Preservation of Genetic Redundancy in The Existence of Developmental Error and Fitness Assignment Error

Ayse Selen Yilmaz
School of Computer Science
University of Central Florida
Orlando, FL 32816, USA
selen@cs.ucf.edu

Annie S. Wu
School of Computer Science
University of Central Florida
Orlando, FL 32816, USA
aswu@cs.ucf.edu

ABSTRACT

Conservation of functionally identical copies of the same gene throughout the generations is not an easy task. In this study, based on the biological evidence that suggests the existence of the developmental error as one of the ways to preserve redundancy, our goal is to investigate the impact of developmental error on a simple problem using a genetic algorithm(GA). Developmental errors exist during the biological development of an individual. The biological models with developmental errors have demonstrated that it is possible to maintain redundant copies in the existence of proper mutation rates and developmental error rates. Our preliminary results with a simple problem demonstrates that using developmental error helps preserving the redundant copies and maintaining a better solution quality for the redundant copies of a gene. Besides the developmental error, we propose a new error type that comes into play during the fitness assignment and enhances the quality of the solutions when used together with developmental error.

Categories and Subject Descriptors: G.1.6 [Numerical Analyses]: Optimization[global optimization]; I.2.8 [Artificial Intelligence]:Problem Solving and Search; J.3 [Life and Medical Scinces]:Biology and genetics

General Terms: Algorithms, design, experimentation, reliability.

Keywords: Genetic algorithms, redundancy, developmental error, evolutionary stability, ontogeny.

1. INTRODUCTION

Redundancy means having more information than what is needed in a usual case. We can see redundant information everywhere in real life as well as in artificial life. An example from our everyday life that we encounter most is human language. By the help of the redundancy that exists in the structure of the language, the texts with typos can still be read and understood. Data compression is another field where the existence of redundancy in data helps in obtaining high compression rates.

We can define genetic redundancy as the existence of multiple genes performing the same function. Biologists have a more detailed classification for different types of redundancy [4]. The inactivation of one of the redundant genes does not affect the fitness, i.e the resulting phenotype, in *true redundancy*, while it slightly reduces the fitness in *almost redundancy*. In *generic redundancy*, the inactivation of one of the redundant genes occasionally results in a fitness decrease, for example in case of developmental error or environmental change. In this paper, we focus on a type of generic redundancy, where developmental failure occasionally causes defects that are not inherited to the next generations, but only affects the fitness during ontogeny; i.e. developmental process.

Recent biological studies explore why genetic redundancy is common and how it remains evolutionarily stable [4, 3].

Krakauer and Nowak [3] consider preserving functionally redundant genes as a challenge due to the following reasons:

- Redundant genes are subject to neutral mutations.
- Redundant genes may evolve into new genes.
- Due to deleterious mutations, imperfect copies of redundant genes might be formed.

They present some of the existing techniques that address these challenging issues. The existence of developmental error during developmental process is among these techniques.

Tautz also [7] states that the information transmission from an egg to an adult organism is subject to errors at every stage of development. Developmental error is an error probability that indicates the probability that a gene will be defected during the developmental process. It is important to point out that developmental error does not have an effect at the genomic level as the mutation does. Unlike mutational errors, developmental errors are not inherited, but rather happen during the developmental process. Krakauer and Nowak [3] show that there is a correlation between the mutation rate and the developmental error rate in order to maintain redundant genes during evolution. Each gene must mutate less frequently than its duplicate experiences a developmental error.

As selection pressure and the accumulation of deleterious mutations on redundant genes tend to counteract the main-

GECCO'05, June 25–29, 2005, Washington, DC, USA.

tenance of the redundancy during evolution, Nowak et. al. [4] has developed a model to explain why redundant genes are common in biological environments. In three out of four cases described in this model, they find that it is possible to maintain redundancy and keep it evolutionarily stable. One of these three cases involves the existence of developmental error. In order to keep the redundancy evolutionarily stable, their mathematical models also indicate that developmental error rates should be larger than the mutation rates used.

To the best of our knowledge, studying the impact of developmental error has not attracted many researchers in EC community. A very recent study by Rieffel and Pollack [5] investigates the effects of developmental error for an evolutionary design task that is based on Artificial Ontogeny. An error during the development process complicates the design task, causing a genotype to map to multiple phenotypes with different fitness values. They find that with the help of ontogenic mechanisms that emerge during developmental process, it's possible to adapt to the existence of the error and produce fit phenotypes. Another aspect of incorporating error into evolutionary process is introducing error to the fully developed phenotype without considering any developmental process [2, 6].

Maintaining the redundant copies of a gene has been an interesting problem in evolutionary algorithms. Incorporating redundancy into evolutionary algorithms not only plays an important role in adaptation to the changes in the environment [1], but it also improves the evolvability of the solutions and escapes from local optima by the help of diversity it introduces [8]. We would like to analyze the effect of a *developmental error, E* on keeping redundant copies of a gene during the evolution and investigate how solution quality is affected by varying mutation rates, M. Although biological studies consider different mutation rates and developmental rates for each redundant copy of a gene, we assume that all redundant genes have the same developmental error or mutation rate to keep things simple and easy for initial analysis. In addition to developmental error, we propose a new error type whose goal is to introduce a form of diversity into the solutions to increase the survival probability of less fit solutions. This new error assigns the fitness of an individual to a lower value than it is supposed to take with a predefined probability, which we call the *fitness assignment error, E2.*

2. ERROR IN GENETIC ALGORITHM

Based on the biological theories on the link between redundancy and developmental error we would like to investigate if similar relationships apply to a simple problem using a genetic algorithm. We examine the impact of three types of error on GA performance: Developmental error (E), fitness assignment error ($E2$) and selection error (K).

