Non-stationary Function Optimization Using Polygenic Inheritance

Conor Ryan, J.J. Collins, and David Wallin

CSIS, University of Limerick
Limerick, Ireland
{conor.ryan,j.j.collins,david.wallin}@ul.ie

Abstract. Non-stationary function optimization has proved a difficult area for Genetic Algorithms. Standard haploid populations find it difficult to track a moving target, and tend to converge on a local optimum that appears early in a run.
It is generally accepted that diploid GAs can cope with these problems because they have a *genetic memory*, that is, genes that may be required in the future are maintained in the current population. This paper describes a haploid GA that appears to have this property, through the use of Polygenic Inheritance. Polygenic inheritance differs from most implementations of GAs in that several genes contribute to each phenotypic trait.
Two non-stationary function optimization problems from the literature are described, and a number of comparisons performed. We show that Polygenic inheritance enjoys all the advantages normally associated with diploid structures, with none of the usual costs, such as complex crossover mechanisms, huge mutation rates or ambiguity in the mapping process.

1 Introduction

Natural Evolution is characterized by the adaptation of life forms to ever changing and sometimes hostile environments. Artificial Evolution, on the other hand, is usually applied to static environments, with fixed training and test cases. While Artificial Evolution has enjoyed much success with these fixed environments, there is a school of thought [6,1,4] that if Artificial Evolution were successfully applied to changing environemnts, the search power could improve.

This kind of behaviour could be useful in Genetic Algorithms for a number of reasons. Some problems exhibit shifting target functions [5,7,10], often with a periodic nature. If using a haploid genotype, it is generally better to start evolution from scratch after encountering one of these changes, but if one could force the existing population to adapt, a satisfactory solution could be found much more quickly. This would be particularly useful if the change had a cyclic nature, as the population could be expected to react more quickly to subsequent changes.

If a population could adapt to environmental changes, it would also be possible to use on-line GAs, where the fitness function could vary over time as

E. Cantú-Paz et al. (Eds.): GECCO 2003, LNCS 2724, pp. 1320–1331, 2003.

conditions change. Indeed, it is possible that the population could adapt to previously unforeseen circumstances, as these populations tend to contain considerably more diversity than standard ones, even after lengthy periods of stability.

The "Red Queen Hypothesis", which was observed by Hillis [6], states that evolution is more powerful when the species evolving is involved in an "evolutionary arms race", that is, the problem being tackled gets progressively more difficult, as both problem and species evolve. Another reason why one might wish to employ a capricious environment could be if one was tackling a problem with an infeasibly large training set. Much time could be saved by training the population on various segments of it at a time, and one could reasonably hope that important lessons learnt on one segment will not be forgotten once it has been replaced with another set.

Most higher life forms [3], from plants up through animals to humans, use diploid genetic structures, that is, chromosomes which contain two genes at each locus. Indeed, many crops, and in particular, commercially developed crops, have polyploid structures - that is, three or more genes at each locus. It is thought that these structures have enabled populations to react to changes in the environment [3,5,10], in particular changes that bring about situations that existed in the species history. Species often exhibit a "genetic memory" [10], where it appears that genes that were once useful can be kept in the population - although rarely apparent in the phenotype - until they are needed again, and this allows the population as a whole to react more quickly to changes in the environment.

Historically, most algorithms used in these types of problems have employed a diploid representation scheme, in which two copies of each gene is maintained. However, as described in Sect. 2.1, there are some issues with this type of representation. This paper describes a haploid scheme which shares many of the characteristics of diploid schemes, but with few of the costs. It then compares this scheme, known as *Shades*, to a well known diploidy representation.

The layout of the paper is as follows. Section 2 gives a brief introduction to diploidy, and some of the more well known systems, before illustrating some of the issues with their implementation. The proposed Shades scheme is described in Sect. 3. In Sect. 4 we describe the two problems, Ošmera's dynamic problem and the constrained knapsack problem, on which our analysis are based. We then proceed with an empirical analysis of the results of the Shades scheme and compare its performance against a haploid and two diploid schemes in Sect. 5.

