

Adaptive Gene Expression Programming Using a Simple Feedback Heuristic

Jonathan Mwaura¹ and Ed Keedwell¹

Abstract. Gene expression programming has been shown to be an important algorithm in the optimisation of complex systems. However, it has many more operators and parameters than standard genetic programming, each of which needs to be set when applying the algorithm to new problems. In this paper, an adaptive approach for setting probabilities for genetic operators in gene expression programming (GEP) using parameter control is investigated. The adaptive approach implements simple functions that regulate the probabilities of using a genetic operator e.g. mutation, in a given generation based on the comparison between the fitness of the parent organism and child organism in the previous generations. Using this method, it is shown that by using an adaptive approach an increase in the success rate of a run and decrease in the number of generations required in order to achieve success can be achieved.

1 INTRODUCTION

Gene expression programming [1] is a relatively novel evolutionary algorithm which aims to be more biologically plausible than its predecessors genetic programming (GP) and genetic algorithms (GA). It has been shown to be capable of complex applications similar to genetic programming but has been shown to be more efficient due to its genome/phenome distinction [1] which is not present in GP or GA.

It is generally accepted that for these algorithms to work, variation has to occur to bring about diversity in the population. In earlier evolutionary algorithms (e.g. GAs [7] and GP [8]), mutation and cross-over have been used to deliver this variation whereas in Gene Expressing Programming (GEP, [1]) there are a total of seven genetic operators that are used. All these operators depend on probabilities that they will occur in a given run, and the problem is that it is difficult to determine which probabilities will work for every single problem. The evolution of the system therefore, not only depends on what genetic operators are used but also on the probabilities used to determine the occurrence of the genetic variation. Whilst there are occasionally some 'rules of thumb' for parameter settings in standard EAs, there is no clear agreement on what probabilities to set for GEP for use even in the same problem domains.

In this study we propose an adaptive method that fine tunes the genetic operator's probabilities using a feedback heuristic that checks the effect of a probability in a previous generation and adapts it by either increasing or decreasing the probability in the next generation.

The remainder of the paper is arranged as follows: GEP concept, related works, methodology and experiments, results, conclusion and future work.

2 GEP CONCEPT

Representation

Gene Expression Programming is a relatively recent genotype/phenotype evolutionary algorithm developed in 2001 [1]. It follows in the footsteps of genetic algorithms and genetic programming in mimicking the Darwinian Theory of evolution to solve complex problems in mathematics, computer science and related fields. GEP starts with forming linear character chromosomes representing the solutions, akin to a standard GA representation, and is referred as the genotype. The linear representation is formed in two domains, i.e. head and tail; the head contains both terminals and functions while the tail contains only terminals. The length of the head is normally provided as a user defined variable during a run while the length of the tail is a function of the head as shown in Equation 1:

$$t = h(n-1) + 1 \quad (1)$$

Where t is the length of the tail, h is the length of the head and n is the maximum number of arguments of any function in the function set. After forming the linear representation the chromosome undergoes translation where the chromosome is decoded to form a coding region, (i.e. the part of the chromosome representing the region that will be used to solve the problem), and a non coding part. The coding region is further translated into a tree-like structure similar in nature to that in GP, this is known as the phenotype and is expressed as a structure known as the Expression Tree (ET). The phenotype can also be expressed as a linear structure known as an Open Reading Frame (ORF) which is similar to natural biology where the ORF covers only the coding region of the genome with non-coding region upstream and downstream. Unlike natural genes though, the ORF in GEP only has non-coding alleles downstream. It is the concept of these coding and non-coding regions that provides the capability of GEP to form correct ETs every time the genotype undergoes a genetic operation. An additional difference between traditional EA representations and GEP chromosomes is that they can be made up of more than one gene, i.e. multigenic chromosomes are possible. The different genes are linked together using a linking function which is chosen prior to a run. This multigenic nature of GEP helps to solve more complex problems by providing new materials every time a run is carried, readers are directed to [1] for more information on GEP.

