Applied Computer Science, vol. 15, no. 3, pp. 21–33
doi:10.23743/acs-2019-18

Submitted: 2019-08-30
Revised: 2019-09-14
Accepted: 2019-09-18

genetic algorithm, neural network, crossover, mutation

Wafaa Mustafa HAMEED [0000-0002-2454-6185][*], *Asan Baker KANBAR*[*]

USING GA FOR EVOLVING WEIGHTS IN NEURAL NETWORKS

Abstract

This article aims at studying the behavior of different types of crossover operators in the performance of Genetic Algorithm. We have also studied the effects of the parameters and variables (crossover probability (Pc), mutation probability (Pm), population size (popsize) and number of generation (NG) for controlling the algorithm. This research accumulated most of the types of crossover operators these types are implemented on evolving weights of Neural Network problem. The article investigates the role of crossover in GAs with respect to this problem, by using a comparative study between the iteration results obtained from changing the parameters values (crossover probability, mutation rate, population size and number of generation). From the experimental results, the best parameters values for the Evolving Weights of XOR-NN problem are NG = 1000, popsize = 50, Pm = 0.001, Pc = 0.5 and the best operator is Line Recombination crossover.

1. INTRODUCTION

Genetic algorithms are a type of optimization algorithm, meaning they are used to find the optimal solution(s) to a given computational problem that maximizes or minimizes a particular function. Genetic algorithms represent one branch of the field of study called evolutionary computation (Koza, 1992), in that they imitate the biological processes of reproduction and natural selection to solve for the 'fittest' solutions (Mitchell, 1998). Like in evolution, many of a genetic algorithm's processes are random, however this optimization technique allows one to set the level of randomization and the level of control (Mitchell, 1998). These algorithms are far more powerful and efficient than random search and exhaustive search algorithms (Koza, 1992; Hameed, 2016; Hameed & Kanbar, 2017), yet require no extra information about the given problem. This feature allows them to find

[*] Assistant lecturer, Department of Computer Science, Cihan University – Slemani, Slemani, Iraq, wafaa.mustafa@sulicihan.edu.krd, asan.baker@sulicihan.edu.krd

solutions to problems that other optimization methods cannot handle due to a lack of continuity, derivatives, linearity, or other features. Genetic algorithm and neural networks are both inspired by computation in biological genetically. Neural networks and genetic algorithms are two techniques for optimization and learning, each having its own strengths and weaknesses. The two have generally evolved along separate paths. (Montana & Davis, 1989; Arjona, 1991; Whitley, 1995), The article investigates the role of crossover in GAs with respect to this problem, by using a comparative study between the iteration results obtained from changing the parameters values (crossover probability, mutation rate, population size and number of generation) system. A good deal of biological neural architecture is determined.

2. PROBLEM DEFINITION

Neural Networks (NN) are biologically motivated approaches to machine learning, inspired by ideas from neuroscience. Recently, some efforts have been made to use genetic algorithms to evolve aspects of NN. (Wright, 1991). A NN consists of layers of processing units called nodes joined by directional links: one input layer, one output layer, zero or more hidden layers in between, and finally, the NN uses bias nodes (in some problems, NN needs no bias nodes) (see Fig. 1).

