)^*$. The initial cooperation move ensures that the machine does not enter the always-defect loop. Looking at the path (1,3,2), there is a D/C response used to determine TFT which allows an opponent to defect twice while the opponent responds once. The evolved machine is harmed once for the ability to do harm twice. The second version of Trifecta does not contain this loop making it robust against new opponents.

10.5.3 Expanded Agent Set

The outcome of the second experiment showed interesting properties of MAGnet. It exhibited both the ability to find the relative difficulty of a problem and correlation between problems.

Fig. 10.3 shows the number of non-empty sub-population nodes in the final state of the MAGnet system after the agents have finished sorting the problem cases. Note that as the number of states used in the finite state machines increases the number of sub-population collapses increases. Larger machines are able to model and defeat more complex collection of players. We would expect them to have more general solutions which the sub-population collapses detect.

Fig. 10.4 shows the overall associations found via MAGnet. In Fig. 10.5 the association is shown for a majority, or at the 6th level. Fig. 10.6 shows the much stricter correlation of having ten of both problems are on

the same node; the 10th level. We will now examine these associations for the remainder of this section.

The various modifications of Tit-for-Tat are highly correlated via MAGnet forming a noticeable square in all of the visual representations. They also form a visible association with the Fortress types. Those playing a TFT like style will defect the requisite number of times and will exploit a single cooperate returned by the Fortresses.

Psycho and Always-cooperate show an association. Both are highly exploitable by a defector; ALLC and PSYCHO is beaten optimally by ALLD. ALLD to a lesser extent shows that it is close to these problems in terms of solution for this reasoning. The Fortresses group also is connected with ALLD when the number of states is higher.

The TFT group is not linked to ALLD as they are able to defend against defection and best scores against them come from being cooperators. This would imply that ALLC would be closer to this group. However, ALLC being highly exploitable proves the difference and it is not grouped with the retaliating TFT. An optimal player against the TFT group will want to cooperate and not be able to fully exploit a cooperator.

Fortress-5 requires five states to represent and, as such, it does not associate well with other problems until the number of states in the machine is increased to six. For two states the machine only correlates with Fortress-4, which is also not able to be represented in two states. It is an example of a hard problem being found with the use of the MAGnet system. None of the populations will wish to work on such a problem and therefore there is no reason for it to be moved. The initial scattering remains till the end of the run. This effect is not as visible looking at lower association levels but becomes apparent as the level increases or when the number of states in the machine is increased. While Fortress-4 and 5 both would seem to act as a ALLD given a low number of states, the machines giving the classification are penalized due to the normalization of the fitness. This allows MAG-

net to differentiate between functionally different yet observationally similar machines.

Fig. 10.4 provides an overall look of the classification as discovered via MAGnet. This account closely shows the same divisions that are found via other existing techniques such as finger printing[3][2]. This shows that the technique of MAGnet is doing well at making classifications between the various player types and should be expanded for other agent types. We have seen that MAGnet gives a classification to the problem instances via the creation of players. It uses the experiences of all of these players to move about the problem instances and classify. If we look into the output players on each node we notice the most fit on a node is a good suboptimal approximation for the problem instances on that node. We are classifying and solving using the same method.

10.6 Trials on Graphs

10.6.1 Experimental Settings

It is assumed the reader has some familiarity with graph theory. This section gives only a short review of the necessary properties to understand the results, see [47] for a good reference. A *simple graph* $G(V, E)$ is a non-empty set V of vertexes or nodes and a set E of unordered pairs of elements of V, called edges. Two distinct nodes, v_1 and v_2, are said to be *neighbours* if $(v_1, v_2) \in E$. The number of edges in E which contain a vertex is called the *degree*. A graph is *connected* if there is a pathway via the edges from any vertex to any other vertex. The graphs selected are all connected graphs with thirty-two vertexes, hence, the examination is on the type of connections, and not on the number of nodes. Two values are examined to measure the graphs' connections: *regularity* and *diameter*. If all vertexes in a graph have the same degree it is said to be *regular*. If the common degree of a graph is k, then the graph is said to be *k-regular*. The *diameter* of a graph is the largest

number of edges in any shortest path between any of the vertexes. It can be considered as the shortest path along the graph.

Fig. 10.7 presents the five graphs used for the experiments. The least connective is the Cycle graph C-32. This graph has a regularity of 2, the lowest, and a diameter of 32, the highest. Also selected are two Petersen graphs, P-(16,1) and (16,5). P-(16,1) is a two-cycle graph of size sixteen with a connection between each of the nodes in the outer cycle with the inner cycle. P-(16,5) similarly has two connected groups, the first is a cycle graph of sixteen nodes, the second inner graph is also a cycle but with each fifth vertex connected. Hence P-(16,1) and (16,5) both have the same regularity of 3, however, their diameters are 9 and 6 respectively. The hyper cube of dimension five, H-5, has a regularity of 5 and a diameter of 5. The final graph selected is the complete graph of thirty-two vertexes, K-32, which acts as the baseline as K-5 was used in the previous work. K-32 has the highest regularity of the selected graphs, 31, and lowest diameter with 1.

A population of three times the number of nodes, i.e. ninety-six, was created of finite state machines represented by their transition tables. These evolving agents are initialized at random, and distributed about the nodes at random. The agents have a space of 6 states, which is not enough memory to overcome the hundred rounds in the game. Evolution takes place when an agent moves to a new node. The chance to move an agent was set to 100% and the chance that an agent takes a problem instance to their new location was set to 50%. A tournament selection which compares two randomly selected agents on a node and returns the more fit of the pair was used as the selection mechanism. The operations of crossover and mutation are always applied if there is a breeding step. The MAGnet is run for one-hundred replicates. Thirty-two problem instances of each type of IPD player where randomly distributed around the graph. This is in order to show that the associations provide a true sorting and not variations caused by random chance.

10.6.2 Canonical Agents

In almost all cases for the complete graph there is a collapse to a single node of all problem cases. There is no ability for MAGnet to separate this graph into disconnected sub-graphs; problems all can drop to the single node and Trifecta can emerge. This differs from the test made on the graph K-4 which found that it would always collapse onto a single node. This is in part due to the size of the graph of thirty-two nodes rather than five which allows for a greater chance of two or three nodes spliting the space. Note that the outcome on the main diagonal is as expected, a high relation of a node with itself. Looking at H-5 we see that it classifies each of the agent types into their own node. It allowed for a full separation between the nodes. Looking at the individual runs for H-5, small families of just TFT, ALLC, and ALLD nodes occur. In a few cases nodes join TFT and ALLC, the nice players, with ALLD going off to its own nodes. The levels of TFT on such nodes are always slightly higher than the ALLC in order to prevent the ESS from being disrupted. As an evolutionary algorithm does not require a replacement if the score is strictly higher, both TFT and ALLC will score the same against a *nice* opponent. This forms a random walk on the levels of TFT and ALLC which can be exploited over the population by a ALLD. This explains the darker links between them and the darker main diagonal between K-32 and H-5.

For the cycle graph and the two Peterson graphs there is no informative association produced via MAGnet. The reason for this is the properties of sub-population collapse based on the regularity and degree of the graphs involved. All of these graphs have a low regularity. If a node loses all of its problem instances, which will probabilistically be likely to happen given the types of agents, then it is in effect removed from the graph; the node has the worst possible fitness score and no agent would choose to move to that location. A node undergoing such a collapse reduces the degree of all neighbouring nodes, and given a few of these nodes occurring will break down

the graph into a number of disjoint subgraphs. These subgraphs will then independently sort the instances given the agents and instances in each subgraph. When a graph is fully connected or has a high regularity, their is little chance of such a disjoint graph occurring. Hence, we see more association as the nodes removed do not change the graph's connectedness.

