Multipopulation Cooperative Coevolutionary Programming (MCCP) to Enhance Design Innovation

Emily M. Zechman
CB 7908, North Carolina State University
Raleigh, NC 27695
emzechma@ncsu.edu

S. Ranji Ranjithan
CB 7908, North Carolina State University
Raleigh, NC 27695
ranji@ncsu.edu

ABSTRACT

This paper describes the development of an evolutionary algorithm called Multipopulation Cooperative Coevolutionary Programming (MCCP) that extends Genetic Programming (GP) to search for a set of maximally different solutions for program induction problems. The GP search is structured to generate a set of alternatives that are similar in design performance, but are dissimilar from each other in the solution (or design parameter) space. This is expected to yield potentially more creative designs, thus enhancing design innovation. Application of MCCP is demonstrated through an illustrative example involving GP-based classification of genetic data to diagnose malignancy in cancer. Four different classifiers, based on highly dissimilar combinations of genes, but with similar prediction performances were generated. As these classifiers use a diverse set of genes, they are collectively more effective in screening cancer samples that may not all properly express every gene.

Categories and Subject Descriptors

I.2.2 [**Artificial Intelligence**]: Automatic Programming Program Synthesis

General Terms

Algorithms, Design

Keywords

Genetic Programming, Evolutionary Programming, Niching, Lymphoma Cancer Classification

1. INTRODUCTION

When faced with a new design problem, humans are likely to follow incremental steps, anchored on prior or existing solutions, typically leading to only marginally creative designs. Being a global search heuristic, Genetic Programming (GP) is able to conduct a broad search outside of obvious designs to identify a potentially more innovative design to a new problem. In early applications, GP performed well for toy problems that tested its ability to re-discover functions from sets of data [14]. Recently, it has been applied to more complex design problems that require creativity and innovation. For example, [15] demonstrates the use of GP to discover patentable circuit designs; GP is now used frequently in generating human-competitive designs for an array of applications. GP is capable of exploring more creative designs than humans can since it is not biased towards traditional or standard designs.

The innovative aspects of a GP-based design procedure can be further enhanced by structuring the procedure to explicitly maximize creativity. It is difficult to optimize for creativity, as it is a subjective quality and is not easily expressed quantitatively in a mathematical statement. An indirect approach for generating more creative designs is to explore the solution space for not only the best (or optimal) design, but also a set of near-optimal alternative designs. An alternative design could be generated by slightly tweaking the design variable values of the best solution; however, such a marginally different design would not necessarily offer a creative or innovative alternative with any possibly patentable ideas. On the other hand, an alternative design with maximally different design variable values, i.e., a design constituted of greatly dissimilar operators or constructors, is likely to represent a relatively more creative design manifested by unbiased and unusual combinations of design variables. Therefore, systematically exploring for a set of solutions with similar design performance but with maximally different design variable values will likely lead to more creative designs.

The search for a set of maximally different alternative solutions has been investigated in the context of numeric optimization problems. As described in [2], the *modeling to generate alternatives* approach implements a systematic exploration to generate a small number of alternative solutions that perform similarly well and are located far apart in the decision space. Several studies (e.g., [3], [4], [5], [10], [16], and [21]) report the development of an array of alternatives generation procedures and their application to a number of realistic problems.

This paper describes the development of an evolutionary algorithm called Multipopulation Cooperative Coevolutionary Programming (MCCP) that extends Genetic Programming to search for a set of maximally different solutions for program induction problems. Application of MCCP is

GECCO'05, June 25–29, 2005, Washington, DC, USA.

demonstrated through an illustrative example involving GP-based classification of genetic data to diagnose malignancy in cancer.

2. METHODOLOGIES FOR GENERATING MAXIMALLY DIFFERENT ALTERNATIVES

2.1 Mathematical Background for Generating Alternatives

The mathematical definition of modeling to generate alternatives for a search problem has been provided in [2]. Let a GP-based search problem be represented as:

$$\text{Minimize } Z = f(X) \tag{1}$$

where $f(X)$ is the function representing the prediction error (a surrogate for performance), which is minimized, and X, a solution to this problem, is the set of instructions or rules represented as a tree. While $f(X)$ is described in terms of error in this paper, the MCCP method presented herein is applicable with any appropriate measure of performance used in a GP-based search. Let X^* be the best tree identified by GP, and Z^* is the corresponding minimum error value. An alternative tree that is maximally different from X^* can be generated by solving the following model:

$$\text{Maximize } D = d(X, X^*) \tag{2}$$

$$\text{Subject to } f(X) \leq T(Z^*) \tag{3}$$

where D is a difference function based on $d(X, X^*)$, which represents a "distance" measure between two trees X and X^*, and T is a target that is specified in relation to the error value Z^*. T represents an allowable relaxation, if any, in the prediction error value. For example, if the lowest error identified is 2%, then the target could be set as two times the lowest error to allow the search for solutions that have at most a 4% error value.

