Exploiting The Path Of Least Resistance In Evolution

Gearoid Murphy and Conor Ryan
Biocomputing and Developmental Systems Group
Computer Science and Information Systems Department
University of Limerick, Ireland

ABSTRACT

Hereditary Repulsion (HR) is a selection method coupled with a fitness constraint that substantially improves the performance and consistency of evolutionary algorithms. This also manifests as improved generalisation in the evolved GP expressions. We examine the behaviour of HR on the difficult Parity 5 problem using a population size of only 24 individuals. The negative effects of convergence are amplified under these circumstances and we progress through a series of insights and experiments which dramatically improve the consistency of the algorithm, resulting in a 70% success rate with the same small population. By contrast, a steady state GP system using a population of 5000 only had a success rate of 8%. We then confirm the effectiveness of these results in a number of arbitrary problem domains.

Categories and Subject Descriptors

I.2 [**Artificial Intelligence**]: Problem Solving, Search—*Genetic Programming*

General Terms

Algorithms, Theory

1. INTRODUCTION

Inconsistency is a common problem for evolutionary algorithms. Much of the variance in performance characteristics is due to the stochastic component of the algorithm. This may inadvertently promote suboptimal genetic content at the expense of relatively unfit but potentially very useful material. The gauntlet of a fixed size population forces the algorithm to marshal its limited resources over the unlimited divergences the population may take as it traverses the expression space. Unfortunately, it sometimes makes the wrong decision, resulting in a catastrophic loss of essential material, also known as premature convergence.

Hereditary Repulsion (HR) [4], a selection method coupled to a Mutual Improvement (MI) fitness constraint, substantially improves the performance consistency in evolutionary algorithms. MI is a replacement strategy which requires that a child be better than both parents before it gets into the next generation. Furthermore, the derived results of HR have a far higher quality with respect to generalisation than standard methods [5]; such as steady state or generational algorithms for a broad spectrum of problems.

This paper takes a closer look at the causes of failure for HR in the Parity domain. A sequence of experiments are carried out which yield insights into the evolutionary dynamics of the system. This produces the Density Tournament operator, a selection method which redirects crossover events towards individuals with few or no offspring. We derive further analysis of this selection method and produce a new algorithm for controlling evolution, one which can efficiently and economically adapt to difficulty in evolution. This works by using the ratio of failed crossover events to successful ones as a ranking which controls the focus of evolution within the algorithm, resulting in an extremely robust and highly consistent evolutionary algorithm.

This paper is organised as follows. Section 2 discusses a number of convergence manipulation techniques. Section 3 gives a technical description of the Hereditary Repulsion algorithm and an analysis of the small population experiments and results. Section 4 describes the Density Tournament algorithm, a simple effective technique for substantially improving the performance of the small populations. An analysis of the few failures from the Density Tournament operator is given in Section 5. This analysis produces the adaptive effort algorithm, which dynamically finds the path of least resistance through the expression space, described and evaluated in Section 6. A validation of this algorithm in a set of real world problem domains is demonstrated in Section 7. Conclusions are given in Section 8.

2. BACKGROUND

One of the first attempts at inhibiting the phenomenon of premature convergence was fitness sharing [7]. This technique sought to prevent the homogenisation of the populations content by forcing similar expressions to *share* fitness. The similarity of the expressions was deduced by an external metric and was unique to each implementation. This method was inspired by the observation that biological systems tend to exhibit niche characteristics. Fitness sharing encouraged the expressions to seek out their own niches so as to maximise their fitness.

Another common convergence manipulation technique is the use of spatial segregation to control the evolutionary

GECCO'08, July 12–16, 2008, Atlanta, Georgia, USA.

dynamics. By physically separating distinct pools of genetic content [1, 8], one can prevent a single solution from becoming dominant, thus inhibiting premature convergence.

One possible risk of such isolated development is the emergence of incompatible solutions on the separate *islands*, thus one of the core considerations when implementing an island model is the method of communication which will be used to maintain cohesion between the separate populations.

Using age as a segregation measure is also described in the Age Layered Population (ALP) [2] algorithm. This highly effective protocol uses cascaded pools of progressively older individuals to sustain evolution over an extremely long period. New individuals are continuously generated and processed down the age segregated populations. Normal evolutionary dynamics are used within each population. As the individuals get older, they replace any less fit individuals in the next layer. Crossover also samples from the current and previous "age" layer, allowing a channel for seamlessly integrating age segregated content.

