Paper appears in: Genetic Programming and Evolvable Machines (2007) 8(2):187-220
Special Issue on "Developmental Systems"

Introducing Probabilistic Adaptive Mapping Developmental Genetic Programming with Redundant Mappings*

GARNETT WILSON gwilson@cs.dal.ca
MALCOLM HEYWOOD mheywood@cs.dal.ca
Faculty of Computer Science, Dalhousie University, Halifax, NS, Canada B3H 1W5

Abstract. Developmental Genetic Programming (DGP) algorithms have explicitly required the search space for a problem to be divided into genotypes and corresponding phenotypes. The two search spaces are often connected with a genotype-phenotype mapping (GPM) intended to model the biological genetic code, where current implementations of this concept involve evolution of the mappings along with evolution of the genotype solutions. This work presents the Probabilistic Adaptive Mapping DGP (PAM DGP), a new developmental implementation that involves research contributions in the areas of GPMs and coevolution. The algorithm component of PAM DGP is demonstrated to overcome coevolutionary performance problems that are identified and empirically benchmarked against the latest competing algorithm that adapts similar GPMs. An adaptive redundant mapping encoding is then incorporated into PAM DGP for further performance enhancement. PAM DGP with two mapping types are compared to the competing Adaptive Mapping algorithm and Traditional GP in two medical classification domains, where PAM DGP with redundant encodings is found to provide superior fitness performance over the other algorithms through it's ability to explicitly decrease the size of the function set during evolution.

Keywords: *developmental genetic programming, cooperative coevolution, genotype-phenotype mapping, neutrality, redundant representation*

1. Introduction

Broadly speaking, the term "developmental genetic programming" (DGP) includes methodologies that explicitly set out to separate the genotype space from the phenotype (or solution) space through a connection (or mapping) between the two spaces [12]. In the literature, the term has been used to encompass systems such as the one presented in this work that inserts a genotype-phenotype mapping (GPM) as a genetic code to establish a relationship between the two spaces. A genetic code in the context of this paper is as defined by Keller and Banzhaf in [12], that is, it is the encoding of a symbol

* This paper is a revised and substantially extended version of: G. Wilson and M. Heywood. "Probabilistic Adaptive Mapping Developmental Genetic Programming (PAM DGP): A New Developmental Approach" in Proceedings of the 9th Parallel Problem Solving from Nature (PPSN IX), Reykjavik, Iceland, Runarsson et al. (eds.), Springer: Berlin, 2006, pp. 751-760.

by one or more codons where a codon is a non-zero contiguous bit sequence from a binary genotype. This work conducts an investigation of the algorithmic framework used to coevolve populations of genetic code mappings and genotypes, as well as identifying the most appropriate GPM model for use in such a coevolutionary framework. In the process of meeting these requirements, a new algorithm for the coevolution of efficient mappings is presented. A preliminary version of PAM DGP was first presented in [30], with considerable improvements introduced in [31] (wherein the performance of PAM DGP was also demonstrated on two regression problems). This work serves as the comprehensive introduction of PAM DGP to the literature, significantly expanding on [31] by introducing a superior (and more developmental) adaptive redundant mapping structure to the algorithm, comparing PAM DGP to other GP algorithms using medical classification benchmarks, and providing an extensive discussion of the developmental nature of PAM DGP and how it addresses previously unrecognized coevolutionary pathologies in previous work of Margetts and Jones [16-18].

The system described in this work has its roots in the DGP algorithm of Keller and Banzhaf [1, 11-13], but separates mapping from genotype (they are united in the implementation of Keller and Banzhaf) to form two populations that co-evolve as in the subsequent algorithm of Margetts and Jones [16-18]. The system implemented here uses a mapping that can be seen as a table relating genotype subsections (binary sequences) to symbol members of a function set, and in that respect it is similar to the mapping structure seen in the implementations produced by both groups of researchers. Their collective work thus serves as the starting context for this work. In terms of describing developmental context (the biological analogue) of the algorithm described in this paper, the system can be considered to model both evolution of a "genetic code" that maps codons to amino acids in biological organisms and evolution of the genotype of the organisms themselves.

The evolution of a genetic code is of great benefit when there is little or no information about the problem space, or when the problem space takes the form of a dynamic environment and adaptiveness is required for survival of individuals. An evolved genetic code, or mapping, is used to adaptively bias towards symbols that are important for the solution during the execution of the algorithm or dictate which symbols are permitted to be used in the construction of a solution. The co-evolution of the mappings and genotypes thus dynamically reduces problem search space and biases solutions towards particular regions of the search space. Furthermore, as noted by Keller and Banzhaf [11], the separation of the two spaces avoids the hindering effect of operators in traditional GP approaches that are restricted so that only legal phenotypes are generated. This built-in restriction of the operators limits the search to the feasible areas of the search space. However, the infeasible regions may contain genetic diversity needed to quickly generate very high quality individuals within fewer generations. The aim of this work is to provide a system that efficiently generates mappings and their corresponding genotypes to realize the benefits of both a more efficient search and a better tailored solution that incorporates and emphasizes the correct symbols in a problem's function set.

Section 2 of this work describes relevant background literature in coevolution of genetic code-type mappings and solutions, the particular developmental nature of PAM DGP, and pathologies of cooperative coevolution to be addressed by PAM DGP. Section 3 details the drawbacks of the earlier Adaptive Mapping DGP algorithm of Margetts and Jones [16-18]. Section 4 describes the PAM DGP algorithm, how it improves on the

previous work, and compares the performance of the two algorithms on the Maximum Output problem previously used to demonstrate properties of DGP algorithms [18]. Having established performance comparisons of PAM DGP and Margetts and Jones's algorithm using equivalent genotypes and mappings, a more developmental adaptive redundant mapping is incorporated in PAM DGP in Section 5 and results for a simple regression problem are provided. Section 6 compares PAM DGP (using both mapping types), Margetts and Jones's algorithm, and Traditional GP on two medical classification benchmarks. The analysis in Section 6 demonstrates the effectiveness of the new mapping with respect to fitness performance, and shows it is achieved through tailoring of the function set, leaving a subset of the function set most appropriate to the problem domain. Section 7 provides a summary of the findings and associated conclusions.

2. Background and Related Work

2.1 The Developmental Nature of PAM DGP

In nature, the decoding of genes occurs in two stages: transcription and translation. During transcription, sections of the double-helix DNA are unwound to expose the genes. The genes are then used as templates to create messenger copies of the genes (mRNAs) in the DNA's analogue chemical language RNA. During the translation phase, the codons (triplets of nucleotides) on the mRNA molecules are recognized by one end (called the "anticodon") of a transfer RNA (tRNA) molecule. The other end of the tRNA molecule corresponds to a binding for a specific amino acid. The correct tRNA molecules continue to bind to successive codons, bringing together the amino acids required to form the protein product for which the gene was originally coded. Multiple codons of the mRNA (and hence, multiple genes of the DNA) can specify the same amino acid. However, each type of tRNA molecule can be attached to only one type of amino acid. Thus, multiple types of tRNA molecules exist with identical anticodons that can carry the same amino acid. This amounts to a redundant code: each amino acid is specified by more than one codon. The biological development equivalent of a genotype-phenotype mapping individual in our DGP is technically the tRNA molecules—they are responsible for mapping codon (via anticodon identifier on the tRNA) to amino acid. Another, simpler way of putting this is that the genetic code (biological codon to amino acid mapping) is being modeled. Since the natural genetic code is redundant, a developmentally accurate DGP analogue ought to be redundant. Regarding research groups to be discussed in this section, Banzhaf and Keller [1, 11-13] and O'Neill and Ryan [20, 21] incorporate redundancy into their mappings, whereas the encoding of Margetts and Jones [16-18] is one-to-one (and hence neither redundant nor developmentally accurate).

The purpose of any coevolutionary developmental system is to evolve both a good genetic code and an optimized solution. But the genetic code is often assumed by the layperson (and modeled in most traditional evolutionary algorithm systems) to be fixed. As it turns out, modern biology indicates that the genetic code is *adaptive*: it is capable of evolving [9, 27]. Furthermore, the genetic code itself has evolved as a product of

adaptive evolution [9], and its evolution is capable of explaining several problems in evolutionary biology [27]. In particular, it has been postulated that the genetic code adapts to minimize negative effects of mutation and gene translation errors while also maximizing the rate of natural selection as a search algorithm [9]. Furthermore, recent work in the field of molecular evolution argues that the particular mapping of codons to amino acids dictated by the current genetic code is a product of the coevolution of the genome and the genetic code [26]. Sella and Ardell [26] have since found that coevolution modeled with errors in replication and translation actually generate codon-amino acid associations rather than simply preserving ancestral versions of the associations as originally supposed by Crick [5]. In addition, the models of Sella and Ardell consistently generated the main organizational features of the standard genetic code. The adaptive nature of the genetic code in biology justifies the application of selection to a population of genetic codes in a developmental system, and the fact that the mapping of the genetic code is suspected to co-evolve with the genome justifies the use of coevolution in the developmental model.

It is still an open question how best to implement this coevolution. The mappings (tRNA molecules) and genotypes (genes) are separate molecular entities within a single moleclule, so it may seem to make sense to evolve them as being paired within an individual (as in [1, 11-13, 20, 21]). However, if we consider this a bit further, such an approach will not accurately model biological entities as a population. At any given time in evolutionary history, each separate individual will possess unique DNA but undergo basically the same transcription and translation processes (with very rare exceptions that slowly forward the evolution of the genetic code). That is, each individual does not possess a unique genetic coding scheme that occurs within their own cells. The aim of the developmental model is to determine how the genetic code has evolved to optimize for the *collective* population of genotypes. In nature, a highly evolved and nearly universal standard genetic code developed very early in the history of life whereas our human genotype is a much newer development. As noted in by Freeland [9], natural selection processes approach optima asymptotically. As the genetic code is now highly adapted, the changes occurring to it are negligible for practical purposes. However, problems posed to an artificial DGP system cannot assume to have been handed a near optimal genetic code as context in which to work. The purpose of the coevolution of the genetic code in DGP systems is to play "catch-up"—evolve an optimized (or as optimized as possible) genetic code in the context of the *population* of genotypes rather than an individual's genotype. From a developmental point of view, it would thus seem more appropriate to evolve candidate genetic codes (mappings) against a collective population of genotypes in a separated coevolutionary population. Researchers who have opted for this approach include [16-18] and the authors.

2.2 Cooperative Coevolution and its Pathologies

Coevolution has been broadly defined by E. de Jong and Pollack as an implementation where a given individual is evaluated using interactions with other individuals that are evolving at the same time [6], and it is usually divided into two types: cooperative and competitive. In the cooperative model, formulated by Potter and K. De Jong [24, 25], two or more populations are evolved where individuals only mate within their own

population[1]. The motivation behind cooperative coevolution is to decompose a complex problem into components, evolve the components in separate populations, and then assemble the components into a total solution. Since the populations are separate, individuals can (but need not) have very different structures so long as they can collaborate. To evaluate a member of one population, collaborations are formed with members of the other populations. The best individual from each of the other populations was chosen as the collaborator in the initial work on cooperative coevolution, and a phenotype was formed by combining the chosen individuals from each population. In the competitive model [6], individuals (called *learners*) are evaluated by being tested against other individuals (called *evaluators*). PAM DGP and Margetts and Jones [16-18] use two population cooperative coevolution where two members of each population evolve and must be combined (collaborate) to produce a phenotype for fitness evaluation. The remainder of this section focuses on pathologies of cooperative coevolution that pertain to this work.

