Accelerating Real-Valued Genetic Algorithms Using Mutation-With-Momentum

Luke Temby, Peter Vamplew and Adam Berry

Technical Report 2005-02
School of Computing, University of Tasmania, Private Bag 100, Hobart

This report is an extended version of a paper of the same title presented at AI'05: The 18th Australian Joint Conference on Artificial Intelligence, Sydney, Australia, 5-9 Dec 2005.

Abstract: In a canonical genetic algorithm, the reproduction operators (crossover and mutation) are random in nature. The direction of the search carried out by the GA system is driven purely by the bias to fitter individuals in the selection process. Several authors have proposed the use of directed mutation operators as a means of improving the convergence speed of GAs on problems involving real-valued alleles. This paper proposes a new approach to directed mutation based on the momentum concept commonly used to accelerate the gradient descent training of neural networks. This mutation-with-momentum operator is compared against standard Gaussian mutation across a series of benchmark problems, and is shown to regularly result in rapid improvements in performance during the early generations of the GA. A hybrid system combining the momentum-based and standard mutation operators is shown to outperform either individual approach to mutation across all of the benchmarks.

1 Introduction

In a traditional genetic algorithm (GA) using a bit-string representation, the mutation operator acts primarily to preserve diversity within the population, ensuring that alleles can not be permanently lost from the population. For this type of representation a simple bit-flipping mutation operator is commonly used.

Some problems are more naturally represented using real-valued alleles within each individual's phenotype. For this representation, bit-flipping mutation is no longer appropriate, and a more continuous form of mutation is required. The most commonly-applied mutation operator in this context is gaussian mutation. When a gene is selected for mutation, a random value is generated based on a gaussian distribution, and this value is added onto the existing value of that gene. The width of the gaussian is a percentage of the range of possible values for that gene and the resulting combined value is truncated if necessary to ensure it remains in the valid range. The use of the gaussian function ensures that the mutated value is likely to be close to the original value but also allows occasional larger changes, preventing the mutation operator from being overly disruptive whilst still maintaining the ability to break free from local optima.

Gaussian mutation is random in nature, being equally likely to either increase or decrease the value being mutated. In recent years interest has grown in the use of non-random mutation operators, either to improve the speed of the genetic search or the quality of the solutions which are found.

2 Directed Mutation

The theory of natural evolution which was the inspiration for genetic algorithms has been based on the assumption that mutation is a random process. This assumption was brought into question by [1] which appeared to demonstrate that bacteria placed in a hostile environment demonstrated a bias towards beneficial mutations; a phenomenon referred to as 'directed mutation'. Whilst the concept of directed mutation in biology has since been brought into doubt [2], the idea has inspired new techniques in evolutionary computation research, where the accuracy of the biological analogy is less important than the optimisation performance of the system. Most early approaches to directed mutation were based on exploiting known constraints within specific problem domains such as timetabling [3-9]. Whilst these systems demonstrated clear benefits within these domains, the need for domain-knowledge limits their more general applicability.

Partially to address such limitations in traditional directed mutation, more recent literature moves from problem-specific biasing to adaptive approaches [10-15]. The central goal in these dynamic techniques is to shift the responsibility for establishing the direction of mutation away from the expert and to allow these to evolve from within the system. While results are limited, due to the fledgling nature of the

procedure, they show significant promise and carry greater general applicability because of their detachment from domain knowledge. However, such generality comes at the cost of complexity – requiring an increase in the number of system parameters and a significantly more elaborate random-number generator.

[15] proposed the Pointed Directed (PoD) mutation operator. This is a co-evolutionary scheme in which each gene is associated with an additional bit, which dictates the direction of any mutation applied to that gene - this bit determines the sign of the half-gaussian mutation operator which is applied. The direction bits themselves are subject to simple bit-flipping mutation, allowing the search direction to evolve during the operation of the GA. Each gene and its associated direction bit are treated as an indivisible unit for the purposes of crossover. PoD mutation is considerably simpler than previous dynamic directed mutation methods in terms of both the number of parameters and computational cost, and has been shown to outperform gaussian mutation across a range of benchmarks.