Age has also been used to protect individuals from premature exclusion from the population [6]. An individual is given a length of time to "live" thus allowing it time to integrate itself with other members of the population.

Significant results have also been generated by Streeter [9] with his Minimum Behavioural Change (MBC) algorithm. Derived out of an elegant analysis of the causes of bloat in genetic programming, MBC dramatically improved the performance and inhibited bloating of a GP algorithm applied to the quartic polynomial domain by only allowing individuals who are markedly different from their parents entry into the population.

3. HEREDITARY REPULSION

Hereditary Repulsion was derived from the observation that the final generation of a population tends to be descended from a few initial individuals [3]. HR attempted to inhibit this occurrence by vigorously mixing the genetic content of the population through facilitating crossover events between individuals who had dissimilar genetic lineages.

The algorithm begins by selecting an individual at random. This individual is used as the reference for the repulsion algorithm. A tournament pool of size N is then filled with random individuals. The shared hereditary history between the individuals in the pool and the reference individual is measured. The pool individual with the smallest hereditary overlap is selected to be crossed over with the reference individual. Figure 1 provides an example of how the overlap is calculated between two individuals.

Given that the repulsion tournament algorithm will place pressure on the dynamics to explore diverse representations, there is a possibility that the quality of the population will degrade as it explores the expression space. To protect the algorithm from the potential deleterious effects of intense exploratory dynamics, a constraint was incorporated which mandated that an individual must be better than both its parents before it can be considered for insertion into the next generation. This constraint shall be referred to as the Mutual Improvement (MI) constraint.

Due to the nature of the algorithm, HR will have a variable number of evaluations per generation. This is caused by the many rejected crossover events for each successful one. For this reason, the HR algorithm is provided with an upper limit to how many evaluations it may make during a run. For the sake of comparison, this upper limit is set equal to the total number of evaluations in a corresponding standard GP run. Also, for the experiments described in this paper, the HR algorithm used a generational style approach to evolution, deriving the next generation completely from the current one.

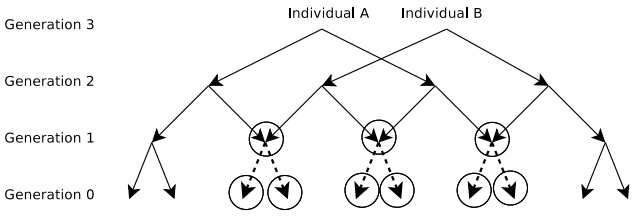

Figure 1: Illustration of Common Hereditary History between 2 Individuals in a Generation System. Parents are at generation 2, grandparents at generation 1 and so on. In this example, individuals A and B share 9 common ancestors.

Experimental analysis has shown that the repulsion tournament only has an appreciable effect when the population is small and that the primary reason for the high performance of the algorithm is the MI constraint. This has been empirically demonstrated in [5].

3.1 Experimental Overview

This section provides an overview of the experimental setup used to generate the initial experimental analysis in this paper.

Our experimental problem domain is the Boolean Odd Parity 5 problem. The Odd Parity problem takes a sequence of input bits and returns a 1 if the number of active bits is odd, a 0 otherwise. The number of input bits used was 5. This amounted to 2^5, 32 training cases.

The primitives used were {AND, OR, NOR, NAND}. This problem domain is notoriously difficult without the XOR or EQUAL functions, thus evolution was allowed to continue for 2500000 evaluations, this is equivalent to a standard population of 5000 evolved for 500 generations. The maximum tree depth was set to 16. The expressions were created using ramped half and half initialisation with a minimum depth of 4 and a maximum depth of 6. Fitness measure was the number of correct outputs (max 32). Mutation was not used in our experiments to eliminate its influence on the convergence behaviour. Details specific to individual experiments will be explicitly stated in each section. All results shown are derived from 100 independent runs for each set of configurations.

3.2 Small Population Analysis

This section presents experimental results from the application of the HR protocol to the parity problem using a population of 24 individuals.

