Prefix Gene Expression Programming

Xin Li[1], Chi Zhou[2], Weimin Xiao[2], Peter C. Nelson[1]

[1]Artificial Intelligence Laboratory
University of Illinois at Chicago
Chicago, IL 60607, USA
+01-312-4134075
{xli1,nelson}@cs.uic.edu

[2]Physical Realization Research Center
of Motorola Labs
Schaumburg, IL 60196, USA
+01-847-5761470
{Chi.Zhou, awx003}@motorola.com

ABSTRACT

Gene Expression Programming (GEP) is a powerful evolutionary method derived from Genetic Programming (GP) for model learning and knowledge discovery. However, when dealing with complex problems, its genotype under Karva notation does not allow hierarchical composition of the solution, which impairs the efficiency of the algorithm. We propose a new representation scheme based on prefix notation that overcomes the original GEP's drawbacks. The resulted algorithm is called Prefix GEP (P-GEP). The major advantages with P-GEP include the natural hierarchy in forming the solutions and more protective genetic operations for substructure components. An artificial symbolic regression problem and a set of benchmark classification problems from UCI machine learning repository have been tested to demonstrate the applicability of P-GEP. The results show that P-GEP follows a faster fitness convergence curve and the rules generated from P-GEP consistently achieve better average classification accuracy compared with GEP.

Categories and Subject Descriptors

[**Genetic Programming**]: Novel representations, algorithm design, and theory.

Keywords

Polish notation, Gene expression programming, genotype-phenotype mapping mechanism, and schema theorem.

1. INTRODUCTION

First introduced by Candida Ferreira [1], Gene Expression Programming (GEP) is improved from Genetic Programming (GP) as a new technique for the creation of computer programs denoting the learned models or discovered knowledge. In GEP, computer programs are represented as linear character strings of fixed-length (called *chromosomes*) which, in the subsequent fitness evaluation, can be expressed as expression trees (ETs) of different sizes and shapes. The separation of genotype and phenotype has endowed GEP with more flexibility and power of exploring the entire search space compared with traditional GP. GEP methods have performed well for solving a large variety of problems, including symbolic regression, optimization, time series analysis, classification, logic synthesis and cellular automata, etc. [2, 4, and 6].

However, the learning procedure of GEP can be improved upon when dealing with complex problems with respect to both time efficiency and solution quality. The biological evolutionary process has revealed the principle of evolving from a self-contained functional single cell to a well-developed entity with numerous specialized components. We are naturally inspired to assume that solutions to complex problems might be built up incrementally from simpler elements. Although the phenotype of expression trees in GEP has retained the structural representation from GP, the linear representation of the genotype conforms to Karva notation [1], under which the genotype-phenotype mapping mechanism does not guarantee that the levels of functional complexity in the phenotype are also directly reflected in the genotype. Since it is the genotype that is subject to the different genetic operations, it is difficult to follow the approach of incrementally forming solutions with the original GEP. Moreover, an evolved good functional structure is very likely destroyed in the subsequent generations not only by mutations but also by crossovers and rotations, which may require much additional computation to recover before an optimal solution is found. Therefore, a more structure friendly genotype is needed to assure the direct linkage among the substructure pieces corresponding to the sub-trees in the phenotype.

In this paper, we propose Prefix Gene Expression Programming (P-GEP), which adopts a new genotype representation scheme and consequently a new genotype-phenotype mapping mechanism following the convention of prefix notation. As distinguished from the Karva notation, P-GEP enjoys the natural contiguousness of substructure components in its genotype representation. This is a fundamental enhancement to the original GEP and it specifically overcomes the aforementioned drawbacks of GEP. We believe P-GEP benefits the evolution in terms of the convergence of a good functional structure.

An artificially structured symbolic regression problem and a set of benchmark classification problems from the UCI machine learning repository [12] have been tested to demonstrate the applicability of P-GEP. The results show that P-GEP follows a faster fitness convergence curve and the rules generated from P-GEP consistently achieve better average classification accuracy than those of GEP.

The next section of this paper gives an overview of related work. Section 3 explains the P-GEP algorithm and its major characteristics. The experiments and qualitative discussion of the results are covered in section 4. Section 5 presents some conclusions and ideas for future work.