3 Mutation-with-momentum

[15] suggests that the success of PoD is due to the implicit, emergent momentum produced by the direction bits - mutation will continue in the same direction which was useful in previous generations. However the magnitude of the mutation is not determined dynamically within PoD. This paper proposes a new dynamic directed mutation algorithm (mutation-with-momentum, or mumentum) based on the concept of explicit momentum, which has previously been explored within gradient-descent training algorithms for neural networks [16]. Momentum involves adding a fraction of the weight change from the previous time-step onto the weight-change calculated at the current time-step. This serves two main roles within a gradient descent algorithm. It allows increased convergence speed in regions of the error-space with a shallow gradient by enabling larger weight changes. It also reduces the likelihood of becoming trapped in a local minima in the error-space.

3.1 Incorporating momentum into mutation

As with PoD, the mutation-with-momentum algorithm requires additional data to be stored for each gene in an individual's genetic representation. This is used to store the momentum term, which records the change which was applied to that gene in its most recent mutation. When the initial population is generated, all momentum terms are initialised to zero, indicating that there has been no prior mutation. The selection and crossover processes are implemented as in a standard genetic algorithm, with the only modification being the requirement that each gene and its associated momentum term are treated as indivisible units during the crossover operation.

Whenever a gene is selected for mutation, that gene's value has both standard gaussian mutation and a proportion of the current momentum term added to it. The momentum term is then updated to reflect this combined mutation value. This process is summarised in the following equations, where $g_{i,o}$ is the original value of gene i, $g_{i,m}$

is the value after mutation, $M_{i,o}$ is the value of the momentum term for gene i, $M_{i,m}$ is the value of the momentum term after mutation, α is the parameter controlling the amount of momentum ($0 \leq \alpha \leq 1$) and G(x) is the gaussian function.

$$g_{i,m} = g_{i,o} + \alpha M_{i,o} + G(x) \quad (1)$$

$$M_{i,m} = g_{i,o} - g_{i,m} \quad (2)$$

To prevent the momentum term from becoming overly large the result of equation 2 may be capped to a maximum magnitude - for the experiments reported in this paper a bound equal to half the range of the alleles was used.

3.2 Hybrid mutation-with-momentum

During early experiments it became evident that whilst mutation-with-momentum produced significant improvements in fitness over early generations, it often had difficulty in performing the 'fine-tuning' required in later generations in order to reach the convergence threshold. Therefore a hybrid system was also developed and tested. This system initially uses the mutation-with-momentum operator for all individuals in the population. Once the population reaches a particular level of performance, the system switches to using the standard gaussian mutation operator for all individuals. For these experiments a simple switching criteria was used - as soon as the best individual in the population achieved a fitness better than a fixed threshold the α parameter was set to zero, thereby eliminating any further use of momentum. We intend to investigate more sophisticated switching criteria in the future.

4 Experimental method

4.1 Implementation and parameters

The GA implementation used for these experiments was GAPlayground [18]. This system was modified to implement the standard gaussian, momentum and hybrid mutation operators. The same settings were used for all three operators (population size of 30, using single-point crossover with a per-gene mutation probability of 0.06). As the range of alleles varies between different benchmarks, the gaussian function's width was set to 1/15 of the range of each allele. The momentum parameter α was fixed at 1. For the hybrid algorithm, the threshold for deactivating the momentum term was set to 0.1 (all the benchmarks used are minimisation tasks, so the hybrid algorithm switched strategies when any individual's fitness fell below the threshold).

For each of the three mutation systems, 100 trials were run on each of seven benchmark problems. A trial was treated as having converged if the best individual in any generation achieved or bettered a threshold of 0.001. In addition a maximum

number of generations was specified for each benchmark - trials which failed to converge within this limit were stopped at that point.

