

Solving Fully Deceptive Problems in Changing Environments

Seamus Hill¹
and Colm O’Riordan¹

Computational Intelligence Research Group
National University of Ireland Galway,
`seamus.hill@nuigalway.ie`
`colmor@nuigalway.ie`

Abstract. This paper examines the performance of a Genetic Algorithm (GA) containing a multi-layered mapping between the *genotype* and the *phenotype* against that of a standard GA over a changing, fully deceptive landscape. The motivation for introducing multiple-layers between the *genotype* and the *phenotype* is to examine the effect of this naturally inspired mapping in maintaining diversity in the population. While running a GA on a static environment its *adaptiveness* is difficult to test, however by allowing the environment to change, we have an opportunity to examine the ability of a GA to readjust the search and look for a new global optimum. This offers us an opportunity to study more closely, the performance of both GAs in adapting to moving landscapes, which exist in many real world problems. The results indicate that the multi-layered mapping GA (mmGA) has the ability to avoid premature convergence on the deceptive attractor and may prove useful in searching particular classes of problems while allowing the use of standard operators.

1 Introduction

Genetic Algorithms (GAs) [11] [5], are probably the best known of the biologically inspired search methods. The standard GA uses a selection policy based on feedback from the environment coupled with recombination and mutation and quickly move towards removing diversity from the population [9]. In GAs the *genotype space* can be defined using distance metrics based on an operator (i.e. single bit-flip for binary spaces) to define a neighbourhood structure over the population of solutions. The phenotype on the other hand can be viewed as the final result, or a search space based on distance metrics between solutions. The neighbourhood structure contained within this space may bear little relationship to the neighbourhood found in the genotype space according to the complexity of the mapping from the representation to the solution [3].

Standard GAs have been used to study population level problems, they cannot effectively replicate genome dynamics because of their *genotype-phenotype* mapping. This is because their genes are position dependent, in other words the contribution of each allele relies on its locus and required that both the number and the allele order be predefined. Genomes in nature, on the other hand have a flexible phenotypic structure as there exists an intermediate level between the phenotype and the genotype i.e. proteins. The GA proposed by the authors also uses binary strings as a representation; however because the multi-layered relationship which exists between the *genotype* and the *phenotype*, the effect of the allele relates not only to its position but also to the allele adjacent to it and in turn with a number of alleles in its neighbourhood, with similar alleles in similar positions having completely different meanings as the algorithm moves from the concept of *DNA* through *transcription* and onto *translation*. Therefore, what is viewed at a *genome* level can appear very different at a *phenotypic* level, as it is not the order of the genome encodings that is important but rather its “meaning”. In other words, a single bit ***1**** at a genome level can have a number (N) of different meanings at the *phenotypic* level. This number (N) is a function of the number of mappings available and the size of of the *protein* alphabet.

The motivation for attempting to create a GA which includes a number of biologically plausible concepts and yet provides a framework based on a binary string, stems from the desire to create a GA which has a multi-layered mapping between the *genotype* and the *phenotype* which is biologically inspired and which differs from previous research and to test how traditional GA operators operate within this environment. A number GAs have attempted to introduce a notion of biological plausibility, these include; Messy GAs [6], Virtual Virus [2], the Proportional GA [17] and the RBF-Gene Model [12]. By running a GA on a static environment its *adaptiveness* is difficult to test. However, by changing the environment we have an opportunity to examine the ability of a GA to readjust the search and look for a new global optimum. This offers us an ability to examine more closely, the performance of both GAs in the face of moving landscapes, which exist in many real world problems.

The purpose of this paper is to examine the performance of a standard GA over a fully deceptive changing environment as GAs are suited to solving problems where the fitness landscape changes over time [10], with that of multi-layered mapping GA (mmGA). The aim is to examine whether or not the mmGA has the ability to out perform a standard GA over the changing thirty-bit fully deceptive problem and to see if the multiple mapping contained in the mmGA can make a difference by maintaining diversity within the population. The paper is organised as follows: section 2 discusses deception and how it relates to GAs. Section 3 describes the Multi-layered Mapping Genetic Algorithm (mmGA), while section 4 discusses the experiments and results and section 5 outlines the conclusions and possible future work.