We begin with the simple *onemax problem* encoded as bit strings where the goal is to maximize the number of ones in the solution. We modify the problem so that we embed a form of redundancy in the solution. We break down each solution of certain length, L into two parts of length $L/2$. Each half is then considered to be a gene. The goal of our GA is to preserve the redundant copies of the string of all ones with length, $L/2$.

Each gene is evaluated separately by simply counting the number of ones. The fitness of the individual is assigned to the fitness of the better half. While the optimal fitness is the length of the individual(L), for a traditional *onemax problem*, the optimal fitness of an individual with length L is $L/2$ for our *modified onemax problem.* So, our fitness function without considering any type of error is:

$$f_{indv} = MAX(f_{gene1}, f_{gene2}), \tag{1}$$

where f_{indv}, f_{gene1}, f_{gene2} denote; fitness of the individual, fitness of the left half and fitness of the right half respectively. f_{gene1}, f_{gene2} are:

$$f_{gene1} = \sum_{i=1}^{L/2} R(i), \tag{2}$$

$$f_{gene2} = \sum_{i=L/2+1}^{L} R(i), \tag{3}$$

where $R(i)$ is the value of each bit i.

In this type of setup, it is not an easy task to obtain strings of all ones with length L, as the fitness of an individual is only based on the fitness of one half. Once half of an individual consists of all ones, the lack of selection pressure on the other half is likely to make it difficult to obtain all ones on the other half. The *modified onemax problem* is a very simple and effective way to understand the effect of different types of errors on keeping redundant genes in a solution.

The first source of error we study in our GA is the inclusion of a developmental error, E. In contrast to mutation, which occurs at the genotype level and is inherited by the new generation, developmental error occurs in the phenotype level and is not heritable. In our *modified onemax problem*, we can model developmental error as misinterpretation of bits, ones as zeros and zeros as ones. Misinterpretation only causes a change in the fitness of the corresponding instance but not a heritable change in the genotype. Each bit has a probability of being misinterpreted, i.e. developmental error rate, E. Given that $R(i)$ equals the genetic value of bit i, the value of the misinterpreted bit will be considered as $1 - R(i)$ during the fitness evaluation. Misinterpretation occurs in all individuals including both optimum and less fit individuals. As a result, we track both the raw fitness values which are based on the genotypic information and the real fitness values which incorporate misinterpretation. As the biological evidence suggests, we set the developmental error rates higher than the mutation rate in order to preserve redundancy.

In addition to the developmental error described above, we also examine the effects of a second type of error, error in assigning the fitness of the individual. We investigate the effect of noise in fitness assignment process. Instead of always taking the fitness of the better gene of a pair of redundant genes, noise in the fitness assignment results in a non-zero probability of occasionally taking the fitness of the inferior gene. Equation 1 gives the fitness calculation for an individual that is free from any assignment error. With fitness assignment error, each individual has $E2$ probability of being assigned to the fitness of the inferior gene instead of the better one. Then with a probability of $E2$, our fitness function is:

$$f_{indv} = MIN(f_{gene1}, f_{gene2}), \tag{4}$$

A third source of error that we examine is the error in selection process, where tournament selection will select the second best individual instead of the best one with probability K.

3. EXPERIMENTS

Our experimental setup is designed to investigate not only the effect of developmental error but also the effect of additional error types, one of which we propose in this study, fitness assignment error.

Our goal is to understand whether different types of errors we introduce have any positive impact on preserving a good solution quality for both genes throughout the evolution. We assume that the redundant genes that we would like to preserve are the genes of all ones with a length of $L/2$.

The basic types of scenarios that we look into are the experimental designs with :

1. No error of any type but mutation
2. Developmental error
3. Fitness assignment error
4. Selection error

For all the scenarios, we keep track of the best individuals in each generation. The fitness of the better and worse performing genes are denoted as *Best Fitness* and *Second Best Fitness* respectively. As we mentioned before, we also keep track of the raw fitness values, counting the number of ones with no misinterpretation, of both genes to see how the existence of developmental error affect the raw fitness of the evolved individuals. The raw fitness of the better and worse performing genes are denoted as *Best Raw Fitness* and *Second Raw Best Fitness* respectively. When the developmental error rate is 0, fitness values are same as the raw fitness values. Raw fitness values are indirectly affected by misinterpretation Although raw fitness values of a given generation are not affected by by the developmental error at that generation, the individuals that are selected from the previous generation to form the given generation are based on the fitness values that undergo developmental error in the previous generation.

3.1 Parameter Settings

The basic parameter settings chosen are based on the best performance obtained after a performance tuning.