Ordinarily, such a small population would be completely inadequate for such a difficult problem using standard steady state or generational algorithms, however; even when a standard system is provided with a far larger population, HR still easily outperforms it. To emphasise this point we contrast the tiny HR population of 24 individuals with a steady state algorithm (SS-GP) initialised with a population of 5000 in-

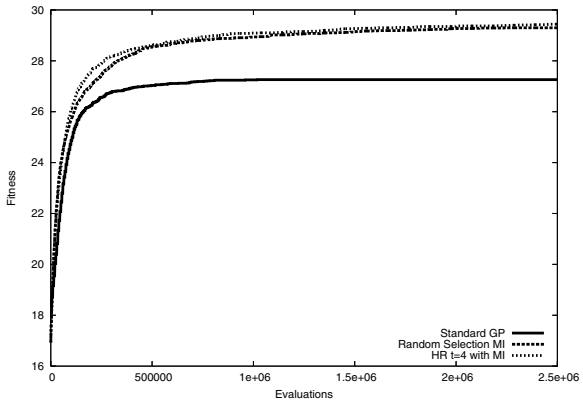

Figure 2: The MI constraint clearly accounts for most of the improvements observed in HR. Both MI implementations easily outperform a steady state GP algorithm of 5000 individuals. HR has a 16% success rate and MI with random selection has a 14% success rate. The SS-GP system has a success rate of only 8%.

dividuals. A tournament size of 4 was used to select individuals in the steady state algorithm. A tournament size of 4 was used for the HR algorithm and an experiment using random selection with the MI constraint was implemented.

Figure 2 shows the average best fitness reached by the respective algorithms for 100 independent runs. On examining the steady state results in detail, it is evident that while it may sometimes do quite well, it is plagued by a tendency to prematurely converge, locking itself into low fitness attractors. The performance for the MI constraint algorithm using random selection is of substantially higher quality, demonstrating a greater density of higher fitness solutions. The HR tournament experiment shows a statistically insignificant improvement in the quality and consistency over the random selection experiment. The relatively equal performance of the random selection with MI and the HR tournament with MI indicate that there is little to be gained with the HR selection method, confirming results in [5].

What is of interest from both HR and the MI with random selection is the occasional low yields from the experiments. Evidently, there are circumstances under which the MI constraint can fail to exploit its genetic content to the highest possible degree.

Our analysis of these circumstances indicate that the MI constraint can sometimes promote relatively low quality expressions *faster* than the high quality expressions. This is especially true in situations where it is difficult to improve on the best fitness in the population. As essential genetic material that may be contained within the high quality expressions is lost, the population finds itself locked into a suboptimal attractor. Therefore, our next experiment sought to inhibit this phenomenon.

```
Density Tournament

1. Select an Individual Randomly
   Set this individual to winner
2. for each N in tournamentSize
     Select a random Individual
     if(random.individual.numOffspring <
       winner.individual.numOffspring)
         winner = random
```

Figure 3: Pseudocode for the density tournament selection operator.

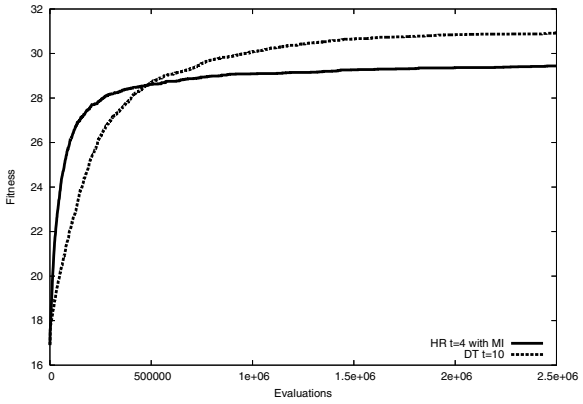

Figure 4: Clear performance gains from using the DT selection mechanism over the HR tournament. The HR experiment successfully solved the problem 16% of the time, the DT experiment had a success rate of 36%.

4. DENSITY TOURNAMENT

The Density Tournament (DT) operator was designed to prevent the saturation of the population with low quality individuals. This is likely to happen as low quality individuals require less *effort* to produce offspring satisfying the MI constraint.

This effect is realised by creating a tournament operator which selects individuals who have relatively little or no offspring in the next generation. It prevents low quality individuals from saturating the population as they will not be selected for crossover once they have produced more offspring relative to the other individuals. Pseudocode for this selection mechanism is given in Figure 3.

