Diploidy for Evolving Neural Networks

Cara Reedy
Rensselaer Polytechnic Institute
110 8th St
Troy, NY, United States of America
reedyc@rpi.edu

ABSTRACT

Genetic algorithms and artificial neural networks are two widely-used techniques that can be combined with each other to produce evolved neural networks. Some research has also looked at the use of diploidy in genetic algorithms for possible advantages over the haploid genetic representation usually used, most notably in the form of better adaptation to changing environments. This paper proposes a diploid representation for evolving neural networks, used in an agent-based simulation. Two versions of the diploid representation were tested with a haploid version in one static and two changing environments. All three genetic types performed differently in different environments.

CCS CONCEPTS

• **Computing methodologies~Neural networks** • **Computing methodologies~Genetic algorithms**

KEYWORDS

Diploidy

ACM Reference format:

Cara Reedy. 2018. Diploidy for Evolving Neural Networks. In *Proceedings of ACM GECCO conference, Kyoto, Japan, July 2018 (GECCO'18)*, 4 pages.
DOI: 10.1145/3205651.3208226

1 INTRODUCTION

Genetic algorithms (GAs) and artificial neural networks (ANNs) are both problem-solving techniques inspired by nature. GAs are a simplified version of evolution and artificial or natural selection, while ANNs were inspired by the structure of the brain and its ability to learn and solve problems.

In the same way that evolution and learning are not entirely separate processes – natural brains evolved, and learned behaviors can influence selection – GAs and ANNs can be combined. GAs can be used instead of or beside conventional learning functions in an ANN. They are typically used to evolve either the weights of the network or both the weights and structure of the network, although other possibilities, such as only evolving the structure or evolving the transfer functions of the network nodes, have also been tested. Evolving the weights can be useful when gradient descent-based learning methods cannot be used or are undesirable, such as when the error function is not differentiable, and evolving the structure of a network is useful because the topology of the network can affect network training and performance. Evolution has been applied to different kinds of ANNs, including feed-forward and recurrent ANNs [1] and deep networks [2].

2 DIPLOID GENETIC ALGORITHMS

Most GAs use haploid genetics, where each individual has one copy of their genetic material. However, in real-life organisms, there are other possible genetic arrangements. Most cells in the vast majority of animals are diploid, or have two copies of their genetic material. Some organisms mix haploidy and diploidy - bees, for example, have diploid females and haploid males - while many plants are polyploid, or have more than two copies. All of these systems have been explored for use in GAs; however, artificial diploidy is the most well-researched of these. It introduces the least additional complexity compared to the default haploidy technique, and its prevalence in real-life organisms implies that it has some kind of advantage in some situations that outweighs the costs of copying and maintaining twice as much DNA. There may be benefits to diploidy in artificial GAs, as well.

The main difficulty with programming a GA with more than one copy of the genome is to figure out, for each gene, which copy should be expressed, or to what degree each copy should be expressed. This is called dominance. In real life, dominance is a consequence of how genes are expressed and the proteins they produce; a simple example is an allele that makes a broken protein being dominant over an allele that produces a properly-working protein, because the broken protein gets in the way of the functioning version. However, in an artificial GA, this must be decided by the creators of the GA. Many methods have been used to decide dominance, and it is not clear which methods are better than others, or if some are better in certain situations than others.

The early work of Goldberg and Smith [3] had a dominance scheme with three evolving alleles that affected dominance. This work was later criticized by Ng and Wong [4], who argued that this scheme was biased, biologically implausible, and, due to the design of the experiment, not actually advantageous over haploid