2. RELATED WORK AND MOTIVATIONS

2.1 Brief Overview of Gene Expression Programming

As is the case with GP, when using GEP to solve a problem, generally five components, i.e., the function set, terminal set (including problem-specific variable names and pre-selected constants), fitness function, GEP control parameters, and stop condition need to be specified. Some details are given as below:

(1) The GEP chromosomes, expression trees (ETs) and the mapping mechanism.

Each chromosome in GEP is a character string of fixed-length, which can be composed of any element (also called *gene*) from the function set or the terminal set. Using the elements {+, -, *, /, sqrt} as the function set and {a, b, c, d, 1} as the terminal set, the following is an example GEP chromosome of length fifteen:

sqrt.*.+.*.a.*.sqrt.a.b.c./.1.-.c.d (2.1)

where "." is used to separate individual genes; *sqrt* denotes the square-root function; *1* is a numeric constant; and *a*, *b*, *c*, *d* are variable (or attribute) names. The above is referred to as *Karva* notation, or *K-expression* [1].

A K-expression can be mapped into an ET following a width-first procedure. A branch of the ET stops growing when the last node in this branch is a terminal. For example, the ET shown in Figure 1 corresponds to the sample chromosome (2.1), and can be interpreted in a mathematical form as (2.2).

The conversion of an ET into a K-expression is also very straightforward, and can be accomplished by recording the nodes from left to right in each layer of the ET in a top-down fashion to form the string.

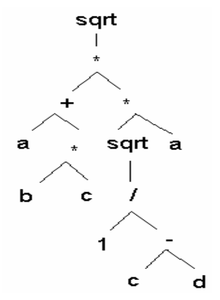

Figure 1. Example of GEP Expression Tree.

$$\sqrt{(a + bc) * a\sqrt{\frac{1}{c - d}}} \tag{2.2}$$

It is important to notice that in GEP all of the chromosomes are of a fixed length, but the size of expression trees can vary. This is because that not every chromosome coincides with a valid K-expression. Very frequently, a chromosome may have some redundant elements which are useless for the chromosome-ET mapping. For example, (2.3) is also a chromosome of length fifteen, however only the underlined part of it forms a valid K-expression which subsequently maps into the ET of size eleven shown in Figure 2. In some other cases, a chromosome may not be able to match any valid K-expression within its length due to the lack of terminals at the end to finish the construction of the ET. In order to guarantee that only legal expression trees are generated, we have applied a validity test proposed by Zhou, et al. [3, 5] to dynamically check if a chromosome is able to encode a legal expression tree within the size limit (for details please refer to [3]) in stead of the original head-tail method [2]. All of the chromosomes randomly generated or reproduced by genetic operators are subject to this test to prevent illegal expressions from being introduced into the population.

$\underline{\text{sqrt.*.+.*.a.*.sqrt.a.b.c.d}}$.1.-.c./ (2.3)

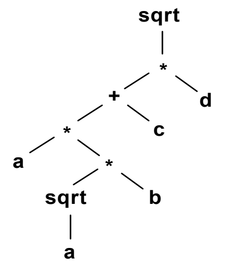

Figure 2. Example of GEP Expression Tree with a size smaller than the chromosome length.

(2) The description of the GEP Algorithm.

The GEP algorithm begins with the random generation of linear fixed-length chromosomes for individuals of the initial population. The chromosomes are represented as expression trees, and the fitness of each individual is evaluated based on a pre-defined fitness function. The individuals are then selected by fitness to reproduce with modification. The individuals of this new generation are, in their turn, subject to the same developmental process, i.e., expression as chromosomes, confrontation in the selection environment, and reproduction with modification. This process is repeated for a pre-specified number of generations or until a solution has been found. In GEP, individuals are often selected and copied into the next generation based on their fitness, as determined by roulette-wheel sampling with elitism [7], which guarantees the survival and cloning of the best individual to the next generation. Variation in the population is introduced by applying one or more genetic operators to selected chromosomes, including:

• Crossover, in which two parent chromosomes are randomly chosen and paired to exchange some elements between them. There are two kinds of crossover: one-point and two-point crossover, working in the same fashion as in the canonical GAs [3].

• Mutation, in which the symbols at any position in a chromosome are subject to a random change according to a certain probability.

• Rotation, in which two subparts of the element sequence in a chromosome are rotated with respect to a randomly chosen point.

Note that all of these operations upon the coding sequence of a chromosome usually drastically reshape the corresponding ET.