A problem relevant to this work is referred to as the "Red Queen Effect" [4], named for the Red Queen character in Lewis Carroll's *Through the Looking Glass* who ran perpetually without moving because the landscape kept up with her. The Red Queen effect occurs in coevolution where traits in one population evolve against traits in the other population that are also evolving, thus little or no progress is made or an adaptation in one population can even undermine the progress of the other. What is hoped for in a coevolutionary implementation is an *arms race*, where progress is made by "mutual and reciprocal adaptations between collaborating groups of individuals" [28]. That is, what is needed is for each population to alternately build on the adaptations achieved by the other to mutually progress toward a solution. Theoretical analysis in the last decade has typically shown that cooperative coevolution evolutionary algorithm (CCEA) implementations are not well-suited for static optimization problems [28]. A number of researchers have currently been working on modifying CCEAs to enable them to perform well on static optimization problems. Bucci and Pollack [3] have attempted to accomplish this using a combination of Pareto selection and memory mechanisms, while Panait, Luke, and Wiegand [22, 23] improve CCEA by biasing evaluation of phenotype toward its actual [22, 23] or expected optimal assessment [22]. This paper introduces a novel way of overcoming the Red Queen for the purpose of evolving efficient mappings in PAM DGP to solve static regression and classification optimization tasks.

2.3 Evolution of Genetic Code Mappings in Developmental Systems

There are three other systems that evolve genetic code-type mappings and solutions in a developmental GP approach. Keller and Banzhaf [1, 11-13, 20, 21], Margetts and Jones [16-18], and O'Neill and Ryan [20, 21] all consider their approaches developmental systems that evolve a mapping that is an analogue of the biological genetic code. Margetts and Jones [16-18] and O'Neill and Ryan [20, 21] both consider their systems to implement coevolution. Keller and Banzhaf [12, 13] do not explicitly state that they implement coevolution, although their method of evolving both mapping and genotype solution as an individual is identical to O'Neill and Ryan and corresponds to the broad definition of coevolution provided in the previous section. Keller and Banzhaf are credited with using coevolution by O'Neill and Ryan in [21], where they state "studies

[Keller and Banzhaf] provide strong evidence demonstrating the effective co-evolution of genetic code and solution." In any case, all research teams implement developmental systems like PAM DGP where genetic codes are evolved along with genotype solutions.

O'Neill and Ryan use a grammar as the structure of their mapping, and thus it differs significantly from the genetic code-based mappings. Their implementation is shown to be viable on symbolic regression problems [21], and it is extended to generate modular GA individuals in [21]. The mapping and genotype in their work are paired for the purpose of reproduction, and thus are not given separate tournaments as unique populations. Banzhaf and Keller [1, 11-13] use arbitrarily assigned mappings of binary sequences to symbols of the function and terminal set. These individual mappings are linked with individual genotypes and are selected and reproduced as pairs throughout a tournament. The mapping has a unique mutation operator compared to the genotype, however. Since each individual's code is an arbitrary codon-symbol mapping, more than one codon can map onto the same symbol. Thus, the code is redundant and emphasizes symbols by assigning them additional codons. The results of Kellar and Banzhaf's seminal work showed that evolution of paired genetic codes and solutions works in principle on easy [12] and hard [13] synthetic problems.

Instead of a DGP algorithm composed of a population of genotypes and population of attached genetic codes as was implemented in [12], Margetts and Jones's algorithm separates the mapping and genotype genomes into two populations and relates the two through a cooperative coevolutionary model [16]. Margetts cites a number of issues with the approach of encoding a mapping with a genotype so that each individual has their own mapping. One downside of this approach as opposed to placing genotypes and mappings in separate populations is that a single population is composed of individuals containing larger amounts of genetic (genotype and mapping) information. Also, each individual must effectively solve two problems at once: it must simultaneously determine a good mapping and it must find a good solution to which the mapping is to be applied. Supposing that an individual is composed of a good solution and a poor mapping (or vice versa), the individual will only possess a single fitness value.

The algorithm of Margetts and Jones uses a population of mapping individuals and a population of genotype individuals that co-evolve in alternating steady state tournaments. In order to evaluate a particular genotype, a member of the mapping population is selected to produce the phenotype for fitness evaluation. The mapping individual chosen to be applied to a genotype is the mapping individual with the current highest fitness in the mapping population. To evaluate a mapping individual, the current best member of the genotype population is used. There is thus only one mechanism for fitness evaluation for genotype and mapping in this implementation: fitness of both mappings and genotypes is evaluated given a phenotype produced from a mapping individual applied to a genotype individual.

In addition to modeling the placement of the genetic code as a separate entity instead of inclusion in the individual, the structure of the genetic code itself is also addressed by Margetts and Jones in [17, 18]. In [18] they introduce their alternative representation, the "adaptive mapping," where they use binary strings of dynamically determined length each corresponding to a symbol. The algorithm chooses its own assignment of binary sequences to symbols using Huffman encoding. Huffman encoding is traditionally a means of allowing frequently-used symbols in a message to be transmitted using a proportionally lower number of bits so that message lengths are reduced for the purpose

of compressed transmission. The Huffman algorithm also ensures that for a given bit length, the most frequent bit sequences correspond to the most frequent symbols.

Given a function set with *s* symbols, each individual in the population of mappings consists of *s* 10-bit binary string sections representing a frequency. Each section of 10 bits is converted to a real valued frequency in the interval [0, 1] using the normalized *countingOnes* function that simply sums all the ones in a given string. In [17], the use of the *countingOnes* function is justified over more intuitive choices (such as binary to decimal conversion) for its ability to produce small changes in phenotype to correspond to small changes in genotype in later phases of search. The symbols, associated frequencies, and genotype are provided as arguments for utilizing Huffman's algorithm, and it returns a one-to-one mapping of varying-length bit strings to symbols. The Huffman mapping encoding process is depicted in Figure 1. In [18], the authors demonstrate that their algorithm and associated adaptive Huffman-encoded mapping outperform a fixed mapping on a DGP variant of the Maximum Output problem. Hereafter, the adaptive mapping and associated algorithm will be denoted the Standard Adaptive Mapping (or simply Adaptive Mapping) DGP.

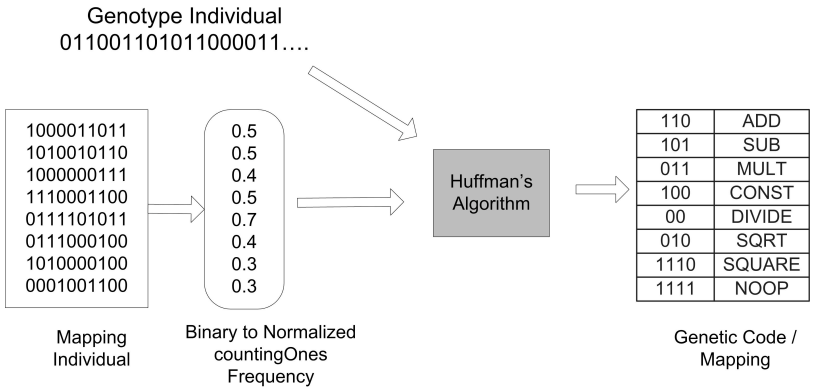

Figure 1. The Huffman-based Adaptive Mapping encoding process.

3 Drawbacks for the Adaptive Mapping DGP

3.1 Introducing the Maximum Output (MAX) Problem

The problem selected to demonstrate the Adaptive Mapping DGP in the literature is a version of the Maximum Output (MAX) problem introduced in [10, 15], as described by Margetts and Jones [16, 18]. This version of the MAX problem is to create a program that returns the largest value possible using the function set within the given program size limit (rather than a tree depth limit as in [10, 15]). An ideal solution to the problem with

the defined function set is a program that repeatedly duplicates a large number and multiplies it by itself, effectively squaring the number as many times as possible.

In the Adaptive Mapping DGP, the MAX problem is posed to a linear (bit string) stack-based version of GP. Each individual is a stack-based machine composed of a general-purpose stack and an output register [16, 18]. A program that changes the state of the machine is a list of instructions from the function set in Table 1. The function set has default codons that are used for a non-mapping benchmark implementation to which the Adaptive Mapping DGP algorithm performance is compared. (Fixed mappings with the default encodings amount to not having a population of mappings, and thus correspond to a traditional GP.) There is no terminal symbol set for this problem specification. Each instruction in the program is processed sequentially, with each having an associated function. For each function call, the required number of arguments is taken from the stack, they are presented to the function, and the return value (if any) is pushed back onto the stack. If there are insufficient arguments on the stack, the function does nothing. We now use the Maximum Output problem that introduced the Standard Adaptive Mapping algorithm to the literature to illustrate its three major drawbacks.

Table 1. Stack-based GP Maximum Output Problem function set.

Symbol	Function Explanation
Plus	Pop 2 items, add, push onto stack.
Times	Pop 2 items, multiply, push onto stack.
Const	Interpret next 10 bits as number, push onto stack.
Dup	Duplicate item on top of stack, push onto stack.
Pop	Remove item at top of stack.
S2R	Copy item at top of stack into output register; does not affect stack contents.
R2S	Push item in output register onto stack.

3.2 Drawback 1: Repeated Loss of Context and Fitness Spiking

While the Adaptive Mapping DGP algorithm was shown to outperform the fixed mapping algorithm corresponding to a traditional non-mapping GP in [16, 18], we show in this section that there are major drawbacks associated with the scheme. The implementation uses the coevolution of a population of genotypes and a population of mappings. An individual from one population cannot be evaluated without a member of the other population. To evaluate the fitness of an individual in either population, the individual is evaluated with respect to the best individual in the other population. This is done to avoid the impractical computational expense of taking the individual we want to know the fitness of and evaluating them against every member of the other population. The authors also state in [18] that evaluating an individual in a population with the best individual in the other creates an algorithm that is "pleasantly symmetric."

The first drawback is that this methodology for evaluating an individual in either population leads the algorithm to regularly disrupt the progress it makes toward a solution. The reason for this is that, for a given individual in a population, the context of

the best individual in the opposite population changes throughout the algorithm. If a brand new best genotype appears, evolution of the mapping population must start its search anew to match it (with the exception of the mapping individual that caused it to happen). That is, every mapping except the one that produces the best fitness with the new best genotype will now likely produce a poor fitness with the new best genotype. The proceeding evolution of the mappings can eventually create a new best mapping in light of the new best genotype, but the previously best genotypes will then fail in the context of this new mapping. The two populations end up struggling to match their search to each other's best individual while at the same time causing the best individual in each other's population to change. The mutual contexts that created a high fitness can quickly become lost with the uncoupling of the relationship of best genotype to best mapping. The same phenomenon can occur with respect to either population. This is a textbook case of the Red Queen Effect: each population struggles against and undermines the progress of the other instead of building on one another's progress.

While elitism to protect the best individuals would seem to be an easy fix for the problem of loss of context, it would hinder the exploration phase of the search considerably given the frequency with which context changes were seen to occur. Despite the problems changing contexts cause, the algorithm relies so heavily on the changing of the context of the best individual for exploration of the search space that attempting to correct the problem by protecting the best individuals in either population would adversely impact the algorithm. The fitness cycles are a necessary effect of the methodology of the Adaptive Mapping algorithm, and are seen in an evolutionary run, as illustrated in Figure 2 for a population size of 8.