We use one-point crossover and each bit has the same probability of being chosen as the crossover point. The probability that a pair of parents will crossover is 1. The mutation used is simple bit flipping. Different mutation rates have been used all being lower than the developmental error rate and fitness assignment error rate.

The length of the individuals, L, is 64 and the maximum fitness value is 32. The population size and the maximum number of generations are set to 5000 and 400 respectively unless otherwise is stated.

We give the details of mutation rate, M, developmental error rate, E and fitnes assignment error rate, $E2$ in each individual scenario as we use varying ranges of values to investigate their effect.

We set the error rate in tournament selection, k, to 0 for all the scenarios but fourth one where we test the effect of error in selection.

For all scenarios but third one, we do not have any error in assigning the fitness of the individual to the fitness of the better performing gene.

All of the results that we present are averaged over 50 runs. As the better performing gene might be either on the right or the left switch in each run, we first average the fitness values of both sides for the last 100 generations in each run to find the better performing side. Then the fitnesses of the better performing genes and the fitnesses of the other side, i.e. worse performing genes, are averaged separately. The same process is done also for the raw fitness values of best and second best(i.e. worse) performing genes.

3.2 Experiments without Error

In order to understand the impact of different types of errors we introduce, we need to first observe the quality of the solutions without any type of error.

Without any error, the best fitness values for each generation we obtain for varying mutation rates are shown in Figure 1. Our goal is to obtain a good solution quality for both genes, which are close to the optimum value 32 on the average. As there is no developmental error, there is no difference between fitness values and the raw fitness values.

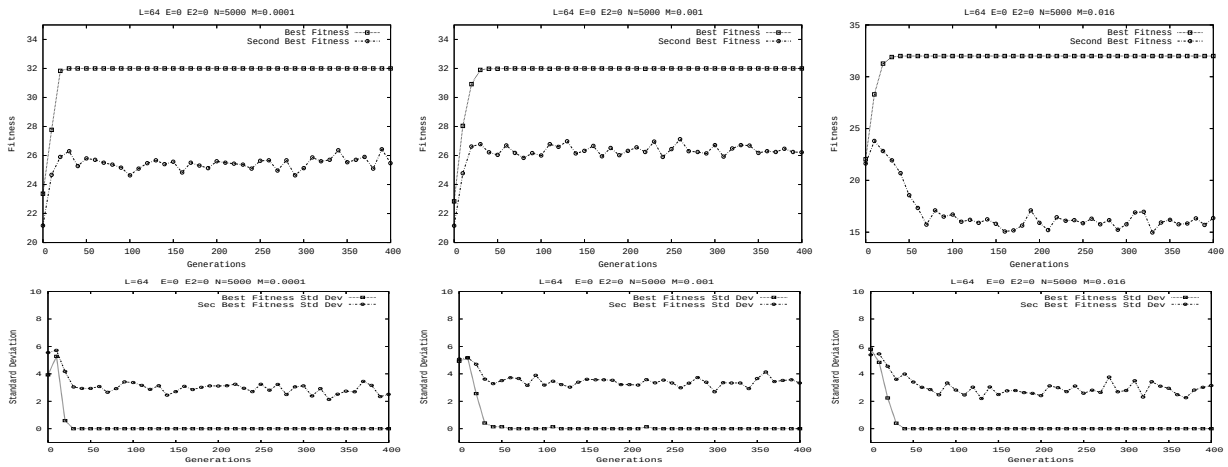
The mutation rate indicates the probability that a bit will be mutated. We begin with a conventional mutation rate of 1 per individual, which in turn results in mutation rate of M=0.016 per bit, since we have individuals of length, L=64. Figure 1(c) shows the results with M=0.016. We observe that although it is possible to obtain a maximum fitness of 32 for one of the genes, it is not possible to keep the fitness value of the other gene even close to the maximum. The second best fitness levels off at around 16, which is approximately half of the value of the maximum fitness and suggests that random selection occurs on the redundant gene. As expected, with no selection pressure on the redundant gene, we can only evolve one good copy of our gene and are unable to maintain multiple redundant copies.

Using lower mutation rates decreases the gap between the fitness values of the genes without compromising the best fitness. The *second best fitness* increases to approximately 25 for M=0.0001 in Figure 1(a) and approximately to 26 for M=0.001 in Figure 1(b). Without any type of error, the highest fitness we can obtain for the *second best fitness* is around 26, when M=0.001. In order to achieve the goal of preserving redundant genes, we need to find a way that helps us to keep genes of all ones at both sides of the individual.

As the best fitness plots shown are average of 50 runs, the standard deviation plots corresponding to each plot are also given in Figure 1. The average best fitness has a standard deviation of 0, showing no deviation from the average, while the average of the second best fitness shows a standard deviation of less than 5 for all mutation rates.

3.3 Experiments with Developmental Error

We are interested in understanding the forces that favor the evolution of redundancy. Each individual consists of two genes and each gene has a length of $L/2 = 32$. Only one of the genes contributes to fitness, the one that has higher fitness. In the onemax problem, fitness is given by the number of ones. Typically, we assume that all genotypic information is interpreted perfectly. In a real world situation, however, misinterpretations can occur and there is a chance that the information can become corrupted during the translation.