4.2 Benchmark problems

Seven benchmarks were chosen from the GA literature, as illustrated in Figure 1. The first three are drawn from De Jong's function set [18] and are relatively simple tasks, with no local optima. The other four functions were chosen as more challenging tasks, as all exhibit local optima of varying complexity.

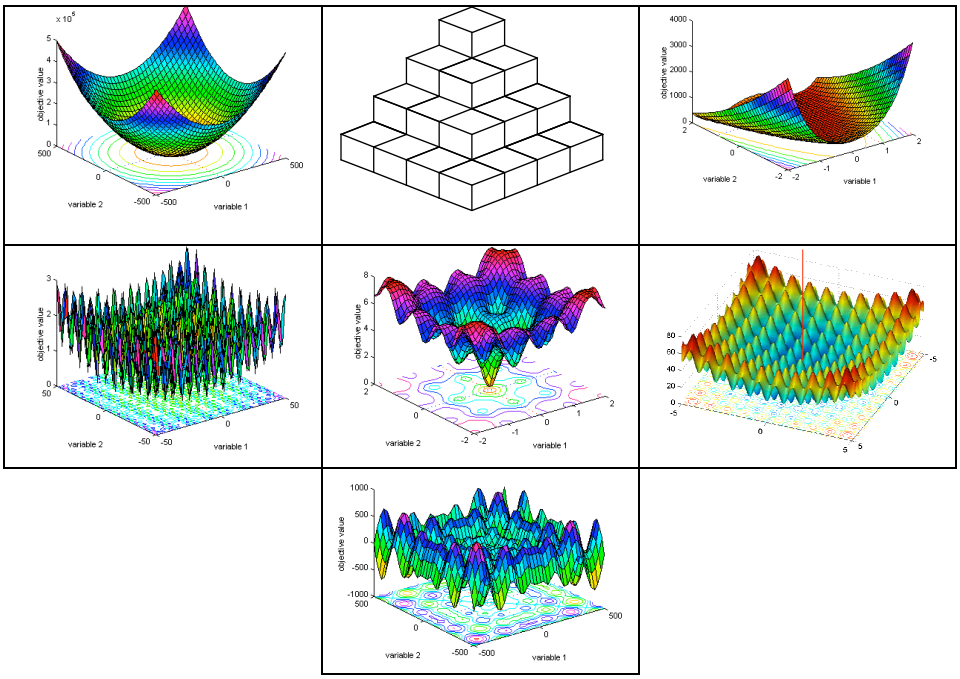

Fig. 1. The seven benchmark functions. Top row (left to right): Sphere, Steps, Rosenbrock; Middle row: Griewangk, Ackley, Rastrigin; Bottom row: Schwefel. For n-dimensional problems, the 2-dimensional instance has been illustrated.

De Jong's Sphere function (equation 3) is the most basic problem. It contains no local optima and provides a smooth gradient towards a broad global optimum. This problem should be easily solvable by a GA. The allele range is -5.12 to +5.12.

$$f(\text{Sphere}) = \sum_{i=1}^{2} x_i^2 \qquad (3)$$

De Jong's Step function (equation 4) contains multiple plateaux. Individuals on the same plateau have equivalent fitness values which can significantly slow the convergence of an optimisation algorithm. The allele range is -5.12 to +5.12.

$$f(\text{Step}) = \sum_{i=1}^{2} \lfloor x_i \rfloor \tag{4}$$

The search space defined by the Rosenbrock function (equation 5) is dominated by a large gradual slope. This slope is raised along one edge to a fine point. Despite the apparent simplicity of this function, it is notoriously hard for genetic algorithms. This is due to the extremely large search space (as defined by an allele range between -2048 and +2048), combined with a relatively small global optimum.

$$f(\text{Rosenbrock}) = 100(x_1^2 - x_2)^2 - (1 - x_1)^2 \tag{5}$$

The Griewangk function (equation 6) is multimodal exhibiting many local optima throughout the search space. The frequency and severity of the optima is decreased as the algorithm approaches the global optimum. Many GAs have difficulty converging on such problems and have a high probability of stagnating early in the process [19].