2 Deception

Deception is often used in testing GAs, as it creates problems which are classified as *GA hard*, which implies that they have the ability to mislead the search [16]. A problem can be considered deceptive if particular hyperplanes lead the search away from the global optimum [5]. Previous research into the use of deceptive problems with GAs include [7], [14], [8] and [16]. A Minimal Deceptive Problem was introduced by Goldberg [4], based upon previous work carried out by Bethke [1], in order to develop an enhanced understanding of problems which are likely to prove difficult for a GA to solve. The *schema* theory [11] helps explain how GAs search difficult search spaces by altering the rate of sampling of hyperplanes [11][5] and describes how low level building blocks relate to the hyperplanes and are recombined moving the search towards a solution which is above the average fitness. When examining a fully deceptive order-3 problem, the information contained in the hyperplanes represented by order-1 and order-2 schemata will lead the GA away from the global optimum and towards the local optimum, which is known as the *deceptive attractor* [16]. A fully deceptive problem of order-N can be viewed as being deceptive when all of the lower-order hyperplanes lead away from the global optimum and towards a deceptive attractor [16]. Goldberg has described such problems as *deceptive* [5].

One failing of the 3-bit fully deceptive problem is that it is too small to really demonstrate a search strategy. The thirty-bit problem as outlined in [7] expands the three-bit problem into ten three-bit deceptive order-three subfunctions. To increase the level of difficulty we include a *loose ordering*, which makes the problem fully deceptive. This is achieved by increasing the defining length to twenty, where the defining length is the maximum distance between two defining symbols in a schema. The effect of this is to make it difficult for the simple GA to solve as it tends to converge prematurely, with the subfunctions converging on 000 rather than the optimum of 111 [7]. If the bits 111 represent the global optimum and the bits 000 represent the *deceptive attractor*, then a full order-3 deception would be similar to that defined by Goldberg, Korb and Deb [7] which is illustrated in Table 1 and shows the fitness values for each bit string. With this in mind we use a fully deceptive order-3 problem as outlined by [7] in this paper.

As was pointed out in [13], diversity is critical for GAs particularly when the landscape is evolving as recombining a homogenous population will not enable the GA to locate the new optimum. To examine the performance of the mmGA over a changing landscape, once the GA reached the half-way point of the search we alter the fitness function which has the effect of changing the landscape and creating a new global optimum. By doing this we can focus the adaptive qualities of the GAs in response to altering the target of the search. Table 2 below, outlines the changes made to the fitness function and shows our new *deceptive attractor* 111, with 000 being the new global optimum. Again, the low

String	Fitness	String	Fitness
f(000)	28	f(010)	22
f(110)	0	f(101)	0
f(001)	26	f(100)	14
f(011)	0	f(111)	30

Table 1. Fully Deceptive Order-3 Problem

level building blocks lead the search away from the optimum, in other words all of the low-order hyperplanes direct the search to the deceptive attractor [16]. We also continue a count of the number of subfunctions discovered throughout the search, with ten being the maximum number achievable.

String	Fitness	String	Fitness
f(111)	28	f(101)	22
f(010)	0	f(100)	0
f(110)	26	f(011)	14
f(001)	0	f(000)	30

Table 2. Reversed Fully Deceptive Order-3 Problem

3 Multi-layered Mapping Genetic Algorithm (mmGA)

DNA as it is found in nature, contains similar levels of the bases *adenine* (**A**), *thymine* (**T**) and similar levels of the bases *guanine* (**G**) and *cytosine* (**C**). RNA acts as the route to bring us from genes to proteins. In relation to the use of genetic information *transcription* copies a gene into an RNA molecule that is complementary to one strand of a DNA double helix. As the RNA sequence is complementary to that of the DNA template strand, it can be viewed as being the same sequence as the DNA coding strand, with *uracil* (**U**) in place of *thymine* (**T**). The copy is removed from the nucleus and placed into the cytoplasm. From there, the process of *translation* uses the information found in RNA to manufacture a protein by aligning and joining specified amino acids.