¹ School of Engineering, Computer Science and Mathematics, University of Exeter, EX4 4QF, UK.
Email: {jm329, E.C.Keedwell}@ex.ac.uk

Operators

Selection is carried out using standard tournament or roulette wheel selection, as is common in GAs and GP. As with GAs and GP, mutation and one and two point recombination are used, however, in addition, GEP also incorporates gene recombination, insertion sequence transposition, root transposition and gene transposition (for a description of how these operators work, please see [1]). These extended techniques ensure that new material is provided in the organisms as the evolution continues and provides a mechanism for sharing information between multiple genes. Regarding selection, roulette wheel, tournament selection and rank based selection can all be used depending on the problem. For the work presented, roulette wheel selection with elitism has been used.

The difficulty with GEP is that whereas there are a number of 'rules of thumb' that exist for the setting of mutation and crossover probabilities in GAs and GP, with seven operators there are no such rules for GEP. With so many operations, many of them potentially able to swap large sections of code and therefore, be somewhat destructive, it will be difficult to find optimal or even reasonable settings for a number of problems. This problem is overcome in this paper with the use of an adaptive technique which is shown to improve the results of the approach and also provides interesting information on the adapted settings throughout a run.

3 RELATED WORK

In GEP, the standard EA method is used whereby a probability is used to determine when each operation in a run will be conducted. In [1] and [2] a probability rate equivalent to two one point mutations is used and a 0.7 probability for one point crossover. In other work, presented in [1], the probabilities implemented are 0.051 for mutation, 0.2 for one point recombination, 0.5 for two point recombination, and 0.1 for all of the other operations. How these probabilities have been pre-selected is not stated, however most of the work carried out in GEP as seen in [2] shows that these probabilities are being used in the majority of applications. It is assumed that some sort of parameter fine tuning has been conducted to arrive at these probabilities as their success in the majority of the work is marked. However, it cannot be assumed that there is always a canonical probability to use for every particular problem at hand; also there is a huge cost of fine tuning parameters particularly when a complex task is being optimised. Adaptive parameter control means that the algorithm can be started with certain probabilities as a seed and during the course of the run the system can be left to regulate them according to fitness success. Using the method shown here; we demonstrate that the choice of the initial probability seed is not a major concern as the system determines finally what works best for a particular problem.

The work in [2] proposes the automatic fine tuning of parameters using GEP as the evolutionary methodology and another meta-heuristic for parameter control, in this case, a GA. This works through reporting success when fine tuning gene head and gene length, but employs a parallel EA run which is computationally costly, and is demonstrated by the requirement of a server and various clients to run the optimization. Note that this was used

for fine tuning gene head and length and similar probabilities in [1] have been used with regard to genetic operators.

In [3] cloud control has been used to control and tune only the crossover and mutation probabilities, the study proposes and eliminates what is known as non-valid individuals, i.e. These are child and parent organisms who have same fitness after the genetic variation was carried out, e.g. if Ind1 and Ind2 are parent organisms, after they undergo cross over we have Ind3 and Ind4, If fitness of Ind1=Ind3 and Ind4=Ind4 or Ind1=Ind4, Ind2=Ind3, then the set {Ind1,Ind2,Ind3,Ind4} is termed as a set of non-valid individuals. Though the exclusion of these non-valid individuals results in a faster search and improved search performance, it nevertheless leads to a loss of good building blocks that are likely to be beneficial to the optimisation. Work in [5] Seeks to optimize real parameters using GEP while [4] uses a momentum-based and standard mutation to form a hybrid system that adapts the mutation rate in a GA system.

Therefore despite this existing research, there remains a need for a simple system that can optimise the multiple parameter settings of GEP throughout an optimisation run.

4 METHODOLOGY

Here it is proposed that by automatically fine tuning probabilities as input parameters, better results can be obtained for GEP. A probability of operation determines how likely every operation will be carried out and thus has a direct impact on the success rate of the run.

4.1 Implementing GEP methodology

The basic GEP algorithm and genetic operators underlying our system are implemented following Ferreira's approach [1].