$$f(\text{Griewangk}) = 1 + \sum_{i=1}^{10}\left(\frac{x_i^2}{4000}\right) - \prod_{i=1}^{10}\left(\cos\left(\frac{x_i}{\sqrt{i}}\right)\right) \tag{6}$$

The Ackley function (equation 7) [20] is a minimisation problem that features deep local minima. This function is notoriously difficult to solve. The function may be expanded to *n* dimensions. In this study10 dimensions were used.

$$f(\text{Ackley}) = 20 + e - 20 * \exp\left(\sqrt{\frac{1}{n} * \sum_{i=1}^{10} x_i^2} - \sum_{i=1}^{10} \cos\left(2\pi x_i\right)\right) \tag{7}$$

The Rastrigin function (equation 8) is another very difficult problem as it defines a very large search space of 20 alleles with values ranging from -5.12 and +5.12. Further, it possesses a very high number of local optima. The combination of these two issues means that many algorithms have great difficulty in achieving an optimal solution. The variables A and ω control the frequency and modulation of the search space. In this study the values of A=10 and ω=2π are used.

$$f(\text{Rastrigin}) = 20A + \sum_{i=1}^{20} x_i^2 - A.\cos\left(\omega.x_i\right) \tag{8}$$

The final benchmark function is the Schwefel function (equation 9). It represents a large search space, with allele values between -500 and +500. The outer region of the search space is riddled with local minima of irregular distribution and severity.

$$f(\text{Schwefel}) = \sum_{i=1}^{10} -x_i . \sin\left(\sqrt{|x_i|}\right) \quad (9)$$

5 Results and Discussion

Table 1 summarises the results achieved by each of the three mutation operators on the test problems. For each problem the table shows the percentage of trials which converged, and the mean number of generations required to achieve that target. For those problems where a significant number of trials failed to converge the table also shows the mean error of the best individual found during each trial. Further insight into the performance of the systems is given by the graphs in Figures 2 to 9, which show the mean error achieved by the best individual in each generation.

		Gaussian	Momentum	Hybrid
Sphere	% of trials converged	100	100	100
	Mean generations to convergence	108	106	71
Steps	% of trials converged	100	100	100
	Mean generations to convergence	174	54	54
Rosenbrock	% of trials converged	82	85	87
	Mean generations to convergence	25,029	22,041	21,912
Griewangk	% of trials converged	0	0	0
	Mean generations to convergence	-	-	-
	Mean error	0.030	0.056	0.031
Ackley	% of trials converged	0	0	0
	Mean generations to convergence	-	-	-
	Mean error	0.007	0.015	0.007
Rastrigin	% of trials converged	7	1	18
	Mean generations to convergence	34,003	47,351	9,405
	Mean error	10.86	0.067	0.008
Schwefel	% of trials converged	0	100	93
	Mean generations to convergence	-	4,450	2,514
	Mean error	786.29	<0.001	3.65

Table 1. Summary of results achieved across 100 trials on each of the benchmark problems.

It can be seen from Figures 2 to 9 that the momentum operator regularly outperforms the gaussian mutation operator in the early generations on each problem. However mutation-with-momentum often has difficulty performing the 'fine-tuning' of the gene values necessary to achieve the convergence criteria, producing overall results similar to that for gaussian mutation. Mutation-with-momentum results in small performance improvements on the Sphere and Rosenbrock functions, but slightly poorer results on the Griewangk and Ackley functions. The results on the Rastrigin function are unclear - gaussian mutation is more likely to converge to a solution, but performs extremely poorly on those trials which do not converge, resulting in a much higher mean error.

The most significant variations between the two mutation operators occur on the Steps and Schwefel functions. The Steps function contains a number of plateaus which pose a major obstacle to a non-directed search. As such it would be expected that momentum would prove beneficial on this problem and this proved to be the case, converging in less than a third of the time required by the gaussian mutation system. The performance improvements on the Schwefel function are even more impressive - the system using gaussian mutation never converges on this problem, whilst the mutation-with-momentum system successfully converges on every trial.

