Computational Intelligence - Evolving a Lunar Lander

40280156

ABSTRACT

Evolutionary Algorithms are a powerful tool used for optimising and searching problems, inspired by the natural selection process. In this project, an EA is used to find the best possible weights of a Multi-Layer Perceptron (MLP) used to control a fleet of identical spacecraft, after using most of their fuel, regardless of their different starting positions and velocities. An extensive search was done to find the best hyper parameters of the EA that could find the most effective weights. However, EAs don't find the single, concrete best solution, as that would require searching the whole search space and would take a very long time, but instead, try to find the lowest minimum (or maximum) in the search space which contains a good enough answer.

KEYWORDS

Evolutionary Algorithms, Computational Intelligence, Neural Networks

1 INTRODUCTION

This project aims the find a combination of MLP weights that will produce the lowest fitness score of 8 spacecrafts at the end of the simulation. The fitness score is a formed by the distance from the ending point, the fuel and the velocity at the end of the simulation, therefore an ideal land would obtain a fitness function of zero.

2 METHODOLOGY

This section describes the design of the Evolutionary Algorithm considered, referenced by the literature reviewed for its development, as well as the design of the new operators added and parameter tuning done of the Neural Network.

A solution to this problem is represented by the weights of the neural network used to control the fleet, and its size changes depending on the amount of hidden nodes. The possible values allowed range from -3 to 3, however, this parameters are evaluated later in section 3. The phenotype represents the fitness of a rocket to land safely with the least amount of fuel possible, while in the genotype space, the solution is represented by an array.

The three essential ingredients of an EA as proposed by the literature [7] are:

- Selection of parents, where there are factors in the environment that favour some individuals over others. These will produce one or more children.
- Crossover of genes in the chromosome of the children, to create a variation of the parents, who were selected as the best ones in the population.
- Mutation, to increase diversity in the population and prevent premature convergence. This will allow individuals to escape local optimums and increase the chance of finding the global maxima.

The operators used to perform the above steps are explained below.

2.1 Initialisation

To start evolving the population of an EA, a population must first be created, and since this is not an apriori algorithm were the algorithm already has some knowledge of the problem, most individuals are initialised with random values. Other methods can be implemented to increase the fitness range quality of the initial population [3]. The drawbacks of seeding the population with initially good solutions is that the chance of premature converge is increased, as there will be more exploitation on the current points of the search space and less random exploration that normally occurs at the start.

2.1.1 Random initialisation. As the name proposes, this process randomly initialises each individual in the population, without evaluating them individually. This creates a population that occupies a wide spread of the search space, increasing exploration at the start of the evaluations.

2.1.2 Double initialisation. A modified version of the random initialisation generates two random individuals, and chooses the fittest one out of both of them. This technique is more computationally expensive as it takes double the amount of work or a random initialisation. This increases the chance of having successful candidates since the beginning and decreases the time required to find a good solution. Diversity is still high as their initialisation is still random, and positions all individuals in different points of the search space. However, if the search space is not rugged, and instead is more flat, the points might be closer to each other and this might produce premature convergence as it increases exploitation since the beginning and reduces exploration.

2.2 Selection

The aim of modifying the selection operators is to produce fitter offsprings. Random selection alone will have a very high diversity rate, but will take longer to converge. The following techniques were implemented.

2.2.1 Random. This method simply selects two random parents from the population for crossover to happen. It brings a low selection pressure to the EA, which allows it to escape premature convergence and might be able to find undiscovered places of the search space. However, the algorithm could my be searching aimlessly for a long time [4]. A lack of an adequate convergence rate could cause the EA to fail, hence a random selection method is only useful with appropriate parameters and use of other operators.

2.2.2 Roulette Wheel Selection Method. Also known as Fitness proportionate selection, contributing to creating a fitter generation [15]. It works by calculating the inverse of the fitness of every member of the population ($\frac{1}{chromosome fitness}$) and creating a spinner.

The better the fitness, the more area it contains on the spinner. The spinner then spins around all of the possible fitness' and lands on one parent, selecting it. This is repeated again to obtain the second parent. This allows for other candidates to also be chosen as parents, even the ones who have a worse fitness score, but better candidates have higher chances of being selected. This is an inconvenience when many chromosomes have similar fitness values, since there will be little selection pressure leading to premature convergence, as the method favours the best individuals [1].