In our investigations, we have used the same symbolic regression and sequence induction equations as used in [1] (see equations 3 and 4 in 4.3 below). Fitness, f_i of an individual i was determined by the absolute error [see [1] 4.1a] as shown in equation two below:

$$F_i = \sum_{j=1}^{C_t} (M - |C_{i,j} - T_j|) \quad (2)$$

Where M is the range of selection, C_t is the number of Fitness cases, $C_{i,j}$ is the fitness of the individual i for fitness case j , and T_j is the target value for fitness case j . In this case the perfect organism will have $C_{i,j} = T_j$ and $f_i = f_{max} = C_t \times M$. The error margin is 0.01 and effectively means that if the distance between C_j and T_j is less than this figure, M is returned as the fitness.

The roulette and tournament selection were both implemented, although to permit comparison with results in [1] all experiments were conducted using roulette wheel with elitism, and cloning the best individual in the previous generation.

The runs were started using different seeds for the operator probabilities, but a set of runs was also conducted using the seed probabilities suggested in [1]. Regarding the stopping criteria, the number of generations (please refer to table 1 for the number of generations used) was used to stop the algorithm or when an organism with maximum fitness was achieved (in symbolic regression problem maximum fitness is 1000 while in the sequence induction, maximum fitness is 200).

To represent chromosomes an array of strings was used, when using multigenic chromosomes the linking function was specified beforehand. In this case the addition function was used for linking the genes.

4.2 Feedback heuristics

A generational approach was used in running the GEP algorithm. In this case there are (n-1) new organisms produced and taken to the next generation, where n is the total population.

A counter was implemented for each of the genetic operators used in GEP. After each operational run, i.e. a run in one generation (for a population of size n, there are n/2 operational runs due to crossover providing 2 children), the counters are updated depending on whether fitness for the child organism has been improved or not, (i.e. one is added to the counter if fitness is improved and one is subtracted if fitness is lowered).

Note that, the fitness of the new organism is calculated after all the genetic operations have been carried out as determined by the probability. Since a probability is used to determine whether a certain genetic operation will be carried out, it means some may not be carried out at all. In the case where the operation did not occur for an individual, there is no effect on the particular counter for the genetic operator in question.

These counters run for the entire generation, after which, the probabilities for genetic operations are updated depending on whether the counter is positive or negative (i.e. accumulated negatives would mean that the operation had a negative effect on multiple organisms). Operations that generally lead to improved fitness are encouraged by increasing their probability and operations that generally provide worse solutions are discouraged by reducing their probability in the next generation. The new probabilities for the genetic operators are used in the subsequent generation. This is repeated until the stop criterion is met. We update the probabilities in small steps as shown by the pseudocode below;

Consider a probability for Mutation; we have kept a counter, mutationCounter

```

gepAdaptMut (mutationCounter)
{
    if (mutationCounter >0)
        proMutation= (proMutation +
        (proMutation*0.2));
    if (mutationCounter <0)
        proMutation= (proMutation -
        (proMutation*0.2));
    if (probability_Mutation>maximum_Prob
    ability)
        proMutation= (proMutation -
        (proMutation*0.2));
    if (probability_Mutation<=minimum_Pro
    bability)
        proMutation= (proMutation +
        (proMutation*0.2));
}

```

Where maximum probability is set as 1 and minimum probability is set as 0. The counter is updated during the course of operational run.

4.3 Experiments

Two functions are optimised as follows:

i) A fourth order symbolic regression,[1].

i.e.

$$F(a) = a^4 + a^3 + a^2 + a^1 \quad (3)$$

Ten values of (a) are given and the corresponding values of f (a), please see [1] for details.

In the first experiment we tested and compared an adaptive GEP using all the genetic operators, with the parameters set as shown in Table 1, Exp 1. This was compared to a standard run. This is a case for a unigenic organism.

In the second experiment a standard run is conducted with all genetic operators and an adaptive run where all genetic operators are adapted. This is a case for a multigenic organism

The Functions and Terminals are set up as in [1].

ii) A Sequence Induction problem given in [1]

$$i.e. N=5a^4 + 4a^3 + 3a^2 + 2a + 1 \quad (4)$$

Ferreira in [1] uses this problem to show how GEP creates constants. In our case since our aim is to show the effect of adaptation, we use the same problem and the constants were created from scratch by the algorithm, i.e. we do not use ephemeral random constants. This is an important capability of GEP as in most problems either electrical or mathematical, the constants needed are not known *a priori*.