GECCO '18 Companion, July 15–19, 2018, Kyoto, Japan

ACM ISBN 978-1-4503-5764-7/18/07$15.00
https://doi.org/10.1145/3205651.3208226

GAs, as Goldberg and Smith claimed. They proposed a new scheme, which they showed was superior to that of Goldberg and Smith and haploid GAs across a range of parameters. Ryan's [5] diploid scheme lacked explicit dominance; instead, each allele was given a value, and the value of an individual's two copies of the gene were added together to determine the trait. This could be extended to polygenic traits, which depend on multiple genes - for example, human eye color. Other work used two additional genes to control dominance for each allele [6], used probability instead of absolute dominance and recessiveness [7], or had an entire chromosome considered dominant or recessive to the other [8]. In some schemes, dominance is fixed, while in others, it can change, either through mutation and evolution, or through population analysis. For example, Uyar and Harmanci [7] tested a system where the dominance value of a given allele can change each generation, depending on the fitness of individuals with that allele during that generation. They found this produced better results than either haploidy or two previous schemes that did not have a dominance change method. Lewis et al [9] had earlier modified these two schemes to produce a similar dominance change based on population fitness, and found this was important for the diploid population to recover after environmental change.

While some diploid schemes have been compared, it is still unclear what features make for a good scheme. Yang [10] examined two features: the number of levels of dominance, from two to eight, and whether or not the mapping from genotype to phenotype was deterministic. It was concluded that while being non-deterministic was a hindrance to performance, having at least four levels of dominance increased it. Bowers and Sevinç [11] looked at whether there was an advantage to Mendelian inheritance, where alleles are completely dominant and recessive, or to complete co-dominance, where each allele contributes equally to the outcome. This was tested both with both diploidy and haploidy, where dominance was used to determine the child's haploid genotype from its parents rather than which gene should be expressed. While there was a drastic difference for the haploid genomes, the results for the diploid genomes were less clear.

The main advantage that has been looked for in diploid GAs is better performance in changing environments. Haploid GAs can struggle when fitness changes, because they have difficulty finding a new solution after having converged on a good one for the initial environment. However, diploid GAs seem to lose less genetic diversity as they adapt [7, 11]. This is because alleles can be preserved and shielded from selection when they are not currently advantageous, if they are recessive and rarely expressed. If the environment changes and that allele becomes advantageous, it is easier for that trait to return in individuals with two alleles and spread through the population, rather than having to wait for mutation to cause the allele to re-appear. Because of this, diploid GAs with a similar phenotype convergence as haploid GAs have more genetic diversity to draw upon.

Another possible advantage for diploid GAs is a greater ability to search complex spaces and find better optima. Bowers and Sevinç [11] speculated about this ability based on how diploid GAs preserved genetic variability even when a haploid GA had entirely converged on a sub-optimal solution. Collingwood et al [12] found that diploidy and higher-level polyploidies could provide an advantage on difficult problems where it was helpful to lose less diversity when converging toward an attractive local optimum, though not when this preservation of diversity was not needed to solve simpler problems. Indeed, Bowers and Sevinç [11] found poorer results from diploid GAs than haploid GAs in simpler, static environments. There may be a solution to improve diploid performance on static problems - changing how crossover works. In real-world biology, crossover mixes genes within a diploid organism before the mixed genome is used to produce haploid gametes, which will combine with other haploid gametes to produce diploid offspring. However, if crossover is changed to mix genes between individuals instead, as is done with crossover operations in haploid GAs, it may improve results on stationary problems without preventing adaptation to changing ones [13].

One last difference between haploid GAs and diploid GAs is that of sensitivity to mutation rate. Mutations can be helpful, but they can also be detrimental. Having too many mutations build up too quickly, before selection can remove bad ones from the population, impedes performance. Diploid GAs have twice as much genetic material as haploid GAs, and thus will build up twice as many mutations given the same mutation rate; however, because mutations can be hidden in recessive genes, the displayed mutation rate in the phenotype will be somewhat lower. Calabretta et al [6] found that diploid GAs performed better with a lower mutation rate than did the equivalent haploid GAs, while Uyar and Harmanci [7] found that two out of three dominance methods were sensitive to mutation rate. The third had a low error rate even at high (30%) mutation rates, leading to speculation that the changing dominance of this method, which the other two lacked, may have provided protection from the mutation rate.