Looking at individual runs in the first experiment, however, tells an interesting story when it comes to why these associations are not available. Looking at the results for the Peterson and Cycle graphs, a large number of nodes still have problem instances, but these nodes are all of a distance of two or more away of any other node with problems. That confirms the idea of a breakdown in the graphs via the sub-population collapse. On the individual remaining nodes there is a large number of a few problem instances of the same type, and usually one or two with a more sizable amount of all the types in a close to equal proportion. This seams to imply that Trifecta has been created on the large nodes and has drawn them together. However, other nodes have been 'stranded' as all of their neighbours flood into them. The diameter plays a smaller role in this than the degree of regularity. There are a few cases in which neighbours stay joined with problem instances. Those problem instances are for the most part the same type of player. This would develop an equilibrium in which the agents would either not move as both of the nodes are equally fit, or in the case of having a single difference, would mutate into each other creating an oscillation of agents and problem cases between the two nodes.

As an additional test of the system, the number of problem instances was doubled to sixty-four and the accepted level of association was halved to 25% before the Peterson graphs and the Cycle graph displayed the ability to classify the problem instances. This is demonstrated in Figure 10.9.

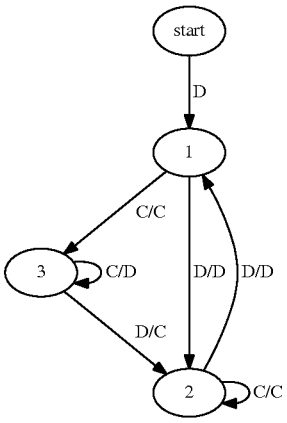
10.6.3 Expanded Agent Set

The second testing set shows very similar properties for each of the graphs as shown with the smaller canonical set. Looking at the levels of association, Fig 10.10, the complete graph K-32 presents an outcome which is close to K-4, though it highlights more hard connections between the two, due to the greater number of nodes. It cleans out some of the noise seen in K-4, due to random chance fluctuations of being placed on the same graph. H-5 shows a cleaner separation than K-32 on some of the agent types. For example, there is a very stark difference visible between 2TFT and TF2T/TFT via the H-5 graph not seen in K-32. This is interesting as there is a method of exploiting 2TFT by making a defection on every other move. This ability for exploitation is not seen in TF2T or TFT. PSYCHO is also separated more from ALLD, and ALLC and ALLD are more separated.

The fortresses stand alone, being not even associated with themselves. The handshaking strategies seem to be more resistant to being discovered by this behavioural technique of classification. Seeing as how they were originally discovered via a deep evolution, and they protect their behaviours closely, this is not surprising that MAGnet is having a hard time associating them to themselves. An agent would have to learn the hidden secret handshake in order to score. This was perhaps easier on a smaller graph than the thirty-two nodes presented to the agents in this case.

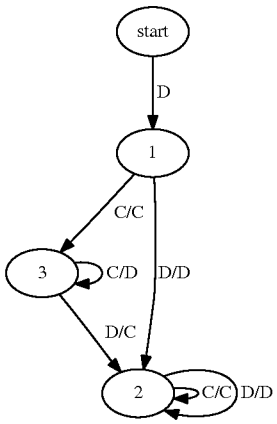
Looking at the higher diameter graphs, again the associations move to nothing. This is most likely due to the sub-population collapse effect cutting the graph into disjoint subsets which will be unable to properly provide a classification. Looking at the individual runs confirms this, as the nodes have been cut off from their neighbours. The cycle graph seems to show the most chaos in how it is sorted after the run. The runs in general show that the fortresses are being classified with each other and themselves but they are very likely to be on small disjoint nodes with only a few of them. This could be due to long deep evolution required, and the strategy for playing against

them not easily emerging.



Initial:	D	
State	If C	If D
1	C→ 3	D→ 2
2	C→ 2	D→ 1
3	D→ 3	C→ 2

(a) Type A



Initial:	D	
State	If C	If D
1	C→ 3	D→ 2
2	C→ 2	D→ 2
3	D→ 3	C→ 2

(b) Type B

Figure 10.2: Trifecta agent types. Note the difference in the final state between returning and a TFT like state. The first can be exploited by a play of $(CDD)^*$, the second will drop into the TFT state and is more defensive.

States	Non-Empty sub-populations	
	2	1
2	96	4
4	71	29
6	64	36

(a) 2 Nodes

States	Non-Empty sub-populations		
	3	2	1
2	91	9	0
4	30	53	17
6	20	57	23

(b) 3 Nodes

States	Non-Empty sub-populations			
	4	3	2	1
2	85	15	0	0
4	13	32	44	11
6	4	21	52	23

(c) 4 Nodes

Figure 10.3: Study of the sub-population Collapse Effect. Number of non-empty sub-populations remaining (nodes with problem instances) at end of evolution for a beginning number of initial nodes.

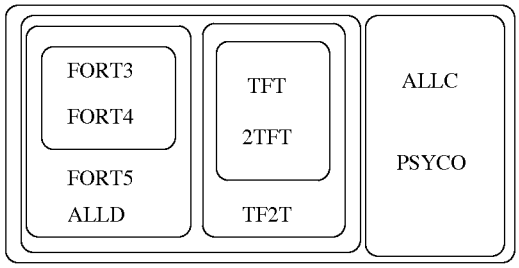
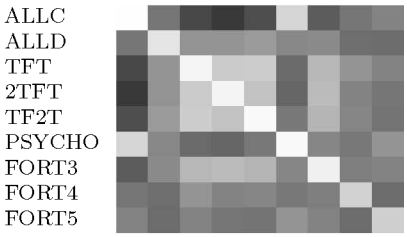