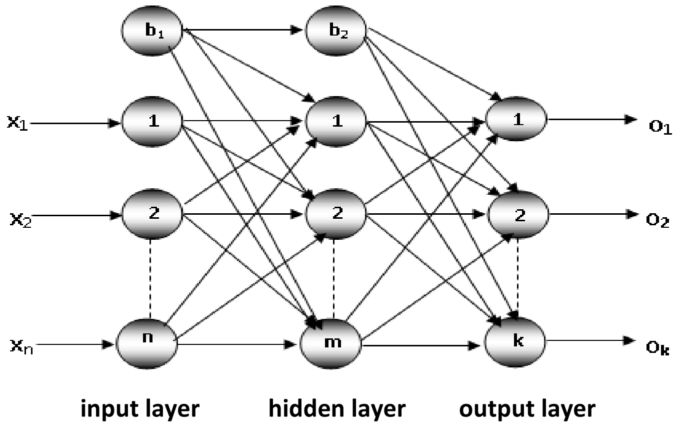

Fig. 1. A schematic diagram of a simple feed forward NN

(Montana & Davis, 1989) took the first approach of evolving the weights in a fixed network. They were using the GA instead of Back-Propagation algorithm and it is desirable to find alternative weight training scheme (Michalewicz, 1996). The simplest Boolean function that is not linearly separable, therefore, this problem cannot be solved by a neural net without the hidden neurons. Table 1 shows the desired relationships between input and output units for this function.

Tab. 1. Training pattern for Exclusive-OR (XOR)

Input		Desired Output
x_1	x_2	
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**0**

3. PROBLEM REPRESENTATION

Each chromosome was a list or vector of 14 weights. Fig. 2 shows how the encoding was done: the weights were read off the network in a fixed order (from left to right and from top to bottom) and placed in a list. Notice that each “gene” in the chromosome is a real number rather than a bit.

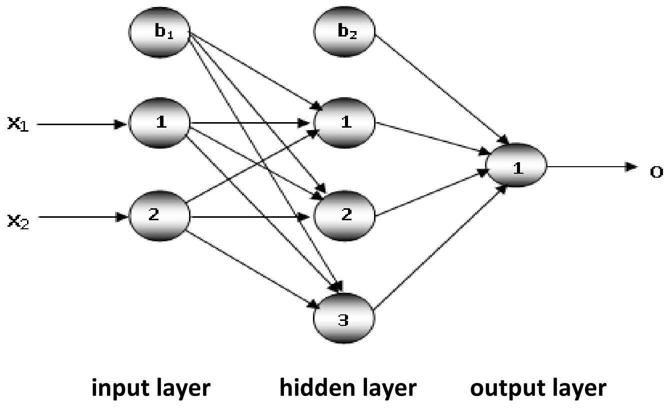

Fig. 2. 2-party XOR (2 – 3 – 1) NN

3.1. Initial Population

The genetic algorithm must create the initial population, which is comprised of multiple chromosome or solutions. An initial population of 13 weight vectors was chosen randomly, with each value based on the proper way to choose the weight W_{ij} in the range of $[-1,1]$ or $[-\frac{3}{\sqrt{k_i}}, \frac{3}{\sqrt{k_i}}]$, where k_i is the number of connection of j to that feed forward to i (the number of input links to i) (Whitley, Starkweather & Fuquay, 1989).

3.2. Evaluation Function

To calculate the fitness of a given chromosome, the weights in the chromosome were assigned to the links in the corresponding network, the network was run on the training set, and the sum of the squares of the errors (collected over all the training cycles) was returned. Here, an “error” was the activation value. The error here is the Mean Squared Error (*MSE*) which represents the square of the difference between the desired output (d_i) activation and actual output (a_i), where $1 \leq i \leq n$, *n* is the number of all possible output values.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(d_i - a_i)^2, \tag{1}$$

For the particular problem, $n = 4$.

Low *MSE* meant a high fitness. In another word, we can obtain the maximum fitness as follows:

$$Fit. = 1 - MSE \tag{2}$$

4. GENETIC OPERATORS

During the alteration phase of the algorithm, we will use the operators described below (Hameed, 2016; Hameed & Kanbar, 2017; Goldberg, 1989).

4.1. Selection Operator

The selection of individuals for crossover and mutation is based toward good individuals. In the classical fitness based roulette-wheel, the chance of an individual. The selected is based on its relative fitness in the population.

4.2. Crossover Operator

Crossover is the operator that creates new candidate solution, in this problem; we can say the One-Point crossover was used. A position is randomly chosen on the string and the two parents are crossed over at this point crossover is mapped, where this occurs at two points along the string.