Figure 4 contrasts the performance of the HR configuration with a tournament size of 4 with a density tournament experiment, of tournament size 5. Both systems use the MI constraint. There is a clear indication of higher densities of high quality expressions being generated by the DT operator, with just a handful of individuals ending up with a low score. The Student T-Test gives a statistical confidence p value of .0005 for these results.

While the results seem to confirm the effectiveness of the density tournament, there were still some disconcerting indi-

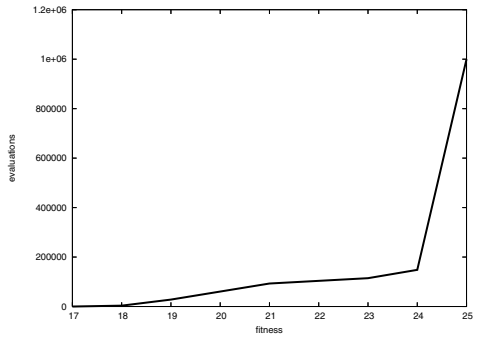

Figure 5: Sharp spike in effort required for improvement indicating evolutionary dead end, or local optimum. This example was taken from a failed run using the configuration from Section 4.

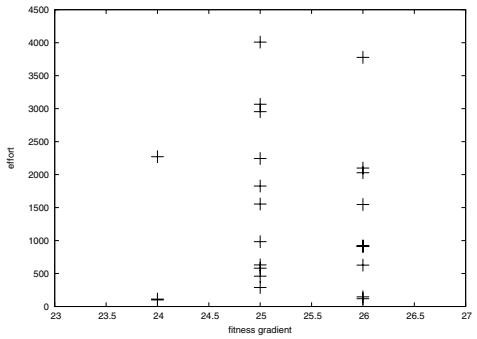

Figure 6: Each point represents the ratio of failed crossovers to successful ones for individuals of a specific fitness in a single generation taken from a DT run. Note that some high fitness individuals require very little effort to generate offspring.

cations of susceptibility to low quality attractors in a small percentage of runs with the DT operator. The next investigation focuses on the defining characteristics of these failed runs.

5. EFFORT ANALYSIS

This section describes the analysis of the failed DT runs. One of the most salient indicators of poor performance was the ratio of effort to the maximum fitness achieved. In this case effort refers to the number of evaluations required to generate a new maximum fitness in the population. Such behaviour is usually unique to each run, an example of which is given in Figure 5. Here we can clearly see the system suffering from a sharp spike in the effort needed to make headway. Such a spike is recognised by the increasingly large numbers of evaluations needed to reach the next highest fitness peak. This particular example was taken from one of the worst performing runs using the DT configuration of the previous section.

If any evasive action is to be taken, this is the point to do it. The question remains however as to what exactly should be done. An initial crude fix would be to roll back the population to some predetermined point in its past and allow evolution to begin from there. This approach is fraught with uncertainty as it is difficult if not impossible to be sure that the population won't get pulled into the same low quality local attractor space again.

A closer analysis yields some interesting insights. Figure 6 shows the effort invested in individuals of a specific fitness for a single generation taken from a run using the DT configuration from Section 4. This diagram indicates that one of the two individuals of fitness 24 in the population required approximately 100 crossovers to produce valid offspring while the other required nearly 2500 crossovers. The several individuals whose fitness was 25 required a large range of computational investments to produce valid offspring, from approximately 400 crossovers at the low end to over 4000 at the high end. A similar distribution exists for the individuals of fitness 26.

The first surprise from Figure 6 was the observation that higher quality individuals do not always require high numbers of evaluations to generate offspring which are better than both parents. The relative proportions of effort needed to generate offspring from two individuals of the same high fitness may also be wildly imbalanced, sometimes one needing 10 times greater effort than the other. The sudden rise in effort needed to progress illustrated in Figure 5 is caused by such unproductive individuals.

The tentative conjecture from this brief analysis is that it may be possible to rapidly follow the path of least resistance through the expression space and that this trajectory could produce high quality individuals. Such an algorithm would use the ratio of failed crossovers to successful crossovers to guide in selecting which individuals are worthy of an investment of computational effort.

