A Comparison of Haploidy and Diploidy without Dominance on Integer Representations

Ayse S. Yilmaz
Department of Computer Science
University Of Central Florida
Orlando, FL 32816
selen@cs.ucf.edu

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

Abstract

Diploidy and dominance mechanisms have primarily been investigated for binary representations and stationary problems. In this work, we propose a new diploid scheme without a dominance mechanism for integer representations. All diploid individuals evolve without using any haploid stage. The performance of a diploid GA is compared with a haploid GA for both stationary and non-stationary environments. Our test problem is an updated form of Traveling Salesman Problem (TSP). For our integer representation, a simple haploid GA without extra recovery mechanisms appears to outperform our diploid GA in terms of computational overhead, solution quality and storage overhead. Diploidy's ability to maintain diversity slows down its adaptation time to a desired target.

1. Introduction

Diploidy and dominance mechanisms are thought to benefit the GA process by providing memory mechanism. Most of the studies conclude that a diploid GA for binary representations in nonstationary environments improves GA performance [1, 2]. Our interest in understanding the behavior of the diploidy mechanism on different representations has motivated us to investigate integer representation for TSP.

Goldberg [1] proposed the triallelic scheme and showed its superiority to a traditional haploid representation. Ng and Wong [2] prove that the triallelic scheme performs better for an isolated example but can not be generalized. They introduce a mechanism of dominance change to make use of the implicit memory of diploidy; however it fails in cases with more than two targets. The comparative study by Lewis *et al.* [3] shows that for problems with multiple targets, a haploid GA with hyper-mutation outperforms diploid GA due to its ability to maintain diversity. The mentioned methods have mostly been applied to bit-coding problems where the locus of a determines its meaning. They are inapplicable to the problems where the relative order of characters is significant, as in the TSP. Binary representation typically requires some repair algorithms for the TSP in order to obtain valid paths after operations. Yoshida *et al.* [4] use a technique called pseudo-meiosis to solve the TSP. A haploid chromosome, formed using recombination operators on diploid chromosomes, undergoes genetic operations. Thus the diploid genotype information is not totally maintained after a haploid chromosome is formed.

We investigate the use of diploidy for a sequence based path representation. Our method uses diploidy without a dominance mechanism. Genetic operations are performed directly on the diploid genotype. The phenotype of each diploid individual, however, is formed by applying a recombination operator on a diploid chromosome pair and selection is based on this phenotype. Nevertheless, there is no haploid stage in this work as seen in pseudo-meiosis [4]. The operators used are the corner-stones of the TSP performance. We adopt the *edge-recombination crossover* [5] which performs best among other eight crossover operators for three different test problems [6] and the three best performing mutation types of *displacement mutation* [7], *simple inversion mutation* [8] and *inversion mutation* [9].

2. Designing A Diploid GA

2.1. System Flow

1. Initialize each chromosome of each individual in the population randomly. Calculate the expressed phenotype of each individual.
2. Evaluate the fitness of each individual in current generation using Eq. 1 in section 2.4.

3. Continue until the maximum number of generations is reached.
4. Calculate the expected number of offspring for each individual in current population.The lower the fitness value the greater the number of offspring is.
5. Select the parents of the next generation.
6. Apply genetic operators to reproduce and form the next generation. Generation based reproduction is used where the entire population of chromosomes are replaced by children. Chromosome i of first parent crosses over with the corresponding chromosome i of the second parent with a probability of crossover rate defined. The resulting offspring are then mutated to obtain the final form of the individuals for the new generation.
7. Go to step 2.

2.2. Test Problem

We use a TSP instance consisting of 20 cities.The cost matrix represents the cost (time or price) of travelling between each pair of cities.The cost matrix remains fixed throughout a run of the stationary TSP. The cost matrix changes during the course of the run of the non-stationary TSP to investigate the performance of GAs in terms of adaptivity to environmental changes.

2.3. GA Representation

Path or *Integer Representation* which is considered to be one of the best performing representation for TSP [6] is used to represent the individuals or tours in this work. Each gene on a chromosome represents the label of a city and the entire sequence of genes represent a full tour. While a haploid individual contains only one tour, a diploid individual is composed of two tours in the genotype.