(a) (b) (c)

Figure 1: Fitness(above)/Standard deviation(below). No developmental (E=0), selection (k=0) or fitness assignment error (E2=0). (a)Mutation rate, M=0.0001 (b)M=0.001 (c)M=0.016

Making an analogy to the developmental error that the organisms face during their life cycle, we define a probability of each bit being misinterpreted. Misinterpretation causes the fitness value to experience noise. As a result, we also track the raw fitnesses that reflect the noise-free fitness values. We use same error rate for both genes.

Based on the biological theory by Krakauer and Nowak [3], we focus on developmental error rates that are larger than the corresponding mutation rates. Figure 2 - 4 show the fitness values and the corresponding standard deviation values below each fitness plot for developmental error rates of 0.03, 0.06 and 0.09, respectively.

In Figure 2, with the help of the developmental error, E=0.03, we are able to obtain higher fitness values for *second best raw fitness* as compared to the same scenario without developmental error shown in Figure 1. With this configuration, the best raw fitness value does reach the optimum value of 32, indicating that we can preserve a gene of all ones on one side. The second best raw fitness values are close but not equal to the optimum value of 32 on the average. Figure 2(a) and Figure 2(b) show the results with different mutation rates of M=0.0001, and M=0.001, respectively. For error rate E=0.03, increasing the mutation rate from M=0.0001 to M=0.001 causes a significant decrease in the *second best raw fitness* values. For this error rate, the performance of a GA highly depends on the mutation rate changes.

We next increase the error rate to E=0.06. As shown in Figure 3, with both mutation rates of M=0.0001 and M=0.001, evolving a gene of all ones on both sides is possible, as the average second best raw fitness values climb up to 32. Higher mutation rate enables a slightly earlier convergence to the optimal values for both best and second best raw fitness values. The standard deviation levels for the higher mutation rate are quite low, leveling off at 0 for both the best raw fitness and the second best raw fitness. Thus, for some mutation/developmental error rate combinations we are able to evolve redundant genes on both sides of our individual most of the time.

Further increase in the developmental error rate, E, does result in a better performance as the results for E=0.09, in Figure 4, show. The solution quality for the best raw fitness remains same, around 32, but the raw fitness of the second gene climbs up to 32 more quickly compared to the results with lower error rate, E=0.06. Increasing the mutation rate from M=0.0001 to M=0.001, allows a little bit earlier convergence to the optimal value for both average best and second best raw fitnesses. An increase in mutation rate also helps us to achieve standard deviation values of zero for the raw fitness values.

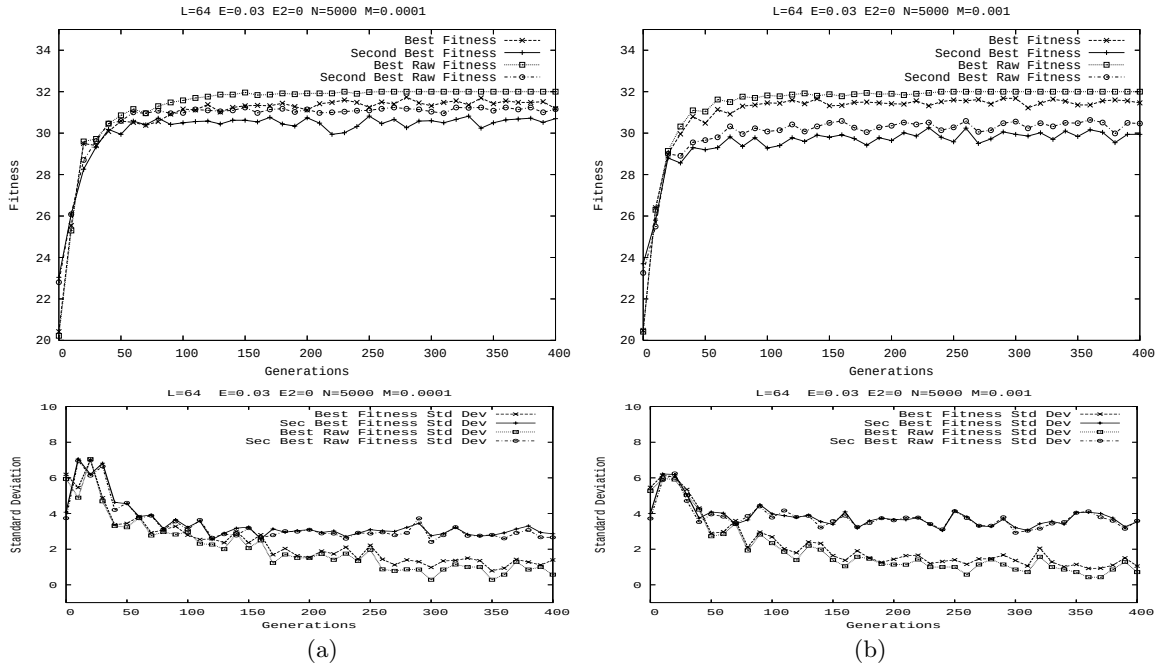
These findings indicate that the impact of developmental error on performance is highly dependent on adjusting the rate correctly, as is the case of many other GA parameters.