6. THE ADAPTIVE EFFORT ALGORITHM

This section describes the concept of the Adaptive Effort (AE) algorithm. The findings from the previous section indicated that the presence of low yield individuals in the population can seriously degrade the effectiveness of the evolutionary process. Our primary design criteria was to develop a means to avoid these individuals.

The solution was derived from an algorithm originally designed for use with a mutation based hill climber. The particular problem domain we were applying the hill climber to was quite difficult and we were consistently getting poor results. On closer examination of the behaviour of the hill climber, we noticed that 95% of the hill climbers' progress was made in the first 10% of the mutation events. This indicated that the hill climber initially made progress quite easily but quickly found itself in a position that required a great deal of *effort* to improve upon, wasting the remaining 90% of the mutation events.

The solution to this problem was to allow the expression to *branch* into multiple points in the search space. This lessened the potential for becoming caught in a particular dead end. Branching strategies can generate exponential increases in the number of points to be searched however, this does not become a problem because the algorithm does not expand each branch equally. If a particular branch requires relatively large numbers of mutations for improvement, it is quickly ignored by the algorithm in favour of branches which require less effort to make progress into the expression space.

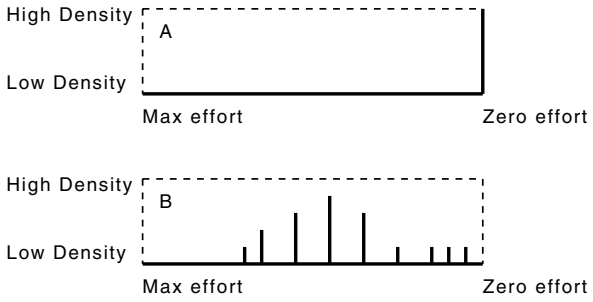

Figure 7: Part A of this figure shows the state of the effort array when the algorithm starts; the entire population is represented as a spike at position zero of the array. Part B shows how the population has been spread over the effort array once some computation has been invested in the search space. Note the trickle of new points in the search space originating from position zero being migrated towards the main population cluster as the algorithm attempts to improve them.

This shift in focus is implemented very simply by using an *effort array*. Each point in the array represents the ratio of failed mutation events to successful ones. The algorithm will always take the first expression it finds from the top of the array, this represents the point in the search space with the least effort *invested* in it. A fixed investment of mutations is made on the expression. Any resulting mutation which improves on its parent is inserted as a branch into the search space. If the parent has reached or exceeded the allowed number of branches, it is removed from the effort array. A diagram illustrating the initial state of the array and its state after some effort has been invested in the search space is given Figure 7.

This algorithm was good at generating high quality solutions in the expression space. It initially exhausted all the easily improved expressions before consistently discovering viable routes to higher fitness through the search space.

Based on the similar observation of highly variable effort needed to produce viable offspring from individuals in a population of GP expressions in Section 5, the AE algorithm was reconfigured for use in an evolutionary algorithm.

The only change that this entailed was that rather than selecting a single expression from the top of the effort array, a small population of individuals was selected instead. A fixed investment of crossover events and evaluations were alloted to that population. The AE algorithm uses the DT selection operator to select parents from within the temporary population.

After the investment of effort, any individual who had reached or exceeded the alloted number of successful offspring was removed from the effort array. Otherwise it was reinserted into the effort array in a position which reflected the effort invested in it.

In this manner, one may have a small population that has the benefit of a large number of dormant points in the search space to fall back onto should it run into difficulty.

When we applied this algorithm to the Parity problem, we observed significant performance increases. We allowed each expression to branch 3 times and fixed the investment made on the population at 1000 evaluations. The size of the population was, as before, 24 individuals. The size of the density tournament was 10 individuals. This result is shown in Figure 8.