2 Background

Diploid GAs maintain two copies of each gene, known as alleles, typically choosing one of them to be expressed as the phenotype. In Mendel's experiments, two alleles were described for each gene, one *recessive* and the other *dominant*. In the case of a homozygous location, i.e. both forms the same, the expressed allele is selected at random. However, in the case of a heterozygous location, where there are two different forms of the gene, the dominant form is expressed, as in Fig. 1. Diploidy can shield genes from selection by holding them in abeyance, i.e.

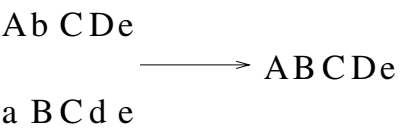

Fig. 1. A simple dominance scheme. Upper case letters represent dominant genes, lower case recessive ones. On the left are a pair of chromosomes, while the right illustrates the expressed alleles

the gene is present but not expressed in the phenotype. During crossover, one of the genes from each locus is passed on so that in this manner, even though a gene may not have been expressed, it can still be inherited by an offspring. This endows the population with a "genetic memory" [10], where genes that were once useful can be kept in the population until they are needed again, and this allows the population as a whole to react more quickly to changes in the environment.

2.1 Previous Work

Goldberg [5] reported on a number of early diploid schemes. Unfortunately, each of these contains an inherent bias, as the resolution of heterozygous pair requires one gene to be chosen over the other.

An alternative to this scheme was proposed by Ng & Wong [10], the Dominance Change Mechanism, in which the dominance relationship between alleles could change over time. Although Lewis et al. [8] showed that it was able to track changes in an environment even after enormously long static periods, the system is dependent on hyper mutation, which must be triggered by a user-specified change in fitness. Typically, this change in fitness is relatively large, in the order of 20%, which means that not only must one be familiar with the environment being examined, but that the system is not suitable for domains in which the changes are relatively small.

2.2 Diploidy without Dominance

A different approach to diploidy avoids the use of dominance by randomly choosing which chromosome to choose a bit from [11]. The intention in this scheme is to let a bias evolve, so, in a locus in which a 1 contributes to the overall fitness, it is likely that a homozygous 11 pair will appear, and similar for the case where a 0 is required. In the case where the most desirable bit varies from generation to generation, a heterozygous pair is expected. Thus, the scheme can be summarised as

$$f(x) = rand(f(x'), f(x''))$$

where $f(x)$ is the phenotype, $f(x')$ the first chromosome and $f(x'')$ the second chromosome. This system is known as Random Diploidy [11]. A second proposal

offered by Ošmera [11] was to perform a logical XOR on the pair of chromosomes. In this scheme, a heterozygous pair produce a 1, while a homozygous pair produce a 0. This scheme is interesting in that it has two representations for each phenotype, e.g. in the case of 1, the two are 10 and 01. This is useful because it means it is possible to mate two individuals of identical phenotypes producing an offspring of a different phenotype, a property which is crucial if a diploidy scheme is to be able to adapt. The XOR scheme can be summarised as below

$$f(x) = f(x') \bigoplus f(x'')$$

This scheme was tested on an objective function which varied with time, which is described in Sect. 4.

3 Polygenic Inheritance – The Shades Scheme

There is no particular reason why any trait should be controlled by a single gene, or gene pair. The first instance of this in natural biology was discovered in 1909 by Nilsson-Ehle [12] when he showed that the kernel colour in wheat, an additive trait, was in fact managed by two pairs of genes. Inheritance of genes of this type is known as polygenic inheritance.

An example of this additive effect comes from Pai [12], who used Four o'clock flowers to illustrate the point. In these flowers, the colour is controlled by a gene pair, and there are two alleles, G_r which corresponds to a red gene, and G_w which corresponds to a white gene. If the pair is G_rG_r the flower is red, while the flower is white if the pair is G_wG_w. If, however, a white parent and a red parent are mated, all the children will have the heterozygous G_rG_w pair. Unlike the other schemes where a conflict is resolved by choosing one of the alleles, *both* contribute to the colour, resulting in an intermediate form, pink.