3.4 Experiments with Fitness Assignment Error and Developmental Error

Having achieved promising results in keeping the redundant copies of two genes of all ones with the help of developmental error, our goal is to improve these findings and find a method to make them more robust and less susceptible to changes in parametric configuration.

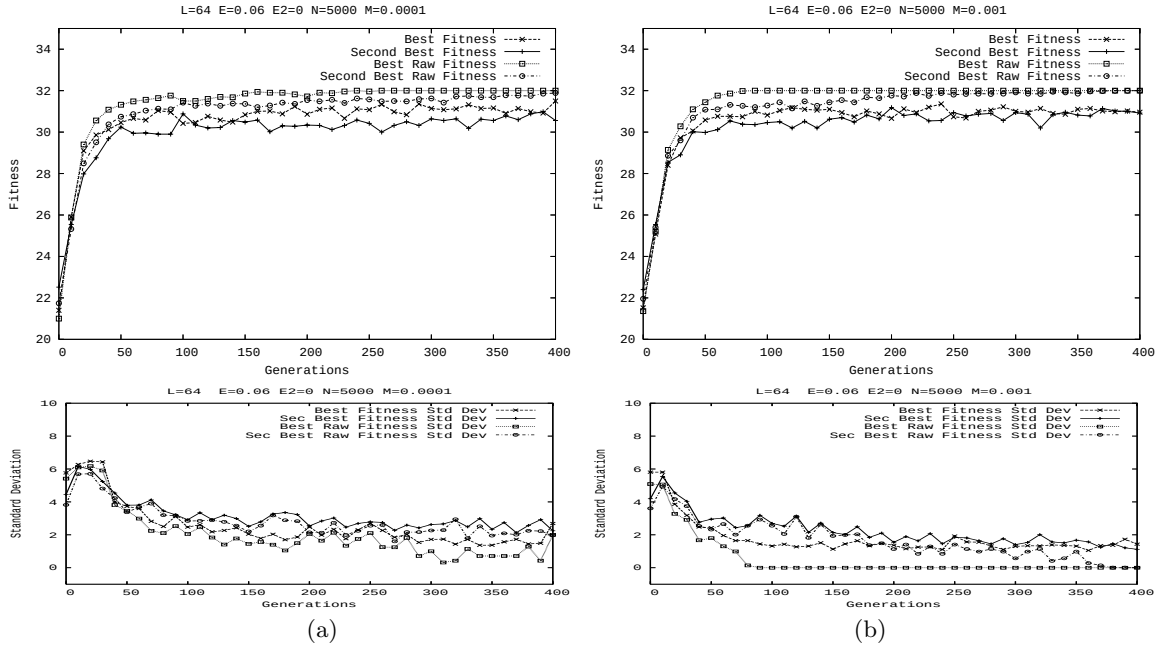
The fitness assignment error, $E2$, comes into play during the fitness assignment process. The fitness of each individual is determined by one of the redundant genes. Our test function is made up of two genes. Each gene is evaluated seperately and the fitness of the better gene is assigned to the fitness of the individual when there is no error in fitness assignment. Fitness assignment error results in assigning the fitness of the worse gene as the fitness of the individual with a predefined error probability, $E2$. Our goal in doing this is to increase diversity in the population and prevent premature convergence to a local optima before finding the redundant genes of all ones.

We include error in fitness assignment process for the two of the experiments we have conducted using developmental error in section 3.3. For the test set with developmental error rate of 0.06 and mutation rate of 0.001, we investigate the effect of two different fitness assignment error rates, $E2$=0.12 and $E2$=0.18. Both error rates result in an improved performance compared to the result shown in Figure



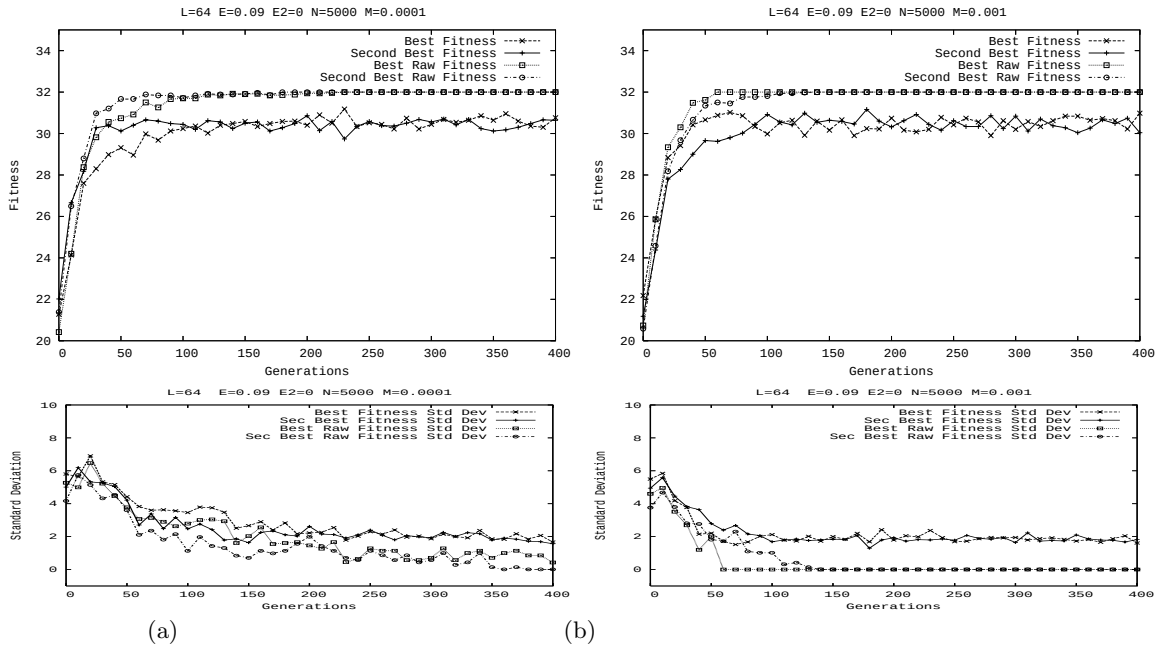
(a) (b)