4.3. Mutation Operator

The mutation operator used in this problem selects *n*-non input units and for each incoming link to those units, adds a random value between (–1.0) and (+1.0) to the weight on the link.

4. GENETIC PARAMETERS

For this particular problem, (Weisman & Pollack, 2002; Al-Inazy, 2005) used the following parameters: population size *popsize* = 20, probability of crossover *Pc* = 0.7, probability of mutation *Pm* = 0.001.

6. EXPERIMENTAL RESULTS

In table 2 we provide the generation number for which we noted improvement in the evaluation function, together with the value of the function. The best chromosome after 1000 generations was:
v_{max} = 2.1433; –2.6102; –0.2982; 4.4594; 4.5946; –0.1168; –4.0; –4.7712; –0.3300; 2.5095; –5.7542; 6.1160; 0.2693. Which is slightly less than 0.0111.

Tab. 2. Results of 1000 generations for evolving weights in NN

Generation number	Evolution function	Fitness
0	0.2610	0.7390
34	0.2457	0.7543
177	0.2195	0.7895
289	0.1883	0.8117
402	0.1325	0.8675
498	0.0861	0.9139
576	0.0364	0.9636
622	0.0262	0.9738
695	0.0159	0.9841
734	0.0121	0.9879
867	0.0111	0.9889

For this problem, a simulation has been constructed in order to apply the GA, using the crossover parameters mentioned above, the v_{min} value has many different sets of weights give *MSE* = 0.010, then the fit. value is 0.99.

7. THE EFFECT OF DIFFERENT TYPES OF CROSSOVER ON EVOLVING WEIGHTS OF XOR-NN PROBLEM

In this part, we will try to study the effect of applying different types of crossover on the reported algorithms, on their performance, speed, and ability to find the solution.

To see the effect of using different types of crossover operators on this problem, Weisman (Weisman & Pollack, 2002; Al-Inazy, 2005) used the Guaranteed Average crossover depending on the following parameters: Pc = 0.7, Pm = 0.001, *popsize* = 20, NG = 1500. Table 3 describes the comparison study of the iterations results between the above crossover and the other kinds which are implemented on this problem. In addition, the table shows the average of iterations results for 10 runs.

Tab. 3. Comparison study of Guaranteed Average crossover and other kinds

Crossover	NG	Fitness
GUA	748	0.9900
ARITH	851	0.9881
DR	870	0.0868
HU	838	0.9900
EX	845	0.9882
IR	762	0.9900
LR	793	0.9900

From table 3, the average iterations results shows that the Guaranteed Average (followed by Intermediate Recombination and Line Recombination) is the best because when a = 0.5 this will makes approximate balance between the vectors x_1 and x_2. The Discrete Recombination is the worst because it generates corners of the hyper cube defined by the parents this may effect on the fitness value.

8. THE EFFECT OF DIFFERENT PARAMETERS ON CROSSOVER

The crossover is an extremely important component of a genetic algorithm. Many GA practitioners believe that if we delete the crossover operators from a GA the result is no longer a GA. In fact; many GA practitioners believe that the use of a crossover operator distinguishes GA from all other optimization algorithms. In this section we will try to study the effect of different genetic parameters on the performance of the proposed algorithms.

9. STUDYING THE EFFECT OF THE PROBABILITY OF CROSSOVER ON EVOLVING WEIGHTS OF XOR-NN PROBLEM

This operator owns a major role in GA, so specifying the probability of crossover, that should not be done randomly, but it must depend on many runs of the simulation to this problem, in order to tune this operator to obtain the fine probability of crossover. We will apply this operator with different values and so other operators. This problem, table 4 shows that the population size, the number of generation and the mutation rate are all fixed, while the crossover probability takes the values 0.0, 0.3, 0.5 and 0.8.

Tab. 4. Crossover probability effect when *NG* = 1000, *popsize* = 50, *Pm* = 0.001.