Figure 10.4: Associated groups found by MAGnet

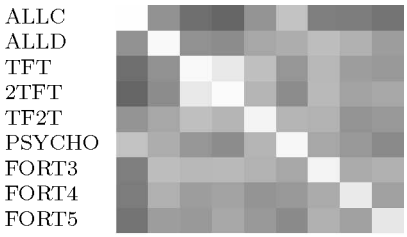


(a) 2 Nodes, 2 States

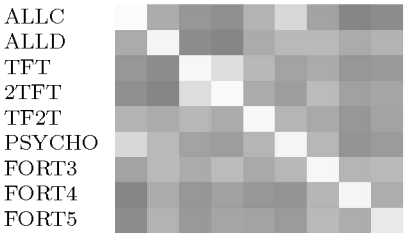
(b) 2 Nodes, 4 States

(c) 2 Nodes, 6 States

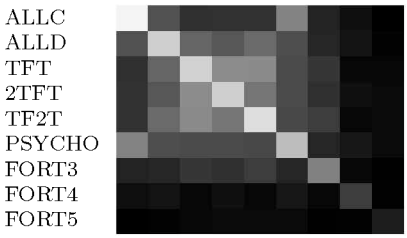
(d) 3 Nodes, 2 States



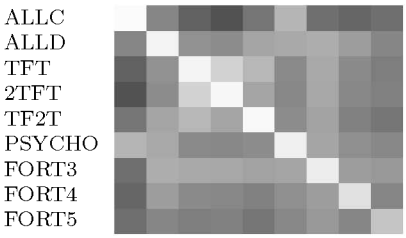
(e) 3 Nodes, 4 States



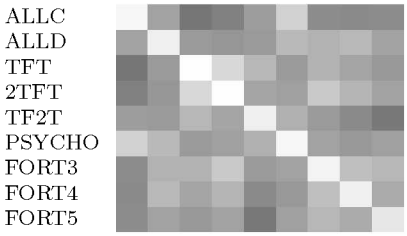
(f) 3 Nodes, 6 States



(g) 4 Nodes, 2 States

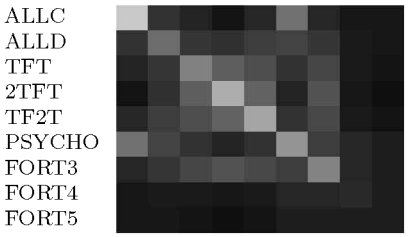


(h) 4 Nodes, 4 States

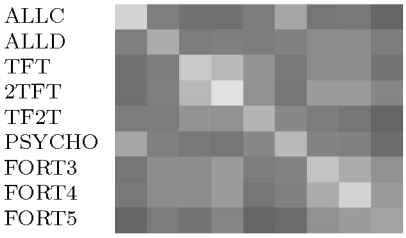


(i) 4 Nodes, 6 States

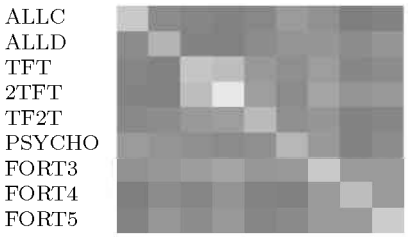
Figure 10.5: Number of Associations at the 6th Level where White is 100/100 and Black is 0/100



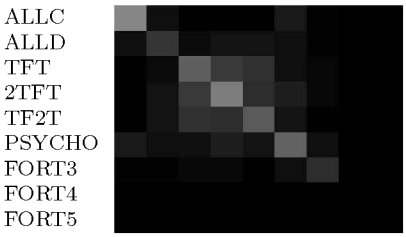
(a) 2 Nodes, 2 States



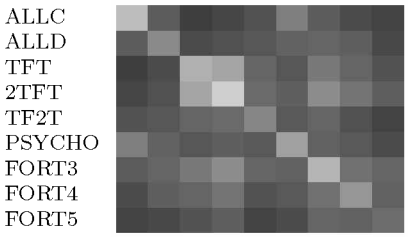
(b) 2 Nodes, 4 States



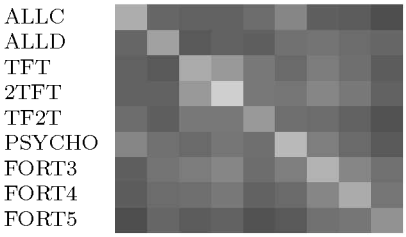
(c) 2 Nodes, 6 States



(d) 3 Nodes, 2 States

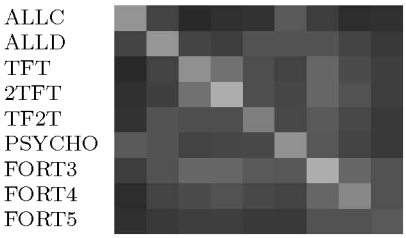


(e) 3 Nodes, 4 States

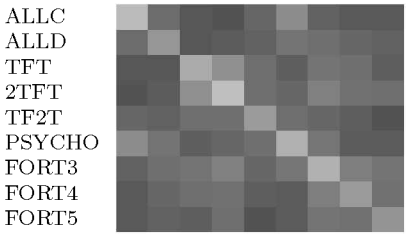


(f) 3 Nodes, 6 States

(g) 4 Nodes, 2 States

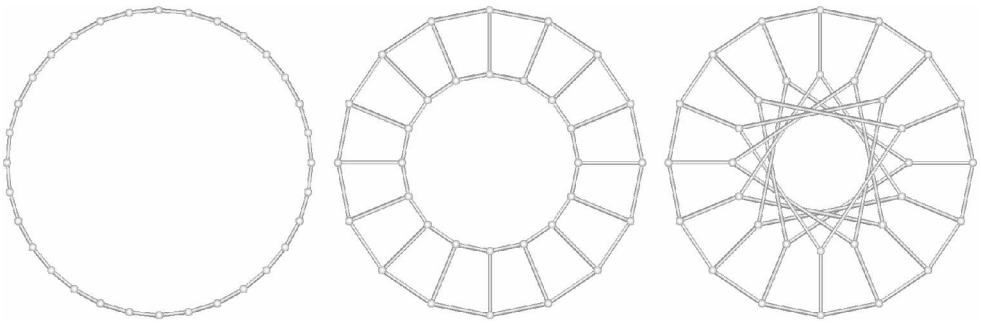


(h) 4 Nodes, 4 States

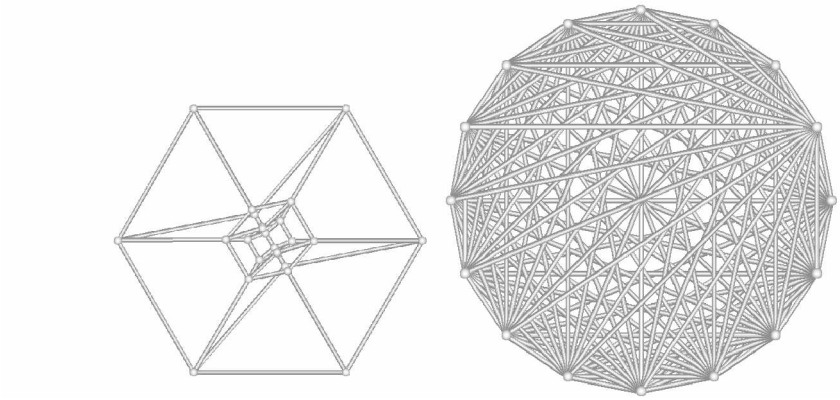


(i) 4 Nodes, 6 States

Figure 10.6: Number of Associations at the 10th Level where White is 100/100 and Black is 0/100



(a) Cycle Graph — C-32 (b) Petersen Graph — P-(16,1) (c) Petersen Graph — P-(16,5)



(d) Hypercube of Dimension Five — H-5 (e) Example Fully Connected Graph — K-16 (Used is K-32)