2.2 Motivations from Biological Evolution, GAs and GP

Complex data mining tasks are characterized by large data sets, high dimensional feature sets and non-linear forms of hidden knowledge within the data. It is very appealing to solve these problems with GEP since the nature of the algorithm fits the challenges well. However, the margin for improving the algorithm is also noticeable due to observations from the following aspects:

(1) Biological evolution

As perceived from the biological evolution in nature, a complex creature is usually evolved from a simple but self-contained functional single cell, and the subparts of this individual are later further specialized or adapted as the response to changes in the environment. This is virtually the modern evolution theory based upon Darwin's theory of natural selection [8]. Since GEP mimics the biological evolutionary process, an analogical hypothesis is that solutions to complex problems can be built up from simpler elements. To achieve this purpose, a structure-preserving representation with respect to genetic operators is in need.

However, the Karva notation of original GEP has the weakness in this aspect because the genes corresponding to the functionally independent sub-trees in the expression tree are not mapped as contiguous segments in the linear chromosome. As the consequence, the solution structures are very fragile when subject to genetic operations, not only by mutations (which are usually intended for producing genetic diversity) but also by crossovers and rotations (which are more generally intended for inheriting the genetic traits from previous generations). The evolutionary process does not well benefit from the explored individuals in terms of reusing the evolved good substructure components in other candidates to form the solution incrementally.

(2) Genetic algorithms (GAs)

The general philosophy of GEP can be traced back to GAs which are also the ancestor of GP. More importantly, GEP shares the linear genotype of fixed-length as in GAs. As one of the foundations of GAs, the schemata theory was developed to analyze how the GAs work [9]. Defined as some patterns of the chromosomes, schemata are used to represent parts of the search space. It was then proved that the evolution of GAs works by continuing to explore the areas of the search space that are likely to produce higher levels of fitness. Following this, when a GA explores new candidate solutions, it should ideally keep the search within the space confined by the fitter schemas. In other words fitter schemata are those that should be transmitted between generations. This subsequently derives the building blocks hypothesis based on the schemata and introduces the linkage (i.e., building block disruption) problem [11]. The performance of GAs would presumably be improved if functionally related bits are more likely stay together in the string under crossover [3].

(3) Genetic Programming (GP)

GP gains its success by taking the tree representation to increase the complexity of the structures undergoing adaptation [10]. At the same time, since the genetic operations are defined upon the sub-trees, the elementary solution structure components are not easily destroyed and over time, this concentrates the search of the solution space into the subspace of ever-decreasing dimensionality and ever-increasing fitness. Meanwhile, GP has tailored the schemata theorem for its tree structured individuals to analyze the performance of the algorithm, where a schema is defined as a set of specified sub-trees [10]. Therefore the overall effect of fitness-proportionate reproduction and crossover is that sub-trees from relatively high-fitness programs are used as building blocks for constructing new individuals in an approximately near-optimal way. These ideas have also been extended into different versions of schema theorems for different derivations of the GP algorithm [13, 14, and 15].

However, the tree representation scheme makes the algorithm difficult to implement and easily trapped by the explosively growing tree size during the evolution. It was also recognized that due to the variable length of the GP representation, none of the existing formulation of a GP schema theorem predicts with any certainty that good schema will propagate during a GP run [16].

All of above has motivated us to develop a new genotype representation for GEP to incorporate the complexity hierarchy of solutions into its flexibility in evolution.

3. POLISH NOTATION BASED GENE EXPRESSION PROGRAMMING

3.1 Genotype, Phenotype and Mapping Mechanism of P-GEP

The major contributions of P-GEP are the adoption of a new linear genotype representation in prefix notation and a resulted different mapping mechanism between its genotype and phenotype. To make it clear, we will use the example from Section 2.1 to illustrate the concepts of genotype, phenotype and the mapping mechanism under P-GEP as well as their differences from GEP.

First of all, the phenotype of P-GEP is the same as in GEP, namely, an expression tree. Secondly, in P-GEP, the genotype is still a linear character string of fixed-length. However, the mapping mechanism between genotype and phenotype conforms to the convention of prefix notation expressions. Therefore, given the same chromosome, the corresponding ET is formed following the preorder instead of width-first fashion as in GEP. For the chromosome of length fifteen in example (2.1), P-GEP will map it into a dramatically different ET as shown in Figure 3. As the result, the arithmetic expression translated from this ET also represents a completely different functionality as shown in (3.1).