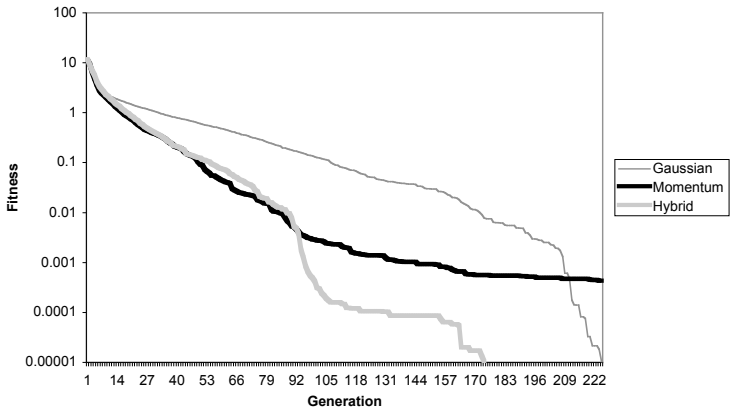

Fig. 2. Mean fitness of the best individual in each generation for each mutation operator on the Sphere problem.

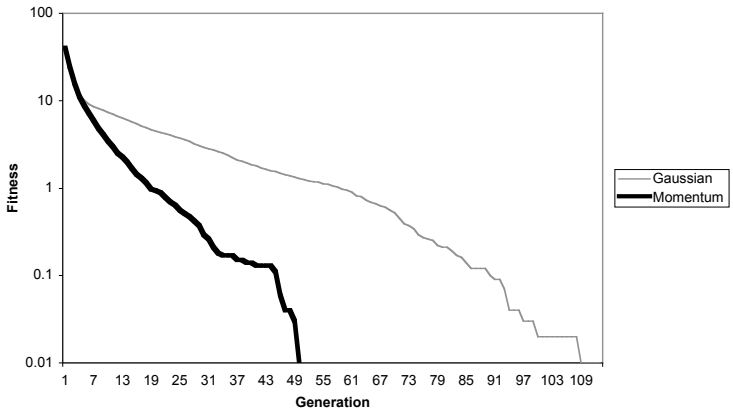

Fig. 3. Mean fitness of the best individual in each generation for each mutation operator on the Steps problem. The hybrid system is not shown as for this problem its results are identical to the momentum system.

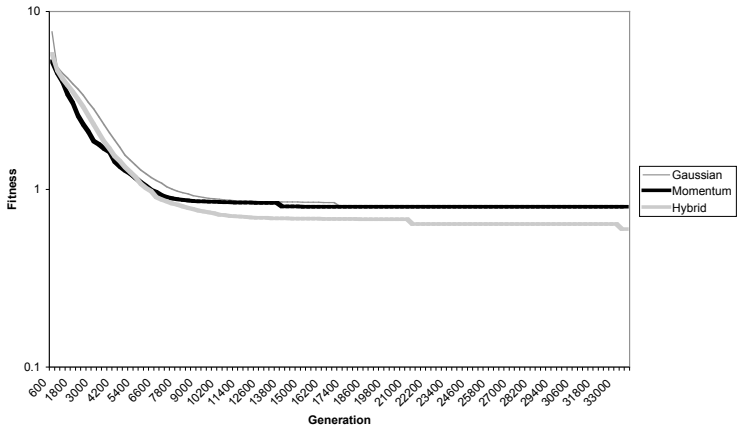

Fig. 4. Mean fitness of the best individual in each generation for each mutation operator on the Rosenbrock problem.