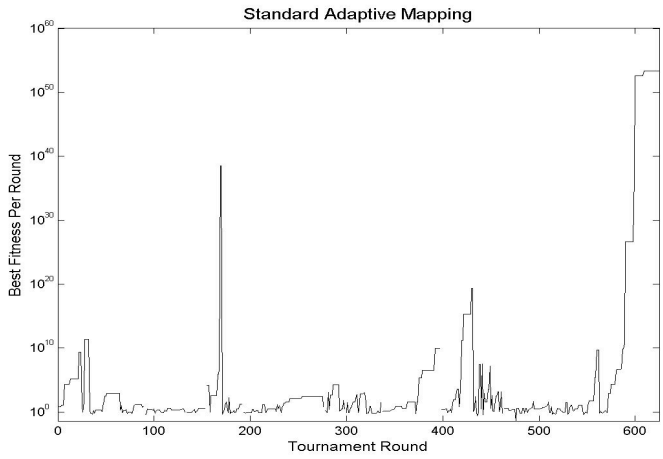

Figure 2. Best fitness per round for the Standard Adaptive Mapping DGP for the MAX Problem. This graph represents a typical run for 200 bit individuals and a population of 8. Fitness scale is logarithmic. Breaks in the graph indicate fitness points ≤ 1 on the log scale.

Previous results reported for the Adaptive Mapping DGP algorithm do not plot the fitness of individuals participating in each round of the MAX tournament in [16, 18]; only

final numerical results are shown for a low number of tournaments. They do provide data for *maximum* fitness at each round [16]. However, such a reporting scheme masks the wide variation in fitness resulting from the loss of context between the two populations. As such, they report steady monotonic improvement in fitness, wherein the fitness spiking of Figure 2 cannot be recognized.

3.3 Drawback 2: Lack of Exploration of the Search Space

Even if there were not an issue involving constant changes in context, the Adaptive Mapping algorithm has a second major drawback. This was the original motivation for creating a new algorithm to improve on the Adaptive Mapping. Since the Adaptive Mapping always evaluates individuals chosen in genotype tournaments against only one mapping, and vice versa, the individuals in either population do not get an adequate chance to evolve in the contexts of other individuals in the opposite population during an arbitrarily chosen stage of evolution. While all members of a mapping population or genotype population may serve their term in establishing a context for the other population, the members of the other population do not have an opportunity to achieve higher fitnesses by combining with the non-best members. Such combinations could yield higher fitness schemas that go unnoticed in the search conducted by the Standard Adaptive Mapping algorithm, i.e., a greedy mechanism for relating the two populations is assumed to be the best design choice by the Adaptive Mapping algorithm.

3.4 Drawback 3: Lack of Fitness-Based Performance

Experiments using the Adaptive Mapping DGP were reported in [16, 18] to outperform a standard GP using a default encoding for each function symbol on the MAX problem for most program size limits (bit lengths of individuals). Given the performance obstacles so far identified in the Adaptive Mapping algorithm, it is of interest to determine how it outperformed GP with default encodings (traditional GP with a fixed mapping). The results in [16, 18] were reported for experiments up to 5000 function evaluations, with just 10 experiments run for each of five program size limits (50, 100, 150, 200, and 250 bits). Evaluating the Adaptive Mapping using a limit of 5000 function evaluations results in a very early stopping criterion: Assuming an average of 3 genotype bits per function call, this is equivalent to 15 rounds (for 250 bit individuals) to 75 rounds (for 50 bit individuals) under a steady state tournament with 4 individuals selected per round. For a more though analysis, runs were conducted over more rounds for the Adaptive Mapping algorithm and a fixed, default encoding GP with a population of fifty 250 bit individuals. The results are given in Figure 3, plotting up to tournament round 20 (where the cited results of [16, 18] go up to 15 rounds for 250 bit individuals). Figure 3 indicates that the Adaptive Mapping DGP does indeed outperform the GP with default mappings. However, if the tournament continues just to round 100 (Figure 4), it is quite evident that the default encoding GP overtakes the Adaptive Mapping DGP. Loss of context and fitness spiking associated with the Red Queen effect does indeed hinder the performance of the Adaptive Mapping algorithm compared to Traditional GP.

In summary, there is an additional implicit overhead in systems employing evolved mappings in that these systems need to discover both the appropriate gene sequence and the best function set, whereas traditional GP need only concentrate on locating the relevant gene sequence. It is thus more difficult problems, often involving function sets including extraneous symbols, which are best suited to evolved mapping algorithms. Under such conditions, DGP might be able to better the traditional GP approach by discovering relevant subsets of symbols and concentrating on forming solutions from this subset of the function set, i.e., the size of the DGP search space decreases. The MAX problem will be used hereafter only as a basis to demonstrate how the PAM DGP algorithm overcomes the drawbacks of the Standard Adaptive Mapping algorithm. Section 6 provides an example of when DGP betters the performance of traditional GP under two benchmark classification problems.

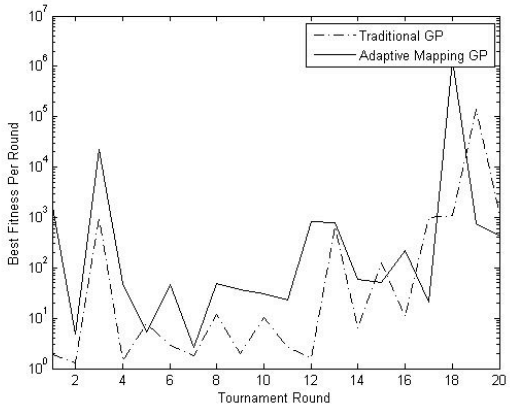

Figure 3. Best fitness per round for Traditional GP and Adaptive Mapping DGP for the MAX problem, population of 50 individuals of size 250 bits, up to tournament round 20.

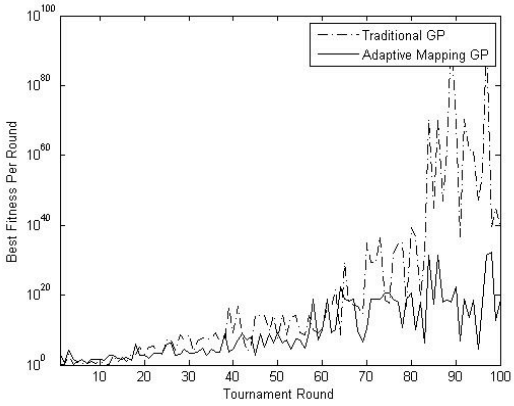

Figure 4. Best fitness per round for Traditional GP and Adaptive Mapping DGP for the MAX problem, population of 50 individuals of size 250 bits, up to tournament round 100.

4 Introducing Probabilistic Adaptive Mapping DGP (PAM DGP)

4.1 The PAM DGP Algorithm

With the above drawbacks of the Standard Adaptive Mapping DGP algorithm in mind, an alternative architecture is proposed in which genotype and mapping population are coevolved through a probabilistic model. Specifically, each individual in the genotype population is allotted a corresponding point on the *x* axis of a probability table, and each individual in the adaptive mapping population is allotted a corresponding point on the *y* axis. The table is initialized so that each column, one corresponding to each genotype point on the *x* axis, sums to unity. All cells in each column are thus initialized to the same probability. For direct comparison with the Standard Adaptive Mapping algorithm, we retain the Huffman based encoding in the mapping population [16-18]; however, in Section 5, the case of a redundant mapping is introduced.

As per the Standard Adaptive Mapping algorithm, a steady state tournament is retained. Individuals are constructed using roulette wheel selection to choose a position in the table, with four separate genotype individuals selected in each tournament round, Figure 5. Each entry in the table indexes a genotype-mapping pair. The genotypes are ranked by fitness after being interpreted in the context of the corresponding mapping. With a steady state tournament size of four, the best two genotype individuals (the parents) are left untouched while the remaining two (genotype) individuals become children. The children become copies of the parents and are subjected to mutation and crossover with corresponding likelihood probability thresholds. (Crossover and mutation operators and associated rates will follow, being discussed with individual problem parameterizations.) The mappings associated with the ranked genotypes receive the ranking of their partnering genotypes during competition in the aforementioned tournament round (Figure 5). The mappings associated with the top two genotype parents are ranked as the top two tournament mappings and kept as the parent mappings, while the mappings associated with the two (genotype) children become copies of the corresponding parent mappings and are subject to mutation and crossover. Since there is only a guarantee during selection that separate genotypes will be selected, naturally, the above process may result in mapping individuals appearing more than once in the ranked list for a tournament round. A mapping appearing twice or more in the ranking list may (and will likely) have a different fitness each time, in virtue of being associated with a different genotype at each placement in the list. If a single mapping is associated with both losing genotypes, the crossover operation is, of course, not performed with respect to mappings for that ranking.

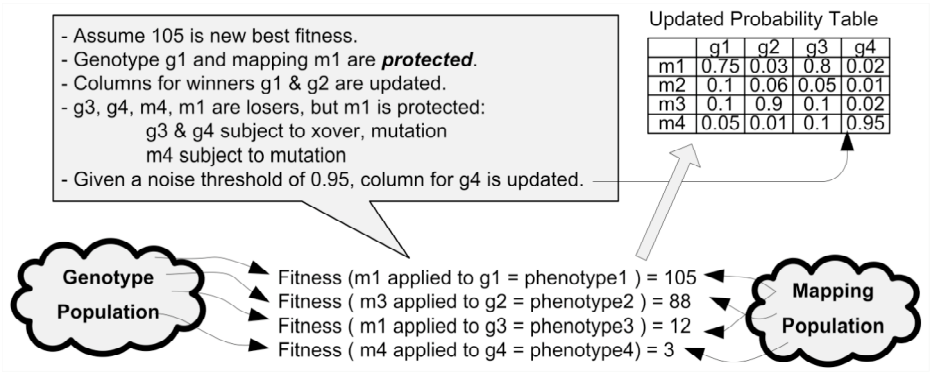

Figure 5. PAM DGP algorithm and data structures.

The algorithm also features elitism in that the genotype individual and the mapping that produces the current highest fitness cannot be replaced. Note that unless both members of that pairing are explicitly protected, either member of the pairing can be selected as a child by being coupled with an alternate member of the other population during roulette wheel selection on the probability table. The position on the table associated with the two winning genotype-mapping combinations is updated according to (1), while the losing combinations in the same column are updated according to (2)

$$\text{Winning Combination: } P(g,m)_{new} = P(g,m)_{old} + \alpha(1 - P(g,m)_{old}) \quad (1)$$

$$\text{Other Combinations: } P(g,m)_{new} = P(g,m)_{old} - \alpha(P(g,m)_{old}) \quad (2)$$

where g is the genotype index, m is the mapping index, α is the learning rate (or how much emphasis is placed on current values as opposed to previous search), and $P(g,m)$ is the probability in location $[g, m]$ of the table. Recall that four separate genotypes, not necessarily four separate mappings, are chosen per tournament round. Therefore, the genotype-associated columns are updated rather than the rows. Equations 1 and 2 are adapted from [8], where their original use was for an ant-based network routing domain. Equations 1 and 2 ensure that the column to which they are applied in the probability table sums to unity following the update. Equation 1 increases the future probability of choosing the winning combination by a value proportional to the learning rate α and to the previous value allotted to the combination. Given a particular α, smaller probabilities will be increased proportionally more than larger probability values to provide expedient biasing toward selection of newly discovered pairings of greater fitness. Equation 2 causes the other values in the column to be allotted a negative reinforcement implicitly by normalization. Roulette wheel selection for the probability table operates by adding the probabilities across the rows associated with mapping individuals, although the manner in which the table is traversed does not matter because each cell is equally likely to be chosen by uniform selection.