Once the first alternative has been identified, additional alternatives can be generated by modifying the difference function D so that the new tree being sought is maximally different from all previous trees, while Eqn. 3 remains unchanged. The search for alternatives stops when either no significantly different new alternative is found or a sufficient number of alternatives is generated. Several algorithms have been designed for generating a sequence of maximally different alternative solutions to numeric optimization problems, based on mathematical programming search methods, including linear programming, nonlinear programming, integer/binary programming, and dynamic programming (e.g., [3] and [5]).

2.2 EA-based Approaches for Generating Alternatives

Several procedures have been developed to use Evolutionary Algorithms (EA) for generating alternatives. The most direct EA-based approach, as suggested by [10], is to solve iteratively the model defined in Eqns. $2-3$ by incrementally updating the difference function D when each new alternative is found. The application of this approach using a genetic algorithm (GA) is described in [10]. This simple approach may become computationally intensive, since repeated execution of the EA is required.

A niching operator can be used within EAs for generating a set of alternative solutions [17]. The niching operator can be used to generate a number of alternatives, where the number of niches (or alternative solutions) is changed by adjusting the sharing distance parameter or the niche count [16]. Niching will identify different solutions, but cannot ensure that the alternative solutions will be maximally different in the solution space. A post-screening step can be used to select from among the niches a few alternatives that are maximally different with respect to an appropriate difference function [16]. Niching has been investigated in the context of GP primarily for the purposes of maintaining diversity in the population to avoid premature convergence to a sub-optimal solution ([8], [12], and [18]). A set of alternative trees may be identified using niching, but the operator is not designed to maximize the difference between the trees and thus will not necessarily yield creative or novel solutions.

A GA-based procedure (GAMGA - Genetic Algorithms for Modeling to Generate Alternatives) presented in [16] extends the use of niching to identify maximally different solutions for numeric optimization problems. Niches are formed around the most different solutions in a population, and restrictive mating is used to encourage solutions to maintain differences in solution space as well as to avoid migrating from one niche to another. The algorithm introduces several additional parameters and algorithmic steps that require careful tuning.

Another EA-based procedure designed for numeric optimization problems, the Evolutionary Algorithm for Generating Alternatives (EAGA) [21], is designed with a minimal number of additional tuning parameters to explicity force solutions to be as different from each other as possible. EAGA uses a set of subpopulations to generate a predetermined number of alternative solutions. The first subpopulation searches for the solution with the best fitness and the secondary subpopulations search for solutions that are distant in solution space from each other as well as the first subpopulation while staying within the specified fitness target (Eqn. 3). As the search progresses, this target changes corresponding to the error value (Z^*) of the current best solution in the first subpopulation. Crossover, reproduction, selection, and mutation occur in each subpopulation separately, with no migration. As the structure of EAGA is independent of the search procedure employed in the subpopulations, it can be used with genetic algorithms or evolutionary strategies for numeric optimization problems. The Multipopulation Cooperative Coevolutionary Programming (MCCP) approach described in this paper extends the concepts of EAGA to symbolic optimization problems that are solved using GP.

3. MULTIPOPULATION COOPERATIVE COEVOLUTIONARY PROGRAMMING

MCCP uses the basic concept of cooperative co-evolution to evolve a set of instructions or rules, represented as trees, to minimize prediction error. Subpopulations collectively search for different alternative solutions, where each subpopulation is guided toward a region in the solution space that is distant from other subpopulations. Information about the location of a subpopulation in the solution space (and therefore the set of common solution-characteristics of a subpopulation) is shared such that the subpopulations cooperate in co-evolving toward different regions of the solution space.

The selection of the surviving solutions in each subpopulation depends upon how well the solutions minimize the prediction error, as well as upon how far they are from the other subpopulations. MCCP is designed to search explicitly for a set of solutions that are as different as possible in the sets of instructions represented by the trees and are within a prediction error target (Eqn. 3). The main steps of the algorithm are described below.

3.1 Algorithmic Steps

Step 1. Create an initial population with P subpopulations (each with a population size of K), where P is the number of alternative solutions being sought. Let SP_p (p=1,..., P) represent the index for subpopulation p. The first subpopulation (SP_1) is dedicated to the search for the tree with the lowest prediction error.

Step 2. In SP_1, evaluate the fitness, or error (Eqn. 1), of each solution, and identify the best solution in the subpopulation with the lowest error. This solution will serve as the benchmark for setting the relaxation constraint Eqn. 3.

Step 3. In SP_p (p=2,..., P), evaluate the error of each individual solution. Solutions that meet the target constraint Eqn. 3 are assigned a feasible flag, and solutions that fail to meet the target are labeled infeasible.