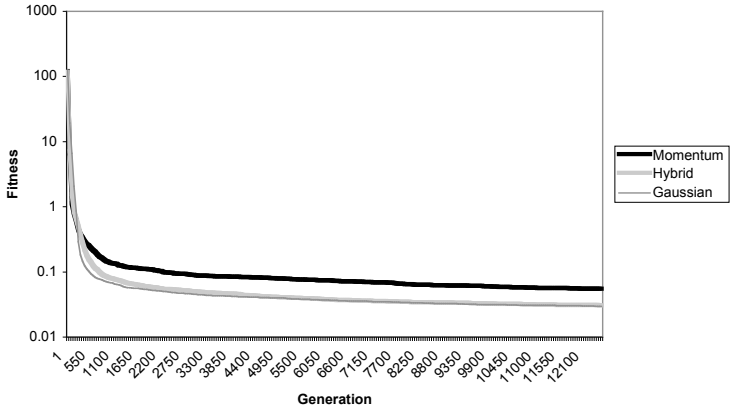

Fig. 5. Mean fitness of the best individual in each generation for each mutation operator on the Griewangk problem.

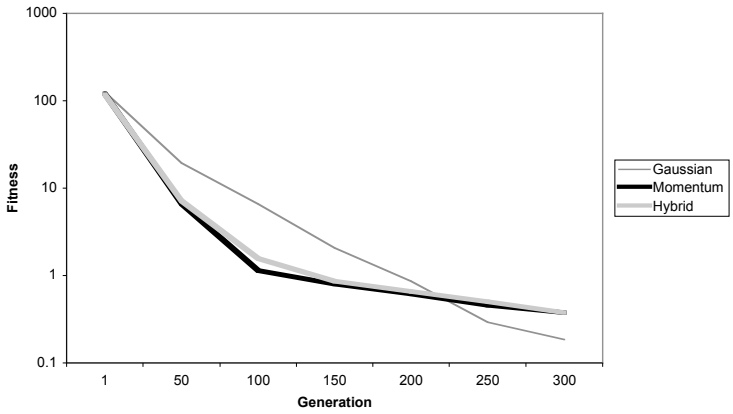

Fig. 6. Mean fitness of the best individual in each generation for each mutation operator on the Griewangk problem, for the first 300 generations only (to make clearer the early performance of the different systems).

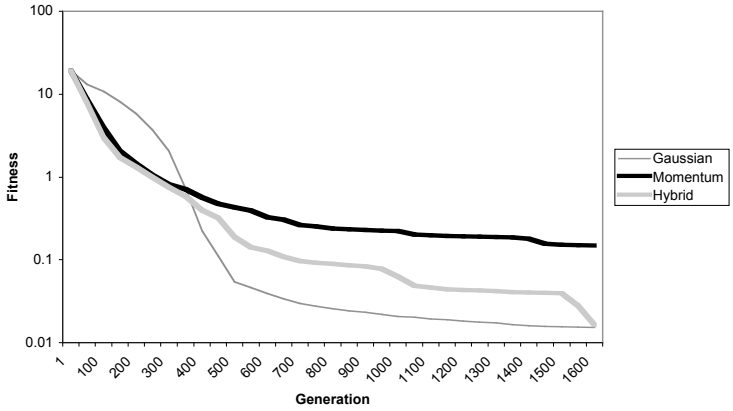

Fig. 7. Mean fitness of the best individual in each generation for each mutation operator on the Ackley problem

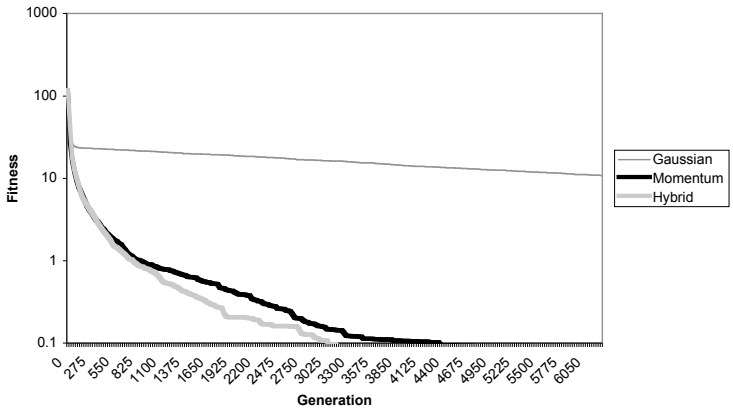

Fig. 8. Mean fitness of the best individual in each generation for each mutation operator on the Rastrigin problem