Using two genes to control each phenotypic trait, where each gene contributes a "shade" of 1 to the trait. We are, depending on the threshold level, likely to get a phenotype of value 0, if the composite shade of the genes is a light shade, or, if the composite shade is a darker shade of 1, the phenotype will likely be 1. We refer to this style of additive polygenic inheritance as *Shades.*

Figure 2 illustrates the calculation of the *shade of one* for different genotypic combinations. In this system, there are three alleles, namely A, B and C, with shades of 0, 1 and 2 respectively. The shade for a locus is simply the sum of its alleles. A sum less than 2 gives a phenotype of 0, while a sum greater than

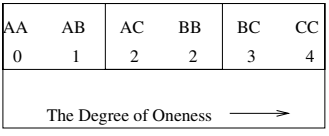

Fig. 2. Varying shades of oneness

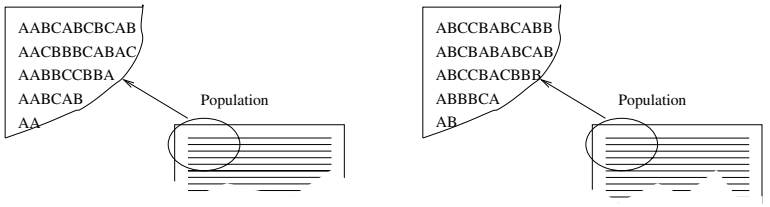

Fig. 3. Traits in a population being fixed due to lack of diversity in homozygous pair (left) and heterozygous pair (right)

2 results in a phenotype of 1. When the total is exactly two, we say that the phenotype is in a "grey area" and the value is chosen randomly.

There is no limit as to the number of genes could be involved in generating a trait, and the system can also represent more than two phenotypes. To distinguish between the different implementations of Shades, we append the number of genes involved with a dash, giving us Shades−2 and Shades−3.

Using the polygenic system, the population can still fixate on certain alleles. Figure 3 illustrates the problem using Shades−2 as an example. Once every individual in the population has a homozygous pair for a particular trait, it cannot change without mutation. In this case, all individuals have a homozygous AA pair controlling the first trait. The pairs that can cause this problem are AA and CC, as these are at the extreme of the phenotypic space.

A second problem also occurs at a pair-level. Even the heterozygous pairs can fixate, due to each pair having its genes in the same order, also illustrated in Fig. 3. This is a particularly interesting problem, as a simple examination of the gene distribution would suggest that there are ample numbers of each gene to permit a change of phenotype. Again, taking the example in Fig. 3, the gene pair controlling the first trait (AB) is identical throughout the population. Despite being heterozygous, the phenotype cannot change, due to the positions of the alleles. If two parents, both of the form AB mated, one would expect that the possible offspring would be AA, AB and BB. However, the fixed order of haploid crossover prevents this from happening. All children will have A in the first position and a B in the second, which effectively stops evolution of that pair. Both these problems were solved by the introduction of simple operators.

3.1 The Force Operator

Despite the continuous phenotypic space introduced by the additive effect of polygenic inheritance, only those shades near the threshold or grey area have the ability to quickly adapt to an environmental change. To alleviate this, the **force** operator is introduced. **Force** is applied to any homozygous pair that occurs at an extreme of the phenotypic space, and forces the mutation of one of the alleles. The mutation is always such that while the phenotype does *not* change, but if the pair is crossed over with an identical pair, the parents can produce

offspring of a different phenotype. It is still possible for them to produce offspring of the same phenotype as themselves. **Force** is applied to every occurrence of extreme homozygotes in Shades scheme. This is similar to the phenomena of hyper mutation in bacteria, where the level of mutation of certain areas of their chromosome can effectively be marked to be mutated than other areas.

3.2 The Perturb Operator

Perturb operates in a similar manner to the rather unpopular inversion operator, but at a much more limited scale of disruption, hence its gentler title. **Perturb** swaps a pair that control a single trait, and therefore doesn't affect the phenotype. **Perturb** is applied randomly to possible candidate locations, and helps alleviate the fixed position problem. Neither the **force** or the **perturb** operator modify the phenotype of the individual affected.