2.4. Fitness Function

Fitness value of each individual is the total length of the tour each individual's phenotype represents. *d(i,j)* is the cost between city i and city j. Our aim is to minimize this fitness value.

$$fitness_i = \sum_{i=0}^{n-1} d(i, (i+1) \bmod n) \qquad (1)$$

2.5. Operators

2.5.1. Genetic Edge Recombination Crossover. This operator works by using an edge map that gives the edges for each city in the parents. This operator always produces valid offspring. The details of the recombination algorithm can be found in Whitley's work [5]. We adopt this operator for the crossover operation occuring between the parents, and we use a modified version of this operator to determine the phenotype of the diploid individual. The modified operator transforms the two "parents" or the two diploid chromosomes into a child which represents the phenotype of the individual. We extend the algorithm so that when the number of entries of the cities in the edge list are equal, the city that has the minimum cost to reach the *current city* is chosen to be the next *current city*. In this way, the diploid mechanism is given the opportunity to express good solutions.

2.5.2. Mutation Operators. The displacement mutation operator [7] selects a subtour and changes its place randomly. The simple inversion mutation [8] selects a subtour randomly and reverses the ordering. In the inversion mutation operator [9] a randomly selected tour is removed from the tour and inserted back in a random place in reverse order.

2.6. Parameters

Unless otherwise specified, the default parameter settings are as follows. Haploid GA population has twice as many individuals as the diploid GA to ensure that both GAs have access to the same amount of genotypic information.

Population size	: 300/150 (haploid/diploid)
Genome length	: 20
Parent Selection	: Tournament, size:5
Expected # of offspring	: 0 to 2
Crossover type	: edge-recombination
Crossover rate	: 1.0
Mutation type	: inversion
Mutation rate	: 0.5 (per individual)
Reproduction rate	: 1.0
Max number of generations	: 300
Number of runs	: 100
Oscillation Period	: 100 (non-stationary)

2.7. Test Sets

We compare the performance of our diploid GA with a traditional haploid GA in both stationary and nonstationary environments.

2.7.1. Stationary Environment Tests. The goals of this set of tests are to investigate:
1. The effects of different mutation & crossover rates.

2. How diversity is maintained in both schemes and its effect on the solution quality.
3. The effect of changing population size and mutation types for both GAs.

2.7.2 Non-stationary Environment Tests. The original cost matrix is composed of the mileage between 20 cities. The minimum cost obtained using the original cost matrix is 9090. In order to simulate changes in the environment we randomly generate cost matrices using uniform distribution. The cost matrix is switched between the different target cost matrices after each oscillation period, i.e. predefined number of generations. The aspects to be investigated are:
1. Diploid GA versus haploid GA under different sudden environmental changes. The minimum cost found for each of the three randomly generated matrices generated are 6976, 6381, 16128 respectively.
2. The effects of increasing the oscillation period. The length of the oscillation period effects the degree to which each GA can converge.The oscillation periods tested are 100, 200 and 400.
3. The effects of varying mutation rate. A haploid GA is solely dependent on mutation operator to adapt to the changes while diploid GA is capable of keeping diversity due to its dual chromosome structure. Using higher mutation rates may increase diversity and eliminate the question of whether results are biased towards diploidy. Mutation rates tested are 0.1, 0.3 and 0.4.
4. The effects of increasing the number of targets. Diploid structure enables a memory mechanism for only two solutions. This test reveals if the diploid mechanism we introduce enables the needed memory mechanism and diversity maintanence to allow for adapting to more than two target changes.

3. Experimental Results

3.1. Stationary Environment

The first set of experiments aims to determine the best mutation and crossover rate combination for the diploid scheme. Table 1 shows that in both the haploid and diploid schemes, a mutation rate of 0.3 and crossover rate of 1 gives us the highest frequency of discovery of the best solution out of 100 runs. In almost all cases however, the diploid GA is not as successful as the haploid GA. We run the second and third set of experiments with the best performing mutation-crossover rate combination.