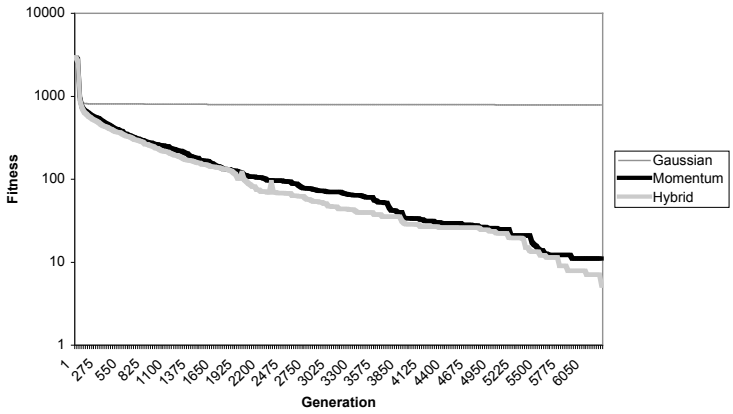

Fig. 9. Mean fitness of the best individual in each generation for each mutation operator on the Schwefel problem

The table and figures also show that the hybrid system successfully merges the early-generation speed of the momentum operator with the fine-tuning capability of gaussian mutation. The hybrid approach equals or outperforms the other two systems in terms of both convergence speed and the ability to find an accurate solution across all of the benchmark problems except for the Schwefel function. For that problem the hybrid system is faster than the momentum system on those trials which do converge, but it does fail to converge in a small percentage of trials whereas the momentum system always converges. In light of the gaussian system's inability to converge on

this problem it would appear that these failures may be due to premature termination of the momentum component of the mutation.

6 Conclusion

In the majority of the problems examined the inclusion of momentum within the mutation process produced beneficial results, ranging from minor improvements (Sphere and Rosenbrock) to major performance improvements (Steps, Rastrigin and Schwefel). On the problems where the momentum term impedes performance (Griewangk and Ackley) the impairment is minor. Even on these trials the momentum term was beneficial early in the evolutionary process as shown in Figures 6 and 7, which would prove useful in any application requiring a near-optimal result to be produced extremely rapidly. It should also be noted that the convergence criteria of 0.001 is quite stringent - a more generous choice of threshold would have further emphasised the benefits of the momentum operator.

The hybrid system further improved performance by exploiting the relative strengths of the momentum and gaussian mutation systems. It significantly outperformed non-directed mutation on five of the benchmark problems, producing benefits both in the ability to converge to a near-optimal solution, and the speed of that convergence. At worst it produced results equal to that of the non-directed mutation operator. This is achieved at little additional computational cost and with no tuning of additional parameters.

7 Future work

The major limitation of this work is the global nature of the test used to switch mutation operators within the hybrid system. Currently a single solution with good fitness will trigger the whole system to cease using momentum when the bulk of individuals are still far from optimal. This can be seen in the Schwefel function, where a premature switch from momentum to guassian mutation may be leading to stagnation. An algorithm that switched the mutation method on an individual basis may provide more reliable results. That is, the particular fit individual and its descendents would use the gaussian system for final convergence but other individuals in the population, whose fitness is poorer, would still use momentum. We also intend to investigate switching strategies which do not rely on static fitness thresholds, to reduce the need for a priori knowledge of the problem domain.

It is also important to test the mutation-with-momentum operator on more realistic problems. One area where optimisation of real-valued alleles is extremely important is in the genetic optimisation of the weights of neural networks, such as in the SANE neuro-evolution algorithm [21]. Given the success of momentum in non-genetic training of neural weights, it may prove fruitful to apply mutation-with-momentum to the task of finding such weights via evolutionary means.