Step 4. For each solution k in subpopulation SP_p ($p \neq 1$), calculate the difference $D^{k,p}$ (defined in Section 3.2) in the solution space between that solution and other subpopulations.

Step 5. Apply an elitism operator to all subpopulations SP_p to preserve the best solution in each subpopulation. In SP_1, the best solution is the solution with the lowest error. In SP_p ($p \neq 1$), the best solution is the feasible solution that is located most distantly in solution space from the other subpopulations. If all solutions in a subpopulation are infeasible, then the best solution is the solution that has the lowest error value.

Step 6. Check for termination criteria. Stop the algorithm if termination criteria (e.g., a maximum number of iterations) are met. Otherwise, go to Step 7.

Step 7. In each subpopulation SP_p, apply a selection operator. In SP_1, the selection is based on how well a solution minimizes error. In SP_p ($p \neq 1$), the selection is based on how well the solution meets the constraint Eqn. 3, as well as on the value of the difference function (as described in Section 5.1).

Step 8. In each subpopulation, apply recombination and mutation operators to the solutions selected in Step 7, and go to Step 2.

3.2 Definition of Difference

The difference function for a solution is based on the distance of that solution to a set of subpopulations. The difference function, $D^{k,p}$, for solution k in subpopulation SP_p is the minimum of the distances between solution k and the other subpopulations SP_q, $q \neq p$. The distance from a solution in one subpopulation to another subpopulation is defined as the average of the distances between that solution and each solution in the other subpopulation. Thus $D^{k,p}$ is expressed as:

$$D^{k,p} = Min\{\frac{\sum_{j=1}^{K} d(X^{k,p}, X^{j,q})}{K}; q = 1, \ldots, P, q \neq p\} \quad (4)$$

where $d(X^{k,p}, X^{j,q})$ is the distance between the two solutions $X^{k,p}$ and $X^{j,q}$, K is the number of solutions in a subpopulation, and P is the number of subpopulations.

For a numeric search problem, the distance d between two vectors of numbers may be easily represented, for example, as the Euclidean distance. The distance between two individual solutions in a tree-based GP is difficult to define, however. A few metrics have been suggested for calculating the distance between two trees. For example, the *edit distance* is based on a dynamic programming algorithm to calculate the minimum number of transformations that will change one tree into the other [19]. Edit distance calculation is computationally intensive and, as a result, typically not used for GP applications. Alternative measures include a metric based on the difference between the symbol or value at each node [8], a node-to-node comparison between trees starting at the root node [7], and a metric that tallies the number of similar subtrees between individuals [13]. Simpler methods, such as a binary distance metric that indicates whether individuals are identical or not [12], and a phenotypic distance based on the difference in the behavior of solutions without consideration of tree structures [18] can also be used.

4. GP-BASED SEARCH FOR CLASSIFIERS FOR A LYMPHOMA DATA SET

Accurate prediction and diagnosis of cancer is important in making appropriate treatment decisions. Because it is difficult to diagnose and predict cancer types, efficient ways to use available information are needed to assist in cancer classification. Large amounts of diagnostic data are generated using the DNA microarray technology. This technique places thousands of genes on a single square inch slide so that the expression of each gene, whether it is turned off or on in a cancer cell, can be visualized. A function is used to convert the visual data into a numeric value to represent the gene expression. The data provided by DNA microarrays can be used to discover rules to classify cancer based on gene expression information. Several approaches, including neural networks, k-nearest neighbor method, support vector machine, and GP [6], have been used to construct rules based on this data.

To diagnose cancer cells, a classifier is constructed as a function of a set of gene expressions, as demonstrated in [11]. The type of cancer may be identified based on the value given by the function, where a positive function value will place the cancer in one class, and a negative value will place the cancer in another class. An example of a sample cell is given in Table 1 with its associated array of genes, represented as $G1, G2, \ldots, G16$, and respective gene expressions. Suppose two classifiers for this type of cancer have been identified, Classifier #1 and #2, which are represented by the trees and the decoded functions shown in Fig. 1. If only Classifier #1 were available, it would be impossible to classify the sample given in Table 1 because the expression of gene $G11$ is not recorded. If Classifier #2 is also available, then this sample could be diagnosed. The availability of alternative rules that use different genes is useful when the expressions of all genes are not recorded for all samples.