After a period of search depending on the learning rate, it was discovered that the probability table can prematurely converge on particular genotype-mapping pairings while other locations in the table have no (or practically no) probability associated with

them. In order to allow the algorithm to continue to explore all genotype-mapping combinations and the underlying binary sequences they make available to the search space, an additive noise source is introduced (Figure 5). This is accomplished by examining each (genotype-associated) column when it is updated to see if any location in that column has exceeded a user-defined threshold γ. If it has, each member of the column has a uniform probability $P(1 - \gamma)$ of having a standard Guassian probability adjustment in the interval [0, 1] added to its current value. (In rarer cases where the Guassian value is outside the interval [0, 1], it is simply re-chosen until it falls in the interval.) The values in the column are then re-normalized so that they sum to unity. For completeness, a summary of the algorithm pseudocode is given in Table 2.

Table 2. Pseudocode for the Probabilistic Adaptive Mapping Developmental Genetic Programming (PAM DGP) Algorithm.

```
Initialize size P mapping & genotype populations
Initialize each value in P x P probTable = 1/P
while (tournamentNotDone && solutionNotFound)
   Use probTable mapping rows for roulette selection
   Rank selectedGenotype & associatedMapping pairings
   Verify or set current bestGenotype & bestMapping
   Apply update to probTable according to Eq. (1) & (2)
   if (element of winning genotype column > γ)
      for (each column element)
         Add Gaussian noise value to element
      Normalize column contents to unity
   for 2 loserGenotypes & 2 loserMappings
      if (respective mutation thresholds are met)
         if (loserIndividual ≠ bestIndividual)
            mutate(loserIndividual = copy of parent)
      if (respective xover threshold is met)
         if (both loserIndividuals ≠ bestIndividual)
            xover(loserIndividuals = copy of parents)
```

4.2 How PAM DGP Addresses the Drawbacks of the Standard Adaptive Mapping

PAM DGP addresses the problem of loss of context and fitness spiking (the Red Queen effect) using two components of the algorithm: elitism and the probability table. Firstly, fitness spiking is stopped by protecting the best genotype mapping *combination* (elitism). Then PAM DGP introduces a small degree of elitism that protects the genotype and mapping in the current best combination from being overwritten, mutated, or crossed over until it is replaced with a new, better genotype/mapping combination. This is accomplished in the algorithm by checking if a losing individual happens to be the best genotype or mapping, and if it is, it does not have its genotype replaced and is not mutated or subject to crossover. The result of this protection of the current best combination is a much more robust algorithm where the evolutionary progress is always retained. This behavior is shown in Figure 6 and can be contrasted with the behavior for

the Standard Adaptive Mapping algorithm on the same experimental trial shown in Figure 2. When comparing performance of the algorithms, note that each round of PAM DGP evaluates 4 genotype-mapping pairings as does each round of the Standard Adaptive Mapping implementation. Actually, all algorithms discussed in this paper use 4 evaluations per round. An any point in this paper, readers interested in performance based on evaluations need only multiply the tournament round metric by a factor of 4. The difference between the algorithms is simply that each round of the Standard Adaptive Mapping is either a genotype or mapping tournament round (as described in Section 2.3), where there is no such distinction in PAM DGP.

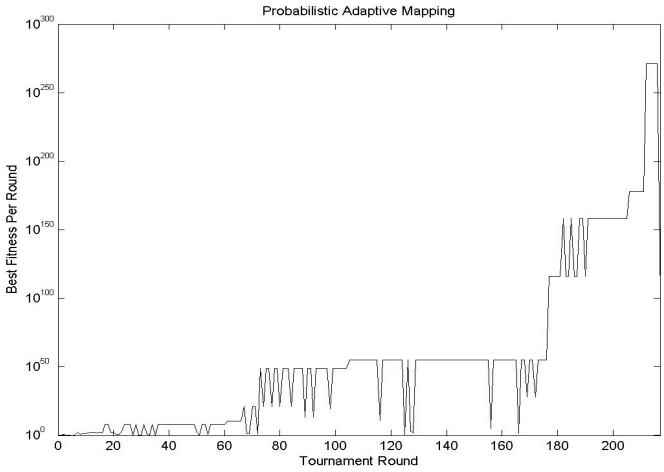

Figure 6. Best fitness per round for the probabilistic adaptive mapping (PAM) DGP for the MAX Problem. This graph represents a typical run for 200 bit individuals and a population of 8. Fitness scale is logarithmic. Breaks in the graph indicate fitness points ≤ 1 on the log scale.

Loss of context leading to the Red Queen Effect is due to reliance on a single individual (that may change) in one population to set the context for the entirety of the other population. The Red Queen loss of context is minimized by the use of the probability table because the algorithm no longer relies on a single individual from the one population as the fitness context for all the individuals in the second population. This also means that the Standard Adaptive Mapping algorithm suffers from a lack of adequate exploration of the search space by not providing a mechanism for evaluating alternative combinations of genotypes and mappings at multiple stages of evolution. The PAM DGP algorithm features a probability table that guides solution search while allowing the combination of any genotype individual with any mapping during a tournament round, with the selection of the combination made in a fitness-proportionate way. A quantification of the search space considered during a tournament round for each algorithm is provided as a proof in Table 3.

Table 3. Quantification of the combinations considered by the Standard Adaptive Mapping Algorithm and PAM DGP.

Let n be the number of individuals in either the genotype population or the mapping population. Let c be a number < n corresponding to a particular individual in either population.

G_i denotes an individual in the genotype population, where $i < n$. M_i denotes an individual in the mapping population, where $i < n$.

Standard Adaptive Mapping DGP:

For any given mapping round in the Standard Adaptive Mapping Algorithm, the genotype in all combinations produced is fixed (the best genotype).
A combination thus takes the form $(G_c, M_{i\ldots n})$ where G_c has 1 possibility and $M_{i\ldots n}$ has n possibilities.
There are thus 1 x n = n possible combinations for $(G_c, M_{i\ldots n})$. Given 4 combinations selected per round, with no duplication of mappings and one genotype, this gives four combinations:

1. $(G_c, M_{i\ldots n})$ _ 1 x n = n possibilities
2. $(G_c, M_{i\ldots n-1})$ _ 1 x $(n-1)$ = $n-1$ possibilities
3. $(G_c, M_{i\ldots n-2})$ _ 1 x $(n-2)$ = $n-2$ possibilities
4. $(G_c, M_{i\ldots n-3})$ _ 1 x $(n-3)$ = $n-3$ possibilities

Result 1. During each mapping round, then, $n + (n-1) + (n-2) + (n-3) = 4n-6$ possibilities are considered. The same argument applies to any genotype round, only substituting G for M.

Probabilistic Adaptive Mapping DGP:
For any given round in the Probabilistic Adaptive Mapping Algorithm, separate genotype individuals are picked and combined with any mapping.
A combination thus takes the form $(G_{i\ldots n}, M_{i\ldots n})$ where $G_{i\ldots n}$ has n possibilities and $M_{i\ldots n}$ has n possibilities.
There are thus n x n = n^2 possible combinations for $(G_c, M_{i\ldots n})$. Given 4 combinations selected per round, with no duplication of genotypes and duplication permitted for mappings, this gives four combinations:

1. $(G_{i\ldots n}, M_{i\ldots n})$ _ n x n = n^2 possibilities
2. $(G_{i\ldots n-1}, M_{i\ldots n})$ _ $(n-1)$ x n = $n^2 - n$ possibilities
3. $(G_{i\ldots n-2}, M_{i\ldots n})$ _ $(n-2)$ x n = $n^2 - 2n$ possibilities
4. $(G_{i\ldots n-3}, M_{i\ldots n})$ _ $(n-3)$ x n = $n^2 - 3n$ possibilities

Result 2. During each mapping round, then, $n^2 + (n^2 - n) + (n^2 - 2n) + (n^2 - 3n) = 4n^2 - 6n = n(4n-6)$ possibilities are considered. Comparing Result 1 and 2, Probabilistic Adaptive Mapping DGP allows n times as many possible combinations as the Standard per round.

Table 3 demonstrates that the PAM DGP algorithm considers n times as many combinations for a population of size n as the Adaptive Mapping algorithm during any given tournament round. The computational expense of evaluating an individual's fitness by considering every individual in the alternate population is avoided by using the probability table to provide a guided selection of fruitful combinations, but the solution search can still consider any combination of genotype and mapping, with no duplication of genotypes but duplication of mappings permitted, at any given tournament round.[2]

To summarize, elitism in PAM DGP allows the algorithm to keep the best genotype and mapping pair that currently generates the best fitness; thus neither the mapping nor the genotype that contribute to the highest fitness can be lost due to changing contexts. The retained top fitness genotype and mapping are not replaced until a better combination appears. The increased exploration of the search space using the probability table allows the exploration of any genotype-mapping combination at any time. This means that a higher fitness combination can be much more readily discovered during the algorithm. Furthermore, the search context for each population is the entirety of the other population throughout the algorithm, preventing the loss of an individual that is central to the context of the other population. PAM DGP thus minimizes an overall lack of fitness-based performance caused by the Red Queen Effect in virtue of its elitism and probability-based selection table. In the remainder of Section 4, we confirm the performance of PAM DGP over the Standard Adaptive Mapping algorithm using the Maximum Output benchmark introduced above.

4.3 The Maximum Output Problem

As described in the Adaptive Mapping DGP implementation of the MAX problem in Section 4.1, the MAX problem is implemented in PAM DGP with a linear (bit string) stack-based version of GP. Each individual is a stack-based machine composed of a general-purpose stack and an output register. A program that changes the state of the machine is a list of instructions from the function set in Table 1, where the program lengths of 50, 100, 150, 200, and 250 bits are tested [16, 18] (the 50 bit case is the most difficult, whereas the 250 bit case is the easiest).

In PAM DGP, the dimensions of the probability table are the respective population sizes. Parameterization of the algorithm thus involves considering the trade-off between the amount of initial genotype and mapping material you want available for the search and the sparseness of the probability grid. Optimization of the probability table dimension is naturally problem dependent. Under the MAX problem, we are able to use PAM DGP with a population of 8 (4 genotypes and 4 mappings), the smallest case. A tournament is stopped when the maximum round limit of 1250 is reached or the success criterion is met. In these experiments, the MAX problem was considered solved when an individual generated a number large enough that it is given the double value of "Infinity" by the Java 2 Runtime Environment, build 1.5.0_05, on a 1.25 GHz PowerPC G4 running

Mac OS X Version 10.4.4. All algorithm parameters are summarized in Table 4 for both DGP algorithms (with learning rate and noise threshold only applicable to PAM DGP).