3.3 Shades−3

Using three alleles gives us a situation as in Table 1. Again we have the grey area, and extreme values as in Shades−2.

In this case, **force** operates slightly differently. Phenotypes at the extremities have their shades adjusted up or down by 2, those near the extremities are adjusted up or down by 1. This ensures that if the individual subsequently mates with another individual of the same phenotype, they can produce an offspring of a different appearance. Table 2 shows the action taken on each genotype in the Shades−3 scheme. Notice that the order for **force** is not important.

For **perturb**, two of the three genes controlling a trait are selected at random and swapped. Notice that no attempt is made to tune the level of mutation, nor is it related the level of change in the environment.

Table 1. Genotype to phenotype mapping using 3 alleles, the order of alleles doesn't affect the shade

Combination	AAA	AAB	AAC	ABB	ABC	BBB	ACC	BBC	BCC	CCC
Shade	0	1	2	2	3	3	4	4	5	6
Phenotype	0	0	0	0			1	1	1	1

Table 2. Using the force operator on three alleles. For the extreme values, two mutations take place

Genotype	After Force
AAA	ABB or AAC
AAB	ABB or AAC
BCC	BBC or ACC
CCC	BBC or ACC

4 Experiments

In previous work, Shades−3 has been shown to outperform both the triallelic and the Dominance Change Mechanism schemes for diploidy [16]. It was quicker to react to a change in the environment and consistently found new optima following a change. We now proceed to empirically analyze the performance of haploid and the two diploid schemes from Sect. 2.2 against the Shades genotypes using two dynamic problem domains.

The first problem is taken from Ošmera [11] which involves trying to track a constantly moving target, while the second is Goldberg's classic constrained knapsack problem [5] in which the goal periodically shifts. The working hypothesis to be verified or invalidated is that Shades will yield similar performance to diploidy without incurring domain specific overheads such as selection of genotype to phenotype mapping or a dominance change mechanism.

4.1 Ošmera's Dynamic Problem Domain I

Ošmera [11] presented two dynamic problems, the first of which is specified by the following function:

$$g_1(x,t) = 1 - e^{-200(x-c(t))^2} \quad \text{where} \quad c(t) = 0.04(\lfloor t/20 \rfloor) \tag{1}$$

$x \in \{0.000, \ldots, 2.000\}$ and $t \in \{0, 1000\}$, where t is the time step, each of which is equal to one generation. Figure 4 shows a three dimensional view of the domain, with x varying linearly with t. Solutions to the problem are derived by finding a value of x at time t that minimizes $g(x,t)$. Examination of the problem shows that it displays quite a predictable pattern, with the possibility of certain loci becoming fixed at 1 without incurring any penalties.

A gray coded bit string of length 31 was used to represent the parameter, and normalized to yield a value in the range $\{0, 2\}$. A generation gap of 1.0 was set as the replacement policy with test parameters of $P_{mut} = 0.01, P_{cross} = 0.7$, and population size $p = 400$. Remainder stochastic sampling without replacement as the selection scheme.

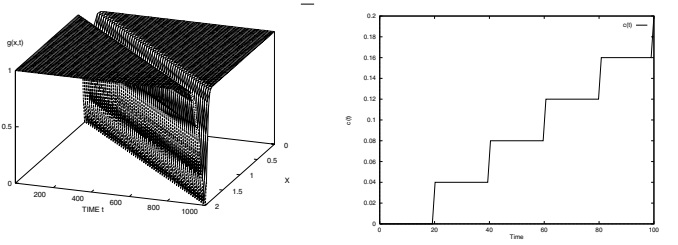

Fig. 4. Ošmera's dynamic problem domain (left), and behaviour of $c(t)$ over time (right)

4.2 The Knapsack Problem

The constrained 17-Object 0-1 knapsack problem is taken from Goldberg & Smith [5], and the weighting is used as specified therein. The population consisted of 150 individuals, made up of 17 pairs of 1 bit genes. A uniform crossover operator was used with a probability of $P_{cross} = 0.5$ for both shades−2 and shades−3, as were $P_{perturb} = 0.175$, while mutation was implemented at a rate of 0.001. Remainder stochastic sampling without replacement was also used here.