The multi-layered mapping Genetic Algorithm proposed by the authors attempts to mimic this natural mapping (at a basic level) and has the advantage of allowing, multiple mappings to exist between *genotype* and *phenotype*. To achieve this the size of the alphabet to be created and the number of representations required must be fixed. The multiple mappings are created by the mmGA and the size of the genome is then calculated by the mmGA. Once the genome (DNA) is

created we carry out the process of *transcription*. The mmGA randomly creates the genome string and this in turn is converted to a *DNA template* using the following mapping $00 \rightarrow A$; $01 \rightarrow C$; $10 \rightarrow G$ and $11 \rightarrow T$. The next step in the mmGA mapping is a conversion from the template to DNA coding using the mapping $G \rightarrow C$; $A \rightarrow T$; $T \rightarrow A$ and $C \rightarrow G$. The final stage of *transcription* are now carried out and the DNA coding strings are now converted into RNA where the only change maps $T \rightarrow U$. We then apply a process of *translation*, where the RNA's found are converted into amino acids which are combined into proteins, using the mapping created by the mmGA at initialisation, which can be used as the building blocks i.e. 0 or 1, to represent solutions for the particular problem in question. The pseudocode for the process is outlined in Algorithm 1 below.

```

initialize mmGA;
r=1; (Number of runs);
for Number of runs do
    Initialise Individual Genomes P(g);
    Transcribe Genome to Amino Acids P(g);
    Translate Amino Acids to Phenotype P(g);
    Evaluate P(g); (Phenotype fitness);
    for Number of Generations do
        g=0; (generations);
        for All members of P do
            Select P(g) from P(g-1);
            Crossover P(g); genotype level;
            Mutation P(g); genotype level;
            Transcribe Genome to Amino Acids P(g);
            Translate Amino Acids to Phenotype P(g);
            Evaluate P(g); (phenotype fitness);
        end
        g+=1;
    end
    r+=1;
end
end mmGA;

```

Algorithm 1: Pseudocode - multi-layered mapping Genetic Algorithm

The mmGA operates using a binary representation which allows the use of standard genetic operators such as crossover and mutation. The difference between the multi-layered mapping GA (mmGA) and the standard representation found in a GA lies in the fact that it contains multiple layers of mappings between the *genotype* to the *phenotype*. The fitness function f , for a traditional GA can be broken down into two parts. The first translates the *genotype* space to the *phenotype* and the second maps the *phenotype* space to the fitness space

[15]. The mmGA differs as it offers a multi-layer mapping between the *genotype* and *phenotype*. By introducing an additional layers between the *genotype* space and the *phenotype* space, the modified mapping introduces a many-to-one relationship.

4 Experiments and Results

The experiments conducted by the authors use a thirty-bit fully deceptive problem whose function changes half way through the search to introduce a changing landscape. Each form of GA has a population of 200; a crossover rate of 0.7; a mutation rate of $1/l$, where l is the length of the chromosome; is executed for 25000 generations and runs 50 times. As there are a infinite number of different parameter settings, the authors have attempted to create a set of experiments to analyse both forms of GA of over the changing landscape. Due to the nature of the problem landscape and the population size, both GAs can achieve a relatively high fitness value, however this high fitness value is normally associated with proximity to the *deceptive attractor* and the difficulty lies in escaping from this neighbourhood.

The results of the experiments are outlined below. Figure 1 illustrates the average best fitness per generation for the multi-layered mapping GA (mmGA). The results indicate that early on in the search the mmGA is drawn towards the *deceptive attractor* as illustrated by the fitness values returned and the number of sub-functions discovered (outlined in Figure 2). As the search continues, the mmGA exhibits the ability to avoid convergence around the deceptive hyper-plane and begins its progression towards discovering the global optimum. As the number of sub-functions discovered increases (shown in Figure 2) so too do the fitness values returned, until we reach a point where the maximum number of sub-functions have been discovered (ten) and the maximum fitness value has been reached. Once we reach the half way point the fitness function changes and creates a different landscape, the mmGA begins searching for the new global optimum. One point to note here is that the position from where the mmGA is beginning the search, is very close to the new deceptive attractor so the fitness function is designed to encourage the search to remain in this neighbourhood. As the search begins we can again see that the mmGA has the ability to avoid converging around the local optimum (Figure 1) and begins to discover new sub-functions. This continues until we have located the new global optimum and found the maximum number of sub-functions (Figure 2).