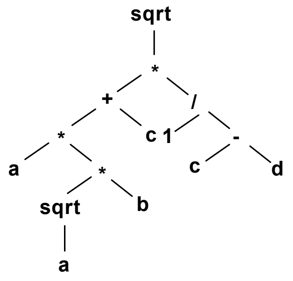

Figure 3. Example of P-GEP Expression Tree

$$\sqrt{(a\sqrt{a}b+c)(\frac{1}{c-d})} \qquad (3.1)$$

On the other hand, the conversion from an ET to a chromosome is also straightforward as reading the tree nodes in preorder. For the expression tree as shown in Figure 1, the corresponding P-GEP chromosome of length fifteen is:

$$\text{sqrt.*.+.a.*.b.c.*.sqrt./.1.-.c.d.a} \quad (3.2)$$

As is the case in GEP, all of the chromosomes in P-GEP are of a fixed length, but the size of expression trees can vary. It is still possible that a chromosome may have some redundant elements which are not useful for the chromosome-ET mapping, or a chromosome does not map to a legal expression tree, so the validity test applies here as well. For the evolutionary process of the algorithm and the genetic operators, P-GEP follows the same definitions as in GEP.

3.2 Major Characteristics of P-GEP

3.2.1 Substructure Preserving Characteristics

In P-GEP, the genotype has a much more direct correlation with its phenotype with respect to the formation of the solution's functionality. Nodes from the same sub-tree of an expression tree appear adjacent to each other on the matching character string, and subroutines of the solution can therefore be easily recognized as segments of the chromosome. This equivalently means that P-GEP solution substructures have a tighter genetic linkage, which results in less destructive crossover and rotation operations compared with GEP. These substructure preserving characteristics can be illustrated by examining the crossover and rotation operators in P-GEP with the example chromosomes of (2.1) and (3.2), which is listed in Table 1 (underscores are used to identify the crossover and rotation points).

Table 1. Genetic Operators for the P-GEP algorithm

Genetic Operators	One-point crossover	Initial Chrom	sqrt.*.+.*.a.*.sqrt.a.b.c./.1.-.c.d (1.1)
			sqrt.*.+.a.*.b.c.*.sqrt./.1.-.c.d.a (3.2)
		Offspring	sqrt.*.+.*.a.b.c.*.sqrt./.1.-.c.d.a (3.3)
			sqrt.*.+.a.*.*.sqrt.a.b.c./.1.-.c.d (3.4)
	Two-point crossover	Initial Chrom	Sqrt.*.+.*.a.*.sqrt.a.b.c./.1.-.c.d (1.1)
			sqrt.*.+.a.*.b.c.*.sqrt./.1.-.c.d.a (3.2)
		Offspring	sqrt.*.+.*.a.b.c.*.sqrt.c./.1.-.c.d (3.5)
			sqrt.*.+.a.*.*.sqrt.a.b./.1.-.c.d.a (3.6)
	Rotation	Initial Chrom	sqrt.*.+.*.a.*.sqrt.a.b.c./.1.-.c.d (1.1)
		Offspring	*.+.*.a.*.sqrt.a.b.c./.1.-.c.d.sqrt (3.7)

For the one-point crossover example, the resulting offspring (3.3) can be mapped into the ET shown in Figure 4. Compared with the ET shown in Figure 3, it is observed that crossover has also reshaped the ET as is the case in GEP, however not that drastically since we see the subroutine of 1/(c-d) remain intact in both the initial chromosome and the offspring, except for its relocation in the solution structure. Similar observations are available for other offspring (3.4), (3.5), (3.6) and (3.7). Thus in P-GEP, crossover and rotation operators can better serve the purpose of passing genetic material from one generation to another as intended. The sub-components of fittest individuals are more likely preserved and transmitted than in GEP. This potentially leads to a faster convergence of the solution structure and finally an optimal solution for P-GEP. This is further examined in section 4.2 with some in-depth discussion and theoretical analysis.

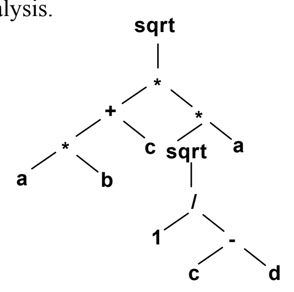

Figure 4. Example of P-GEP Expression Tree produced by one-point crossover.