In the first experiment an adaptive GEP with Mutation (Pm= 0.022) and one point crossover (P=0.7) and a unigenic gene with h=6 and no of genes =7 was compared to a standard run.

This same experiment was then conducted using all genetic operators, adapting all of them and then compared with a standard run.

The addition function was used as the linking function for all experiments with more than one gene.

Table 1. Parameters used in the experiment involving Symbolic Regression and Sequence Induction

	Exp 1	Exp 2	Exp 3	Exp 4
Population	30	30	50	50
Generations	50	50	100	100
Function Set	/,*,-,+	/,*,-,+	/,*,-,+	/,*,-,+
Terminal set	a	a	a	a
Head Length	24	6	6	6
Number of genes	1	3	7	3
Mutation Rate	0.051	0.051	0.022	0.3
One Point Cross-over rate	0.2	0.2	0.7	0.7
Two Point Cross-over rate	0.5	0.5	0.1	0.2
Gene Cross-over rate	0.1	0.1	0.1	0.1
IS Transposition rate	0.1	0.1	0.1	0.1
Ris Transposition rate	0.1	0.1	0.1	0.1
Gene Transposition Rate	0.1	0.1	0.1	0.1

Table 1 shows all parameters used for the runs for both experiments. Please note that for all standard runs the actual parameters shown on the table are used whereas in the adapting case, these parameters act as a seed.

5 RESULTS

In each of the following experiments, the system was run 100 times with two features recorded; the success rate (expressed as a percentage) which is calculated as the number of runs which resulted in the correct result, and the average number of generations required to obtain a given result. This is indicative of the performance of the system as those runs which have found the optimal result will be terminated, whereas those which do not are restricted to the generation limit.

Fig1 compares a standard run using all genetic operators as determined by the probabilities in Table 1, Exp 1 and compares it with same probabilities used as initial probabilities for adaptation. This is a unigenic case.

Fig 2 compares a standard run using all the seven genetic operators probabilities as seen in Table 1, Exp 2 and compares this to a run with the genetic operators probabilities adapting with generations. This is a multigenic case.

Fig 3 compares a standard run using all the seven genetic operators probabilities as seen in Table 1 and compares this to a run with all the genetic operators probabilities adapting with generations, this is done in the case of sequence induction.

Fig 4 shows the effect of probability rate as they adapt during the course of one run. Note the oscillatory dynamics we get as the run progresses. This is a single multigenic run using a $P_m = 0.051$, $P_{1r} = 0.2$, $P_{2r} = 0.5$, P_{gr} , P_{is} , P_{ris} , $P_{gfm} = 0.1$ (table 1, Exp 2)

Fig 5 shows the effect of probability rate as they adapt during the course of one run. In this case we repeat use the same set up as used in Fig 4 but with different initial probabilities, see table 1, Exp 4.

Fig.1. Comparison of success rate and average number of fitness of the population using parameters undergoing adaptation versus standard/fixed ones in unigenic organisms on the symbolic Regression problem

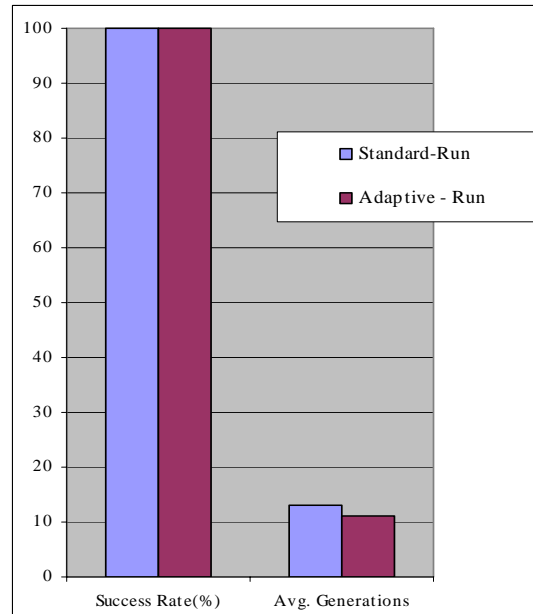


Fig.2. Comparison of success rate and average number of fitness of the population using parameters undergoing adaptation versus standard/fixed ones in multigenic organisms on the symbolic regression problem.