Table 4. . Maximum Output parameterization of Adaptive Mapping and PAM DGP

Tournament Style	Steady State, 4 individuals for each round
Maximum Rounds	1250 (5000 individuals processed)
Experiments	50 independent runs
Function Set	+, *, const, dup, pop, stack2Register, register2Stack
Genotype structure	Stack-based w/ register; 50, 100, 150, 200, 250 bits
Mapping structure	Adaptive, 70 bits (10 bits per function set symbol)
Genotype mutation	Point mutation, threshold = 0.01
Mapping mutation	Point mutation, threshold = 0.01
Genotype crossover	Equal-sized blocks, threshold = 0.09
Mapping crossover	Equal-sized blocks, threshold = 0.09
Population size	4 or 25 individuals in each population
Fitness	Output register content after evaluation.
Objective	Generate largest number possible.
Termination	Infinity (success) or maximum rounds.
Learning rate	0.1
Noise threshold	0.95

The number of experiments out of 50 independent trials that solved the problem is given in Table 5. As noted in the third column, restricting the Standard Adaptive Mappiing DGP to a population of 8 to match PAM DGP hindered its performance, so it was permitted a starting population of 50 (25 individuals in each population). On equal basis of respective optimal population sizes, PAM DGP still dramatically outperforms the Standard Adaptive Mapping algorithm (Table 5). The function set symbol content of the solutions as a percentage of total symbols is shown in Table 6 along with p-values and acceptance or rejection of the hypothesis that the symbol frequencies in solutions of the two algorithms are equal. There is no significant difference at the 0.95 or 0.99 confidence intervals for 5 out of 7 symbols, where PAM DGP used less constants and more multiplication. Since the best solutions only require repeated duplication or multiplication following initial presence of a constant, it follows that PAM DGP is more effective at locating the optimal instruction types than the Standard Adaptive Mapping.

Table 5. Number of Maximum Output solutions in 50 independent experiments.

Symbol	PAM DGP, Pop. 8	Standard, Pop. 8	Standard, Pop. 50
50	0	0	0
100	10	0	1
150	38	0	6
200	47	9	22
250	48	21	28

Table 6. . Mean symbols as percentage of total solution over 50 trials with associated p-values for 200 bit, Population 8 MAX Problem using PAM DGP and Adaptive Mapping algorithms

	CONST	POP	DUP	S2R	R2S	PLUS	TIMES
PAM DGP	0.0292	0.0314	0.232	0.166	0.121	0.203	0.218
Adaptive	0.0471	0.0371	0.241	0.166	0.115	0.205	0.189
p-value	0.000249	0.275	0.575	0.986	0.591	0.882	0.00771

The mean best fitness for each tournament round over all 50 experiments is plotted in Figure 7 for both algorithms using their respective optimal populations for 50, 100, 150, and 200 bits. (250 bits represents a trend similar to 200 and was omitted for brevity.) Since a successful value of "infinity" is impossible to plot, the fitness measure for any experiment not yet achieving success at a round is used to determine the mean. The PAM DGP algorithm (solid line) outperforms the Standard Adaptive Mapping algorithm consistently throughout all tournament rounds. The algorithm is also more robust, as far fewer fitness spikes are evident in the PAM DGP trend line.

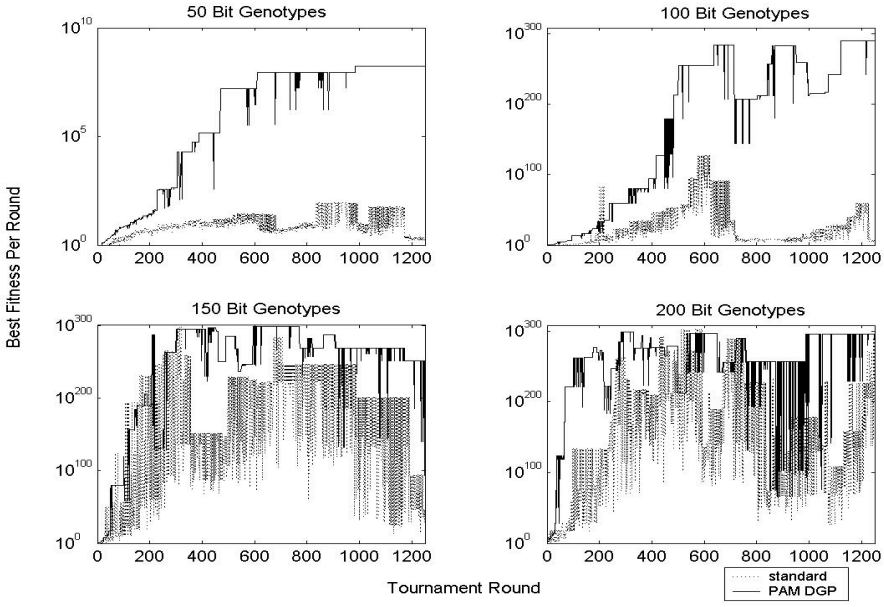

Figure 7. Mean best fitness per round over 50 independent runs for PAM DGP, population of 8, and the Standard Adaptive Mapping algorithm, population of 50, on the MAX problem.

5 Introducing the Adaptive Redundant Mapping to PAM DGP

5.1 Expected Benefits of a Redundant Adaptive Mapping

So far, the only structure used for mapping individuals in the PAM DGP framework has been the adaptive mapping scheme based on binary strings interpreted with the countingOnes function described in section 2.3, which simply sums all the ones in a given string to yield a frequency for Huffman's compression algorithm. Keeping the mapping structure in PAM DGP and the Adaptive Mapping algorithm constant thus far has allowed a comparison of the algorithm component of PAM DGP with the algorithm of the Adaptive Mapping DGP independent of differing mapping types. Having demonstrated the superiority of PAM DGP's algorithm component in the previous Section, we now move to improving the encoding process of its mapping individuals. There are a number of benefits that alternative mappings (encoding schemes) could provide that are not considered in works defining the Standard Adaptive Mapping [16, 18]. The most obvious of the benefits overlooked by assuming a Huffman encoding is that for any given unique genome (binary permutation) in the genotype space, there is only one phenotype (binary permutation) in the phenotype space that corresponds to it. That is, there is a one-to-one mapping from genotype to phenotype and the mapping is not *redundant* at all. As mentioned earlier, the genetic code in nature is redundant (or, as biologists say, it is "degenerate"). Practical algorithm engineering considerations aside, redundant mappings are more representative of the developmental paradigm as a whole. Specifically, from a biological perspective, many different genotypes result in phenotypes of comparable functionality [1]. According to the neutrality theory of evolution [14], most of natural evolution at the molecular level is due to mutations that are practically neutral with respect to selection. That is, variations in genetic material are typically neither advantageous nor disadvantageous, allowing evolutionary exploration of alternative genomes without a severe fitness cost to individuals possessing the alternate genomes. By using a redundant representation of the genetic code, it can be expected that solution search could benefit from neutral variations (although the presence of neutral variation is not within the scope of this work).

Given the description of a redundant mapping provided by Keller and Banzhaf in [12], we consider a mapping (encoding) to be *redundant* if it can map more than one codon (genotype subsequence) onto the same symbol. We should be careful to be clear here; we are speaking of redundancy at the level of the encoding and this work investigates the benefits of a redundant mapping at that level of representation. That is, a mapping/encoding is redundant when it allows two or more binary sequences to map to a single symbol. However, if any symbol of the function set is mapped to by more than one genotype, then it is the case that more than one distinct genotype (entire binary sequence representing an individual) can map to the same phenotype. All that is necessary for this to occur is that the two genotypes be identical except for the binary subsequence corresponding to two different codons that map to the same symbol in both phenotypes. Thus, if a representation is redundant at the mapping encoding level, then it is also redundant at the level of the genotype-phenotype individuals themselves. There is thus

little concern about confusion, for the neutrality literature often calls a genotype-phenotype mapping *redundant* if it maps multiple genotypes to the same phenotype.

The Huffman mapping structure was advocated largely for its ability to allow refined exploration during later (exploitative) phases of the solution search [16-18]. However, the mapping scheme of Keller and Banzhaf provides a more transparent, but severe, method of emphasizing symbols of the function set. As described in Section 3.1, each symbol of the function set may be encoded with more than one bit sequence. In other words, any given symbol of the function set can be *redundantly* encoded. The emphasis of a symbol of the function set then results from the number of bit sequences corresponding to it, as well as the content of those bit sequences (the more times the sequence is occurring in the building blocks of the fittest individuals, the better). In the work of Banzhaf and Keller, the function set symbol allotted to each of the mapping's possible binary encodings is chosen arbitrarily [1, 11-13]. It is possible for a mapping to dictate that certain function set symbols are never to be read when interpreting the genotype, or even in the extreme case that only one function set symbol is ever to be read. In contrast, the adaptive mapping scheme using Huffman encoding always ensures that every symbol of the function set gets some encoding allotted to it, although symbols of the function set with low frequencies in the mapping will be given more obscure, often longer, encodings. The ability of the redundant mapping to effectively trim and more readily emphasize symbols in the function set is expected to produce solutions to harder problems using tailored function sets.

In addition to the potential for more efficient search using neutral mutations, better adherence to developmental considerations, and appropriate emphasis of function set symbols, the use of redundant mappings provides lower computational cost. Consider the complexity analysis of Traditional GP as a benchmark. Assume that a GP tournament runs for t rounds before the stop or success criterion is met. Given the evaluation of n individuals for each round, a Traditional GP under a fixed length representation only involves execution of some constant k number of instructions n times, giving linear time $O(n)$ for evaluation of all individuals. Following evaluation in each round, we will assume that an efficient sorting mechanism is used in each tournament round for ranking of n individuals, such as heap sort which is known to have complexity $O(n \log n)$. Breeding operations (reproduction on $n/2$ individuals, crossover and mutation on $n/2$ individuals given n individuals per round) are then performed in linear time $O(n)$ following ranking. The overall complexity of the Traditional GP can thus be said to be $O(n (n + n \log n)) = O(n^2 + n^2 \log n) \in O(n^2 \log n)$. Based on the preceding assumptions of evaluation, sorting, and breeding, both the Standard Adaptive Mapping algorithm and PAM DGP using Huffman encoding take $O(n^3 \log n + n^3 + n^2 \log n + n^2) \in O(n^3 \log n)$ time given that Huffman encoding has complexity $O(n + n \log n)$. PAM DGP using redundant mappings replaces Huffman encoding with a direct encoding scheme (to be explained below), providing a cost of $O(n)$ instead of the cost of Huffman's algorithm. The overall complexity of PAM DGP using redundant encodings is $O(n^3 + n^2 \log n + n^2) \in O(n^3)$, which is a savings of $O(\log n)$ time compared to PAM DGP using Huffman-based mappings. Interested readers will find a proof of the complexities in [29].

5.2 Adaptive Redundant Mapping Encoding Process

Keller and Banzhaf introduced the concept of redundant mappings in which each genotype individual is paired with its mapping for the entire tournament [11-13]. This scheme is not compatible with the coevolutionary architecture of the Adaptive Mapping algorithm or the PAM DGP framework. Thus, we design an *adaptive redundant mapping* for the PAM DGP framework with an associated encoding mechanism and structure suited to the PAM DGP algorithm. The proposed adaptive redundant mapping is summarized by Figure 8. Firstly, a list of binary sequences is chosen so that each symbol of the function set has the potential to be represented by one or more unique fixed length bit sequences. That is, a mapping individual is set to consist of $b \geq s$ binary strings of length 10, where b is the number of binary sequences required to represent a function set of size s. For example, a function set of size 4 would require a list of four sequences of 2 bits each, and a function set of size 7 would require a list of eight sequences of 3 bits each (with one extra encoding). Instead of each of the binary strings in the mapping individual representing frequencies, as in the Huffman encoding, they directly represent indices of the ordered function set. Each of the b 10-digit binary strings is interpreted by converting them to their decimal representation and normalizing to the range [0…1]. They are then mapped onto an ordered function set *index* (where indices are numbered 0 to $s - 1$ for a function set of size s) by multiplying them by s and truncating to an integer value. (In the rare case where the normalized decimal for a binary string is 1, a reference beyond the last index is avoided by referencing the last index.) The ordering of the function set is arbitrary upon initialization and remains unchanged throughout the algorithm.