All experiments were run for 400 generations, with a period of 15 generations, and the fitness varied between 87 and 71, depending on the oscillation.

5 Results

Results are presented in two different ways. An overview is given in the following tables, which indicate the average error, while more detailed graphs give more detailed view. In both experiments, the best performer was Shades, although, surprisingly, Shades−2 outperforms Shades−3 in the Knapsack problem.

5.1 Ošmera's Function

Table 3 shows the performance of each of the approaches on Ošmera's test function. One surprising aspect is the good performance shown by the haploid scheme, however, this is probably due to the monotonically increasing nature of the problem, which means that the changes are not only regular, but also relatively small.

Figures 5, 6, and 7 indicate the manner in which each of the approaches track the error. Notice how similar the behaviour of haploid and random diploid are, with the greatest difference being the number of large errors made by the latter, giving it a larger standard deviation than the haploid system. XOR diploid, on the other hand, is consistently better across the entire run.

The performances of the two shades systems are shown in Fig. 7. In this case, we can see that shades-2 performs similarly to random diploid, but with a greater number of relatively large errors. The best performer on this problem was shades−3.

5.2 Knapsack Function

Table 4 shows the relative performances on the Knapsack problem. In this case, the haploid approach reverts to the expected, and fails to track the environment, generating an average error of almost 50%. The two diploid perform considerably better than it, but, again, Shades shows the best peformance. Curiously, in this case, Shades−2 actually outperforms Shades−3. The mean tracking error for each method are plotted in Figs. 8, 9, and 10.

In this case, the haploid scheme fails to map to the lower weight because, even by the first oscillation, it doesn't contain enough diversity to make the

Table 3. Mean tracking error

Ošmera Test Function I		
Genotype	**Diploid Mapping**	**% Error**
Haploid	–	0.043
Diploid	Random	0.026
	XOR	0.173
	Random	0.079
	XOR	0.030
Shades−2	–	0.025
Shades−3	–	0.016

Table 4. Mean tracking error

Constrained Knapsack Problem			
Genotype	**Diploid Mapping**	**Mean Fit.**	**% Error**
Haploid	–	44.67	49.38
Diploid	Random	77.89	3.15
	XOR	73.44	7.61
Shades−2	–	78.910	1.746
Shades−3	–	77.867	3.074

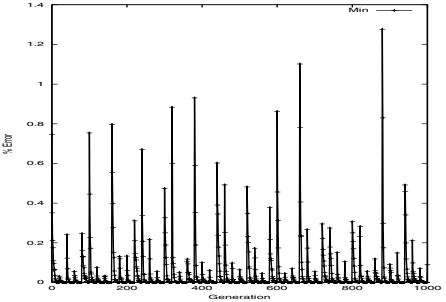

Fig. 5. Ošmera's dynamic problem. Error plotted against generation for haploid

change. The relative performances of the XOR and random diploid systems are inverted with this problem, with random performing substantially better, and successfully tracking the problem after the first few oscillations. XOR, on the other hand, never manages to track the *higher* weight. It does, however, evolve towards it, but the problem changes too quickly for it to get there.

The relative performances of shades−2 and shades−3 are also inverted, although both succeed in finding the target on each oscillation. As indicated in Table 4, the two shades methods still perform better than the other methods, although there is no statistically significant difference between random diploid and shades-3 on this problem.