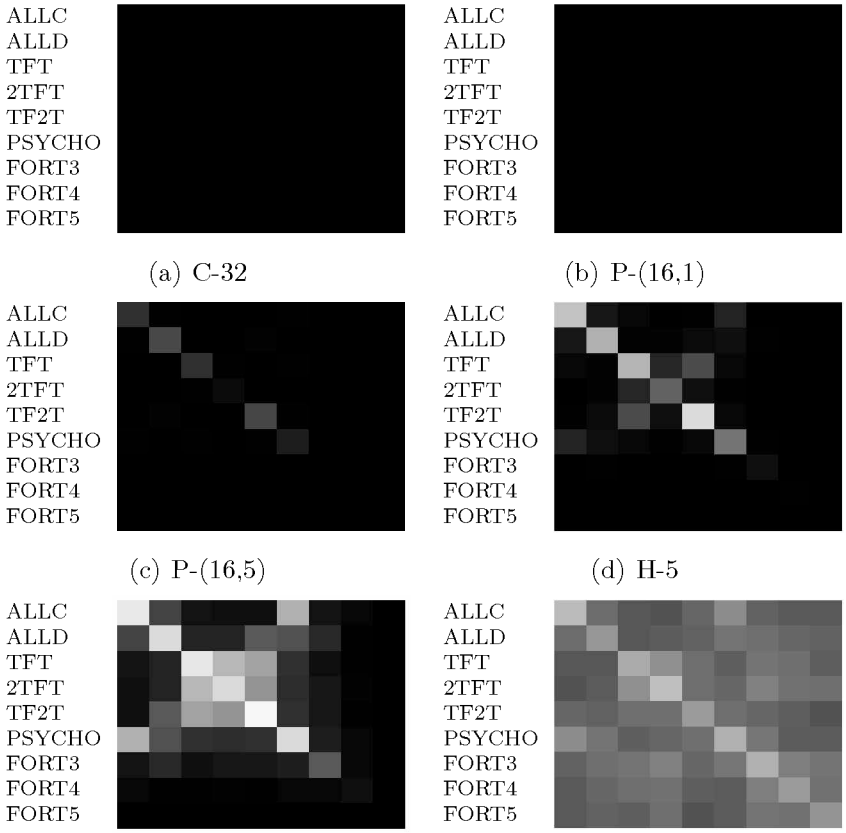
Figure 10.7: Graphs used for the MAGnet experiments from least to most regularity

(a) C-32 (b) P-(16,1) (c) P-(16,5) (d) H-5 (e) K-32

Figure 10.8: Association at the 50% level (16-problem instances on the same node out of 32) for the set of canonical agents: ALLC, ALLD, TFT.

(a) C-32 (b) P-(16,1) (c) P-(16,5)

Figure 10.9: Association at the 25% level (16-problem instances on the same node out of 64) for the set of canonical agents with increased number of problem instances.



(a) C-32 (b) P-(16,1)

(c) P-(16,5) (d) H-5

(e) K-32 (f) K-4, 6 states

Figure 10.10: Association at the 50% level (16-problem instances on the same node out of 32) for the five graphs using the second larger set of agent types. Presented in 10.10(f) is K-4 with 6 states from the previous work to allow for comparison tool. Settings for K-4 can be found in [21], and differ from these tests.

Chapter 11

Dominators

The results from this chapter were first presented in *Domination in Iterated Prisoner's Dilemma* [26].

11.1 Evolutionary Stable Strategies

Evolutionary Stable Strategies [75] have been shown not to be stable for a number of evolutionary algorithm populations [42][39][12]. This failure is most notably caused by the evolutionary algorithms having finite population, compared to the infinite expansion required for the determination of an ESS. Also ESS only allow for an organism to persist if it has a strictly higher fitness than the others in the population. However, the majority of EAs allow for those which are equally fit to persist in the population. Dominators are an alternative description of the properties which an ESS is trying to model, which can be used when the assumptions required to have an ESS fail. Dominators will present given a finite population of a set of agents, where the evolution tend towards.

11.2 Domination

We now begin our look at the idea of domination starting with the definition.

Remark 11.2.1. Let $v(\mathcal{S}, \mathcal{T})$ denote the average value a finite state strategy $\mathcal{S}$ receives playing against a set of finite state strategies $\mathcal{T}$ for an indefinitely long number of iterations.

Definition 11.2.2. A finite state strategy $\mathcal{S}$ is a *dominator* of a set of finite state strategies $\mathcal{T}$, denoted by $\Omega(\mathcal{S}, \mathcal{T})$, if and only if there does not exist a finite state strategy $\mathcal{S}'$ such that:

$$v(\mathcal{S}, \mathcal{T}) < v(\mathcal{S}', \mathcal{T})$$

Definition 11.2.3. A finite state strategy $\mathcal{S}$ is a *self dominator* if and only if $\Omega(\mathcal{S}, \mathcal{S})$.

Definition 11.2.4. A finite state strategy $\mathcal{S}$ is *nice* if and only if $\mathcal{S}$ never defects first.

Note that unlike classical dominant strategies, there is no concept of a strict dominance. Dominators in finite state machines are in actuality a potentially infinite set, which can be described as an equivalence class with a minimal machine. Note that some areas of these machines may not be accessed by those dominated. However, it could be exploited or allow exploitation of other machines. For example, $\Omega(ALLC, TFT)$ and $\Omega(TFT, TFT)$ implies that in a world consisting of only people playing TFT a cooperator can exist harmoniously as ALLC is indistinguishable in terms of behaviours. Yet, ALLC is the most exploitable player type, as no matter how many times you stab it in the back it will continue to cooperate. TFT never exploits others and therefore, it will ignore large areas of functionality of another *nice* player. This property becomes useful when we wish to define behaviours. The set of all dominators can be represented by grouping equivalent domina-

tors of a given set of strategies into an equivalence class which is represented with a single machine with a minimal number of states.

This idea of a *dominator* allows for a formalization of properties which a IPD playing finite state machine can have. Axelrod's idea of *nice* is easily represented as the set of all *dominators* of the singleton set of Tit-for-Tat. Note that we assume scores are those resulting after a large number of plays, effectively the asymptotic score.

Theorem 11.2.5. *A state strategy $\mathcal{S}$ is* nice *if and only if $\mathcal{S}$ is a* dominator *of the singleton set containing Tit-for-Tat.*

Proof. Tit-for-Tat is *nice* but will defect given a single defection. If $\mathcal{S}$ is a *nice* strategy then it will receive C in each round playing *TFT*. If $\mathcal{S}$ is not nice it will defect receiving T; it may then at some point return to cooperation receiving S. As $T + S < 2C$ and $D < C$ this is not optimal and it cannot be a dominator. Otherwise, $\mathcal{S}$ defects forever at a certain iteration. Then $\mathcal{S}$ and Tit-for-Tat will at least tie on round $\lfloor x \rfloor$ since the first defection where:

$$T + (x - 1)D = xC \tag{11.1}$$

$$x = \frac{D - T}{D - C} \tag{11.2}$$

On the $\lfloor x \rfloor + 1$ round it receives D. As $D < C$ the machine plays non optimally on the $\lfloor x \rfloor + 1$ round and beyond and cannot be a *dominator*. Therefore, the only *dominators* are *nice*. □

This proof is interesting beyond giving a formation for *nice*. It shows that the payoff values for finite state machines matter in terms of the dominators which will be expressed. This gives a gives a rigourous foundation for investigation of the empirical findings by Ashlock and Kim [4] that changes to the payoffs will result in differing final sets of agents being created. In a co-evolutionary algorithm of IPD agents, a single member of the population will have a high fitness by creating the dominator of the current population.

By changing the payoffs there may be a change in dominators, which leads to differing final populations dependant on payoff.

As we have the property of *nice* we can also imply the existence of and give a formalization for *anti-nice*.

Definition 11.2.6. A finite state strategy $\mathcal{S}$ is *anti-nice* if and only if $\mathcal{S}$ never cooperates first.

Theorem 11.2.7. *A finite state strategy $\mathcal{S}$ is* anti-nice *if and only if $\mathcal{S}$ is a* dominator *of the singleton set containing Always Defect.*

Proof. Always Defect is *anti-nice* as it will never cooperate. If $\mathcal{S}$ is *anti-nice* it will score D on each round of play. Otherwise, $\mathcal{S}$ will cooperate during a iteration and score S. As $S < D$ it cannot be a *dominator* of Always Defect and the only *dominators* of Always Defect will hence be *anti-nice*. □

These proofs give examples as to how the notion of dominator can be used in order to formalize behavioural concepts of IPD agents in a single mathematical framework.