The second set of experiments compares the ability of haploidy and diploidy in maintaining diversity. The diploid GA clearly excels, with the best fitness over 100 runs levels off after generation 100. The haploid GA levels off as about generation 30. In both GAs, the best fitness continues to improve after leveling off, but at a very slow rate.

Table 1: Percent optimum found for varying mutation-crossover rates, X:Crossover rate

	Mutation Rates									
	Haploid GA					Diploid GA				
X	0.1	0.3	0.5	0.7	1	0.1	0.2	0.3	0.4	0.5
0.1	24	82	42	67	2	0	1	1	3	0
0.5	38	85	57	80	0	11	23	39	37	6
1	68	93	91	83	1	42	59	61	34	3

Table 2: Percent optimum found for varying mutation types and population size. SIM:Simple inversion, DM:Displacement, IVM:Inversion

	Mutation Types					
	Haploid GA			Diploid GA		
Pop_Size	SIM	DM	IVM	SIM	DM	IVM
100	22	47	71	44	30	35
200	54	64	90	50	46	38
300	62	77	93	54	62	63

In the third set of experiments we test three different population sizes of 100, 200 and 300. Table 2 indicates the number of runs the best solution found out of 100 runs as mutation type and population size varies. For the haploid GA, inversion mutation outperforms other mutation types independently of population size. For the diploid GA, small population sizes appear to be more effective using simple-inversion mutation, while larger population sizes perform better with inversion mutation. In both schemes, increases in the population size result in performance improvement.

3.2. Non-stationary Environment

We test the effectiveness of the diploid integer representation for non-stationary environment. The first set of experiments compare the behavior of diploid and haploid GAs in non-stationary environments with sudden oscillatory changes. We perform the tests with two different sets of cost matrices: the first set oscillate between targets of 6381 and 9090 and the second between 6381 and 16128, both with an oscillation period of 100. The results of the former experiment are given in Figure 1. In both examples, the haploid GA is better at finding the minimum fitness for both targets and shows better behavior in terms of the population average. The diploid GA

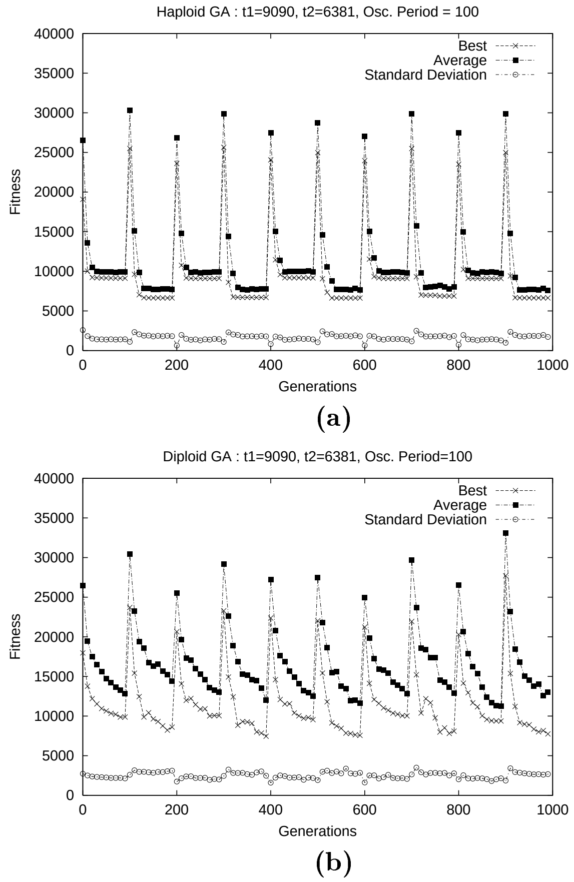

Figure 1: Best/average fitness vs. generation for 2 target oscillating TSP t1=9090, t2=6381. Haploid GA in (a), Diploid GA in (b). Oscillation period=100.