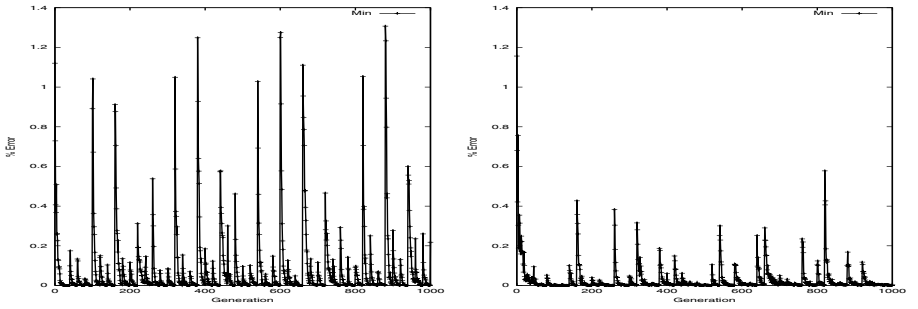

Fig. 6. Ošmera's dynamic problem. Error plotted against generation for random diploid (left) and XOR diploid (right)

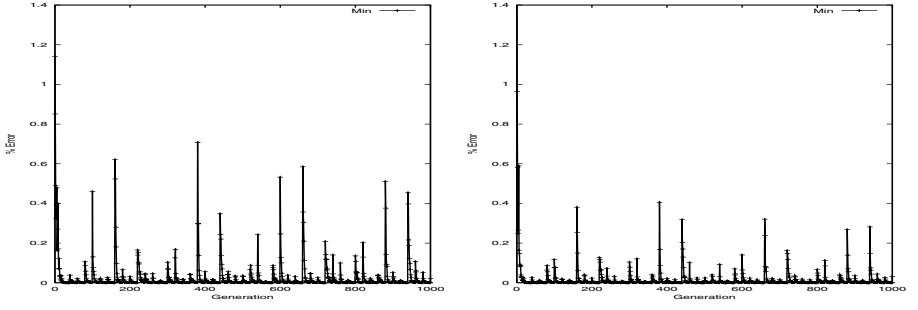

Fig. 7. Ošmera's dynamic problem. Error plotted against generation for shades−2 (left) and shades−3 (right)

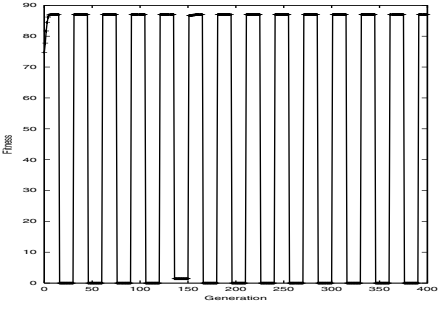

Fig. 8. The Knapsack problem. Fitness plotted against generation for haploid. Notice the different scale required compared to the plots for diploid & shades

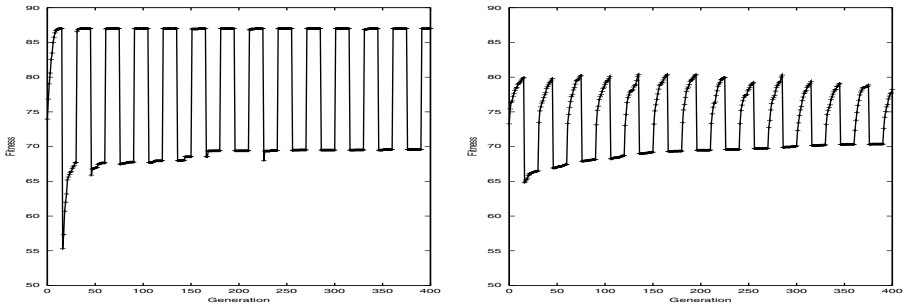

Fig. 9. The Knapsack problem. Fitness plotted against generation for random diploid (left) and XOR diploid (right)

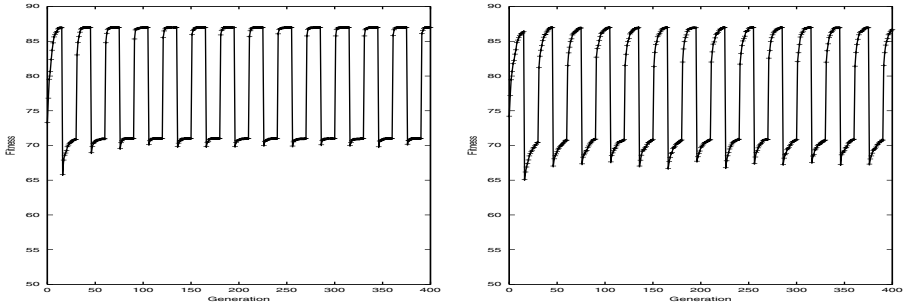

Fig. 10. The Knapsack problem. Fitness plotted against generation for shades−2 (left) and shades−3 (right)