3.2.1 Inherent Hierarchy in Forming the Solution

The genotype based on prefix notation in P-GEP makes the structural hierarchy of the solution not only apparent in expression tree, but also inherent within the linear character strings. In both representations, the hierarchies are formed incrementally and a higher-level structure is built upon the lower ones naturally. This can be clearly illustrated with the example chromosome (2.1) and its corresponding P-GEP ET in Figure 3. Both are redrawn in Figure 5. In the ET representation, root nodes of each valid sub-tree are annotated with circled numbers marking their hierarchy (i.e., modularity levels) within the tree. A smaller number refers to a lower hierarchy and the hierarchy of a sub-tree is always higher than that of any one of its component sub-trees. These numbers also accompany the matching character segments (called *substructures*) in the linear string representation, which are identified with underlines.

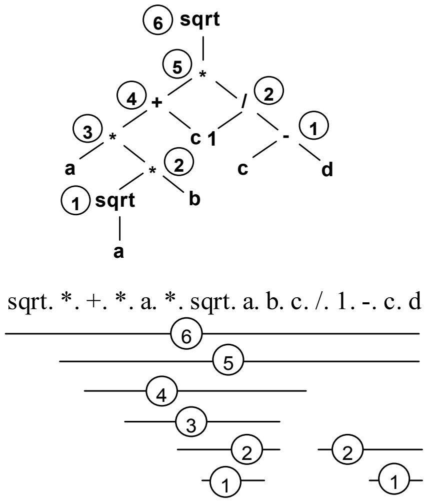

Figure 5. Illustration of inherent hierarchy in genotype and phenotype of P-GEP.

Figure 5 conveys another important feature of P-GEP: the genotype and phenotype encode the functional complexity in a similar way. Once the root of the sub-tree is found, it is equally convenient to pick up the whole functional branch either in the ET representation or the linear chromosome representation. In other words, the linear chromosome is no less expressive than the ET in terms of the functionality of the solutions they both encode. This is another significant advantage of P-GEP since now the tree representation is actually not necessary for implementing the genetic operations and fitness evaluations. Operating on the linear chromosome is much more efficient than working on the tree structures. However, we keep the ET representation for the clarity of the illustration on the solutions.

4. EXPERIMENTS AND DISCUSSIONS

4.1 Experiments and Results

In order to justify the advantage of P-GEP as compared to GEP, we first experimented on the two algorithms with an artificially structured symbolic regression problem. Later a suite of classification tests were conducted on the four testing datasets that are cited from the UCI Machine Learning repository.

(1) A simple symbolic regression problem.

The simple symbolic regression problem used for the experiment is shown in (4.1). Since the final solution has inherently defined substructures and hierarchy, evolutionary processes with emphasis in preserving substructure components should enjoy some benefits under our assumption. A set of twenty-one fitness cases equally spaced along the x axis from -10 to 10 were chosen for the tests.

$$y = 3 * (x+1)^3 + 2 * (x+1)^2 + (x+1) \qquad (4.1)$$

The general experiment setup is summarized as follows: the chromosome size is 128; the population size is 1000; the maximum number of generations is 500; the crossover probability is set as 0.7, and the mutation and rotation probability is set as 0.02; the function and terminal sets are selected as {+, -, *, /} and {1, 2, 3, 5, 7} respectively. Each experiment has been run for ten times and the results were averaged.

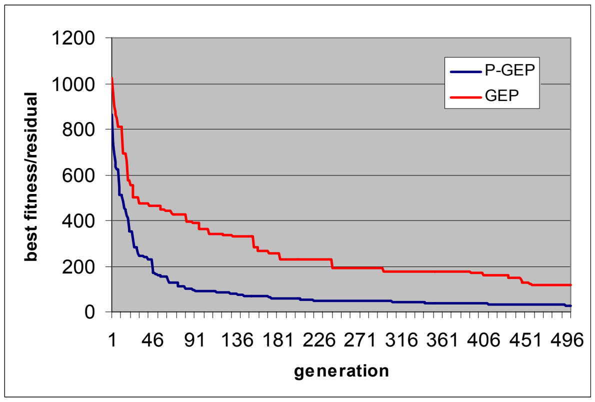

Figure 6. The fitness convergence curves of P-GEP vs. GEP for the given symbolic regression problem.

Since the ultimate goal for both techniques is to find the optimal solution effectively, we decided to examine the fitness convergence curves revealed by every evolutionary process. In our experiments for this symbolic regression problem, the fitness of a solution is defined as the residual, which is better when smaller. The best fitness value among all of the individuals was recorded at every generation for all of the ten runs, and the average is plotted against the generation in Figure 6 for both GEP and P-GEP.