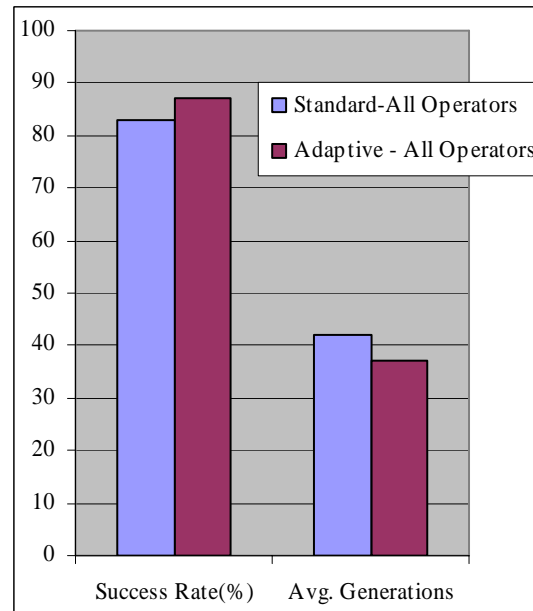
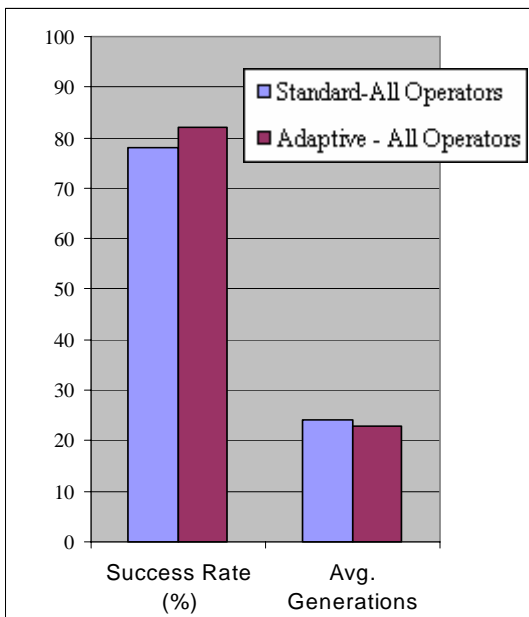


Fig.3. Comparison of success rate and average number of fitness of the population using parameters undergoing adaptation versus standard/fixed ones in multigenic organisms on the sequence induction problem