Figure 2: Fitness(above)/Standard deviation(below) versus generation. Developmental error rate, E=0.03, no selection(k=0) or fitness assignment error(E2=0). (a) Mutation rate, M=0.0001 (b)M=0.001



(a) (b)

Figure 3: Fitness(above)/Standard deviation(below) versus generation. Developmental error rate, E=0.06, no selection(k=0) or fitness assignment error(E2=0). (a) Mutation rate, M=0.0001 (b)M=0.001



(a) (b)

Figure 4: Fitness(above)/Standard deviation(below) versus generation. Developmental error rate, E=0.09, no selection (k=0) or fitness assignment error(E2=0). (a) Mutation rate, M=0.0001 (b)M=0.001

3(b) obtained without fitness assignment error. Both Figure 5(a) and Figure 5(b) show that the second best raw fitness level increases to the optimal value of 32. With the optimal fitness value achieved for the best raw fitness level, we are able to maintain the two redundant genes of all ones. The corresponding standard deviation level decreases to zero for both fitness assignment error rates.

Repeating the tests with a developmental error rate of E=0.09 results in similar behavior. For both fitness assignment error rates of $E2$=0.18 and $E2$=0.27, both the best raw fitness and the second best raw fitness show a little improvement as shown in Figure 6(a) and Figure 6(b) as compared to the test cases with no fitness assignment error. The standard deviation levels decrease slightly with the usage of fitness assignment error. These results suggest that, we achieve a better performance in obtaining the redundant genes of all ones on the average. The drops in standard deviation levels also suggest that, the solutions that we obtain by the inclusion of fitness assignment error are robust and evolutionarily stable.

3.5 Experiments with Selection Error and Developmental Error

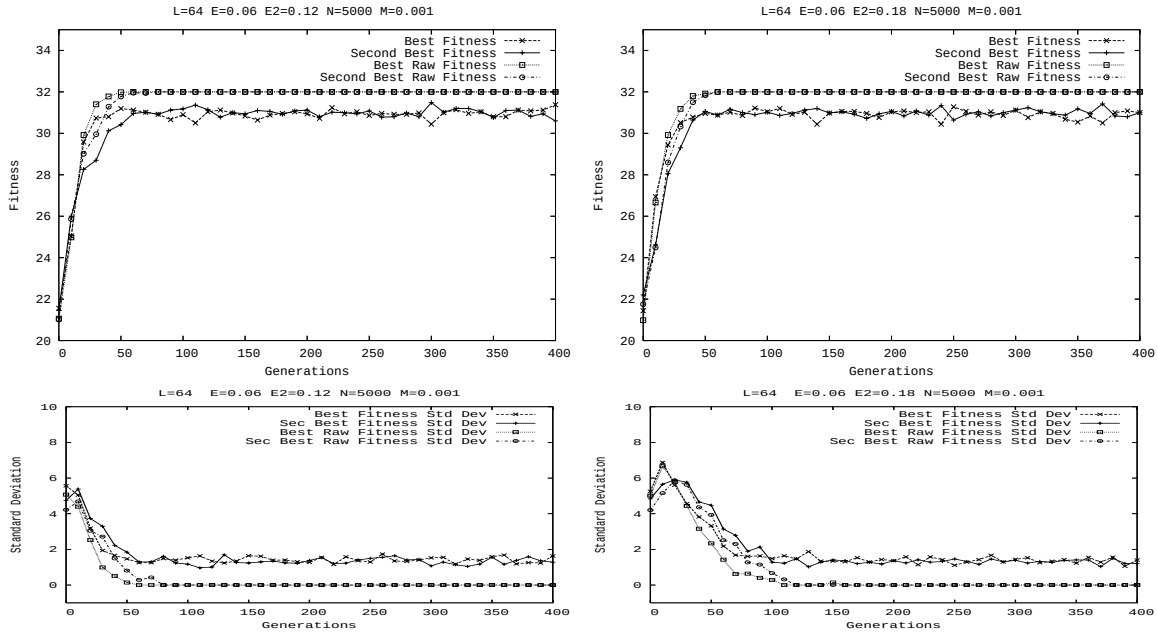
We also would like to consider whether an error in the selection along with the developmental error will improve our results. We use tournament selection. When there is no error in the selection, the best individual out of a tournament is always selected to be a parent in the next generation. When we introduce an error in the selection, the second best individual is selected with a certain predefined probability (K), from among the tournament candidates.