2.2.3 Tournament Selection Method. randomly selects a subset of the population and then makes the members 'compete', choosing only the strongest members to be the parents. This can be done once, by selecting one subset and selecting both parents from there; or twice, by selecting two random subsets and one parent from each. This is a simpler method that does not require fitness scaling and selection pressure can be easily be changed with different tournament sizes [15]. The higher the tournament size, the higher the chance of selecting the best individuals, improving the population.

2.3 Crossover

Crossover methods create offsprings by combining the genes of 2 parents chosen previously. In a one-point crossover, a random crossover point is chosen in the chromosome, the genes up to that point are copied from parent 1, and the remaining from parent 2. The second child is made up of the unused genes, the first set come from parent 2 and the last set from parent 1. This keeps genes from good parents and also represents unseen solutions, increasing diversity and therefore the chance of finding a better solution. The methods implemented were one-point crossover, as discusses above, and multi-point and uniform crossover, described below.

2.3.1 Multi-point crossover: similar to a simple crossover, but instead of having only one crossover point, it has N points. In this case, 2 crossover points were applied to the chromosomes for simplicity and because it has been proven in the literature to still give significant better results than only one-point crossover [14].

2.3.2 Uniform crossover: in this method, the chromosome isn't divided into segments, but instead, every gene is treated separately. A random probability is used to decide if that gene is taken from one parent or the other. However, to increase selection pressure, the probability can be biased to have a higher chance of choosing genes from the better parent [16], though this could lead to premature convergence.

2.4 Mutation

Mutation methods introduce new areas of the search space that were not explored before, they introduce new genes and therefore increase diversity this allows the population to not get stuck in a local optima and also consider other points of the search space. It is suggested that mutations only happen based on a probability, so that fitter solutions can be found, instead of dispersing the search too much. It has been proven to produce better results with a small population, while crossover produces fitter offsprings in big populations [2]. In this case, the population used for the testing was 50 solutions, making mutation an important factor to find a fitter generation.

2.4.1 Random Mutation. Loops through every gene of the chromosome with a probability of changing it by a small specified mutation change.

2.4.2 Swap Mutation. As the name implies, this method swaps two genes from a chromosome. It does not introduce new genes so there is less exploration and limits the scope of the search space in which the algorithm can explore. This operator can change two independent genes or a subset of the chromosome with another subset.

2.4.3 Limited Mutation. This method loops through every gene, just like the random mutation method, and with a small probability, it adds or subtract a specified mutation change. The new solution is then evaluated and the change is only accepted if there is an improvement in the fitness. This method was implemented with the aim of increasing exploitation of good solutions.

2.5 Replacement

When fitter offsprings are found by crossover or mutation, it is vital to replace them with members of the current population to drive evolution forward and create fitter generations. They can be replaced by random, the oldest or worst members.

2.5.1 Random. Randomly selecting a member of the population to be replaced by the offpring. This method has the disadvantage that the member selected has the best fitness of the population, as sometimes crossover or mutation methods do not find a fitter offspring.

2.5.2 Fitness Based. These methods can be used to replace the worst members of the population. The two methods implemented were

- replacing the worst member of the population
- selecting a subset of the population and replacing the worst one from there (tournament replacement), maintaining diversity but decreasing exploitation

2.5.3 Aged Based. These methods aim to find new areas of the search space by keeping track of how long each member of the population has been in the population, and replacing the oldest ones. These methods aim to emulate problems in a more natural way, and have been proven to solve complex optimization problems very effectively [8].

2.6 Diversity

Diversity methods introduce new individuals into the population, allowing the algorithm to search new areas of the search space and increasing the chances of finding a better solution. The algorithms used for this problem were the following.

2.6.1 Hill climber. The hill climber method is a local search algorithm used to optimize a single solution. It is ran after the EA has completed its evolutions. The best solution found is passed to the hill climber local search algorithm and it keeps mutating it until it finds a better solution. Then, the new solution is mutated until the maximum number of iterations is reached.