The curves show that generally in the very early generations, both evolutionary processes experience a quick reduction in their best residual values. However, GEP seems to have comparably much higher residual values than P-GEP for the same generation. This indicates that the substructure preserving characteristics of P-GEP actually do improve its problem solving ability in this intentionally structuralized problem. Since in quite a lot of situations, the problem solutions have underlying structures, P-GEP is promising in fulfilling these tasks.

(2) Classification tests.

In our previous benchmark classification testing for GEP in comparison with traditional machine learning methods and canonical tree-based GP classifiers, GEP has already exhibited better performance in terms of the classification accuracy, compactness of evolved rules and running time (for details please refer to [3, 5]). Therefore, here we focus on testing the performance improvement of P-GEP over P-GEP in executing classification tasks.

The datasets for classification tests are summarized in Table 2. The experiment setting was as follows: the chromosome size is 80 for iris, 128 for abalone and 100 for the rest; the population size is 1000; the maximum number of generations is 1000; the crossover probability is set as 0.8, and the mutation and rotation probability is set as 0.02; the functional and terminal sets are selected as {IF, +, -, *, /, sqrt} and {1, 2, 3, 5, 7} respectively. We have applied the five-fold cross validation method for the testing of iris, zoo and wine. For abalone, we followed the past usage of the data set to separate the first 3133 cases into the training set and the remaining 1044 into the testing set. Finally, each experiment has been repeated for five independent runs and the results were averaged.

Table 2. Summary of the classification data for P-GEP vs. GEP

Dataset	Description
Iris	4 numeric attributes; 150 cases; 3 classes with class distribution {50, 50, 50}
Zoo	15 boolean and 1 numeric attributes; 101 cases; 7 classes with class distribution {41, 20, 5, 13, 4, 8, 10}
Wine	13 numeric attributes; 178 cases; 3 classes with class distribution {59, 71, 48}
Abalone	1 nominal and 7 numeric attributes; 4177 cases; 3 classes with class distribution {1407, 1323, 1447}

Table 3 summarizes the experimental results, where *average* refers to the average of the averaged classification accuracy for each run of five-fold cross validation over the total five runs (except that for abalone, it directly shows the general average classification accuracy over the five runs) and *best* refers to the average of the classification accuracy over the five folds in the best run (except that for abalone, it directly shows the best testing classification accuracy out of the five runs). Each number (except the two in the entries of best for abalone) shown in the table is in terms of percentage and associated with its 95% confidence interval.

As seen from Table 3, for the average values over the five runs, P-GEP has consistently achieved a higher classification accuracy value and a smaller confidence interval in all of the tests. We further performed the one-tailed t-tests with an unequal variance assumption and a 0.05 significance level. The results suggest there are statistically significant differences between the average classification accuracy values, and therefore P-GEP outperforms GEP for the selected classification problems in this sense. Regarding the best classification accuracy (averaged over five folds) out of the five runs, although for iris and abalone datasets P-GEP has yielded apparently better values, in general both methods are just comparable and the differences are not statistically significant under the t-tests. This is understandable as both P-GEP and GEP are very flexible by adopting the linear genotype representation, and therefore the algorithms could ultimately approximate equally good solutions. However, with the structure-preserving representation, P-GEP can quickly converge to a better solution than GEP given the same amount of resources.

Table 3. Summary of the test results for P-GEP vs. GEP

Dataset	Average		Best	
	P-GEP	GEP	P-GEP	GEP
Iris	**96.0±0.7**	93.7±1.4	**96.7±4.1**	95.3±5.7
Zoo	**94.4±0.5**	93.0±0.6	95.0±6.2	94.0±5.7
Wine	**92.2±2.6**	89.8±2.8	95.5±4.5	93.3±3.7
Abalone	**56.7±3.4**	50.1±4.1	**59.9**	53.6

4.2 Theoretical Discussion

The experimental results have shown that the performance of P-GEP fits into our hypothesis about its characteristics presented in section 3. Since P-GEP combines the representation advantages of both GAs and GP, we are inspired to conduct a similar schema analysis on how P-GEP forms the solution incrementally by composing useful subcomponents together. Because each valid individual in the search space of P-GEP corresponds to an expression tree, the search space of P-GEP can be represented by all of the possible expression trees constructed with the pre-selected functional and terminal sets within the specified chromosome length. It is then convenient to define schema in P-GEP in terms of substructures, each of which is equivalently a complete sub-tree in the expression tree representation. The main activity of the search process of P-GEP is to find an optimal combination of these schemata.