We introduce an error with a probability of K=0.2. When the tournament size, T=2, K=0.2 indicates the probability of selecting the worse parent out of every two parent candidates. Figure 7, shows the results with K=0.2 and E=0.06. Compared to the result without any selection error shown in Figure 3(b), we are able to obtain a slightly better performance in terms of second best raw fitness. Similarly, as shown in Figure 8(a), introducing an error in selection with the development error rate of 0.09 enable a similar solution quality for both the best raw fitness and the second best raw fitness. Increasing the selection pressure by selecting a higher tournament size of 5 in Figure 8(b) causes the fitness levels to drop. We conclude that error in selection results in either similar or a slightly improved performance in our results.

4. CONCLUSIONS

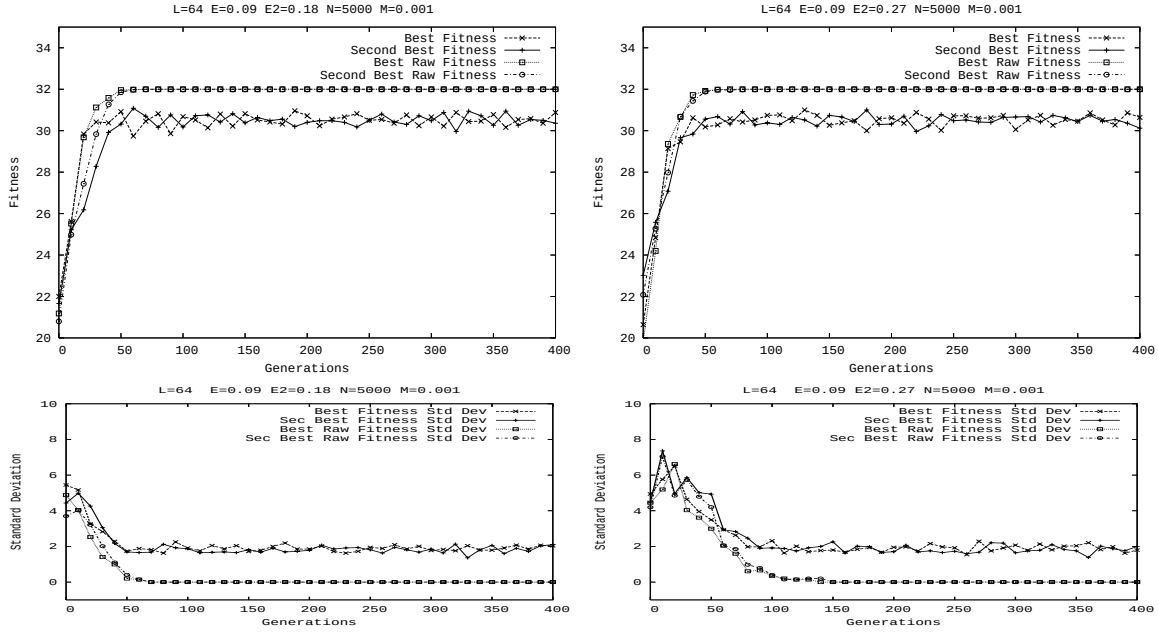
Given the biological evidence we have on the effect of developmental error in preserving functionally redundant genes during evolution, we explore how these effects apply to a simple problem using a genetic algorithm and how we can benefit from the results we obtain. We modify the one-max problem in order to obtain a form of redundancy and perform our tests on this *modified onemax problem*. We construct a variety of scenarios to test different types of errors including developmental error, fitness assignment error and selection error. Our goal is to see if we are able to obtain identical copies of two genes in a solution and preserve it throughout evolution. Compared to the scenarios without developmental error, the results we obtain using developmental error are successful in maintaining a better solution quality for redundant genes and keeping it stable. The addition of a fitness assignment error causes noise which enables us to further improve the results obtained using developmental error.

The results we obtain are preliminary but promising and need to be tested on additional types of problems. We have considered only two redundant genes performing same function. This work can be extended to examine increased num-