The countingOnes method is not used to interpret the mappings, as when mapped onto the ordered function set it would greatly favor symbols at the centre of the function ordering and very rarely choose symbols at the start or end of the ordered function set. (Consider that in a countingOnes interpretation of 10 bits, there are many possibilities of encoding a frequency of 0.5 where five of the ten bits are ones, but only one possibility of encoding the frequency of 1.0 where all bits are set to one. Using normalized binary to decimal conversion, however, there is a uniform chance of choosing any value from 0.0 to 1.0.)

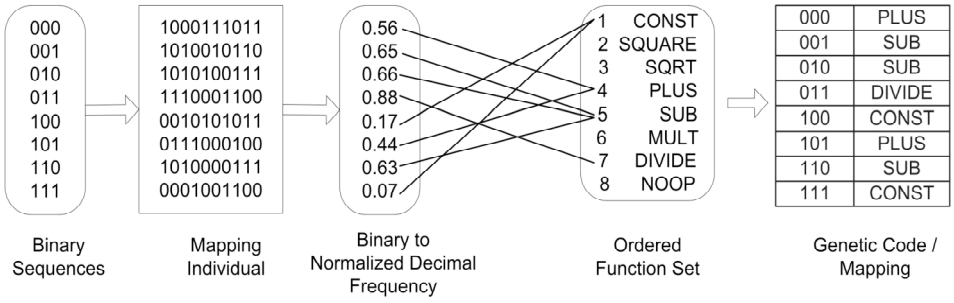

Figure 8. Interpretation of adaptive redundant mapping individuals in PAM DGP.

An elaboration on the term *redundancy* as it applies to the redundant adaptive mapping may be helpful to avoid confusion. As mentioned previously, we adopt the term *redundant* in the sense of Keller and Banzhaf [12] in particular: a mapping is redundant if

it "*may* map more than one codon onto the same symbol." We call our adaptive mapping redundant because it has the *potential* to map more than one phenotype to a single genotype; although it may not do so in all cases. By allocating distinct binary encodings to each symbol of the function set when the problem's function set can be represented by exactly s binary sequences ($s=b$), there is no redundancy introduced at all. If the problem's function set cannot be represented by exactly s binary sequences (it is required that $b > s$), then there will be some problem-dependent minimal degree of redundancy. The maximum redundancy for every function set of size s (every problem) is always the case where every function set symbol has the same encoding (although this would not allow a very fit solution in most cases).

Because the mapping is adaptive, it can range from the problem-dependent lower limit of redundancy to the upper limit throughout the algorithm. Thus, the redundant adaptive mapping can trim the function set when necessary or keep all function set symbols. Moreover, the redundant adaptive mapping has the added benefit that it can adaptively set its level of redundancy and may for some problem cases produce solutions that opt not to use redundancy at all—recalling the definition adopted from Keller and Banzhaf [12], it may (or may not) map more than one encoding onto the same symbol. Mathematically speaking, the mapping is non-injective and non-surjective: every element of the function set can be mapped to by one or more binary sequences, but not every element of the function set will necessarily have a binary sequence mapped to it, respectively.

5.3 Comparative Mapping Performances on the Maximum Output Problem

In this section we now empirically investigate the performance of the Huffman encoded and redundant mapping schemes in the PAM DGP framework for the Maximum Output benchmark. All algorithm parameters remain identical to those described in Table 4; only the mapping structures are being compared. Both implementations use a population of 50. Table 7 shows that the Huffman mapping produces more solutions than the redundant mapping in the PAM DGP framework at all bit levels. Table 8 shows that there is no significant difference in the function set symbol content of the solutions for the two mappings at the 0.95 or 0.99 confidence intervals for 5 out of the 7 symbols. Both mapping types place a healthy emphasis on DUP, PLUS, and TIMES—all very useful for creating large outputs. The redundant mapping uses more load constant to register (CONST) symbols, while the Huffman encoding uses more register to stack (R2S) transfers (both significant at the 0.99 confidence interval). When used properly, either function set member can be an effective part of a solution to generate large numbers.

Table 7. Number of Maximum Output solutions in 50 independent experiments.

Symbol	Huffman	Redundant
50	0	0
100	10	3
150	38	23
200	47	20
250	48	30

Table 8. Mean symbols as percentage of total solution over 50 trials with associated p-values for the 200 bit, population 8 MAX Problem using adaptive and redundant mappings in PAM DGP.

	CONST	POP	DUP	S2R	R2S	PLUS	TIMES
Huffman	0.0291	0.0314	0.232	0.166	0.121	0.203	0.218
Redundant	0.182	0.0228	0.212	0.146	0.0547	0.192	0.190
p-value	1.99×10^{-9}	0.396	0.436	0.244	0.000236	0.619	0.166

While it appears from the simple MAX problem that Huffman mappings outperform redundant encodings, the problem is too simple to confirm anything other than the redundant encoding may introduce too much search overhead to efficiently solve trivial problems as quickly. By their nature, regression problems in general have many alternative solutions using numerous combinations of function set symbols that allow an approximation to fitness cases. Thus, regression problems are not necessarily the best problems to realize the benefits of tailored function sets that a redundant mapping offers. Solutions to the maximum output problem can easily be found without even reducing the function set, let alone requiring the degree of emphasis afforded by the redundant mapping. The Maximum Output Problem was used largely to benchmark and develop PAM DGP against the Standard Adaptive Mapping algorithm of Margetts and Jones because they used that problem to introduce their algorithm to the literature [18]. We now consider two more difficult benchmarks taken from the medical classification domain to test the potential benefits of the PAM DGP redundant adaptive mapping.

6. Results for Medical Classification Benchmarks

6.1 Problem Definitions and Parameterizations

In order to test adaptive mappings using redundant encodings in PAM DGP in a classification domain, we used two medical data sets from the well known UCI machine learning repository [19]. The first data set used was the Heart Disease data collected at the Cleveland Clinic Foundation [7]. The database contained 303 instances (164 negative, 139 positive), each consisting of 13 attributes, with a 14^{th} indicating positive or negative diagnosis. The second consisted of 699 instances of Breast Cancer data (458 negative and 241 positive) obtained at the University of Wisconsin Hospitals, Madison [32]. Each instance contained 9 useful attributes, with a tenth classifying it as positive or negative.

Additional parsing of the file was performed prior to the experiments: Unknown values were replaced by the mean value of data recorded for the relevant attribute in the database, and a positive or negative classification was changed to '1' or '0' for all instances. Finally, the experiments used four-fold cross-validation to verify accuracy of the findings. The data set was partitioned so that 25% was used as a test set, with 75% used as the training set. Each of the four partitions used a unique set of instances for the training and test partitions, and both the training and test set retained a class distribution

representative of the entire data set. For partitions 1-3 of the Breast data set, this resulted in 524 training instances and 175 test instances; for partition 4, there were 525 training instances and 174 test instances. For partitions 1-3 of the Heart data set, this resulted in 227 training instances and 76 test instances; for partition 4, there were 228 training instances and 75 test instances. If the output of an evaluated individual (program) was less than 0 on a fitness case, the case was considered to have been classified as negative; if the individual evaluated to greater or equal to 0, the case was classified as positive.

The division of the data and parameterization are shown in Table 9. Lower mapping mutation and crossover rates for the mapping population than the genotype population provide a more stable background for solution search among genotypes. Preliminary experiments showed that no gain in training was accomplished for the Heart Disease data after 30 000 rounds, so this was chosen as the tournament limit for training the classifier. Breast Cancer was given a limit of 50 000 rounds for training. Note that the function set is larger than would typically be employed for a classification problem (for instance [2]), since we are interested in evaluating the ability of the mapping to suitably focus the search on the most applicable subset of symbols.

Tournament Style	Steady State, 4 individuals for each round
Training Rounds	Heart=30 000, Breast=50 000
Cross validation	4-fold, each with 75% Training and 25% Test
Data Set Characteristics	Heart: 303 instances, Breast: 699 instances
Function Set	+, *, -, % (protected), SIN, COS, EXP, LOG (base 10), SQRT, NATLOG
Genotype structure	Instruction sequence with 4 registers; 320 bits
Mapping structure	100 bits (10 bits per function set symbol)
Genotype mutation	XOR mutation, threshold = 0.5
Mapping mutation	Point mutation, threshold = 0.1
Genotype crossover	Equal-sized blocks, threshold = 0.9
Mapping crossover	Equal-sized blocks, threshold = 0.1
Population size	25 genotypes, 25 mappings (50 for traditional)
Wrapper	IF ((out ≥ 0) AND (label == TRUE)) OR ((out < 0) AND (label == FALSE)) THEN classification++
Fitness	Classification count
Objective	Highest classification accuracy possible on test set.
Learning rate	0.1
Noise threshold	0.95

Table 9. Medical Classification Problem Parameterization. "Out" is an individual's output and "label" refers to the classification of a given fitness case.

6.2 Medical Classification Performance Results

The classification accuracy of the Heart and Breast classifiers after 30 000 and 50 000 rounds of training, respectively, over 50 independent trials are shown in Figures 9 and 10, respectively. Results describe performance for Traditional (linear) GP (Traditional), the Adaptive Mapping algorithm (Standard), PAM DGP using Huffman encoding (Huffman), and PAM DGP using redundant adaptive mappings (Redundant). The results are based on four-fold cross-validation, so the median and spread shown in the boxplot correspond to the mean accuracy across all four unique test sets over the 50 trials. Each box indicates the lower quartile, median, and upper quartile values. If the notches of two boxes do not overlap, the medians of the two groups differ at the 0.95 confidence interval. Points represent outliers to whiskers of 1.5 times the interquartile range.

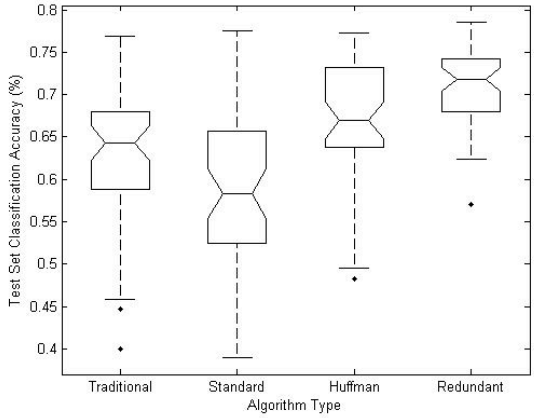

Figure 9. Boxplot of mean classification accuracy for the Cleveland Heart data set over 50 trials using four-fold cross-validation. Each partition was 75% training, 25% test.

It is evident from Figure 9 that the redundant adaptive mapping outperforms the Huffman mapping in PAM DGP at the 0.95 confidence interval, as well as outperforming all other GP-based algorithms examined at the 0.95 confidence interval (there is no overlap between Redundant's notch and the other algorithms). In fact, the redundant mapping also boasts the best median, general spread of data, best accuracy achieved (note top of upper whisker). The Standard (Original) Adaptive Mapping algorithm of Margetts and Jones is the worst performer of all the algorithms at the 0.95 confidence interval, as well as when considering both median and general spread of the data.