Currently we define a *schema* in P-GEP as a set of substructures. There are two more restrictions to this definition. The first restriction is that the substructures in a schema are intended as non-overlapping ones in the instance chromosome (however they can be identical ones as long as they reside in different positions within the chromosome) and as a result the sum of their lengths does not exceed the predetermined chromosome length. Secondly, each substructure is associated with its starting position in the chromosome, so the schema is position dependent. Note that under this definition, every individual in the P-GEP search space actually belongs to at least one schema among all the possible ones which is the one defined by the expression tree encoded by the individual itself.

Following the terminologies used in the schema theorem of GAs, *the defining length δ(H)* of a schema H in P-GEP is the distance between the outmost elements of all the specified substructures in this schema, and *the number of specified positions O(H)* is the number of the positions in the chromosome that have specified elements by the schema, whose value is the sum of the lengths of the substructures in the schema. However, *the order of the schema* is defined as the maximum hierarchical level among all of the substructures included in the schema. We can illustrate these concepts with the example chromosome (2.1). Table 4 has included some of the possible schemata that chromosome (2.1) could be an instance of, along with their respective defining length and order. It is not difficult to understand that a schema composed of lower-order defining substructures that are physically close to each other will have a smaller defining length. Besides the extreme cases of schemata composed of a single terminal, those defined by a single primitive substructure (composed of only one operator and the corresponding parameters) also tend to have the smallest defining length.

Table 4. Illustration of schema, its defining length and order

Schema	δ(H)	O(H)	Order
{-.c.d: 13}	2	3	1
{*.sqrt.a.b: 6, d: 15}	9	5	2
{a: 5; a: 8; b: 9;c: 10; c: 14}	9	5	0
{*.+.*.a.*.sqrt.a.b.c./.1.-.c.d: 2}	13	14	5

For a given problem, if the chromosome length is set as L, then the (L-1) interstitial positions are uniformly chosen at random for the crossover operation, out of which δ(H) will cause the disruption of schema H. Moreover, there's a probability of p_c that the chromosome is subject to crossover. In general the probability ε_c of the disruption of a schema H due to the crossover is approximately:

$$\varepsilon_c = p_c \frac{\delta(H)}{L-1} \tag{4.1}$$

For the purpose of simplification, only one-point crossover is considered in (4.1). The result for two-point crossover could be similarly derived.

Since the rotation operation will always yield changes in the starting positions of the substructures, the probability ε_r of the disruption of a schema H due to the rotation is approximately equivalent to p_r which is the probability that a chromosome undergoes the rotation operation:

$$\varepsilon_r = p_r \tag{4.2}$$

There is an additional note about rotation, i.e., although this operation is very destructive to the schemata under our definition, it is still likely to protect the substructures defining the schema as long as the rotation point is not chosen as a specified position of the schema. These substructures may be reorganized by the evolutionary process to discover new useful schemata, which is better than forming candidates all from primitive genes.

At the same time, every position in the chromosome has a mutation probability of p_m. Only when the positions specified by the schema are mutated, can the schema be destroyed. Consequently, the probability ε_m of the disruption of a schema H due to mutation is approximately:

$$\varepsilon_m = 1-(1-p_m)^{O(H)} \approx p_m O(H) \tag{4.3}$$

And the fitness-proportionate reproduction yields the estimation of the expected number m(H, t+1) of occurrences of every schema H at the generation t+1 as (4.4), where f(H, t) is the fitness of the schema H at generation t (defined as the average fitness of individuals belonging to the schema H); and $\overline{f(t)}$ is the average fitness of the population at generation t.

$$m(H, t+1) \geq \frac{f(H, t)}{\overline{f(t)}} m(H, t)(1-\varepsilon_c)(1-\varepsilon_r)(1-\varepsilon_m) \tag{4.4}$$

(4.4) shows that under the fitness-proportionate reproduction and appropriate genetic operation probability settings with small values, the schemata that can receive an exponentially increasing number of trials (near optimal) are those which have a higher fitness value, a shorter defining length, and fewer defined positions. This implies that schemata defined by a single low-order substructure are most likely the basic building blocks in the evolution of P-GEP. The exploration of the search process further examines the candidates containing these small functional components and tries to find an optimal combination of them to get the final solution.