Crossover	Pc	Iteration	Max. Fit.	Min. Error
GUA	0.0	967	0.99	0.0
	0.5	800	0.99	0.0
	0.8	820	0.99	0.0
ARITH	0.0	980	0.99	0.0
	0.5	725	0.99	0.0
	0.8	646	0.99	0.0
DR	0.0	947	0.99	0.0
	0.5	836	0.99	0.0
	0.8	814	0.99	0.0
HU	0.0	988	0.99	0.0
	0.5	728	0.99	0.0
	0.8	706	0.99	0.0
EX	0.0	904	0.99	0.0
	0.5	911	0.99	0.0
	0.8	837	0.99	0.0
LR	0.0	955	0.99	0.0
	0.5	696	0.99	0.0
	0.8	835	0.99	0.0
IR	0.0	923	0.99	0.0
	0.5	887	0.99	0.0
	0.8	821	0.99	0.0

From table 4 we note the following analytic aspects:

1. The worst iteration results (high iteration levels) are be obtained when the effect of the crossover probability is eliminated (when *Pc* = 0.0).
2. In the most kinds of the used crossovers, including the Guaranteed Average crossover, the iteration results is be improved when using *Pc* = 0.5.
3. the results which appeared shows that the Guaranteed Average crossover, which is used previously, is better than the other operators, which are used to solve this problem.

10. STUDYING THE MUTATION RATE EFFECT ON CROSSOVER FOR EVOLVING WEIGHTS OF XOR-NN PROBLEM

This operator plays a dual role in genetic algorithm, it provides and maintains diversity in a population, so that other operators can continue to work and it can work as a search operator in its own right. We will apply this operator with different numbers of mutation rate and so other operators. In this problem, table 5 shows that the population size, the number of generation and the crossover probability are all fixed, while the mutation rate takes the values 0.0, 0.001 and 0.003.

Tab. 5. Mutation rate effect when *NG* = 1000, *popsize* = 50, *Pc* = 0.8.

Crossover	Pm	Iteration	Max. Fit.	Min. Error
GUA	0.0	888	0.99	0.0
	0.001	820	0.99	0.0
	0.003	1000	0.789	0.25
ARITH	0.0	798	0.99	0.0
	0.001	646	0.99	0.0
	0.003	1000	0.739	0.75
DR	0.0	815	0.99	0.0
	0.001	814	0.99	0.0
	0.003	1000	0.745	0.5
HU	0.0	824	0.99	0.0
	0.001	806	0.99	0.0
	0.003	1000	0.99	0.0
EX	0.0	807	0.99	0.0
	0.001	737	0.99	0.0
	0.003	1000	0.831	0.25
LR	0.0	745	0.99	0.0
	0.001	535	0.99	0.0
	0.003	1000	0.745	5.0
IR	0.0	878	0.99	0.0
	0.001	821	0.99	0.0
	0.003	1000	0.728	1.0
	0.003	**1000**	**0.728**	**1.0**

From table 5 we note the following analytic aspects:

1. When increasing the value of *Pm*, no positive results are gotten.
2. When using *Pm* = 0.001, the Line Recombination and Extended crossovers will give good results.

11. STUDYING THE EFFECT POPULATION SIZE ON CROSSOVER FOR EVOLVING WEIGHTS OF XOR-NN PROBLEM:

The operation which determines the population size is depending on the nature of the problem, that we require solving it. When increasing the complexity of search space, then it needs to a large population. In general, we cannot estimate the real size, but we can give the domain of it. In this operator we use different populations with other operators

In this problem, table 6 shows that the number of generation, the mutation rate and the crossover probability are all fixed, while the population size takes the values 20, 50 and 100.

Tab. 6. Population size effect when *NG* = 1000, *Pm* = 0.001, *Pc* = 0.8.