3 DIPLOID NEURAL NETWORKS

Evolving ANNs have been used in many applications, including robotics, control systems, game player AI, and artificial life [14]. It may be possible to use diploidy with evolving ANNs to combine their strengths, such as the ability of diploidy to adapt to changing environments and perhaps search complex spaces and that of evolving ANNs to solve reinforcement learning problems. This combined approach appears to have been tried only once before. Calabretta et al [6] evolved the weights of networks controlling simulated robots. The robots were scored based on how much of their environment they explored, but had to return to a food area periodically. In one condition, this area was fixed; in the other, it moved every 25 generations. While diploids had lower average fitness in the fixed condition, they performed better than haploids in the changing one. In a follow-up study [15], the pace of environmental change was drastically sped up to every other generation. While haploids had difficulty adapting to this, diploids appeared to find a solution for both environments at once.

The system proposed here, programmed in Python 3.6 and using Caffe for its neural networks, is meant to extend this

previous work by Calabretta et al. For example, it is intended to be able to evolve both the weights and the structure of neural networks, although only the weight evolution was tested here. The previous work had no sexual reproduction; new individuals were created by duplication with mutation. The proposed system allows for recombination between two diploid genomes, including crossover between parent chromosomes within each parent. Although there has been speculation that crossover operations are detrimental to evolving neural networks because they could break up sub-structures in the network, it is not clear that this is necessarily the case in practice, and other evolving ANNs have successfully used crossover [2]. In our system, crossover can also be constrained to try to avoid breaking sub-structures.

Based on considerations from previous work discussed in section 2, four chromosomes are used – two for encoding layers, and two for weights. Each gene contains the information needed to make a layer or a weight, along with a mutation rate, which itself can mutate, and flags for allowing or disallowing mutation and duplication. Following the previous finding that having at least four levels of dominance is helpful for diploid genomes, dominance is an integer ranging from one to five. There are also several other adjustable system parameters, such as the default mutation rate used when creating new genes, and the probability of the different kinds of mutation. These mutations include changes to weight, layer size, mutation rate, and dominance.

To create the network, the two layer chromosomes are first compared against each other. Where there are differences, the network checks that both layers can be created. If one of them cannot – for example, if one gene codes for a layer that depends on an input that does not exist in the network – then the other is used. Otherwise, dominance is used to determine which gene is used; if there is a tie, then one is chosen at random. Once this is finished, the network is created, and the weights must be altered following the weight genes. The two weight chromosomes are compared to each other; here, if two alleles coding for the same weight have the same dominance, the resulting weight is the average of those encoded by the two alleles. The random network generated at creation can also be used to create a set of weight genes, if the genome does not have any already.

4 RESULTS AND DISCUSSION

This genetics system was tested in an agent-based simulation environment. The environment consists of cells and wraps around on itself. In it are several objects, including the agents. Each agent, meant to represent a biological creature, has its own set of drives – most importantly, fullness. Agents that reach zero fullness are considered to have starved to death and are removed from the simulation; agents also have a chance of dying dependent on their age, and are removed if they reach a threshold of old age at an age of 130,000 time steps. Agents can move around the environment and perform actions on other objects, such as eating a food source. These movements and actions are controlled by the agent's neural network. The input to this network consists of what the agent can see right in front of it, its own drives, and a short-term memory of its previous actions. An agent can also attempt to mate with another agent it encounters, although whether this is successful depends on each agent's current level of fertility.

The genetics system was tested, with only weight genes able to mutate, with three versions of the genetics, in three versions of this environment. Because random networks tend to perform very poorly in this environment, almost always starving to death as soon as possible at 300 time steps, eighteen diploid genomes which survived for a long time (>=30,000 time steps) in preliminary experiments were first collected and their two weight chromosomes saved separately. The genomes that began the first generation of every iteration of the experiment were drawn from this gene pool. The first genetic variant was a haploid version of the genetics system. The second and third were two variations on diploidy. With the first, 'plain diploid', each beginning agent was given two copies of the same chromosome from the gene pool, while with the second, 'diverse diploids', agents received two different random chromosomes from the gene pool.