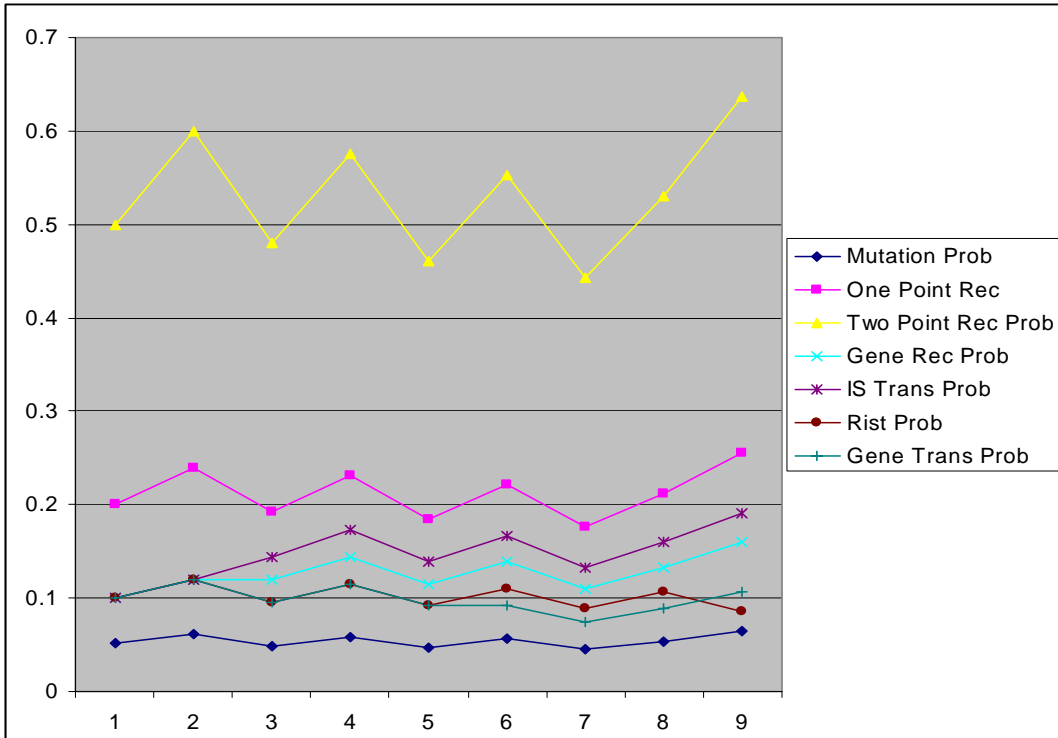


Fig.4. Progression of adaptation of probabilities of genetic operators. A typical case in Experiment 2.

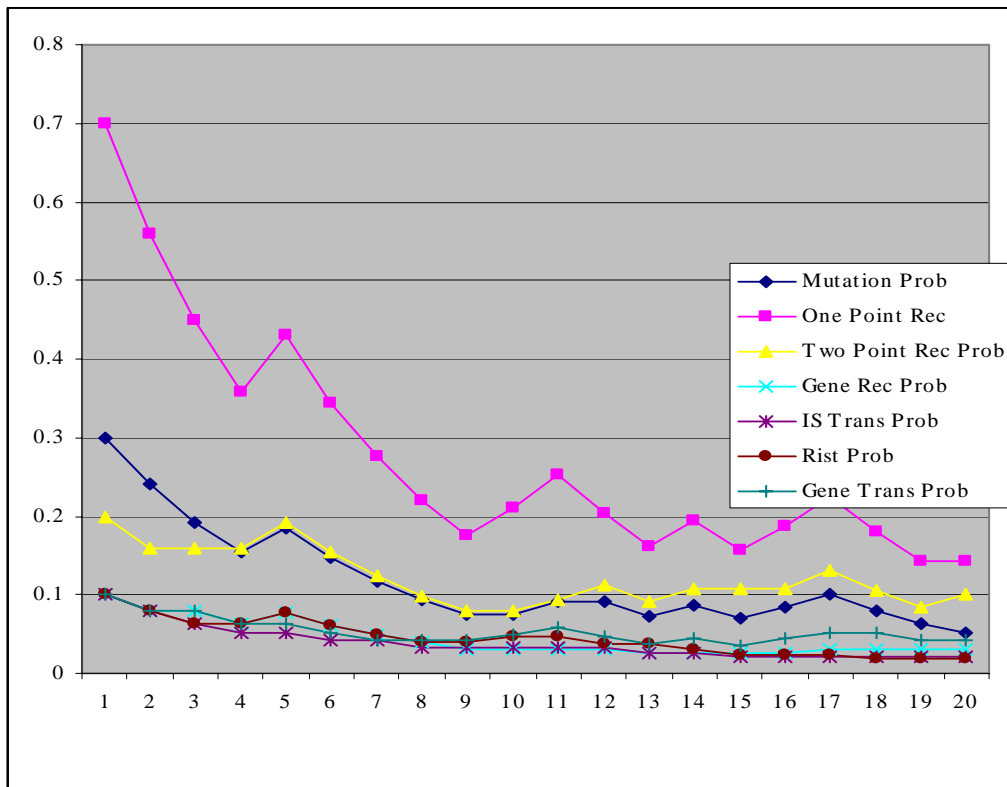


Fig.5. Progression of adaptation of probabilities of genetic operators. A typical case in Experiment 3. Please note this is a similar set up with Fig 4 above but with changed initial genetic operators' probabilities.