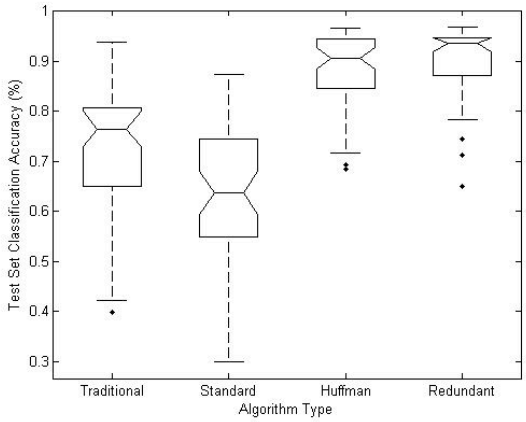

Figure 10. Boxplot of mean classification accuracy for the Wisconsin Breast data set over 50 trials using four-fold cross-validation. Each partition was 75% training, 25% test.

In the case of the easier Breast data set, Figure 10, while redundant mappings do not actually outperform Huffman mappings at the 0.95 confidence interval, the redundant mapping does outperform the Huffman mapping when considering median and general spread of the data (two rightmost box plots). It is certainly the case that the redundant mapping is not outperformed by the Huffman alternative in any respect. The redundant mapping, however, outperforms the two other GP-based algorithms at the 0.95 confidence interval. As was the case for the Heart Disease data, the Standard Adaptive Mapping, followed by Traditional GP, had the worst performance out of all four algorithms given 0.95 confidence interval, median, and general spread of the data. The PAM DGP implementations thus clearly outperform the other GP alternatives in both the Breast and Heart medical classification domains, with the redundant mapping being preferable.

6.3 Function Set Analysis

While redundant mappings are shown to outperform other algorithms in these medical classifier domains, it remains to be shown that the redundant encoding is trimming the symbols in the function set to yield this improved performance (recall that Huffman encoding cannot trim the function set). It is also of interest to see whether or not there is any significance to the particular symbols chosen by the redundant mappings to remain as useful members of the function set. The average number of unique function set symbols used in the classification solutions to the experiments described above are given in Figures 11 and 12.

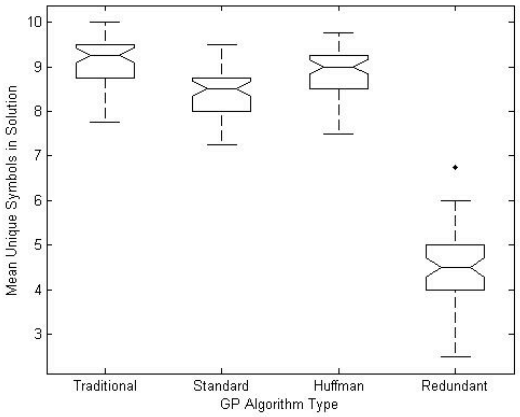

Figure 11. Boxplot of mean number of unique function set symbols used in the classifier for the Cleveland Heart data set over 50 trials using four-fold cross-validation. Each partition was 75% training, 25% test.

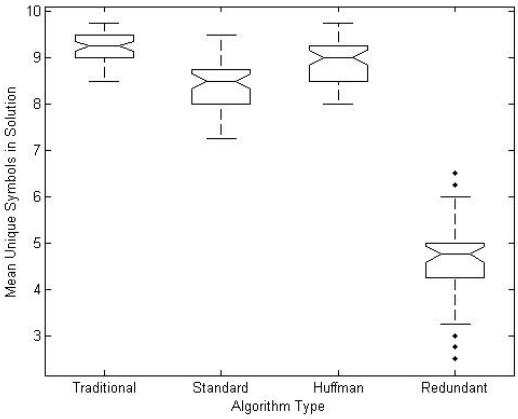

Figure 12. Boxplot of mean number of unique function set symbols used in the classifier for the Wisconsin Breast data set over 50 trials using four-fold cross-validation. Each partition was 75% training, 25% test.

It is evident in both Figures 11 and 12 that the redundant mapping provided a significant reduction to the size of the function set used by the solutions. In both datasets, the highest outlier (most symbols) is still below the lowest number of symbols used in all other algorithms. In some cases, in both classification problems the average over the 4 partitions was fewer than 3 symbols per solution. The reduction of the function set used, in conjunction with the improved classification performance, indicates that the redundant mapping is improving classification performance (beyond any algorithm using Huffman encoding or Traditional GP) by trimming the function set.

In terms of the symbols chosen to remain in the function set, the results are summarized in Figures 13 and 14 for Heart and Breast data, respectively. On the Heart classification problem, the redundant mappings produce a statistically significant difference in symbols for solution composition at the 95% confidence interval for every symbol when compared to Huffman. Thus, the redundant mappings for the Heart problem not only tend to use less symbols in a solution on average, but the symbols it chooses for a solution are significantly different than those used by the Huffman encoding. Needless to say, symbols identified by the redundant mapping are also significantly different from those utilized by Traditional GP. Figure 13 shows that the Huffman mapping simply produces a very even distribution of the available symbols across all solutions, even more so than that identified by Traditional GP. In contrast, the Redundant mapping favors a distinctive combination of symbols, using multiplication, base 10 log, and natural log to create a classifier, which has both superior classification accuracy and a reduced function set size.

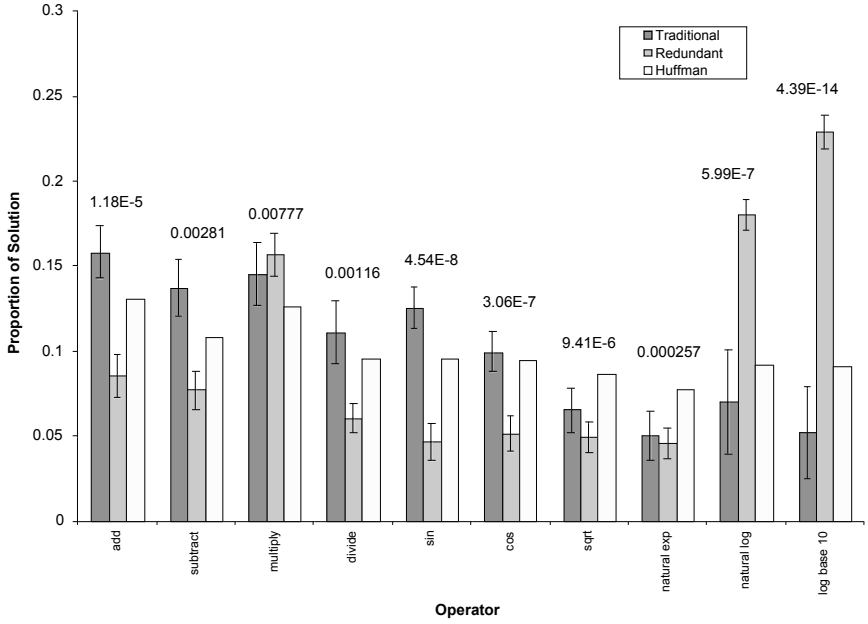

Figure 13. Mean symbols as a proportion of total solutions over 50 trials for the Cleveland Heart Problem, population 50. Error bars reflect two-tailed t-distribution for the 0.95 confidence interval. P-values corresponding to Huffman and Redundant mappings are displayed above each set of data points.

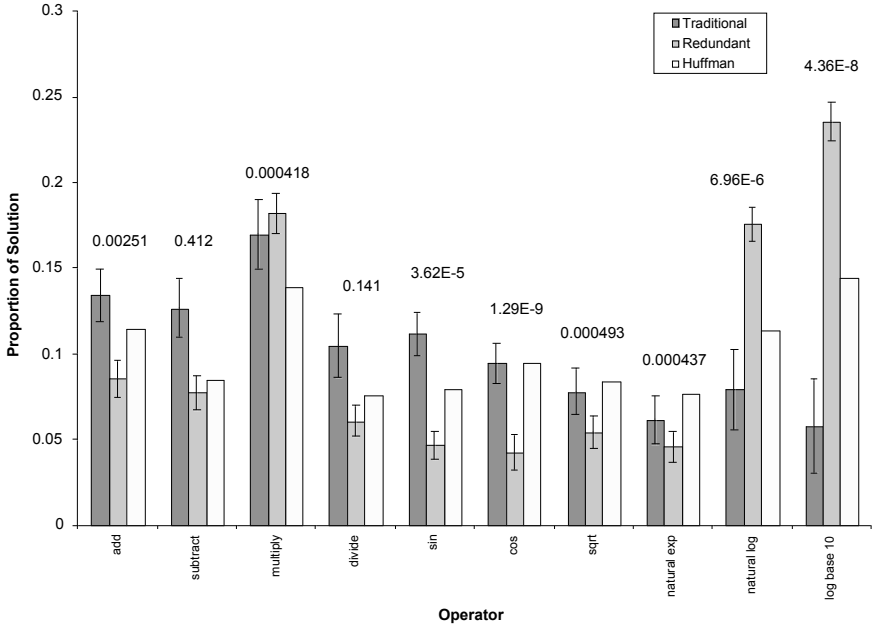

Figure 14. Mean symbols as a percentage of total solutions over 50 trials for the Wisconsin Breast Problem, population 50. Error bars reflect two-tailed t-distribution for the 0.95 confidence interval. P-values corresponding to Huffman and Redundant mappings are displayed above each set of data points.

Figure 14 for the Breast classification data again shows a comparatively uniform utility of function set symbols when using a Huffman mapping, compared to Traditional GP or the redundant mappings. The classifier produced using the redundant mappings has again favored the function set members multiplication, base 10 log, and natural log. We can see that for both classification problems, the redundant mapping has led to reduced function set size and greater emphasis of particular function set symbols to produce higher performance classifiers. It is also interesting that across both of these classification problems that the redundant mapping implementations have chosen to heavily emphasize the same subset of three particular symbols to generate higher fitness classifiers.

7. Summary and Conclusions

This work presented a new developmental GP algorithm that models the coevolution of the genetic code (biological codon to amino acid) and genotype in nature. Previous similar developmental systems are discussed, as were pathologies of coevolution that require addressing in order to ensure the design of an efficient search algorithm. Shortcomings of a previous coevolutionary algorithm, the Adaptive Mapping algorithm of Margetts and Jones [16-18], were then exposed and illustrated empirically. In particular the previous Adaptive Mapping algorithm was demonstrated to suffer from a

lack of exploration of the search space and the Red Queen Effect, with associated repeated loss of context, fitness spiking, and an overall lack of fitness-based performance.

The PAM DGP algorithm was then introduced, initially operating on populations of genotypes and mappings using the same structure and encoding process as the Adaptive Mapping algorithm. The PAM DGP algorithm was shown to outperform the Adaptive Mapping algorithm on the Maximum Output benchmark used to introduce the latter algorithm to the literature [18]. The PAM DGP algorithm components responsible for overcoming the drawbacks of the Adaptive Mapping algorithm, and the Red Queen Effect in general, were combined elitism and a novel probability table that allowed dynamic fitness proportionate selection of promising genotype-mapping pairs. Having established the performance of the PAM DGP algorithm itself, a population of new (more developmentally sound) adaptive redundant mappings were incorporated into the PAM DGP framework. While PAM DGP with the adaptive redundant mappings provided too much search overhead to be effective at the very simple Maximum Output regression problem, it was found to yield highly effective classifiers for the much more difficult benchmark medical classification problems by tailoring the available function set. In effect, by concentrating on explicitly identifying the most suitable symbols from the function set, the PAM DGP algorithm (with redundant mapping) was able to provide an efficient mechanism for reducing the size of the search space relative to both traditional GP or the original Adaptive Mapping algorithm of Margetts and Jones.