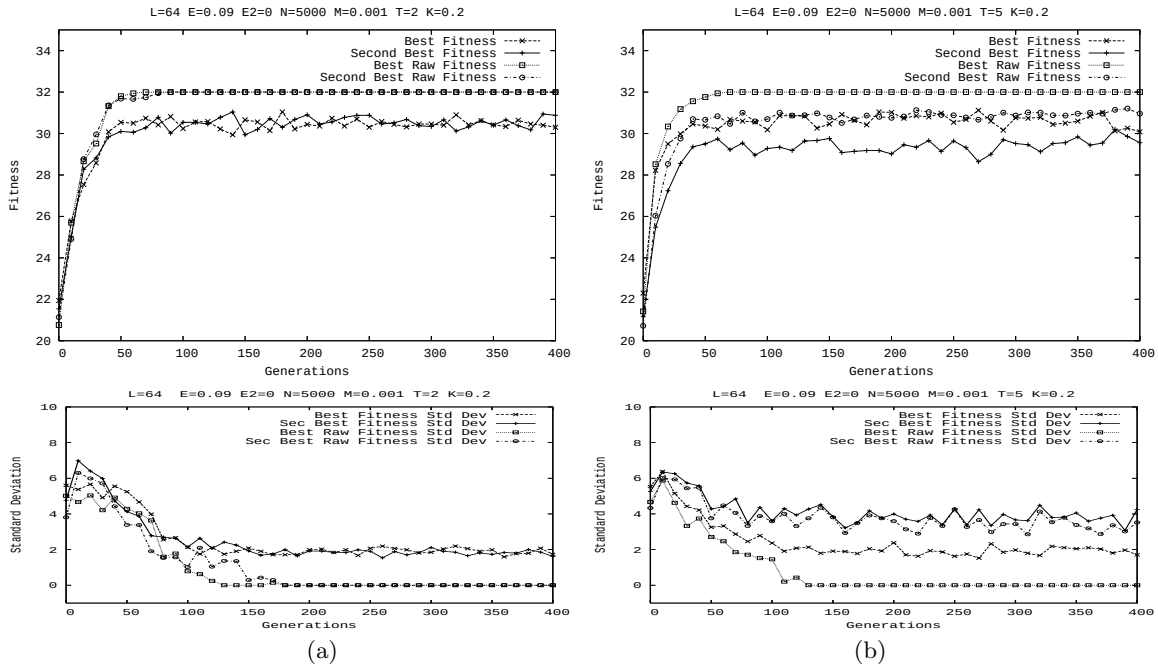
(a) (b)

Figure 5: Fitness(above)/Standard deviation(below) versus generation. Developmental error rate, E=0.06, no selection error, (k=0), (a) Fitness assignment error rate, E2=0.12 (b)E2=0.18



(a) (b)

Figure 6: Fitness(above)/Standard deviation(below) versus generation. Developmental error rate, E=0.09, no selection error, (k=0), (a) Fitness assignment error rate, E2=0.18 (b)E2=0.27



(a) (b)

Figure 8: Fitness(above)/Standard deviation(below) versus generation. Developmental error rate, E=0.09, Selection error, k=0.2, no fitness assignment error(E2=0) (a)Tournament size, T=2 (b)T=5

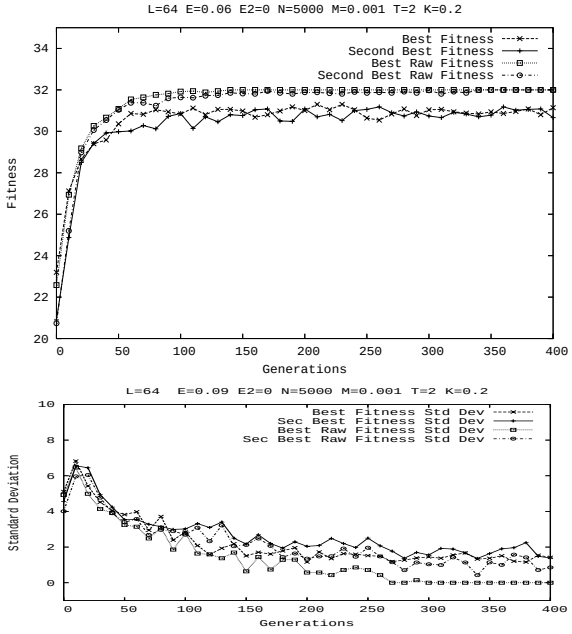


Figure 7: Fitness/Standard deviation versus generation. Developmental error rate, E=0.06, Selection error, k=0.2, no fitness assignment error(E2=0)

ber of genes and greater redundancy. We leave investigating the number of redundant genes that can coexist based on the developmental error as a future work.

5. REFERENCES

[1] M. A. Huynen, P. F. Standler, and W. Fontana. Smoothness with ruggedness: The role of neutrality in adaptation. In *Proc. of National Academy of Science*, pages 397–401, 1996.

[2] N. Jakobi, P. Husbands, and I. Harvey. Noise and the reality gap: The use of simulation in evolutionary robotics. In *Proc. of the Third European Conference on Artificial Life (ECAL'95)*, pages 704–720, 1995.

[3] D. C. Krakauer and M. A. Nowak. Evolutionary preservation of redundanct duplicated genes. *Cell and Developmental Biology*, 10(0337):555–559, 1999.

[4] M. A. Nowak, M. C. Boerlijst, J. Cooke, and J. M. Smith. Evolution of genetic redundancy. *Nature*, 388(scdb.1999.0337):167–171, July 1997.

[5] J. Rieffel and J. Pollack. The emergence of ontogenic scaffolding in a stochastic development environment. In *GECCO'04, Genetic and Evolutionary Computation Conference Proceedings*, June 2004.

[6] K. Sims. Evolving virtual creatures. In *Proceedings of the 21st annual conference on Computer Graphics and interactive techniques*, pages 15–22, 1994.

[7] D. Tautz. Redundancies, development and the flow of information. *BioEssays*, 14:1263–266, July 1997.

[8] M. Toussaint and C. Igel. Neutrality: a necessity for self-adaptation. In *Proc. of 2002 Congress on Evol. Comput. (CEC)*, pages 1354–1359, 2002.