11.3 Domination of Common Agents

The set of $\{ALLC, ALLD, TFT\}$ is dominated by *Trifecta* (Fig. 10.2). This player type has some interesting properties. *Trifecta* while a dominator, is not a self dominator and will act as a *ALLD* when playing itself. Further, $\Omega(Trifecta, ALLD)$ so it is *anti-nice*. Note that it not true that $\Omega(Trifecta, TFT)$ or $\Omega(Trifecta, ALLC)$, showing the existence of a Dominator of a set which does not dominate the singletons.

While both types will solve the problem of dominating the given set, interesting is the two final structures. In the *Trifecta* type B (Fig. 10.2(b)) state 2 is a sink and plays the strategy of *TFT*. It is therefore more robust when meeting an opponent not within the dominated set. Examining the

Type A (Fig. 10.2(a)) version of *Trifecta* the dominating machines will play $(CDD)^*$. The initial cooperation move ensures that the machine does not enter the always-defect loop. Looking at the path (1,3,2), there is a D/C response used to determine TFT which allows an opponent to defect twice while the opponent responds once. It is harmed once for the ability to do harm twice. The Type B has a dominator of TFT and not containing this loop makes it robust against new opponents which it has not encountered as it will play a TFT.

Chapter 12

Conclusions

12.1 Discussion

K-models, a generalization of the K-means algorithm, is a starting point for partitioning regression, that is the ability to classify using models to fit a set of data. K-means has shown itself to be a widely used in a variety of areas for clustering data types, making it a good method to generalize. Where a different statistically model is known, K-models gives a more meaningful representation and model to the data. However, where the model is not immediately determinable, evolutionary methods can provide a solution to the issue of breaking down a set of data via a meaningful set of models.

With this end in mind, two novel EAs have been developed, Multiple Worlds and Multiple Agent Genetic Algorithms. Both have the ability to make classifications for partitioning, regression, and model discovery. There use in modeling regressions has developed into partitioning regression which uses regression in order to act as a mean of partitioning. There intended use in bioinformatics has produced results. In modeling of demographics there has been an examination of a pedagogical example, with a more detailed model this could provide useful in real world applications. Finally, they provide interesting divisions of game playing agents, and have discovered

new agent types, such as *Trifecta*.

The creation of these algorithms have lead to the rethinking of a variety of issues in Game Theory leading to a new formalization known as Dominator Theory. It allows for a better formalism of a number of known properties in IPD, such as *nice*. It does not make assumptions about the population of the system being infinitely large and expanding, problems which preclude the use of ESS for analysis of evolutionary algorithm populations.

The process of sub-population collapse shows itself to be a valuable process for the discovery of the number of categories within a data set. This is especially evident when there is not a direct error reward as used in the MWM v. GA comparison. In cases such as the HLA motif finding, the difference between having a good classifier is achieved by using a number of classes is one more than the actual. The addition of this extra class provides a similar benefit as the error term. Both error terms and extra classes for the populations not only to be just better than the other populations, but also better at the problem. The error term is an explicit requirement of a need to do well on the problem, the extra population is an implicit method which forces the populations to hold onto their own niches of the data. This extra class, however might be a subset of a class in the system.

12.2 Future Directions

12.2.1 Multiple Model Representations

K-models has only been examined in cases where there is the same model for each group. The algorithm tests off all points on their error to a model, then adjusts each model independently. Hence, there is no limitation on having a single model represent the data. A combination of models could be applied when there is known statistical differences. This also has implications for MWM having multiple representations.

Again, the type of model has been set as static between the populations

for MWM. There is no transfer of genetic material between the populations, only a test of fitness between them which amounts to an objective function. If we can give the same objective function or at least a fairly compaired objective function, then there is no limitation on having each population having the same representation. Multiple different representations could be attempted, and if the number of representation times was larger than the number of classes in the data they could represent, then there is an implicit test on which representation is better on the data using sub-population collapse as a discriminator.

MAGnet, as there is genetic transfer between members would be more difficult to incorporate a change in the model. However, if the crossover operation is removed or limited to models with the same representation, treating each representation like a species, or only mutation is allowed, such as seen in ES and EP, then it would be possible to have multiple models moving about the network, sorting problem instances.

12.2.2 MWM Collapse Analysis

We have looked at some of the properties of a collapse in MWM and how it allows in many cases for a suitable number of partitions to be found. Expansions should be made in order to allow the algorithm to automatically restart/diverge when there is too powerful a collapse event, such as only having a single population take all of the points with a model of high error. This would be an explicit creation of a new species where there is a niche available. Further, analysis of resultant classifiers should be automated or formalized to see if a collapse is a result of the model or just a fluke. For motif finding we found motif classified which were subclassifications of other good motifs. Note that the degenerate motifs have a structure which makes such subclassifiers likely, each loci forms a poset under the 'or' relation, so such issues are representation dependant.

12.2.3 Cooperative Coevolution with MWM

MWM to this point has been used for the evolution of interacting models which test their fitnesses against each other. However, a slight change can be made for a co-evolutionary version of MWM which would allow for a design with multiple cooperating parts — a shared fitness for a world. Each of the model segments would evolve based on a fitness evaluation, yet the interactions with the other parts would have an implicit effect on the direction of the evolution. Unlike a chromosome with multiple loci, the loci being evolved separately avoids some of the issues of genetic draft, where an allele may increase its frequency by being connected to a gene which is undergoing a positive selection, as seen in the MWM v. GA exercise.

For example, an artistic application of such a model would be the creation of fractals with iterative applications. Feeding the output of an escape value from a fractal into another fractal equation produces a iterated fractal. Each of these fractal equations would be an evolving population, and the combined fitness of the world, determined by an objective fitness evaluation based on the final escape values, would be given to each of the segments. This is a cooperative rather than competitive use of the system.

12.2.4 Error Transform

The creation of regressors can allow for the classification of harder data sets. Once regressors are chosen, they can be used to perform an error transform. This transform maps a data point to the vector of the modeling error of multiple regressors on the data point. The transformed data should (i) be often far more separable than the original data, (ii) be coerced to have a higher or lower dimension, depending on the application, (iii) permit a simple classification of new data points that did not participate in selection of the regressors. Industrial process data is one possible real world application for such techniques.

12.2.5 Motif Finding and Biological Data

Previous application of the MWM to motif finding centred on the discovery of motifs in biological data. It has been found that models taking into account short substring motifs were found more often than those with a reverse complement. This is due to the representational problems of using the chromosome of a degenerate motif expression. Representation of a motif in an evolving structure will need to be examined further in order to allow the system to examine such sequences. Such use of different representations may allow for a better final classification via an error transform.

In terms of biological data an application can be seen in the classification of sequences in *tetrahymena*. This organism is an eukaryotic ciliate. One of the most interesting biological features of this model organism is the existence of a nuclear dimorphism — that is, it has two cell nuclei, one of them being a MAC (macronucleus) created as somatic expansion, decompression, of a small germline MIC (miconucleus). Both have great structural and genetic differences. During the expansion of a MIC into a MAC, two types of sequences are present: internal eliminated sequences, which are removed and are not present in the MAC, and sequences which code for the MAC.

12.2.6 Dominator Theory

Dominator theory should be expanded to discover other properties which can be defined by a optimal player. Further, the current dominators looked at have been based on a single agent against a set of agent types. This could be generalized to a single agent against a multiset, or a set of agents against a set. Further, dominators are now defined on an average score over an infinite game. As is seen in the proof about dominating TFT, the payoffs based on the number of rounds for which game runs can affect the outcome of what is a dominate strategy. A generalization to dominators could take into account games of a finite number of moves k, which would be called a k-dominator.