The performance of the standard GA however, fails to detect the global optimum as the search converges towards the *deceptive attractor* (Figure 3) and the number of subfunctions discovered for the first half of the search are quite low (Figure 4). This would appear to be consistent with the theory outlined

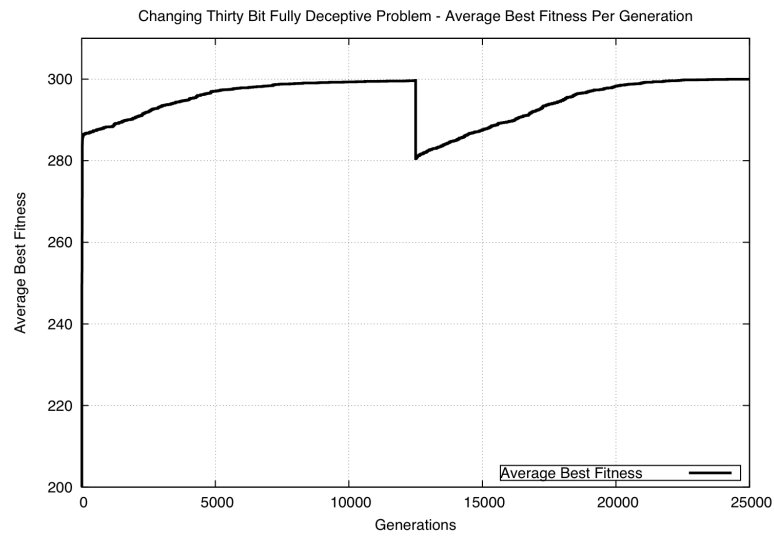


Fig. 1. Changing Thirty Bit Fully Deceptive -Avg. Best Fitness Per Generation Multi Layered GA

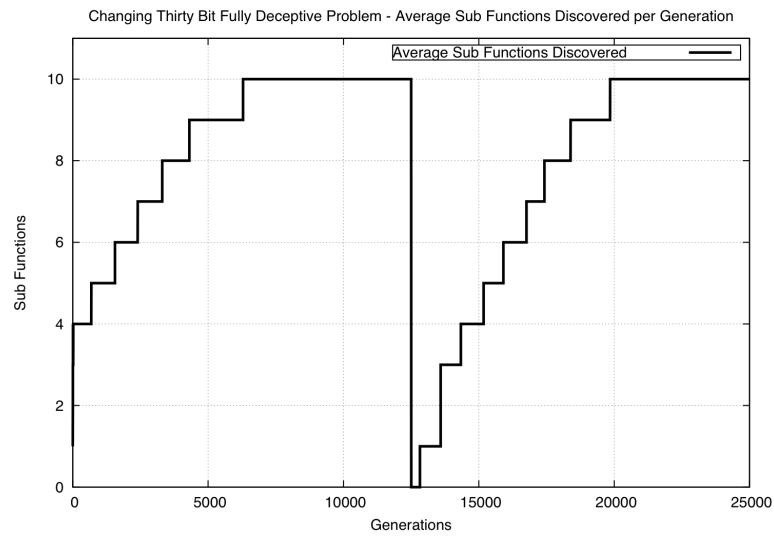


Fig. 2. Changing Thirty Bit Fully Deceptive - Sub Functions Discovered Multi Layered GA