The above could not be achieved otherwise with Karva notation of GEP, which may probably account for the relatively worse performance of GEP in the conducted experiments.

5. CONCLUSIONS AND FUTURE WORK

This paper has presented Prefix Gene Expression Programming (P-GEP) which adopts a novel genotype representation derived from prefix notation expressions, in order to naturally encode the solution structures incrementally from simpler elements within the linear chromosomes with respect to the genetic operations. The positive experimental results and the follow-up theoretical discussion have shown that P-GEP is a fundamental enhancement to the original GEP algorithm. The genetic operations such as crossover and rotation become more protective for substructures, which leads to a better evolutionary process. Structural components in the linear chromosome representation correspond to contiguous segments of the character strings, which has the potential of identifying useful structural information emergent in the evolutionary process.

P-GEP framework has opened a new entry to define an effective evolutionary process based on the solution structure information. Future research on P-GEP will mainly focus on the following: (1) to propose a framework to reuse the building blocks as emergent substructures in the P-GEP process, (2) besides the substructure components, to define the solution structure based on P-GEP's genotype, and (3) to design a method to evaluate a solution structure and further combine this structure fitness information into the selection procedure of P-GEP, so that emergent good solution structures will be promoted. And the augmented method will be tested with real large benchmark datasets.

6. ACKNOWLEDGMENTS

We highly appreciate the *Physical Realization Research Center of Motorola Labs* for providing funding for this research work.

7. REFERENCES

[1] Ferreira, C. Gene Expression Programming: a New Adaptive Algorithm for Solving Problems. *Complex Systems*, 13, 2 (2001), 87-129.

[2] Ferreira, C. *Gene Expression Programming: Mathematical Modeling by an Artificial Intelligence*. Angra do Heroismo, Portugal, 2002.

[3] Mitchell, M. *An Introduction to Genetic Algorithms (Complex Adaptive Systems)*. MIT Press, 1996.

[4] Zhou, C., Xiao, W., Nelson, P. C. and Tirpak, T. M. Evolving Accurate and Compact Classification Rules with Gene Expression Programming. *IEEE Transactions on Evolutionary Computation*, 7, 6 (2003), 519-531.

[5] Zhou, C. *Gene Expression Programming and Rule Induction for Domain Knowledge Discovery and Management*. Doctoral dissertation, Department of Computer Science, University of Illinois at Chicago, Chicago, IL, 2003.

[6] Xie, Z., Li, X., Eugenio, B. D., Xiao, W., Tirpak, T. M. and Nelson, P. C. Using Gene Expression Programming to Construct Sentence Ranking Functions for Text Summarization. *Proceedings of the 20th International Conference on Computational Linguistics (COLING-2004)*. Geneva, Switzerland, 2004.

[7] Goldberg, D. E. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Pub. Co., 1989.

[8] Weismann, A. *The Evolution Theory (Vol. 1)*. AMS Press, New York, NY, 1983.

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

[10] Koza, J. *Genetic Programming: On the Programming of Computers by Natural Selection*. MIT Press, Cambridge, MA, 1992.

[11] Harik, R. G. and Goldberg, E. D. Learning linkage. In Belew, R. and Vose, M., editors, *Foundations of Genetic Algorithms IV* (247-262). Morgan Kaufmann, San Mateo, California, 1996.

[12] Blake, L. C. and Merz, J. C. *UCI Repository of machine learning databases* [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science, 1998.

[13] Whigham, A. P. A schema theorem for context-free grammars. *Proceedings of IEEE Conference on Evolutionary Computation,* 1(1995), 178-181. Perth, Australia.

[14] Langdon, B. W. and Poli, R. (1997). An analysis of the MAX problem in genetic programming. *Proceedings of the Second Annual Conference of Genetic Programming (GP '97)*. Morgan Kaufmann, San Francisco, CA, 1997, 222-230.

[15] Rosca, P. J. Analysis of complexity drift in genetic programming. *Proceedings of the Second Annual Conference of Genetic Programming (GP '97)*. Morgan Kaufmann, San Francisco, CA, 1997, 286-294.

[16] Banzhaf, W., Nordin, P., Keller R. E. and Francome F. D. *Genetic Programming: An Introduction on the Automatic Evolution of Computer Programs and Its Applications*. Morgan Kaufmann, San Francisco, CA, 1998.