The data set used to demonstrate MCCP involves the classification of diffuse large B-cell lymphoma (DLBCL) using a microarray dataset for lymphoma cancer [1] (available at http://llmpp.nih.gov/lymphoma/). DLBCL can typically

Table 1: Sample data for genes and gene expressions

Gene	Expression	Gene	Expression
$G1$	-0.47	$G9$	0.04
$G2$	-0.04	$G10$	-0.35
$G3$	0.98	$G11$	unknown
$G4$	-0.14	$G12$	-0.18
$G5$	-0.07	$G13$	0.07
$G6$	0.35	$G14$	0.60
$G7$	0.67	$G15$	0.23
$G8$	unknown	$G16$	-0.87

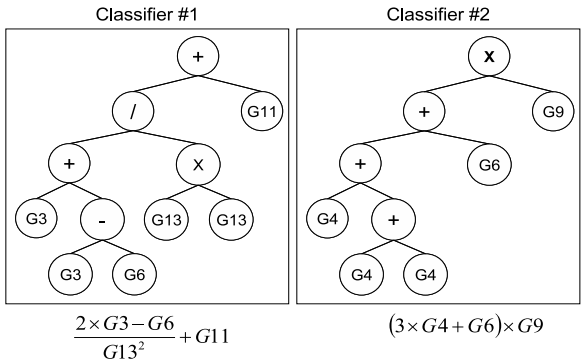

Figure 1: Two sample classifiers

be classified as "activated B-like" or "germinal centre B-like" (GC), where patients with GC B-like DLBCL show a significantly better overall survival than patients with activated B-like DLBCL. The goal is to use the gene expression information to classify the lymphoma, leading to potentially more appropriate treatment decisions. The microarray dataset consists of 24 samples of GC B-like and 23 samples of activated B-like. Each sample consists of 4026 expressed genes.

GP was used by [11] to identify lymphoma classifiers based on a selected set of the genes available in the DLBCL microarray dataset for lymphoma cancer. Signal-to-ratio feature selection was used to choose 30 genes, labeled F1, F2, ..., F30, out of the 4026 genes for classification. A selected set of the available 47 samples was used as the training dataset to calculate the training error when developing the classifiers. Each classifier was represented as a tree, which decodes to a simple arithmetic function, similar to the example demonstrated above. The function was evaluated for each sample, and a sample was classified as active B-like if the function value was negative, and as GC B-like if positive. A correctly classified sample was labeled as a *hit*. The number of misclassifications represents the error, and was calculated using the number of hits out of the training dataset:

$$\text{Error} = samples - hits \tag{5}$$

where *samples* is the size of the dataset. The samples not used for training were used as a validation dataset to assess a classifier's predictive capabilities. The validation error associated with a classifier was calculated using Eq. 5, where *samples* was set as the size of the validation dataset.

The GP approach used by [11] produced first a classifier using the 30 selected genes. An alternative classifier was produced by using an alternative set of functions for building the trees. A third alternative classifier was produced by altering the tree representation to include a weighting at each terminal node [11]. These classifiers were constructed using a leave-one-out cross-validation process based on 46 out of the 47 samples for training and the remaining one for validation. Instead of changing the GP implementation to generate the alternatives, MCCP is applied as described below to generate simultaneously a set of different classifiers.

Table 2: Settings for the MCCP Implementation Described in Section 5.1

Parameter	Setting
No. of Generations	100
No. of Subpopulations (P)	4
Subpopulation Size (K)	200
Function Set	$+$, $-$, $\times$, $\div$, ln, exp
Terminal Set	**R**, Genes {F1, F2, ..., F30}
Max. Initial Depth	8
n_{max}	11
β	5
Selection	Elitist Graduated Overselection
Mutation	5%
Crossover	90%

5. IDENTIFYING A SET OF ALTERNATIVE CLASSIFIERS USING MCCP

5.1 Implementation

MCCP was applied to the lymphoma dataset to identify a set of four maximally different classifiers. The 30 genes selected for use in [11] were used as the set of terminal nodes. Forty-one out of the 47 samples were used as the training dataset, and the training error was defined using Eq. 5 with *samples* set to 41. The validation error corresponds to the prediction error corresponding to the remaining six samples. The target constraint (Eqn. 3) was set as two times the error of the best classifier in Subpopulation 1. The following sections describe the key MCCP operators that were implemented for this problem. The parameter values used in the implementation are given in Table 2.

5.1.1 Initialization

The trees in the subpopulations are initialized by varying the probability of selecting functional or terminal nodes with increasing depth. The probability that a terminal node will be selected at the root node is set to 0.01, and increases linearly to 1.0 at a specified depth (noted as Maximum Initial Depth in Table 2). The terminal set consists of the variables representing genes and a real number (**R**) between zero and one, as shown in Table 2. The initial probability that a unary function will be chosen is 0.1, and the initial probability that a binary function will be chosen is 0.89. The probability of selecting a unary function and the probability of selecting a binary function both decrease linearly to 0.0 at the Maximum Initial Depth.

5.1.2 Penalty Function

Tree sizes tend to grow excessively large in a GP, resulting in long, complex equations for symbolic regression problems.

An additive penalty function was used to limit the number of nodes in a tree and directed the search toward fit solutions with more compact representations. The penalty function was applied as follows to trees consisting of a larger number nodes than a prespecified treshhold, n_{max}:

$$\text{Fitness} = -\text{Error} \times (1 + \beta \times (n - n_{max})) \quad (6)$$

where n is the number of nodes in the tree, n_{max} is the maximum number of nodes allowed in a tree and β is a user-defined constant that represents a relative weight of the penalty associated with the fitness value.