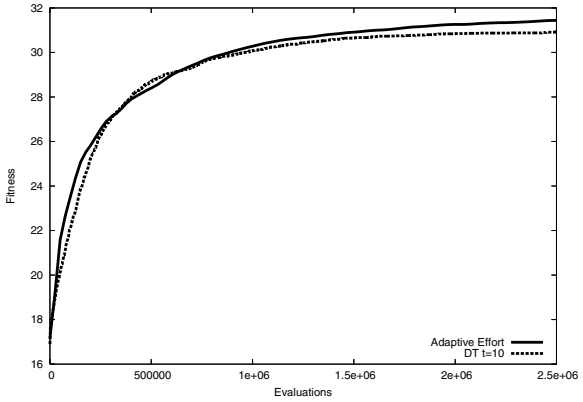

Figure 8: While there does not seem to be a wide margin of difference between the AE and DT algorithms in this graph, AE has a much higher success rate of 70%, compared to 36% for DT. The superiority of AE over DT on the Parity 5 problem has been calculated to a p value of .0005.

Problem	AE-GP	DT-GP	HR-GP	SS-GP
bupa	0.77	0.76	0.76	0.67
indian	0.80	0.80	0.80	0.77
ion	0.92	0.92	0.92	0.87
sonar	0.83	0.83	0.81	0.73
wpbc	0.77	0.73	0.77	0.72

Table 1: Average Best Fitness. AE consistently produces high quality solutions that match or exceed the performance of the other implementations. The low performance of DT in the WPBC problem is due to the fact that this problem domain needs certain individuals to contribute higher than average numbers of offspring, which is inhibited by the Density Tournament.

The AE algorithm solves the Parity problem 70% of the time. This high consistency results in a low standard deviation which provides a very high level of statistical confidence when comparing AE against DT, p=.0005.

7. VALIDATION

So far we have remained relatively insulated from the mercurial properties of the abundant diversity of real world problem domains. Whilst Parity is an excellent problem domain for analysing the performance of a particular evolutionary strategies, these results will only have merit if they can be reproduced consistently in arbitrary problem domains.

To prove the virtue of our algorithms, we have applied them to a wide variety of binary classification problems taken from the UCI Machine Learning repository. Our binary classification GP classifies a particular sample as positive if it is above a .5 and negative if it is below .5; The problem domains we used were as follows; BUPA Liver Dis-

Problem	AE-GP	DT-GP	HR-GP	SS-GP
bupa	0.01	0.01	0.01	0.06
indian	0.01	0.01	0.01	0.02
ion	0.01	0.01	0.02	0.03
sonar	0.03	0.03	0.03	0.06
wpbc	0.02	0.01	0.03	0.01

Table 2: Ratio of Std Deviation of Fitness to Average Fitness. All implementations except SS-GP are generally consistent.

Problem	AE-GP	DT-GP	HR-GP	SS-GP
bupa	34.91	30.26	57.26	7.54
indian	33.70	38.96	56.39	13.70
ion	33.92	45.98	67.19	17.47
sonar	36.90	52.78	74.08	15.80
wpbc	13.12	3.74	50.43	2.12

Table 3: Average Nodes per Tree. AE is highly resistant to bloat yet always produces high quality results. Note that the HR and DT implementations can vary considerably in tree size. The parsimony pressures on the SS-GP system accounts for the small tree sizes, SS consistently performs the worst.

Problem	AE-GP	DT-GP	HR-GP	SS-GP
bupa	0.22	0.16	0.41	1.65
indian	0.17	0.19	0.25	0.74
ion	0.14	0.13	0.26	0.63
sonar	0.15	0.19	0.30	0.97
wpbc	0.28	0.05	0.40	0.88

Table 4: Ratio of Std Deviation of Tree Size to Average Tree Size. AE produces *consistently sized* GP solutions of high quality. DT is also highly consistent in its tree size but may sometimes produce relatively low quality GP expressions and it is more susceptible to larger tree sizes than AE.

order, Pima Indian Diabetes, Wisconsin Prognostic Breast Cancer, Ionosphere Radar Returns and the Sonar Discrimination problem.

The experimental setup used is as follows; a standard steady state algorithm (SS-GP) using a tournament size of 5, a HR algorithm with a repulsion tournament size of 5, a DT algorithm with a density tournament size of 10 and finally an AE algorithm using a density tournament of size 10, a branching factor of 3 and an investment limit of 1000. Each system used a population of 100.

Each setup used ramped half and half initialisation of minimum depth 2 to a maximum depth of 4. The maximum tree depth allowed was 12. A total of two and half million evaluations per run and 100 independent runs were used for each experimental result. The functional primitives used were {*, %, +, -, exp, sqrt}. The SS-GP system used parsimony pressure to regulate bloat, this pressure favoured smaller individuals when fitness was equal and was applied in the tournament and replacement parts of the algorithm.