Crossover	popsize	Iteration	Max. Fit.	Min. Error
GUA	20	738	0.99	0.0
	50	520	0.99	0.0
	100	793	0.99	0.0
ARITH	20	877	0.99	0.0
	50	646	0.99	0.0
	100	1000	0.759	0.25
DR	20	810	0.99	0.0
	50	814	0.99	0.0
	100	1000	0.896	0.0
HU	20	881	0.99	0.0
	50	706	0.99	0.0
	100	851	0.738	0.75
EX	20	909	0.99	0.0
	50	737	0.99	0.0
	100	957	0.99	0.0
LR	20	772	0.99	0.0
	50	635	0.99	0.0
	100	850	0.944	0.0
IR	20	846	0.99	0.0
	50	821	0.99	0.0
	100	1000	0.897	0.0

From table 6 we note the following analytic aspects:

1. For most kinds of the used crossovers, the best results obtained when *popsize* parameter equals 50.
2. The best two crossover operators from all kinds of the used crossovers are the Guaranteed Average and Line Recombination crossovers.

12. STUDUING THE EFFECT OF NUMBER OF GENERATION ON CROSSOVER FOR EVOLVING WEIGHTS OF XOR-NN PROBLEM

In this problem, table 7 shows that the population size, the mutation rate and the crossover probability are all fixed, while the number of generation takes the values 500, 1000 and 1500.

Tab. 8. Number of generation effect when *popsize* = 50, *Pm* = 0.001, *Pc* = 0.8.

Crossover	NG	Iteration	Max. Fit.	Min. Error
GUA	500	500	0.90	0.0
	1000	820	0.99	0.0
	1500	752	0.99	0.0
ARITH	500	500	0. 738	0.75
	1000	646	0.99	0.0
	1500	776	0.99	0.0
DR	500	500	0.739	0.75
	1000	814	0.99	0.0
	1500	756	0.99	0.0
HU	500	500	0.824	0.25
	1000	806	0.99	0.0
	1500	824	0.99	0.0
EX	500	500	0.743	0.5
	1000	737	0.99	0.0
	1500	764	0.99	0.0
LR	500	500	0.879	0.0
	1000	535	0.99	0,0
	1500	699	0.99	0.0
IR	500	500	0.803	0.25
	1000	821	0.99	0.0
	1500	803	0.99	0.0

From tables 6, 7 and 8 we note the following analytic aspects:

1. It is important to mention that, there is a relation between the execution time and the control parameters (number of generation and population size).
2. In XOR Problem by NN, there is no solution to be obtained, when the low levels of generation (NG = 500) are to be chosen.

13. PARAMETRIC STUDY FOR ALL CROSSOVER OPERATORS

From our experiences and experimental results, the best parameters values for all crossover kinds are chosen to make a comparative study for each kind of crossover, for Evolving Weights of Xor-Nn problem and for several runs. The minimum iteration results are shown in tables with statistical diagram to illustrate these results.

For this problem, the best values are: when *NG* = 1000, *popsize* = 50, *Pm* = 0.001 and *Pc* = 0.5. Table 9 shows the comparison between iterations results for 10 runs and Fig. 3 Illustrates the statistical diagram of the comparison study results.

Tab. 9. Comparison study of crossover iterations results for evolving weights of XOR-NN problem

Crossover	NG	Fitness
GUA	816	0.9900
ARITH	871	0.9868
DR	882	0.0881
HU	850	0.9900
EX	872	0.9880
IR	879	0.9882
LR	775	0.9900

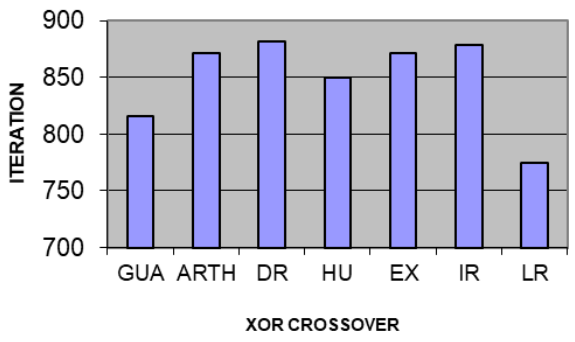

Fig. 3. Statistical diagram of the comparison study results for evolving weights of XOR-NN problem

In this particular problem, from table 9 and Fig. 3, we notes that the operator is the Line Recombination because its selects only one value of the alpha that is generates any point on the line define by the parents and keep the values of chromosome closed to each other.