5.1.3 Distance between two trees

Because the purpose of this application is to identify alternative classifiers that use maximally different sets of genes, the distance (d) between two trees was based on the characteristics of terminal nodes only. It was defined as the sum of the number of different genes, or genes not duplicated between two trees. For example, the distance between the two trees represented in Fig. 1 is five.

5.1.4 Selection

The results presented in this paper are based on an implementation of MCCP that uses generational elitism and an elitist graduated overselection strategy [9]. The elitist graduated overselection strategy is used to select solutions for reproduction and crossover. The solutions in a subpopulation are ranked. In subpopulation SP_1, the solutions are ranked based on their error values. For selection in subpopulations SP_p ($p \neq 1$) a subpopulation is partitioned into a group of feasible solutions and a group of infeasible solutions, where the feasibility of a solution is determined using Eqn. 3 (as described in Step 3 in Section 3.1). Feasible solutions are ranked based on the difference function, and infeasible solutions are subsequently ranked, based on error values. A pool of candidate solutions consisting of the five best solutions in the subpopulation is created. A solution is selected from the pool with equal probability for all candidate solutions. Each time a solution is selected, it is replaced, and the next best solution from the ranked vector of solutions is added to the pool. Thus, the pool size grows by one solution each time a solution is selected.

5.1.5 Mutation

A solution selected for mutation undergoes either mutation of real number values at the terminal nodes or tree mutation. The type of mutation a solution undergoes is chosen at random. Real number mutation changes the real numbers in the terminal nodes to a new random number between zero and one with a uniform distribution. Tree mutation generates a new sub-tree to replace the sub-tree at a randomly selected node in the existing tree.

5.1.6 Recombination

For recombination, constrained complexity crossover [20] is used. As described in [20], the complexity of each node in a tree can be calculated such that function nodes are weighted more heavily than terminal nodes, and the complexity of a node is based on the complexity of each node below it. Two trees are randomly chosen for crossover, and a node is selected at random from both trees. The node is defined as the root node of the new sub-tree that will undergo crossover. If the nodes chosen for crossover are within two units of complexity, then the two new sub-trees are swapped. This crossover ensures a level of similarity between sub-trees that undergo crossover, minimizing the disruptive effects of crossover and suppressing excessive code bloat.

5.2 Results

MCCP was executed for 20 random trials, where each trial generated four alternative classifiers. The performance of the algorithm for a single trial is demonstrated in Fig. 2 and Fig. 3. Fig. 2 shows the convergence of each subpopulation toward solutions with low error values (Eqn. 5). In all subpopulations, the lowest error converges in approximately the first 30 generations. The average errors of Subpopulations 2, 3, and 4 stabilize around the target error (Eqn. 3), with respect to the best solution in Subpopulation 1.

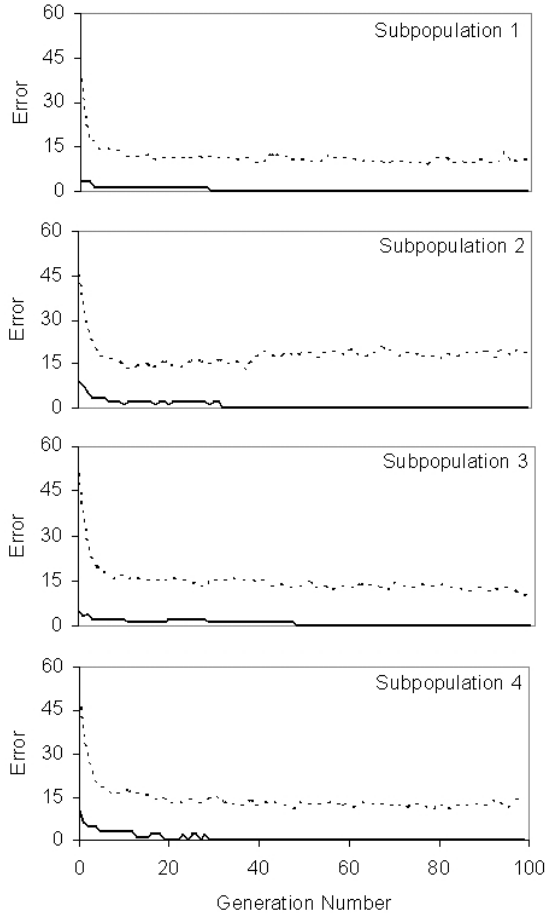

Figure 2: Convergence of the four subpopulations. Bold lines represent the error of the best individual in a subpopulation. Dotted lines represent the average error of the subpopulation.