12.2.7 Other Matrix Games

IPD has been the focus of this study as it is a well known matrix game. However, a wide variety of matrix games exist, and in general few have been examined for strategic player types such as Tit-for-Tat or Fortresses in IPD. Such games as Hawk-Dove, Chicken, or Rock-paper-scissors(-lizard-Spock), etc. when played iteratively may have interesting properties and players. Both MAGnet and Dominator Theory could be applied in the search for player types and the determination of their behavioural properties.

Bibliography

[1] D. Ashlock and L. Guo. Evolutionary parameter setting of multi-clustering. In *Proceedings of the 2007 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology*, pages 25–31. IEEE Press, 2007.

[2] D. Ashlock, Eun-Youn Kim, and W.K. vonRoeschlaub. Fingerprints: enabling visualization and automatic analysis of strategies for two player games. In *Evolutionary Computation, 2004. CEC2004. Congress on*, volume 1, pages 381–387 Vol.1, June 2004.

[3] D. Ashlock and E.Y. Kim. Fingerprinting: Visualization and analysis of prisoner's dilemma strategies. *IEEE Transactions on Evolutionary Computation*, 12(5):647–659, 2008.

[4] D. Ashlock and E.Y. Kim. Prisoner's dilemma: the payoff values matter. In *Proceedings of the 2010 IEEE Conference on Computational Intelligence in Games*, pages 219–226, 2010.

[5] Daniel Ashlock. *Evolutionary Computation for Modeling and Optimization.* Springer, 2006.

[6] Daniel Ashlock. Training function stacks to play the iterated prisoner's dilemma. In *2006 IEEE Symposium on Computational Intelligence and Games*, pages 111–118, 2006.

[7] Daniel Ashlock, Joseph Alexander Brown, and Steve Corns. K-models clustering, a generalization. In *Intelligent Engineering Systems Through Artificial Neural Networks*, volume 20, pages 485–492. AMSE Press, 2010.

[8] Daniel Ashlock and Andrew McEachern. Ring optimization of side effect machines. In *Intelligent Engineering Systems Through Artifical Neural Networks*, volume 19, pages 165–172. AMSE Press, 2009.

[9] Daniel Ashlock and T. von Konigslow. Evolution of artificial ring species. In *IEEE Congress on Evolutionary Computation*, pages 653–659, 2008.

[10] Daniel Ashlock, T. von Konigslow, E. Clare, and W. Ashlock. Transience in the simulation of ring species. In *Proccedings of the 2008 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology*, pages 256–263. IEEE Press, 2008.

[11] W. Ashlock. Why some representations are more cooperative than others for prisoner's dilemma. In *2007 IEEE Symposium on Foundations of Computational Intelligence (FOCI)*, pages 314–321, 2007.

[12] W. Ashlock and D. Ashlock. Changes in prisoner's dilemma strategies over evolutionary time with different population sizes. In *Proceedings of the 2006 Congress On Evolutionary Computation*, pages 1001–1008, 2006.

[13] Douglas Adriano Augusto, Helio José Corrêa Barbosa, and Nelson Francisco Favilla Ebecken. Coevolutionary multi-population genetic programming for data classification. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, GECCO '10, pages 933–940, New York, NY, USA, 2010. ACM.

[14] Robert Axelrod. *The Evolution of Cooperation.* Basic Books, New York, 1984.

[15] T.L. Bailey and C. Elkan. Fitting a mixture model by expectation maximization to discover motifs in biopolymers. In *Proceedings of the Second International Conference on Intelligient Systems for Molecular Biology*, pages 28–36, 1994.

[16] M. Balnaves, L. Ferrier, G. Phillips, and T. O'Regan. Introducing ratings in transition. *Media International Australia*, (105):6–9, 2002.

[17] Colleen Belk and Virginia Borden Maier. *Biology : Science for Life.* Benjamin Cummings, third edition, 2010.

[18] Ricardo Bianchini and Christopher M. Brown. Parallel genetic algorithms on distributed-memory architectures. Technical report, University of Rochester, Rochester, NY, USA, 1993.

[19] Christopher M. Bishop. *Pattern Recognition and Machine Learning.* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[20] Joseph Alaxander Brown. Examination of graphs in multiple agent genetic networks for iterated prisoner's dilemma. In *2013 IEEE Conference on Computational Intelligence in Games (CIG 2013)*, 2013.

[21] Joseph Alexander Brown. Multiple agent genetic networks for iterated prisoner's dilemma. In *2011 IEEE Symposium on Foundations of Computational Intelligence (FOCI)*, pages 7–14, 2011.

[22] Joseph Alexander Brown. Multiple worlds model for motif discovery. In *2012 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*, pages 92–99, 2012.

[23] Joseph Alexander Brown. More multiple worlds model for motif discovery. In *2013 IEEE Symposium on Computational Intelligence in*

Bioinformatics and Computational Biology (CIBCB), pages 168–175, 2013.

[24] Joseph Alexander Brown, D. Ashlock, J. Orth, and S. Houghten. Autogeneration of fractal photographic mosaic images. In *Evolutionary Computation (CEC), 2011 IEEE Congress on*, pages 1116–1123, June 2011.

[25] Joseph Alexander Brown and Daniel Ashlock. Using evolvable regressors to partition data. In *Intelligent Engineering Systems Through Artificial Neural Networks*, volume 20, pages 187–194. AMSE Press, 2010.

[26] Joseph Alexander Brown and Daniel A. Ashlock. Domination in iterated prisoner's dilemma. In *24th Canadian Conference on Electrical and Computer Engineering (CCECE)*, pages 1125–1128, 2011.

[27] K.M. Bryden, D.A. Ashlock, S. Corns, and S.J. Willson. Graph-based evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 10(5):550 –567, oct. 2006.

[28] E. Cantu-Paz. A survey of parallel genetic algorithms. Technical Report 95004, Illinois Genetic Algorithms Laboratory, Urbana, IL, 1997.

[29] R. H. Coase. Payola in radio and television broadcasting. *Journal of Law and Economics*, 22(2):269–328, Oct 1979.

[30] Sylvain Cussat-Blanc, Fabien Viale, Herve Luga, Yves Duthen, and Denis Caromel. Genetic algorithms and grid computing for artificial embryogeny. In *Proceedings of the 10th annual conference on Genetic and evolutionary computation*, GECCO '08, pages 281–282, New York, NY, USA, 2008. ACM.

[31] Charles Darwin. *The Variation of Animals and Plants Under Domestication — II*, volume 8 of *The Works of Charles Darwin*. AMS Press, New York, 1896.

[32] Charles Darwin. *On the Origin of Species : by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life*. Sterling, New York, illustrated edition, 2008.

[33] B.V. Dasarathy and B.V. Sheela. A composite classifier system design: Concepts and methodology. *Proceedings of the IEEE*, 67(5):708 – 713, may 1979.

[34] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *Evolutionary Computation, IEEE Transactions on*, 6(2):182–197, 2002.

[35] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 1(39):1–38, 1977.

[36] T Dobzhansky. A critique of the species concept in biology. *Philosophy of Science*, 2:344–355, 1935.

[37] T Dobzhansky. *Genetics and the Origin of Species*. Columbia Univ. Press, New York, 3rd edition edition, 1951.