14. CONCLUSIONS

This research concludes the following points. For evolving weights of XOR-NN problem, the Line Recombination was the best to be applied. For the parametric study, the research concludes: The worst iteration results (high iteration levels) are obtained when the effect of the crossover probability is eliminated (when $Pc = 0.0$), since the search in this state is closed to the random search. For the population size parameters, the result reveals that due to the founder effect, the GA's cannot always locate the peaks of the fitness landscape, even with higher crossover rate. Although, the mutation rate parameter preferred to be chosen as minimum as possible ($Pm \leq 0.01$), but it is still related to the problem which is discussed. It's natural that, there is a relation between the number of generation and population size. This parameter specification is related to the problem which is discussed.

REFERENCES

Al-Inazy, Q. A. (2005). *A Comparison between Lamarckian Evolution and Behavior Evolution of Neural Network* (Unpublished M.Sc. Thesis). Al- Mustansriyah University, Baghdad, Iraq.

Arjona, D. (1996). A hybrid artificial neural network/genetic algorithm approach to on-line operations for the optimization of electrical power systems. In *IECEC 96. Proceedings of the 31st Intersociety Energy Conversion Engineering Conference* (pp. 2286–2290 vol. 4). Washington, DC, USA. doi:10.1109/IECEC.1996.561174

Goldberg, D. E. (1989). *Genetic Algorithms in search, Optimization, and Machine Learning*. Boston, MA, USA: Addison–Wesley Longman Publishing Co., Inc.

Koza, J. R. (1992). *Genetic programming: on the programming of computers by means of natural selection.* Cambridge, MA, USA: MIT Press.

Michalewicz, Z. (1996). *Genetic Algorithm + Data Structure = Evolution Programs*, 3rd Revised Extended Edition. New York, USA: Springer – Verlag Berlin Heidelberg.

Mitchell, M. (1998). *An Introduction of Genetic Algorithms*. Cambridge, MA, USA: MIT Press.

Montana, D., & Davis, L. (1989). Training Feed Forward neural networks using Genetic Algorithms, In *IJCAI'89 Proceedings of the 11th international joint conference on Artificial intelligence* (pp. 762–767). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Hameed, W. M., & Kanbar, A. B. (2017). A Comparative Study of Crossover Operators for Genetic Algorithms to Solve Travelling Salesman Problem. *International Journal of Research – Granthaalayah*, *5*(2), 284–291. doi:10.5281/zenodo.345734

Hameed, W. M. (2016). The Role of Crossover on Optimization of a Function Problem Using Genetic Algorithms. *International Journal of Computer Science and Mobile Computing, 5*(7), 425–429.

Weisman, O., & Pollack, Z. (2002). *Neural Networks Using Genetic Algorithm*. Retrieved from http://www.cs.bgu.ac.il/NNUGA.

Whitley, D., Starkweather, T., & Fuquay, D. A. (1989). Scheduling Problems and Traveling Salesman: The Genetic Edge Recombination Operator. *ICGA*.

Whitley, D. (1995). Genetic Algorithms and Neural Networks. In J. Periaux & G. Winter (Eds.), *Genetic Algorithms in Engineering and Computer Science* (pp. 191-201). John Wiley & Son Corp.

Wright, A. H. (1991). Genetic Algorithms for Real Parameters Optimization. *Foundation of Genetic Algorithms*, *1*, 205-218. doi:10.1016/B978-0-08-050684-5.50016-1