References

1 Cairns, J, Overbaugh, J, and Miller, S (1988), The Origin of Mutants, *Nature*, no. 335, pp. 142-5.

2 Hall, B (1990), Spontaneous point mutations that occur more often when advantageous than when neutral, *Genetics*, no. 126, pp. 5-16.

3 Craenen, B.G.W. (1998), *An Experimental Comparison of Three Different Heuristic GAs for Solving Constraint Satisfaction Problems*, Internal Report 98-21, Department of Computer Science, Leiden University.

4 Burke, E.K. and Petrovic, S. (2002), *Recent Research Directions in Automated Timetabling*. European Journal of Operational Research.

5 Ross, P., Corne, D., and Fang, H. (1994), *Improving Evolutionary Timetabling with Delta Evaluation and Directed Mutation*. PPSN III.

6 Paechter, B., Ranking, R.C., Cumming, A., Fogarty, T.C. (1998), *Timetabling the Classes of an Entire University with an Evolutionary Algorithm*. PPSN V, LNCS, 1998. p. 865-874.

7 Wiggins, G.A., *The Use of Constraint Systems for Musical Compositions*. ECAI98 Workshop on Constraints and Artistic Applications.

8 Li, L., Louis, S.J., and Brune, J.N. (1995), *Application of Genetic Algorithms to 2D Velocity Inversion of Seismic Refraction Data*. Proceedings of the Third Golden West International Conference on Intelligent Systems.

9 Baron, P.J., Fisher, R.B., Mill, F., Sherlock, A., Tuson, A.L. (1997), *A Voxel-Based Representation for the Evolutionary Shape Optimisation of a Simplified Beam: A Case-Study of a Problem-Centred Approach to Genetic Operator Design*. Soft Computing in Engineering Design and Manufacturing.

10 Schwefel, H.-P. (1994), *Evolution and Optimum Seeking*, New York: John Wiley & Sons.

11 Hilderbrand, L. (2002), *Asymmetrische Evolutionsstrategien*. PhD Thesis, Department of Computer Science, University of Dortmund.

12 Berlik, S. (2003), *A Polymorphical Mutation Operator for Evolution Strategies*. In Proceedings of the International Conference EUSFLAT, p. 502-505.

13 Berlik, S. (2004), *A Directed Mutation Framework for Evolutionary Algorithms*. Proceedings of the International Conference on Soft Computing, MENDEL.

14 Berlik, S., (2004), *Directed Mutation by Means of the Skew-Normal Distribution*. International Conference on Computational Intelligence, FUZZY DAYS.

15 Berry, A. and Vamplew, P. (2004), 'PoD Can Mutate: A Simple Dynamic Directed Mutation Approach for Genetic Algorithms', Proceedings of AISAT2004: International Conference on Artificial Intelligence in Science and Technology, Hobart, Tasmania.

16 Plaut, D. C., Nowlan, S. J., and Hinton, G. E. (1986). Experiments on learning by back propagation (Technical Report CMU-CS-86-126), Computer Science Department, Carnegie-Mellon University, Pittsburgh, PA.

17 Dolan, A (1998), *Artificial Life and Other Experiments*, viewed April 2005, <http://www.aridolan.com/default.aspx>.

18 De Jong, K (1975), 'An analysis of the behaviour of a class of genetic adaptive systems', PhD thesis, University of Michigan.

19 Digalakis, J. & Margaritis, K. (2000), 'An Experimental Study of Benchmarking Functions for Genetic Algorithms', paper presented to Proceedings of IEEE International Conference on Systems, Man, and Cybernetics, Nashville, Tennessee.

20 Ackley, D.H., (1987), *A connectionist machine for genetic hillclimbing*, Kluwer Academic Publishers, Boston

21 Moriarty, D. and Miikkulainen, R. (1997). Forming neural networks through efficient and adaptive coevolution. *Evolutionary Computation*, 5:373–399.