Fig. 3 demonstrates the behavior of Subpopulations 2, 3, and 4 in maximizing the differences in the trees. Although the difference is relatively large at the beginning of the search, the solutions are not good with respect to the error values (Fig. 2). After the error value is improved in the first 10-15 generations in each subpopulation, the selection pressure is on the improvement of the difference of the

solutions in the subpopulation. Around generation 30, the first subpopulation identifies a solution with an error value of 0.0. The secondary subpopulations minimize the error to meet the tightened constraint, and the average difference of the population drops. Once the secondary subpopulations identify solutions that meet the target error, then the search applies pressure to identify more different solutions, and the average difference improves again.

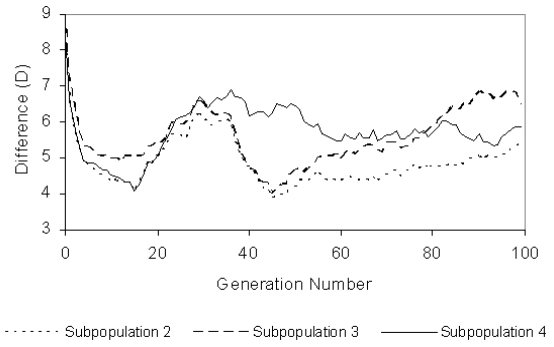

Figure 3: Convergence of Average Difference

The overall performance of the classifiers generated in the 20 random trials is summarized in Fig. 4 and Fig. 5. Fig. 4 shows the average and the range of the prediction performance for the training dataset (of 41 samples). The prediction accuracy represents the percentage of correct hits for the given training dataset used in developing the classifiers. All classifiers show relatively similar average prediction performance, and they vary only slightly in range. Fig. 5 shows the average and the range of the prediction performance for the validation dataset (of six samples). The overall performance of the classifiers is approximately similar when diagnosing the samples in the validation dataset.

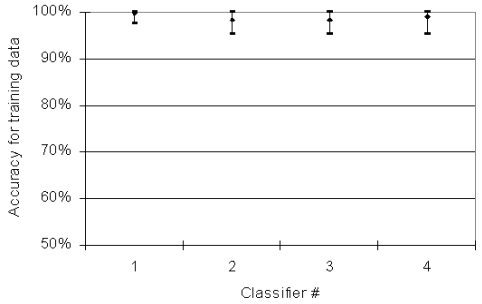

Figure 4: Average and range of prediction accuracy (% of correct hits) based on training dataset for 20 random trials.

A typical set of four classifiers generated by MCCP is shown in Fig. 6 and the corresponding training and validation error values are listed in Table 3. Each of the four classifiers uses a different set of genes to diagnose a lymphoma sample. This may be an advantage when the expression of a gene used in a classifier is unknown or unrecorded. For example, for samples where the gene expression for F1 is not

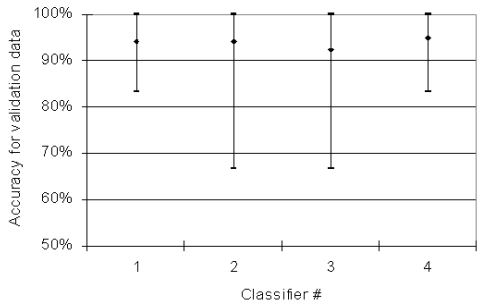

Figure 5: Average and range of prediction accuracy (% of correct hits) based on validation dataset for 20 random trials.

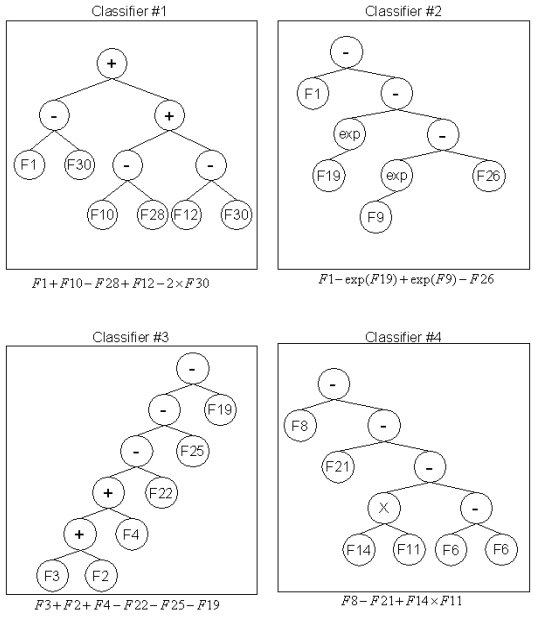

Figure 6: Tree representation of a typical set of four classifiers found by MCCP

available, Classifier #3 or #4, instead of #1 or #2, could be used for diagnosis.