The first environment was static, with only one type of food source. In the second, the changing environment, at 100,000 time steps, the food source was switched to a different object, and the simulation would switch back at 200,000 steps if it was still running. In the third, called the 'nutrition' environment, two food objects were present around the environment from the beginning. One type gave one-third of the normal raise in fullness when eaten. Every 40,000 time steps, this switched, so that the type of food which gave one-third of the normal raise began to give the normal amount of fullness, and the type which had given the normal raise began to only give one-third the normal raise. Each environment was set to 13 by 13 cells, and the experiments begun with 20 agents, with the population capped at 40.

Table 1: Results for Success Measures
Values in bold highlight the best performers.

Log(age) mean	Haploid	Plain diploid	Diverse diploid
Static	2.91	2.88	**3.00**
Changing	2.92	2.89	**2.95**
Nutrition	2.87	**3.01**	2.91
Log(age) median			
Static	**2.82**	2.68	2.81
Changing	**2.74**	2.69	2.70
Nutrition	2.64	**2.82**	2.62
Survival mean			
Static	0.51	0.48	**0.59**
Changing	**0.54**	0.49	0.50
Nutrition	0.44	**0.58**	0.42

Although there are several ways to measure success in this environment, such as number of mating attempts, children produced, or how many generations the simulation reaches before the population dies out, here we look only at age at death and whether each agent survived for at least 500 time steps (coded as

0 for death and 1 for living), which is slightly beyond the point where they can first starve to death (Table 1). Although most agents did die within their first few thousand time steps, a few were able to live much longer. Because of the skew of the distribution of lifespans, analysis was done on log of ages.

The initial hypotheses were that the haploid agents would do best in the static environment, while either kind of diploidy would have an advantage over haploidy in the changing environments, and that the changing environments would be more difficult for all of the genetic systems, particularly the 'nutrition' one. A 2-way analysis of variance (ANOVA) on the log lifespans of the agents (Table 2) found a significant effect of both genetics and environment, as well as a significant interaction effect.

Table 2: ANOVA on Lifespan Results

	SumSq	DF	F	PR(>F)
Genetics	8.96	2	17.19	<0.001
Environment	14.62	2	28.05	<0.001
Gen x Env	3.35	4	3.22	0.012
Residual	13073.59	50184		

The results (Table 1) did not quite follow the initial expectations. Haploidy actually tended to perform better than 'plain' diploidy in both static and changing environments, while it was outperformed by 'diverse' diploidy. 'Diverse' diploidy performed best in static environments, worse in changing ones, and had its worst performance in the 'nutrition' environment, though it did better than haploidy in all three environments. However, 'plain' diploidy actually performed best of all three genetic systems in the 'nutrition' environment, and had longer lifespans there than in the other two environments tested.

5 CONCLUSIONS AND FUTURE WORK

Previous work on diploid GAs found that haploid genomes outperformed diploidy in static environments, while diploidy performed better in changing ones. Here, we tested this hypothesis with evolving ANNs in three environments and two kinds of diploidy. However, the haploid genome's performance was in-between that of the two diploids. It is not clear why there is a difference in behavior seen between the diploid agents which started with two copies of the same genes, and those which began with two different chromosomes, and why one performs best in the environment that the other performs worst in. A possible reason for the better performance of the haploidy could be that because of the nature of this task, there are not many places in the search space that work well, and that because haploid genomes converge on solutions faster than diploid ones, it was able to adapt to those few places better. The 'diverse' diploid type could be more successful in the static and changing conditions than the 'plain' diploid type because it started off with twice as much genetic diversity than either of the other conditions, allowing it to explore more possibilities faster; however, why does it then perform worse in both of the two changing conditions than the static ones? Further exploration of how the genomes adapt and converge over time could provide insight into why these differences are seen, and when different genetic systems may perform better than others.

The code for the system proposed here is available on GitHub.[1] In the future, we will further test its ability to evolve network structure, as well as the capability of diploidy in comparison to haploid genetic structures. To better understand what genetic structures work best in which domains, we need to explore more complex environments, different kinds of environmental change, and different models of diploid evolution.