6 DISCUSSION

The study investigates a method of tuning input parameters during the course of the run to try to counter the problem of starting with parameters that do not suit the problem at hand; rarely is there a case where the optimal parameters to use are known *a priori* before solving the problem. As shown from the results obtained using this method, the system is able to vary the parameter settings in such a way that performance is improved over the standard run by 5% in terms of success rate, also comparing with the average generations required to get a successful organism; the method decreases this by more than 5%.

With adaptive parameters we can also infer that the algorithm avoids the problem of descending into local optima, as the increasing or decrease of the probability rate affects the search space. The results show that although convergence speed is increased, the quality of results does not suffer, indicating that the systems does not descend into local minima. This might be due to the idea that an increase or decrease in probability will tend to improve diversity in the population thereby giving better models every time a good organism is obtained.

Figures 4 and 5 shows how the adaptive algorithm works with different initial probabilities. Starting with small probabilities, the probabilities increase with small steps as the organisms formed are better than in previous generations, but as the algorithm progresses through the generations, organisms become better and we have constant probabilities. It would appear that the majority of the operators have a level at which they are most effective. This is best seen with the mutation which, despite starting at 0.3 (or 30%) in Figure 5 diminishes to a more plausible value of ~0.05. This trend is also seen with the one point crossover which has a tendency to descend and hover around the 0.2 mark despite the seed value provided. The main counter-example to this is two-point recombination which appears to remain constant in this single experiment, an observation which is not repeated when the averages are plotted (not shown).

Starting with higher initial probabilities, organisms formed could be worse than those existing, this is generally because of the deleterious effects of per-gene mutation in particular, the probabilities as seen in Fig 5 will therefore tend to decrease sharply and as generations progresses, they become more constant.

These two experiments also show how the effect of all the genetic operators working together. An increase or decrease in any of the probabilities means the other operators get affected, i.e. if the effect of mutation is negative, the algorithm may opt to use insertion sequence transposition or root transposition or any other operator which appear to guide the evolution to better solutions. This in turn creates the oscillations as shown in Figs 4 and 5 above.

7 Conclusions and future work

All the genetic operators in the standard GEP affect the convergence and success rate of the evolution. In this study an adaptive method is proposed to adapt the set of probabilities rates used in any given problem with an aim of avoiding local optima as well as increasing performance and reducing

computing time. We have shown that this works by providing a simple feedback heuristic in the standard algorithm.

In our future work we seek to extend this study by combining this with an adaptation of the number of genes as well as the gene length.

References:

- [1]. Ferreira, C. (2001) *Gene Expression programming: A new Adaptive Algorithm for Solving Problems*. Complex Systems, 13(2): 87-129
- [2]. Park, Ho-Hyun et al (2008). *Parallel hybrid evolutionary Computation: Automatic tuning of parameters for parallel gene expression programming*. Science Direct. Applied Mathematics and Computation 201: 108-120.
- [3]. Jiang, Yue et al (2008). *Adaptive Gene Expression Programming Algorithm Based on Cloud Model*. International Conference on BioMedical Engineering and Informatics. IEE computer society (2008) doi 10.1109/BMEI.2008.42
- [4]. Temby, L. Vamplew, P. Berry, A (2005). *Accelerating Real-Valued Genetic Algorithms Using Mutation-With-Momentum*. The 18th Australian Joint Conference on Artificial Intelligence, Sydney, Australia, 5-9 Dec 2005
- [5]. K, Xu et. al. (2008). *A novel method for real parameter optimization based on Gene Expression Programming*. Applied soft computing journal 2008, doi:10.1016/j.asoc.2008.09.007
- [6]. Ferreira, C (2002). *Analyzing the Founder effect in simulated evolutionary processes using gene expression programming*. In A. Abraham, J. Ruiz-de-Solar, and M. Koppen (eds), *Soft Computing Systems: Design, Management and Applications*, pp. 153-162, IOS Press, Netherlands.
- [7]. Goldberg D.E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison Wesley Longman Inc.
- [8]. Koza, R.J (1992). *Genetic Programming: on the Programming of Computers by Means of natural Selection*, The MIT Press, Cambridge, MA.