2.6.2 Sawtooth. This method decreases the population size at a steady rate until it reaches a specified size and then is refilled with completely new and random individuals. The more added individuals, the more diversity and the higher the chances of finding a fitter solution. This method was implemented because of its proven effectiveness [5] and simplicity.

2.7 Parameter Optimisation

No algorithm has a set of parameters that works for every single problem instance. For an EA to be effective, it has to be adapted to the dataset used. This EA works by constantly changing the parameters of the NN, however, there are some parameters that can't be changed as the algorithm runs, these are the hyper parameters, the ones defininf the initial EA structure. These hyper parameters have to be evaluated to find the best ones that train on the dataset without overfitting and then produce similar results on the test set. The parameters tested in section 3 were the following:

- The possible range of values in the gene
- The mutation change done to the genes
- The mutation rate or probability
- The number of hidden nodes of the Neural Network
- The population size

2.8 Activation Function

Once the input signals (weights) of a neural network are calculated, an activation function is applied, that will determine the inner state of the neuron and produce the output signal of it. The function can be linear or non-linear, which simulates more precisely the non-linear transfer characteristics of the neuron. It is usually non-linear so the NN can potentially approximate any function (given 2 hidden layers with sufficient numbers of neurons). The Activation Function itself is a number between 0 and 1. The closer it is to 1, the more the neuron signal is passed (it is more activated).

The activation functions tested in this case were the following:

- tanh - similar to the commonly used Sigmoid function but accepts smaller values as it ranges from -1 and 1, while sigmoid only has values between 0 and 1
- Rectified Linear Unit - ReLU - it ranges from 0 and infitiny. When values are less than 0, they stay at zero, but if they are higher, they stay as they are.
- Scaled Exponential Linear Units - SELU - this method was first used in this paper [13] where the authors fix the problem of the dying ReLUs
- Step function - also called as binary step activation as it produces an output of 1 if the value passes the threshold and a 0 if it doesn't.

3 RESULTS

To evaluate the best hyper parameters, each operator was tested separately, regardless of the other ones. The algorithm ran 10 times, 10000 iterations each, and the best fitness of each time was saved to a file so it could then be averaged and the results evaluated and compared. The paremeters initially used were:

- Number of hidden layers: 10
- Min and max gene values: -3 and 3
- Population size: 50
- Tournament size: 7
- Sawtooth
  - Remove member of population every 5 iterations
  - Refill population when population size reaches 25
- Activation Function: tanh
- Mutation rate: 0.04
- Mutation change: 0.1
- Initialisation: random
- Selection: tournament
- Crossover: uniform
- Mutation: random
- Diversity: only sawtooth, no hillclimber
- Replacement: replace worst candidate

3.1 Initialisation

For the two initialisation methods proposed (random and best out of 2), they both seemed to have similar results.

Operator	Training set fitness	Test set fitness
RANDOM	0.124993173	0.173425794
BEST OF 2	0.127592351	0.168324569

It was mentioned before that the *best of 2* method would probably produce better results as better members of the population were chosen to start with. As supposed to that theory, the random initialisation method resulted in a fitter generation. This could be due to the increase in initial diversity or it could mean that initialisation methods do not have an impact on the final outcome. This should be further explored in future work. The results of both the training set and the test set where compared and evaluated using the Shapiro Wilk Test and T-test, which will not be explicitly explained as they are well known tests. For all the initialisation methods, the p value was more than 0.05, meaning that the null hypothesis that the samples come from a population which has a normal distribution cannot be rejected, i.e. the models are consistent even though they have a stochastic start every time they are initialised as the results of the 10 experiments all ended up in the same range of fitness scores.

3.2 Selection

The different selection methods implemented achieved the following results:

Operator	Training set fitness	Test set fitness
TOURNAMENT	0.103536354	0.135867723
RANDOM	0.129903487	0.144856287
ROULETTE	0.098816173	0.100034864

As the literature mentioned above suggests, the results are true to it and the Roulette method results in a fitter generation by favouring fitter members of the population to be chosen as parents. Randomly selecting parents seems to do more exploration rather than exploitation. The t test was used to test the differences between the 3 methods and they were all less than 0.05, meaning that the null hypothesis can be rejected and therefore the results do not have identical averages scores [12]. This proves that the results are reliable and that roulette selection certainly results in a fitter generation.