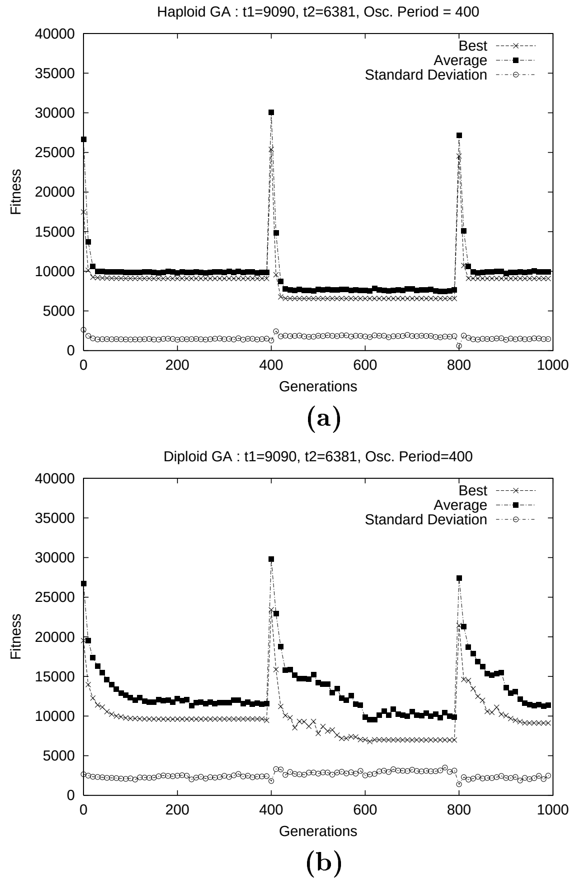

Figure 2: Best/average fitness vs. generation for 2 target oscillating TSP, t1=9090 t2=6381. Haploid GA in (a), Diploid GA in (b). Oscillation period=400.

appears to have slower adaptation which results in worse fitness values at the end of each oscillation period. The variance for diploid GA is higher than that of a haploid GA indicating a greater diversity.

As the diploid GA converges later than the haploid GA, we expect to improve the solution quality for diploidy by increasing the oscillation period. Our second set of experiments focuses on variations in the oscillation period. We alternate between two targets that give the minimum fitness values of 9090 and 6381. The oscillation periods tested are 200 and 400. Figure 2 shows the results for oscillation period of 400. Short oscillation periods do not give the diploid GA a chance to find the best fitness. Increasing the oscillation period results in better solution quality at the time of the target changes. Despite this improvement, we still do not observe an overwhelming performance of the diploid GA over a haploid GA. Haploid GA still has the less variance with better convergence to the two values of 9090 and 6381.

Our third set of experiments focuses on varying mutation rates. Increasing the mutation rate is the only way for a haploid GA to gain diversity while a diploid GA can maintain diversity due to redundancy in representation. We observe that increased mutation rate in a haploid GA does not considerably improve the quality of the best solution obtained. Any deviation from the mutation rate of 0.3 causes both schemes to perform worse in terms of finding good solutions.

Most diploid and dominance mechanisms in binary representations lack the ability to perform well in cases of more than two targets. We examine the performance of both the haploid and diploid schemes on a problem with three targets. We use cost matrices that enable minimum fitness values of 9090, 6138 and 11177. Figure 3 shows that both schemes are able to find all targets but the haploid scheme performs better in finding optimum individuals.

4. Conclusion

In binary encoded GAs, diploidy has been shown to improve the GA performance for nonstationary problems while having little apparent benefit in stationary problems. We test this claim on integer representation.

For the stationary version of TSP where the optimum fitness to be achieved does not change during the course of the run, we have tested different parameter settings. We conclude that for the best parameter settings, a diploid GA performs fairly well, but not as good as a haploid GA. The higher diversity achieved by a diploid GA does not appear to help increase its solution quality.