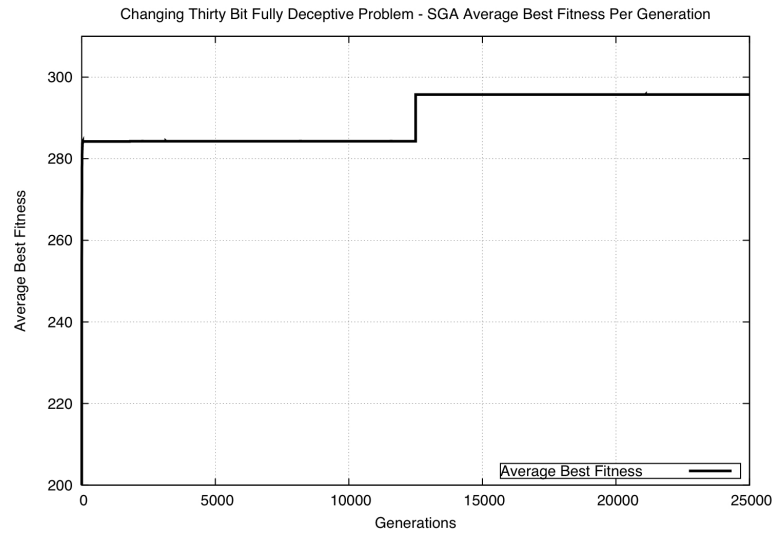


Fig. 3. Thirty Bit Fully Deceptive -Avg. Best Fitness Per Generation Standard GA

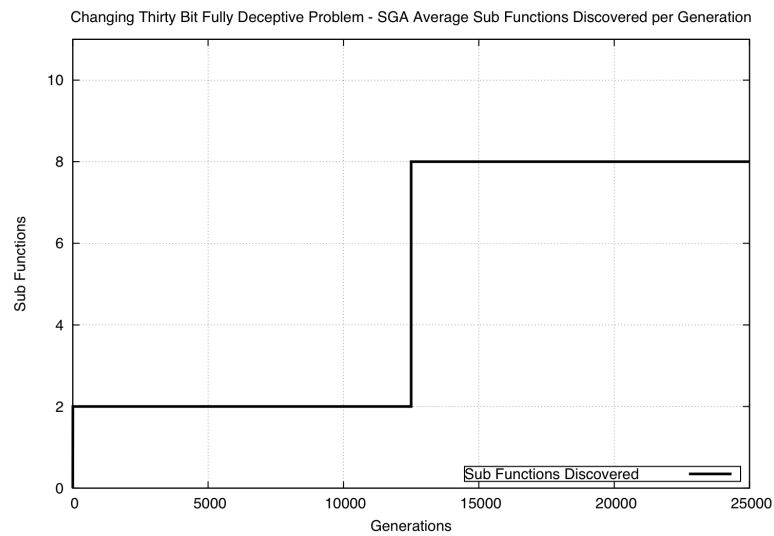


Fig. 4. Thirty Bit Fully Deceptive - Sub Functions Discovered Standard GA

in previous research, whereby the ability of the standard GA is limited, due to premature convergence on the deceptive attractor. This convergence on the local optimum happens very quickly in the search and the standard GA doesn't possess the ability to move out of the neighbourhood. One interesting point is that on changing the fitness function, the standard GA population is now in a position where it is closer to the global optimum and automatically discovers a large number of subfunctions. However, it never improves and the global optimum still remains elusive for the standard GA with the search moving towards the new deceptive attractor, with the average best fitness level (Figure 3) never reaching the maximum fitness value and the average highest number of subfunctions discovered (Figure 4) never reaching the maximum.

5 Conclusion

Living organisms can be viewed as a duality of their genotype and their phenotype and standard GAs have been used to study population level problems, however they cannot effectively replicate genome dynamics because of their *genotype - phenotype* mapping. Genomes in nature, on the other hand have a flexible phenotypic structure, this is because there exists an intermediate level between the phenotype and the genotype i.e. proteins it is this concept that the mmGA is modeled upon. The results of the experiments conducted in this paper may indicate that there are advantages, particularly over difficult landscapes, to introducing a flexible phenotypic structure into a GA. The results of the experiments conducted on the changing thirty-bit fully deceptive problem appear to indicate that multi-layered mapping GA (mmGA), overcomes the lure of the deceptive function, as it exhibits a greater flexibility to change and avoid premature convergence. We believe this is due to the bio-inspired multi-layered mapping which exists between the *genotype* and the *phenotype* of the mmGA, and its ability to maintain diversity in the population. By using a binary representation for the *genotype* in conjunction with the multi-layered mapping, the mmGA uses standard operators, whose behaviours are well understood and operate at the *genotypic* level. Another advantage, we feel, of introducing additional layers between the *genotype* space and the *phenotype* space, is that it allows the modified mapping to introduce a many-to-one relationship, which assists in increasing the level of flexibility in the phenotypic structure.