For the tournament method, the size of the tournament was also tested (as part of the parameter testing) and the following results are shown in Figure 1. Which resulted in a size of 5 producing the best results, probably because it decreases selection pressure and increases diversity.

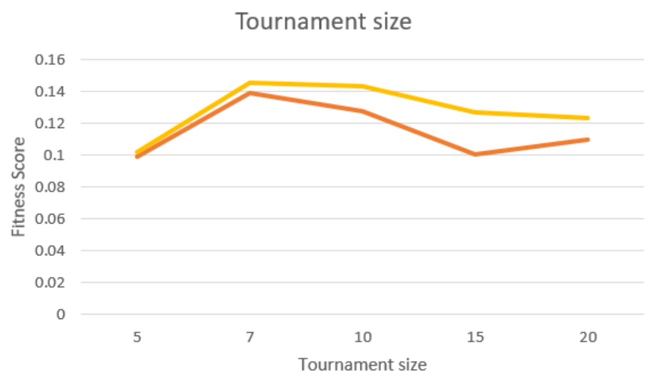

Figure 1: Number of hidden nodes VS Average fitness scores. (Orange - Training set results, Yellow Test set results)

3.3 Crossover

The crossover results showed that uniform crossover produced better results overall, in both cases, by a significant amount. Both methods had a chance of 0.5 of happening or not in each iteration, hence why to test crossover and mutation operators, 20 iterations were ran instead of only 10. This allowed for a more precise average.

Operator	Training set fitness	Test set fitness
UNIFORM	0.144351509	0.185160162
DOUBLE POINT	0.170916361	0.227819264

3.4 Mutation

The three mutation methods tested were random, swap and constrained mutation. It was believed that constrained would produce the best results.

Operator	Training set fitness	Test set fitness
RANDOM	0.146511148	0.18234274
SWAP	0.071791384	0.093481548
CONSTRAINED	0.114576185	0.124862550

Surprisingly, the swap method produced the best outcome, by a significant difference. This could be due to the fact that the chromosomes are less mutated, as only 2 genes are swapped, and the rest are kept the same. Even though constrained only kept the new mutation if it had a better fitness, the new candidate could have been found on a different hill of the search space, losing the one we had before, which could have been better. Therefore it seems that exploiting the current members of the population rather than exploring other areas of the search space produced better results.

As part of the parameter testing, the mutation rate and change were evaluated and can be seen on Figure 2. The tests show that the higher the mutation, the higher the chances of finding a better member, which contradicts what was described above. However, these parameter only change the amount of change of each gene and the probability with which they are changed, but the methods still determine how these mutations occur.

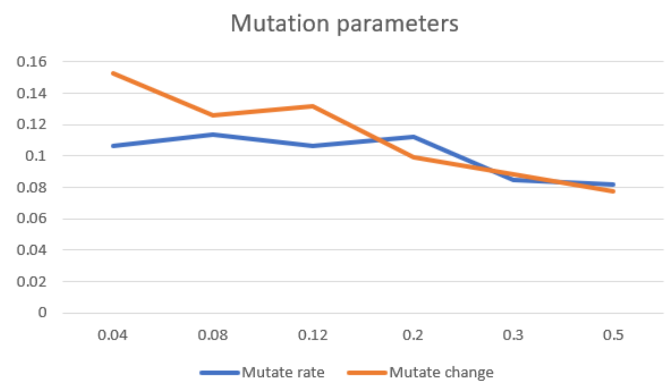

Figure 2: Mutation parameters VS Average fitness scores.

3.5 Replacement

As expected, replacing the worst member of the population was more efficient than using other methods. It was believed that the higher the diversity, the better the chances of finding a better solution. However, depending on the selection type used, replacing the worst or other member of the population would have no effect if worst members were not selected to be parents. Therefore, to make these tests fair, the tournament selection method was used, as stated before.

Operator	Training set fitness	Test set fitness
TOURNAMENT	0.155322893	0.172344975
RANDOM	0.244417189	0.284655389
REPLACE WORST	0.139874164	0.150238977

The tournament size was tested along with the tournament size of the selection method and hence produced the results shown in Figure 1.