The claimed benefits of diploidy for binary representations in nonstationary environments with [2] and without [1] any extra dominance change mechanism do not appear to extend to the integer representation. What is more interesting about our integer representation GA for the TSP is that a haploid GA can outperform a diploid GA, eliminating the need for additional recovery mechanisms that are commonly required in binary representations immediately after target changes [3]. The use of a haploid GA over a diploid GA leads to significant computational savings including about one third of the storage space required for the diploid scheme, one less chromosome and its associated computational effort. Diploid GA is good at keeping diversity so that it can prevent converging to a local optimum which is not observed frequently for the TSP we tested. Comparison of this diploid scheme versus a haploid scheme on different and larger problems with multiple targets are left as future work.

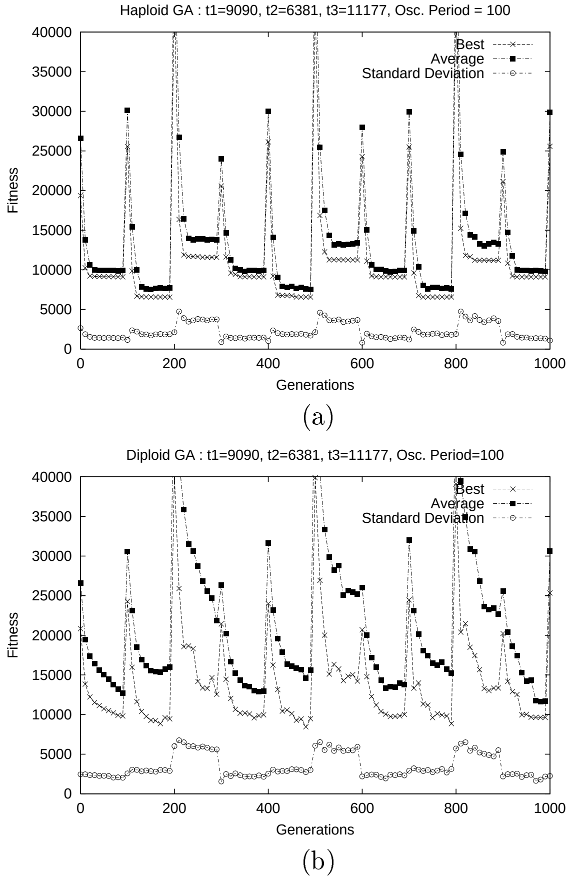

Figure 3: Best/average fitness of the average of the 100 runs vs generation for 3 targets, haploid GA in (a) and diploid GA in (b). t1=9090 t2=6381 t3=11177

References

[1] D.E. Goldberg and R.E. Smith, "Nonstationary function optimization using genetic algorithms with dominance and diploidy," in *Proceedings of the second ICGA*, 1987.

[2] Khim Peow Ng and Kok Cheong Wong, "A new diploid scheme and dominance change mechanism for non-stationary function optimization," in *Proceedings of the Sixth International Conference on Genetic Algorithms*, 1995, pp. 159–166.

[3] J. Lewis, E. Hart, and G. Ritchie, "A comparison of dominance mechanisms and simple mutation on non-stationary problems," in *Parallel Problem Solving from Nature*, 1998, vol. 1498, pp. 139–148.

[4] Kukiko Yoshida and Nobue Adachi, "A diploid genetic algorithm for preserving population diversity," in *Parallel problem solving from nature: PPSN III*, 1994, pp. 36–45.

[5] D.Whitley, T. Starkweather, and D. Shaner, "The travelling salesman and sequence scheduling: Quality solutions using recombination," in *Handbook of Genetic Algorithms*, 1999, pp. 350–372.

[6] P. Larranaga, C. Kuijpers, R. Murga, I. Inzi, and S. Dizdarevic, "Genetic algorithms for travelling salesman problem: a review of representations and operators," *Artificial Intelligence Review*, vol. 13, pp. 129–170, 1999.

[7] Z. Michalewicz, "Genetic algorithms + data structures = evolution programs," in *Springer Verlag, Berlin Heidelberg*, 1992.

[8] J.J. Grefenstette, "Genetic algorithms and their applications," in *Proceedings of the second international conference*, 1987.

[9] D.B. Fogel, "Applying evolutionary programming to selected tsp problems," in *Cybernetics and systems*, 1993, vol. 24, pp. 27–36.