From the results outlined in this paper, future work will include an in-depth investigation into diversity and how it relates to the mmGA; an examination of a potential form of position independence associated with the multi-layered mapping found in the mmGA; the introduction of additional operators within the layered structure of the mmGA (in a very basic way similar to a form of mutation found at the mRNA level in nature) and also the application of the mmGA to a number of real-world applications.

References

1. A. D. Bethke. *Genetic algorithms as functional optimisers*. PhD thesis, (Doctoral dissertation, University of Michigan) Dissertation Abstracts International, 41(9),3503B. (University Microfilms No. 81-06101), 1980.
2. Donald S. Burke, Kenneth A. De Jong, John J. Grefenstette, Connie Loggia Ramsey, and Annie S. Wu. Putting more genetics into genetic algorithms. *Evol. Comput.*, 6(4):387–410, 1998.
3. A. E. Eiben and J. E. Smith. *Introduction to evolutionary computing*. Springer, 2003.
4. D. E. Goldberg. Simple genetic algorithms and the minimal deceptive problem. *Genetic Algorithms and Simulated Annealing*, pages 74–88, 1987.
5. D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Publishing Company, Inc., Reading, MA, 1989.
6. D. E. Goldberg, K. Deb, H. Kargupta, and G. Harik. Rapid, accurate optimization of difficult problems using fast messy genetic algorithms. *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 56–64, 1993.
7. D. E. Goldberg, B. Korb, and K. Deb. Messy genetic algorithms: Motivation, analysis, and first results. *Complex Systems*, 3(5):493–530, 1990.
8. J. J. Grefenstette and J. Baker. How genetic algorithms work: a critical look at implicit parallelism. In *Proc. of the Third. International. Conf. on Genetic Algorithms*. Morgan Kaufmann, 1989.
9. J. J. Grefenstette and H. G. Cobb. Genetic algorithms for tracking changing environments. In *Proc. of the 5th Int. Conf. on Genetic Algorithms and their Applications*, pages 523–530. Morgan Kaufmann, 1993.
10. J. J. Grefenstette. Evolvability in dynamic fitness landscapes: A genetic algorithm approach. In *Proc. 1999 Congress on Evolutionary Computation (CEC 99)*, pages 2031–2038, Washington, DC, 1999. IEEE Press.
11. J. H. Holland. *Adaptation in natural artificial systems*. University of Michigan Press, Ann Arbor, 1975.
12. V. Lefort, C. Knibbe, G. Beslon, and J. Favrel. A bio-inspired genetic algorithm with a self-organizing genome: the rbf-gene model. In *Genetic and Evolutionary Computation Gecco*, 2004.
13. Ronald W. Morrison and Kenneth A. DeJong. Measurement of population diversity. In *In 5th International Conference EA, 2001, volume 2310 of Incs*. Springer, 2002.
14. Tansey R. Distributed genetic algorithms. In *Proceedings of the THird International Conference on Genetic Algorithms*. Morgan Kaufmann, 1989.
15. Franz Rothlauf, editor. *Genetic and Evolutionary Computation Conference, GECCO 2009, Proceedings, Montreal, Québec, Canada, July 8-12, 2009, Companion Material*. ACM, 2009.
16. L. Darrell Whitley. Fundamental principles of deception in genetic search. In Gregory J. Rawlins, editor, *Foundations of genetic algorithms*, pages 221–241. Morgan Kaufmann, San Mateo, CA, 1991.
17. Annie S. Wu and I. Garibay. The proportional genetic algorithm: Gene expression in a genetic algorithm. In *Genetic Programming and Evolvable Hardware*, 2002.