6 Conclusions and Future Work

Shades, a new version of haploidy, coarsely modeled on naturally occurring phenomena, has been described, as have its associated functions. We have tested it and two diploidy methods on two standard benchmark problems, and demonstrated that it outperforms standard haploid and two diploid schemes on these. Perhaps somewhat surprisingly, on one of these problems, a standard haploid scheme performed better than one of the standard diploid schemes. It has been shown that a diploid genetic structure is not a prerequisite for a population to survive in a dynamic problem space.

Despite the fact that Shades−2 outperformed Shades−3 on the Knapsack problem, our belief is that the more genes involved in the phenotype mapping increases the performance on dynamic problem domains. Although there probably is a law of diminishing returns applicable to this, it should be investigated further.

The potential benefits of using dynamic environments have yet to be realized. By providing an unbiased scheme that can quickly react to environmental changes, we hope to be able to convert static problems to dynamic ones.

References

1. Jürgen Branke, "Evolutionary Approaches to Dynamic Optimization", In *Evolutionary Algorithms for Dynamic Optimization Problems*, 2001, pp. 27–30.
2. J.J. Collins and M. Eaton, "Genocodes for genetic algorithms", In *Procs. of Mendel'97*, 1997, pp. 23–30.
3. G. Elseth and K. Baumgardner, Principles of Modern Genetics, West Publishing Company, 1995.
4. A. Ghosh, S. Tsutsui and H. Tanaka, "Function Optimization in Nonstationary Environment using Steady State Genetic Algorithms with Aging of Individuals", In *Proc. of the 1998 IEEE International Conference on Evolutionary Computation.*
5. D. Goldberg and R.E. Smith, "Nonstationary function optimisation with dominance and diploidy", In *Procs. of ICGA2*, 1987, pp. 59–68.
6. D. Hillis, "Coevolving parasites improves simulated evolution as an optmisation procedure", In *Proceedings of ALife II*, 1989.
7. R.B. Hollstein, "Artificial genetic adaptation in computer control systems", PhD Dissertation, University of Michigan, 1971.
8. J. Lewis, E. Hart, and G. Ritchie, "A Comparison of Dominance Mechanisms and Simple Mutation on Non-stationary Problems", In *Proc. of Parallel Problem Solving from Nature – PPSN V*, 1998, pp. 139–148.
9. K.E. Mathias and L.D. Whitley, "Transforming the search space with gray coding", In *Proc. of IEEE Int. Conf. on Evolutionary Computing.*
10. K. Ng and K. Wong, "A new diploid scheme and dominance change mechanism for non-stationary function optimisation", In *Proc. of ICGA-5*, 1995.
11. P. Ošmera, V. Kvasnička and J. Pospìchal, "Genetic algorithms with diploid chromosomes", In *Proc. of Mendel '97*, 1997, pp. 111–116.
12. A. Pai, Foundations of Genetics : A Science for Society. McGraw-Hill, 1989.
13. M. Ridley, The Red Queen: Sex and the Evolution of Human Nature, Viking London, 1993.
14. C. Ryan, "The Degree of Oneness", In *Proceedings of the 2nd Workshop on Soft Computing*, 1996.
15. C. Ryan, "Reducing Premature Convergence in Evolutionary Algorithms", PhD Dissertation, University College Cork, Ireland, 1996.
16. C. Ryan, "Shades : A Polygenic Inheritance Scheme". In *Proceedings of Mendel '97*, 1997, pp. 140–147.
17. C. Ryan, and J.J. Collins. "Polygenic Inheritance – A Haploid Scheme that Can Outperform Diploidy", In *Proc. of Parallel Problem Solving from Nature – PPSN V*, 1998, pp. 178–187.