The predicted classifications given by each of the four classifiers for the six validation samples are shown in Table 4. Collectively using the set of alternative classifiers may provide a more robust diagnosis of the data. For example, Sample #3 is classified incorrectly as B-like by Classifier #1, but correctly as GC B-like by Classifiers #2, #3, and #4. Using only the first classifier leads to the wrong classification, while using all four classifiers gives the user some confidence in the classification of the sample as GC B-like.

Among this typical set, classifiers are constructed with as low as four genes in Classifiers #2 and #4 and with as high as six genes in Classifier #3 (Table 5). Only two genes are used in more than one classifier (e.g., $F1$ in Classifier #1 and Classifier #2, and $F19$ in Classifier #2 and Classifier #3). This can be attributed to the explicit operations in MCCP that maximize the differences among the sets of genes used to construct the alternative classifiers. Collectively the four

Table 3: Performance characteristics of a typical set of four classifiers found by MCCP

Classifier #	Function	Training Error (Eqn. 5) (Out of 41 samples)	Validation Error (Eqn. 5) (Out of 6 samples)
1	$F1 + F10 - F28 + F12 - 2 \times F30$	0	1
2	$F1 - exp(F19) + exp(F9) - F26$	0	0
3	$F3 + F2 + F4 - F22 - F25 - F19$	0	0
4	$F8 - F21 + F14 \times F11$	0	0

Table 4: Performance based on the six validation samples for the four classifiers shown in Fig. 6

Sample Number	Sample Name	True Classification	Predicted classification by Classifier #1	#2	#3	#4
1	SUDHL6	GC B-like	GC B-like	GC B-like	GC B-like	GC B-like
2	DLCL-0010	GC B-like	GC B-like	GC B-like	GC B-like	GC B-like
3	DLCL-0020	GC B-like	B-like	GC B-like	GC B-like	GC B-like
4	OCI Ly3	B-like	B-like	B-like	B-like	B-like
5	DLCL-0014	B-like	B-like	B-like	B-like	B-like
6	DLCL-0036	B-like	B-like	B-like	B-like	B-like

classifiers use a total of 17 different genes out of the 30 that were considered for building the functions (Table 5). This indicates that no one gene is dominantly associated with the classification of this sample dataset. More different classifiers may be found if more subpopulations are used by appropriately setting a larger value for the MCCP parameter P.

6. FINAL REMARKS

MCCP is structured to explore for a set of good designs that constitute unusual and dissimilar combinations of design variable values. Operators in MCCP are introduced to co-evolve subpopulations toward maximally different regions (or niches) in the solution space. The subpopulations cooperatively identify a set of designs with the most dissimilar characteristics, representing potentially creative and innovative designs. These alternative designs, which perform similarly well, are not biased by prior or existing solutions, unlike incrementally different designs generated by humans. Thus, MCCP can help generate innovative and competent designs in a systematic manner.

Results were presented from an illustrative application of MCCP to identify a set of classifiers for diagnosing the type of cancer based on gene expression information. As all gene expression information may not be recorded for each sample, a classifier that is based only on a few genes may not be robust. By exploring for alternative classifiers that are based on as many different genes as possible, a set of innovative classifiers with a more robust diagnostic performance was generated by MCCP. Each of these classifiers uses at most only one gene that is common to another, and collectively they use about 57% of the available genes to cover a broad spectrum of the available gene expression information. Consequently, the collective classification performance shows a more robust diagnosis of the validation data set.

MCCP is structured in a generic manner to enable its application with any GP-based design procedure. Thus, it can be readily used to extend existing GP implementations for

Table 5: Genes used in the construction of the set of typical classifiers

Gene	Gene Ref. Number [1]	Classifier #1	#2	#3	#4
$F1$	75	X	X		
$F2$	2439			X	
$F3$	2417			X	
$F4$	2244			X	
$F5$	2438				
$F6$	2412				
$F7$	2205				
$F8$	2243				X
$F9$	682		X		
$F10$	2416	X			
$F11$	2436				X
$F12$	2437	X			
$F13$	2415				
$F14$	2206				X
$F15$	2263				
$F16$	1276				
$F17$	1277				
$F18$	1279				
$F19$	1281		X	X	
$F20$	1278				
$F21$	1317				X
$F22$	1291			X	
$F23$	1275				
$F24$	1280				
$F25$	1321			X	
$F26$	1316		X		
$F27$	1315				
$F28$	1320	X			
$F29$	1284				
$F30$	1312	X			

[1] As reported in [1]

a variety of problems, e.g., in the discovery of alternative functions to model data representing natural physical processes, in developing rules or instructions for robot designs, and in generating creative designs for electrical circuits or computer programs. Further investigation is needed to explore the full potential of MCCP in generating innovative human-competitive designs.

7. ACKNOWLEDGEMENTS

The research leading to the reported results was supported partly by grants from the National Science Foundation (BES-0312841) and the US Environmental Protection Agency (R82946101). The viewpoints presented here are those of the authors and do not necessarily reflect those of the funding agencies. The authors would like to thank Jin-Hyuk Hong at Yonsei University for providing the lymphoma dataset.