PAM DGP (using adaptive redundant mappings) is a developmental system that models the coevolution of genetic code and genotype, as well as the redundant nature of the biological genetic code. In so doing, PAM DGP provides an effective means of dealing with the Red Queen Effect in two population cooperative coevolution and yields classifiers for medical classification benchmarks capable of outperforming Traditional GP, Standard Adaptive Mapping DGP, and PAM DGP using Huffman encoding. Future work will investigate the use of adaptive redundant mappings in PAM DGP to solve other difficult optimization tasks, including the automatic generation of recursion from machine-language function sets. We also plan to experiment with the evolution of problem-tailored genetic codes in a larger developmental framework.

Acknowledgements

The authors gratefully acknowledge the support of a NSERC PGS-B and Izaak Walton Killam scholarship (Garnett Wilson), and the CFI New Opportunities and NSERC research grants (Dr. M. Heywood).

Notes

1. Assumes a symbiotic as opposed to a speciation model of cooperative coevolution. The latter is a distinct form of the symbiotic model, and widely used within evolutionary multi-objective optimization to encourage multiple solutions from a single population. It is thus not relevant to this work.

2. That is, provided the noise threshold is set to anything less than 1.0, so that no column in the probability table will ever completely exclude any combination by setting the probability of its roulette selection it to 0.

References

1. W. Banzhaf, "Genotype-Phenotype Mapping and Neutral Variation," in Parallel Problem Solving from Nature III, Jerusalem, Israel, Y. Davidor, H.-P. Schwefel and R. Manner (Eds.), Springer-Verlag: Berlin, 1994, pp. 322-332.
2. M. Brameier and W. Banzhaf, "A Comparison of Linear Genetic Programming and Neural Networks in Medical Data Mining," IEEE Transactions on Evolutionary Computation, vol. 5, no. 1, pp. 17-26, 2001.
3. A. Bucci and J. Pollack, "On Identifying Global Optima in Cooperative Coevolution," in Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2005), Washington, DC, USA, H.-G. Beyer, U.-M. O'Reilly, D. Arnold, W. Banzhaf, C. Blum, E. Bonabeau, E. Cantu-Paz, D. Dasgupta, K. Deb, J. Foster, E. d. Jong, H. Lipson, X. Llora, S. Mancoridis, M. Pelikan, G. Raidl, T. Soule, A. Tyrrell, J.-P. Watson and E. Zitzler (Eds.), ACM Press: New York, 2005, pp. 539-544.
4. D. Cliff and G. Miller, "Tracking the Red Queen: Measurements of adaptive progress in co-evolutionary simulations," in Advances in Artificial Life: Proceedings of the Third European Conference on Artificial Life (ECAL 95), Granada, Spain, F. Moran, A. Moreno, J. Merelo and P. Chacon (Eds.), Springer-Verlag: London, 1995, pp. 200-218.
5. F. Crick, "The origin of the genetic code," Journal of Molecular Biology, vol. 38, no. 3, pp. 367-379, 1968.
6. E. de Jong and J. Pollack, "Ideal Evaluation from Coevolution," Evolutionary Computation, vol. 12, no. 2, pp. 159-192, 2004.
7. R. Detrano, A. Janosi, W. Steinbrunn, M. Pfisterer, J. Schmid, S. Sandhu, K. Guppy, S. Lee and V. Froelicher, "International application of a new probability algorithm for the diagnosis of coronary artery disease," American Journal of Cardiology, vol. 64, pp. 304-310, 1989.
8. G. Di Caro and M. Dorigo, "AntNet: Distributed Stigmergetic Control for Communication Networks," Journal of Artificial Intelligence Research, vol. 9, pp. 317-365, 1998.
9. S. Freeland, "The Darwinian Genetic Code: An Adaptation for Adapting?," Genetic Programming and Evolvable Machines, vol. 3, no. 2, pp. 113-127, 2002.
10. C. Gathercole and P. Ross, "An Adverse Interaction between Crossover and Restricted Tree Depth in Genetic Programming," in Genetic Programming 1996: Proceedings of the First Annual Conference, Stanford, California, J. Koza, D. Goldberg, D. Fogel and R. Riolo (Eds.), MIT Press: Cambridge, MA, 1996, pp. 291-296.
11. R. Keller and W. Banzhaf, "Genetic Programming using Genotype-Phenotype Mapping from Linear Genomes in Linear Phenotypes," in Genetic Programming 1996: Proceedings of the First Annual Conference, Stanford, California, J. Koza, D. Goldberg, D. Fogel and R. Riolo (Eds.), MIT Press: Cambridge, MA, 1996, pp. 116-122.
12. R. Keller and W. Banzhaf, "The Evolution of Genetic Code in Genetic Programming," in Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 1999), Orlando, Florida, W. Banzhaf, J. Daida, A. Eiben, M. Garzon, V. Honavar, M. Jakiela and R. Smith (Eds.), Morgan Kaufman: San Francisco, 1999, pp. 1077-1082.
13. R. Keller and W. Banzhaf, "Evolution of Genetic Code on a Hard Problem," in Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2001), San Francisco, California, L. Spector, E. Goodman, A. Wu, W. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. Garzon and E. Burke (Eds.), Morgan Kaufman: San Francisco, 2001, pp. 50-56.

14. M. Kimura, "Evolutionary rate at the molecular level," Nature, vol. 217, pp. 624-626, 1968.
15. W. Langdon and R. Poli, "An Analysis of the MAX Problem in Genetic Programming," in Genetic Programming 1997: Proceedings of the Second Annual Conference, Stanford, California, J. Koza, K. Deb, M. Dorigo, D. Fogel, M. Garzon, H. Iba and R. Riolo (Eds.), Morgan Kaufman, 1997, pp. 222-230.
16. S. Margetts, Adaptive Genotype to Phenotype Mappings for Evolutionary Algorithms, Ph.D. thesis, School of Computer Science, Cardiff University, Wales, Great Britain, 2001.
17. S. Margetts and A. Jones, "Phlegmatic Mappings for Functional Optimisation with Genetic Programming," in Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2000), Las Vegas, Nevada, L. Whitley, D. Goldberg, E. Cantu-Paz, L. Spector, I. Parmee and H.-G. Beyer (Eds.), Morgan Kaufman, 2000, pp. 82-89.
18. S. Margetts and A. Jones, "An Adaptive Mapping for Developmental Genetic Programming," in Proceedings of the Fourth European Conference on Genetic Programming (EuroGP 2001), Lake Como, Italy, J. Miller, M. Tomassini, P. Lanzi, C. Ryan, A. Tettamanzi and W. Langdon (Eds.), Springer Verlag: Berlin, 2001, pp. 97-107.
19. D. Newman, S. Hettich, C. Blake and C. Merz, UCI Repository of machine learning databases [http://www.ics.uci.edu/~mlearn/MLRepository.html], University of California, Department of Information and Computer Science, Irvine, CA, 1998.
20. M. O'Neill and A. Brabazon, "mGGA: The meta-Grammar Genetic Algorithm," in Proceedings of the 8th European Conference on Genetic Programming (EuroGP 2005), Lausanne, Switzerland, M. Kaijzer, A. Tettamanzi, P. Collet, J. Hemert and M. Tomassini (Eds.), Springer: Berlin, 2005, pp. 311-320.
21. M. O'Neill and C. Ryan, "Grammatical Evolution by Grammatical Evolution: The Evolution of Grammar and Genetic Code," in Proceedings of the Seventh European Conference on Genetic Programming (EuroGP 2004), Coimbra, Portugal, M. Keijzer, U.-M. O'Reilly, S. Lucas, E. Costa and T. Soule (Eds.), Springer: Berlin, 2004, pp. 138-149.
22. L. Panait, S. Luke and R. Wiegand, "Biasing Coevolutionary Search for Optimal Multiagent Behaviors," IEEE Transactions on Evolutionary Computation, vol. 10, no. 6, pp. 629-645, 2006.
23. L. Panait, R. Wiegand and S. Luke, "A sensitivity analysis of a cooperative coevolutionary algorithm biased for optimization," in Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2004), Seattle, WA, K. Deb, R. Poli, W. Banzhaf, H.-G. Beyer, E. Burke, P. Darwen, D. Dasgupta, D. Floreano, J. Foster, M. Harman, O. Holland, P. Lanzi, L. Spector, A. Tettamanzi and D. Thierens (Eds.), Springer: Berlin, 2004, pp. 573-584.
24. M. Potter, The design and analysis of a computational model of cooperative coevolution, Ph.D. thesis, Department of Computer Science, George Mason University, Fairfax, VA, 1997.
25. M. Potter and K. De Jong, "Cooperative Coevolution: An Architecture for Evolving Coadapted Subcomponents," Evolutionary Computation, vol. 8, no. 1, pp. 1-29, 2000.
26. G. Sella and D. Ardell, "The Coevolution of Genes and Genetic Codes: Crick's Frozen Accident Revisted," Journal of Molecular Evolution, vol. 63, no. 3, pp. 297-313, 2006.
27. G. Wagner and L. Altenberg, "Perspectives: Complex Adaptations and the Evolution of Evolvability," Evolution, vol. 50, no. 3, pp. 967-976, 1996.
28. R. Wiegand and M. Potter, "Robustness in Cooperative Coevolution," in Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2006), Seattle, Washington, M. Keijzer, D. Arnold, V. Babovic, C. Blum, P. Bosman, M. Butz, C. Coello, D. Dasgupta, S. Ficici, J. Foster, A. Hernandez-Aguirre, G. Hornby, H. Lipson, P. McMinn, J. Moore, G. Raidl, F. Rothlauf, C. Ryan and D. Thierens (Eds.), ACM Press: New York, 2006, pp. 369-376.
29. G. Wilson, Probabilistic Adaptive Mapping Developmental Genetic Programming, Ph.D. thesis, Faculty of Computer Science, Dalhousie University, Halifax, NS, Canada, 2007.
30. G. Wilson and M. Heywood, "Probabilistic (Genotype) Adaptive Mapping Combinations for Developmental Genetic Programming," in Proceedings of the 2006 IEEE Congress on

Evolutionary Computation (CEC 2006), Vancouver, Canada, IEEE Press, 2006, pp. 8667-8674.

31. G. Wilson and M. Heywood, "Probabilistic Adaptive Mapping Developmental Genetic Programming (PAM DGP): A New Developmental Approach," in Proceedings of the 9th International Conference on Parallel Problem Solving from Nature (PPSN IX), Reykjavik, Iceland, T. Runarsson, H.-G. Beyer, E. Burke, J. Merelo-Guervos, L. Whitley and X. Yao (Eds.), Springer-Verlag: Berlin, 2006, pp. 751-760.
32. W. Wolberg and O. Mangasarian, "Multisurface method of pattern separation for medical diagnosis applied to breast cytology," Proceedings of the National Academy of Sciences, USA, vol. 87, pp. 9193-9196, 1990.