[38] Ryan Edwardson. *Canadian content: culture and the quest for nationhood*. University of Toronto Press, 2008.

[39] S. G. Ficici and J. B. Pollack. Effects of finite populations on evolutionary stable strategies. In *Proceedings of the 2000 Genetic and Evolutionary Computation Conference*, pages 927–934, 2000.

[40] D. B. Fogel. Evolving behaviors in the iterated prisoner's dilemma. *Evol. Comput.*, 1(1):77–97, 1993.

[41] D. B. Fogel. On the relationship between the duration of an encounter and the evolution of cooperation in the iterated prisoner's dilemma. *Evol. Comput.*, 3(3):349–363, 1995.

[42] D. B. Fogel, G. B. Fogel, and P. C. Andrews. On the instability of evolutionary stable strategies. *BioSystems*, (44):135–152, 1997.

[43] G.B. Fogel, G.W. Greenwood, and K. Chellapilla. Evolutionary computation with extinction: experiments and analysis. In *Proceedings of the 2000 Congress on Evolutionary Computation*, volume 2, pages 1415 –1420 vol.2, 2000.

[44] Lawrence J. Fogel, Alvin J. Owens, and Michael J. Walsh. *Artifical Intelligence through Simulated Evolution.* John Wiley & Sons, New York, 1966.

[45] D. E. Goldberg and J. Richardson. Genetic algorithms with sharing for multimodal fitness optimization. In *Proceedings of the 2nd ICGA*, pages 41–49, 1987.

[46] David E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley Professional, January 1989.

[47] E. G. Goodaire and M.M. Parmenter. *Discrete Mathematics with Graph Theory (3rd Edition).* Prentice-Hall, Inc., 2005.

[48] B. Rosemary Grant and Peter R. Grant. The feeding ecology of Darwin's ground finches. *Noticias de Galápogos*, pages 14–18, 1979.

[49] B. Rosemary Grant and Peter R. Grant. Niche shifts and competition in Darwin's finches: *Geospiza conirostris and Congeners. Evolution*, 36:637–657, 1982.

[50] B. Rosemary Grant and Peter R. Grant. Fission and fusion in a population of Darwin's finches: An example of the value of studying individuals in ecology. *Oikos*, 41:530–547, 1983.

[51] G.W. Greewood, G.B. Fogel, and M. Ciobanu. Emphasizing extinction in evolutionary programming. In *Proceedings of the 1999 Congress on Evolutionary Computation CEC 99*, pages 666–671, 1999.

[52] Thor Hanson. *Feathers: Evolution of a Natural Miracle.* Basic Books, New York, 2011.

[53] Michael Hemesath. Cooperate or defect? Russian and American students in prisoner's dilemma. *Comparative Economics Studies*, 176:83–93, 1994.

[54] G. Z. Hertz and G. D. Stormo. Identifying dna and protein patterns with statistically significant alignments of multiple sequences. *Bioinformatics*, 15(7):563–577, 1999.

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

[56] John E. Hopcroft, Rajeev Motwaniand, and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation (3rd Edition).* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2006.

[57] D. E. Irwin, S. Bensch, and T. D. Price. Speciation in a ring. *Nature*, 409:333–337, 2001.

[58] H. Ishibuchi, H. Ohyanagi, and Yusuke Nojima. Evolution of strategies with different representation schemes in a spatial iterated prisoner's dilemma game. *IEEE Transactions on Computational Intelligence and AI in Games*, 3(1):67–82, 2011.

[59] A. K. Jain. Data clustering: 50 years beyond k means. *Pattern Recognition Letters*, 31(8):651–666, 2010.

[60] J. Jayachandran and S. M. Corns. A comparative study of diversity in evolutionary algorithms. In *2010 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–7, 2010.

[61] K. A. De Jong. *An analysis of the behaviour of a class of genetic adaptive systems.* PhD thesis, University of Michigan, 1975.

[62] E. Y. Kim, S.Y. Kim, D. Ashlock, and D. Nam. Multi-k: accurate classification of microarray subtypes using ensemble k-means clustering. *BMC Bioinformatics*, 10(260):1–12, 2009.

[63] Bruce C. Klopfenstein. Audience measurement in the vcr enironment : An examination of rating methodologies. In Julia R. Dobrow, editor, *Social and Cultural Aspects of VCR Use*, chapter 3, pages 45–72. Routledge, 1990.

[64] John R. Koza. *Genetic Programming: on the Programming of Computers by Means of Natural Selection.* MIT Press, Cambridge, MA, USA, 1992.

[65] David Lack. *Darwin's Finches.* Cambridge University Press, 1947.

[66] C. Liu, Y. Song, M. Garuba, and L. Burge. Hmmf: An hidden markov model based approach for motif finding. In *3rd International Conference on Bioinformatics and Biomedical Engineering*, pages 1–3, 2009.

[67] F.F.M. Liu, J.J.P. Tsai, R.M. Chen, S.N. Chen, and S.H. Shih. Fmga: finding motifs by genetic algorithm. In *IEEE Fourth Symposium on Bioinformatics and Bioengineering*, pages 459–466, 2004.

[68] J.S. Liu, A.F. Neuwald, and C.E. Lawrence. Bayesian models for multiple local sequence alignment and gibbs sampling strategies. *J. Am. Stat. Assoc.*, 90(432):1156–1170, 1995.

[69] X. Liu, D.L. Brutlag, and J.S. Liu. Bioprospector: discovering conserved dna motifs in upstream regulatory regions of co-expressed genes. *Pacic Symposium of Biocomputing*, 6:127–138, 2001.

[70] Stuart P. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–136, 1982.

[71] Ricardo Lopes and Rafael Bidarra. Adaptivity chalanges in games and simulations: A survey. *IEEE Transactions on Computational Intelligence and AI in Games*, 3(2):85–99, June 2011.

[72] David J. C. Mackay. *Information Theory, Inference & Learning Algorithms.* Cambridge University Press, June 2002.

[73] Sylvia S. Mader. *Inquiry into Life.* McGraw-Hill, tenth edition, 2003.

[74] C. Marshall. Mass extinction probed. *Nature*, (392):17–20, 1998.

[75] J. Maynard-Smith. *Evolution and the Theory of Games.* Cambridge University Press, Cambridge, UK, 1982.

[76] E. W. Mayr. *Systematics and the Origin of Species from the Viewpoint of a Zoologist.* Cambridge: Harvard University Press, 1942.

[77] E. W. Mayr. What is a species, and what is not? *Philosophy of Science*, 63:262–277, June 1996.

[78] R. J. McEliece. *The Theory of Information and Coding.* Cambridge University Press, New York, NY, 1977.

[79] G. H. Mealy. A method of synthesizing sequential circuits. *Bell Systems Technical Journal*, 34:1054–1079, 1955.

[80] N.J. Mehdiabadi, U. G. Mueller, S. G. Brady, A. G. Himler, and T. R. Schultz. Symbiont fidelity and the origin of species in fungus-growing ants. *Nature Communications*, 3(840), 2012.

[81] Gregor Mendel. Versuche über plflanzenhybriden. *Verhandlungen des naturforschenden Vereines in Brünn, Bd. IV für das Jahr 1865*, Abhandlungen:3–47, 1866.

[82] Zbigniew Michalewicz. *Genetic algorithms + data structures = evolution programs (2nd, extended ed.)*. Springer-Verlag New York, Inc., New York, NY, USA, 1994.

[83] Julian F. Miller and Pete Thomson. Cartesian genetic programming. In *Proceedings of the European Conference on Genetic Programming*, pages 121–132. Springer-Verlag, 2000.