8. REFERENCES

[1] A. A. Alizadeh, M. B. Eisen, R. E. Davis, C. Ma, I. Lossos, A. Rosenwald, J. Boldrick, H. Sabet, T. Tran, X. Yu, J. I. Powell, L. Yang, G. E. Marti, T. Moore, J. Hudson, Jr., L. Lu, D. B. Lewis, R. Tibshirani, G. Sherlock, W. C. Chan, T. C. Greiner, D. D. Weisenburger, J. O. Armitage, R. Warnke, R. Levy, W. Wilson, M. R. Grever, J. C. Byrd, D. Botstein, P. O. Brown, and L. M. Staudt. Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling. *Nature*, 403(6769):503–511, 2000.

[2] E. D. Brill, Jr. Use of optimization models in public-sector planning. *Management Science*, 25(5):413–422, 1979.

[3] E. D. Brill, Jr., S.-Y. Chang, and L. D. Hopkins. Modeling to generate alternatives: The HSJ approach and an illustration using a problem in land use planning. *Management Science*, 25(2):221–235, 1982.

[4] E. D. Brill, Jr., J. M. Flach, L. D. Hopkins, and S. Ranjithan. MGA: A decision support system for complex, incompletely defined problems. *IEEE Transactions on Systems, Man, and Cybernetics*, 20(4):745–757, 1990.

[5] S.-Y. Chang, E. D. Brill, Jr., and L. D. Hopkins. Use of mathematical models to generate alternative solutions to water resources planning problems. *Water Resources Research*, 18(1):58–64, 1982.

[6] S.-B. Cho and H.-H. Won. Machine learning in DNA microarray analysis for cancer classification. In *Proceedings of the First Asia-Pacific Bioinformatics Conference (APBC 2003)*, pages 193–206. Australian Computer Society, Inc., 2003.

[7] E. D. de Jong, R. A. Watson, and J. Pollack. Reducing bloat and promoting diversity using multi-objective methods. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 11–18. Morgan Kaufmann Publishers, 2001.

[8] A. Ekart and S. Nemeth. Maintaining the diversity of genetic programs. In *Proceedings of the 5th European Genetic Programming Conference*, pages 162–171. Springer-Verlag, 2002.

[9] T. Fernandez and M. Evett. The impact of training period size on the evolution of financial trading systems. In *Genetic Programming 1997: Proceedings of the Second Annual Conference*. AAAI Publishers, 1997.

[10] L. Harrell. *Methods for Generating Alternatives to Manage Water Quality in Watersheds*. PhD thesis, North Carolina State University, Raleigh, NC, USA, 1998.

[11] J.-H. Hong and S.-B. Cho. Lymphoma cancer classification using genetic programming with SNR features. In *Genetic Programming: Proceedings of the 7th European Conference, EuroGP 2004*, pages 78–88. Springer, 2004.

[12] J. Hu, K. Seo, S. Li, Z. Fan, R. Rosenberg, and E. Goodman. Structure fitness sharing (SFS) for evolutionary design by genetic programming. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 780–787. Morgan Kaufmann Publishers, 2002.

[13] M. Keijzer. Efficiently Representing Populations in Genetic Programming. In *Advances in Genetic Programming: Volume II*, pages 259–278. MIT Press, 1996.

[14] J. R. Koza. *Genetic Programming*. The MIT Press, Cambridge, Massachusetts, 1986.

[15] J. R. Koza. *Genetic Programming IV*. The MIT Press, Cambridge, Massachusetts, 2004.

[16] D. H. Loughlin, S. Ranjithan, J. W. Baugh, and E. D. Brill, Jr. Genetic algorithm approaches for addressing unmodeled objectives. *Engineering Optimization*, 33(5):549–569, 2001.

[17] S. W. Mahfoud. *Niching Methods for Genetic Algorithms*. PhD thesis, University of Illinois at Urbana-Champaign, Urbana-Champaign, Illinois, USA, 1995.

[18] R. McKay. Fitness sharing in genetic programming. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 435–442. Morgan Kaufmann Publishers, 2000.

[19] U.-M. O'Reilly. Using a distance metric on genetic programs to understand genetic operators. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, Computational Cybernetics and Simulation, Vol. 5*, pages 4092–4097. IEEE Computer Society Press, 1997.

[20] A. H. Watson and I. C. Parmee. Improving engineering design models using an alternative genetic programming approach. In *Proceedings of the International Conference on Adaptive Computing in Design and Manufacture*, pages 193–206. Springer-Verlag, 1998.

[21] E. M. Zechman and S. Ranjithan. An evolutionary algorithm to generate alternatives (EAGA) for engineering optimization problems. *Engineering Optimization*, 36(5):539–553, 2004.