Only testing results are shown for the generated fitnesses of the experimental configurations. These results were derived from evaluating the evolved expressions on an unseen data sample, a much better measure of the quality of the system.

The average best fitness for each experiment over all the problem domains is shown in Table 1. As the problems are relatively easy, there is little difference in fitness between the AE, DT and HR experiments. AE produces results of the highest quality to a high consistency as illustrated by Table 2. This table shows the *ratio* of the standard deviation to the average fitness, showing the scale of the variability in performance. Whilst DT and HR are also highly consistent, they are more susceptible to lower quality performance.

One of the most prevalent forms of negative performance measures in GP is the susceptibility of the system to bloat, the tendency of the system to produce unnecessarily large expressions. Despite the fact that AE, DT and HR have no "awareness" of the underlying GP representation, they are considerably resistant to this undesirable phenomenon.

Table 3 shows the average tree size for each experiment over all the problem domains. SS-GP is notable for the small size of its low quality expressions however these sizes are highly inconsistent as shown by Table 4. This table shows the ratio of the standard deviation to the average tree size.

Notably, these tables show that AE produces high quality compact expressions in a consistent manner. DT and HR suffer from a tendency to bloat in some problem domains. DT produced the smallest expressions for the WPBC problem but these expressions are of poor quality.

These results clearly demonstrate that the high quality performance of AE in the parity domain manifests in a consistent manner across a wide variety of problem domains. Furthermore, this method provides excellent bloat regulation as an organic, natural feature of the algorithm. The resulting expressions lend themselves quite readily to interpretation of functional relationships in the application domain due to their compact size.

Interestingly, the DT selection operator has a tendency to suffer in problem domains that require higher than average numbers of offspring from specific individuals. Clearly, the Parity problem alone gives a distorted view of its ability.

8. CONCLUSIONS

We have presented a series of experimental analyses using the context of a small population on a difficult regression problem. As the potential for premature convergence is greatly amplified in this situation it highlights any aspect of the algorithm which contributes to lower performance.

This strategy resulted in a fruitful investigation which yielded the Adaptive Effort algorithm, a high performance system which produced a 70% success rate at the Parity 5 problem using only 24 individuals.

The quality of AE was demonstrated to be applicable in a diverse set of real world GP problems, manifesting as both high fitness and a resistance to bloat.

Future work will examine the emergence of dominant solutions in the population and how these forms influence the nature and quality of convergence in the *phenotypic* space. A detailed analysis of the bloat regulation mechanism in AE is also planned.

9. REFERENCES

[1] D. Eby, R. C. Averill, B. Gelfand, W. F. Punch, O. M. III, and E. D. Goodman. An injection island ga for flywheel design optimization. In *5th European Congress on Intelligent Techniques and Soft Computing*, volume 3, 1997.

[2] G. S. Hornby. Alps: the age-layered population structure for reducing the problem of premature convergence. In *Gecco 2005*, 2005.

[3] N. McPhee and N. Hopper. Analysis of genetic diversity through population history. In *Gecco*, pages 1112–1120, 1999.

[4] G. Murphy and C. Ryan. Manipulation of convergence in evolutionary systems. In *Genetic Programming Theory and Practise*. Springer US, 2007.

[5] G. Murphy and C. Ryan. A simple powerful constraint for genetic programming. In *EuroGP*, 2008.

[6] F. A. Naoyuki Kubota, Toshio Fukuda and K. Shimojima. Genetic algorithm with age structure and its application to self organising manufacturing system. In *IEEE Symposium on Emerging Technologies and Factory Automation*, 1994.

[7] B. Sareni and L. Krahenbuhl. Fitness sharing and niching methods revisited. In *IEEE Transactions on Evolutionary Computation*, volume 2. IEEE Computational Intelligence Society, 1998.

[8] D. J. Siddique, M. X.-Y. Lin, R. M., and C. J. Automated detection of nodules in the ct lung images using multi-modal genetic algorithm. In *IEEE Transactions on Evolutionary Computation*, volume 1. IEEE Computational Intelligence Society, 2003.

[9] M. J. Streeter. The root causes of code growth in genetic programming. In *EuroGP*, 2003.