REFERENCES

[1] Xin Yao. 1999. Evolving Artificial Neural Networks. *Proceedings of the IEEE* 87, 9 (September 1999), 1423–1447. DOI:http://dx.doi.org/10.1109/5.784219

[2] Risto Miikkulainen et al. 2017. Evolving Deep Neural Networks. *arXiv preprint* arXiv:1703.00548.

[3] David E. Goldberg and Robert E. Smith. 1987. Nonstationary Function Optimization Using Genetic Algorithms with Dominance and Diploidy. *ICGA* (October 1987), 59–68.

[4] Khim Peow Ng and Kok Cheong Wong. 1995. A New Diploid Scheme and Dominance Change Mechanism for Non-Stationary Function Optimization. In *Proceedings of the 6th international conference on genetic algorithms*. Morgan Kaufmann Publishers Inc., 159–166.

[5] Conor Ryan. 1997. Diploidy Without Dominance. *Third Nordic workshop on genetic algorithms*. 63–70.

[6] Raffaele Calabretta, Riccardo Galbiati, Stefano Nolfi, and Domenico Parisi. 1996. Two is better than one: A Diploid Genotype for Neural Networks. *Neural Processing Letters* 4, 3 (1996), 149–155. DOI:http://dx.doi.org/10.1007/bf00426023

[7] A. Şima. Uyar and A. Emre Harmanci. 2005. A New Population Based Adaptive Domination Change Mechanism for Diploid Genetic Algorithms in Dynamic Environments. *Soft Computing* 9, 11 (February 2005), 803–814. DOI:http://dx.doi.org/10.1007/s00500-004-0421-4

[8] Boris Shabash and Kay C. Wiese. 2015. Diploidy in Evolutionary Algorithms for Dynamic Optimization Problems. *International Journal of Intelligent Computing and Cybernetics* 8, 4 (November 2015), 312–329. DOI:http://dx.doi.org/10.1108/ijicc-07-2015-0026

[9] Jonathan Lewis, Emma Hart, and Graeme Ritchie. 1998. A Comparison of Dominance Mechanisms and Simple Mutation on Non-Stationary Problems. *Lecture Notes in Computer Science Parallel Problem Solving from Nature — PPSN V* (September 1998), 139–148. DOI:http://dx.doi.org/10.1007/bfb0056857

[10] Shengxiang Yang. 2006. On the Design of Diploid Genetic Algorithms for Problem Optimization in Dynamic Environments. *2006 IEEE International Conference on Evolutionary Computation* (2006). DOI:http://dx.doi.org/10.1109/cec.2006.1688467

[11] Robert Ian Bowers and Emre Sevinç. 2006. Preserving Variability in Sexual Multi-agent Systems with Diploidy and Dominance. *Engineering Societies in the Agents World VI Lecture Notes in Computer Science* (October 2006), 184–202. DOI:http://dx.doi.org/10.1007/11759683_12

[12] E. Collingwood, D. Corne, and P. Ross. 1996. Useful Diversity Via Multiploidy. *Proceedings of IEEE International Conference on Evolutionary Computation* (May 1996), 810–813. DOI:http://dx.doi.org/10.1109/icec.1996.54270

[13] Mayada F. Abdul-Halim and Abbas F. Abdul-Kader. 2006. A New Recombination Scheme for Diploid Genetic Algorithms. *IEEE International Conference on Computer Systems and Applications, 2006.* (March 2006), 274–280. DOI:http://dx.doi.org/10.1109/aiccsa.2006.205101

[14] Joel Lehman and Risto Miikkulainen. 2013. Neuroevolution. *Scholarpedia* 8, 6 (2013), 30977. DOI:http://dx.doi.org/10.4249/scholarpedia.30977

[15] Raffaele Calabretta, Stefano Nolfi, Domenico Parisi, and Riccardo Galbiati. 1998. Diploid Robots Adapting to Fast Changing Environments. *ICANN 98 Perspectives in Neural Computing* (1998), 1145–1150. DOI:http://dx.doi.org/10.1007/978-1-4471-1599-1_180

[1] https://github.com/Lekyn/dgeann