[84] Melanie Mitchell. *An introduction to genetic algorithms*. MIT Press, Cambridge, MA, USA, 1996.

[85] Charles R. Moore. Regulation of deceptive practices by the federal trade commission. *Food Drug Cosmetic Law Journal*, 16:102–115, 1961.

[86] Michael Muhlbaier, Apostolos Topalis, and Robi Polikar. Ensemble confidence estimates posterior probability. In *Proceedings of the 6th international conference on Multiple Classifier Systems*, MCS'05, pages 326–335, Berlin, Heidelberg, 2005. Springer-Verlag.

[87] Peter Nordin and Wolfgang Banzhaf. A method for behavioural organization for autonomous robots based on evolutionary optimization of utility functions. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engin*, (217):249–258, June 2003.

[88] J. Paredis. Steps towards co-evolutionary classification neural networks. In *In Proc. of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*, pages 102–108, 1994.

[89] R. Polikar. Ensemble based systems in decision making. *Circuits and Systems Magazine, IEEE*, 6(3):21–45, quarter 2006.

[90] William Poundstone. *Prisoner's Dilemma*. Anchor Books, 1993.

[91] Markus Prior. The immensely inflated news audience : Assessing bias in self-reported news exposure. *Public Opinion Quarterly*, 73(1):130–143, 2009.

[92] William M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850, December 1971.

[93] D. Raup and J. Sepkoski. Mass extinctions in the marine fossil record. *Science*, (215):1501–1503, 1982.

[94] I. Rechenbreg. Cybernetic solution path of an experimental problem. Technical report, Ministry of Aviation, Royal Aircraft Establishment, 1965.

[95] I. Rechenbreg. *Evolutionsstrategie: Optimierung Technischer Systemenach Prinzipien der Biologischen Evolution*. Frommann-Holzboog, Stuttgrat, 1973.

[96] Likert Rensis. A technique for the measurement of attitudes. *Archives of Psychology*, (140):1–55, 1932.

[97] F.R. Roth, J.D. Hughes, P.E. Estep, and G.M. Church. Finding dna regulatory motifs within unaligned non-coding sequences clustered by whole-genome mrna quantitation. *Nature Biotechnology*, 16(10):939–945, 1998.

[98] Duncan Roy. Learning and the theory of games. *Journal of Technical Biology*, 204:409–414, 2000.

[99] H.-P. Schwefel. *Evolutionsstratgie und numbersche Optimierung.* PhD thesis, Technische Universität Berlin, 1975.

[100] H.-P. Schwefel. *Numerische Optimierung von Computer-Modellen mittels der Evolutionsstratgie.* Basel, Birkhäuser, 1977.

[101] Xiong Shengwu, Wang Weiwu, and Li Feng. A new genetic programming approach in symbolic regression. In *Proceedings of the 15th IEEE International Conference on Tools with Artifical Intelligence*, pages 161–165, 2003.

[102] Karl Sigmund and Martin A. Nowak. Evolutionary game theory. *Current Biology*, 9(14):503–505, 1999.

[103] Y. Song, D. Che, and K. Rasheed. Mdga: Motif discovery using a genetic algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 447–452, 2005.

[104] Frank J. Sulloway. Darwin and his finches : The evolution of a legend. *Journal of the History of Biology*, 15(1):1–53, Spring 1982.

[105] Kazuo Suzuki, Toshiaki Shijuuku, Toshihiko Fukamachi1, John Zaunders, Gilles Guillemin, David Cooper, and Anthony Kelleher. Prolonged transcriptional silencing and CpG methylation induced by siRNAs targeted to the hiv-1 promoter region. *Journal of RNAi and Gene Silencing*, 1(2):66–78, 2005.

[106] M. T. Tu, E. Wolff, and W. Lamersdorf. Genetic algorithms for automated negotiations: a fsm-based application approach. In *11th International Workshop on Database and Expert Systems Applications*, pages 1029–1033, 2000.

[107] Michael Twardos. Probability distribution of song length in a collection of itunes libraries, Nov 2011. theinformationdiet.blogspot.ca/2011/11/probability-distribution-of-song-length.html.

[108] August Weismann. The supposed transmission of mutilations. In *Essays upon Heredity*, volume I, chapter VIII. Oxford University, London, England, 1889.

[109] Darrell Whitley, Soraya Rana, and Robert B. Heckendorn. Island model genetic algorithms and linearly separable problems. In David Corne and Jonathan Shapiro, editors, *Evolutionary Computing*, volume 1305 of *Lecture Notes in Computer Science*, pages 109–125. Springer Berlin / Heidelberg, 1997.

[110] Darrell Whitley, Soraya Rana, and Robert B. Heckendorn. The island model genetic algorithm: On separability, population size and convergence. *Journal of Computing and Information Technology*, 7:33–47, 1998.

[111] Xiao-Feng Xie, Wen-Jun Zhang, and Zhi-Lian Yang. Hybrid particle swarm optimizer with mass extinction. In *IEEE 2002 International Conference on Communications, Circuits and Systems and West Sino Expositions*, volume 2, pages 1170 – 1173 vol.2, 2002.

[112] S. Zhong and J. Gosh. A unified framework for model based clustering. *Journal of Machine Learning Research*, 4:1001–1037, 2003.

Appendix A

Proof that Self-Driving FSMs can create non-regular languages where output symbols are multiple input symbols

Lemma A.0.1. *A language is regular if and only if there exists a deterministic finite state machine (FSM) which accepts the language.*

See [56] for proof.

Theorem A.0.2. *A self-driving finite state machine allowing for the output of more than one symbol can create non-regular languages.*

Proof. We will prove via contradiction using Lemma A.0.1. Take the self-driving machine defined by $Q = \{q_0\}$, $I = \{C, G\}$, $Z = \{C, GG\}$, $\delta = \{(q_0, C) = q_0, (q_0, G) = q_0\}$, and $\omega = \{(q_0, C) = GG, (q_0, G) = C\}$, with initial state q_0 and input string C. See Figure A.1 for the state machine diagram.

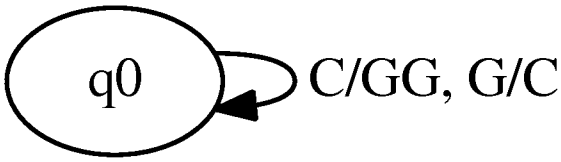


Figure A.1: A single state self-driving machine with multiple outputs which creates a non-regular language as an output.

This creates a string $G^2C^2G^{2^2}C^{2^2}G^{2^3}C^{2^3}\ldots G^{2^k}C^{2^k}\ldots$ where k is a positive integer. Assume there exists a deterministic finite state machine which can recognize this string. This machine requires the FSM to count a string of Gs and then Cs of length 2^k. As k tends to infinity, then it would have to count an arbitrarily large string, which would require more than any finite number of states. As a FSM has finite states there exists a contradiction based on the definition of a FSM. Hence, no such machine can exist. As no FSM can recognize this string, then the language created by this self-driving FSM must be non-regular as per the contrapositive of Lemma A.0.1. Therefore, there exists a non-regular language created by a self-driving FSM. □

It is interesting that this self-driving machine is of only one state, hence extremely simple in terms of the number of unique stings it can produce, yet it would require an infinite number of states to recognize. However, if a non-deterministic machine was allowed to take multiple input symbols, as inputs, then self-driving machines could be recognizable by a finite number of states.