3.6 Diversity

The sawtooth method, supposed to be necessary to increase diversity in the population, did not produce a significant difference when it was not used. Again, this could be due to the chosen selection method. It could also prove that mutation is more important in cases were the population is smaller (as it was in this case, with 50 members), and crossover is when the populations are greater, as mentioned before in Section 2.4.

Operator	Training set fitness	Test set fitness
SAWTOOTH	0.113858039	0.149645213
NO SAWTOOTH	0.123735204	0.167835629
HILL CLIMBER	0.092365748	0.121248923
NO HILL CLIMBER	0.111349245	0.124634289

The hill climber algorithms finds many solutions when the best fitness found during the evolution process is not optimal. However, when the EA finds very low solutions, the hill climber barely finds a better solution, therefore it does not cause much difference.

3.7 Activation Function

The different activation functions were tested and the results are shown below. Even though the SELU function is technically an improved version of the ReLU function, it turned out to produce worse fitness scores. ReLU is the most common used activation function in Neural Networks [9] and these tests prove why.

Operator	Training set fitness	Test set fitness
TANH	0.106300864	0.131927459
RELU	0.104907376	0.129345782
SELU	0.117617576	0.156732490
STEP	0.112843472	0.153462893

3.8 Parameter testing

The results obtained when testing the operators still did not achieve the best fitnesses found while developing it. This is probably as a result of the general hyper parameters still not being experimented with.

3.8.1 Population size. Most studies suggest that a large population can help increase diversity and move an EA forward, however, other studies have proven the opposite [6] and are more computationally expensive. A rule of thumb is to choose the size of the population as 10 times the number of dimensions, which in this case is a number that keeps changing depending on the number of hidden nodes. Therefore the following population sizes were tested (shown in Figure 3).

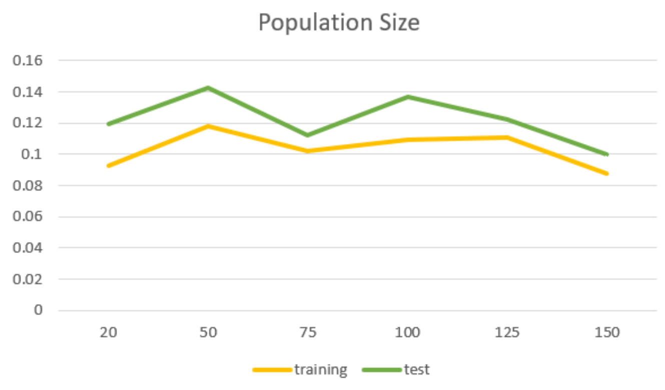

Figure 3: Population size VS Average fitness scores.

The two boundaries provided the best results (a population size of 20 and 150). However, since 50 was the amount chosen to test all operators and did not produce the lowest results, 150 was the number chosen for the final model.

3.8.2 Number of hidden nodes. The number of hidden nodes was found to increase the fitness of the training set as it was increased. However, this produced overfitting on the training test and resulted in worse results on the test test. Therefore a balance had to be found between both of them and for this reason, the amount of hidden nodes chosen was 15.

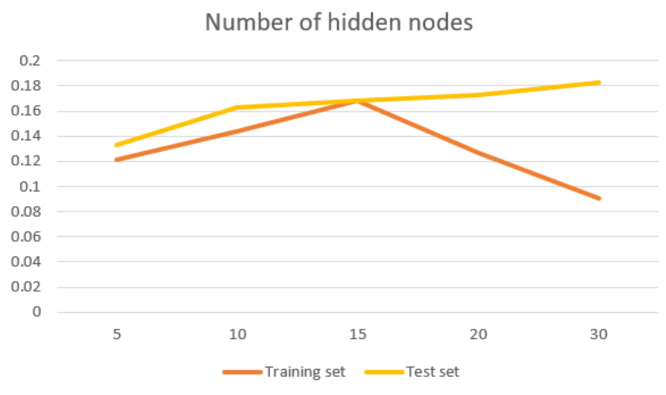

Figure 4: Number of hidden nodes VS Average fitness scores.

3.8.3 Minimum and maximum values for each gene. The initial values of the gene turned out to be good values to produce fit members. However, after comparing both training and test set, it was clear that the optimum values were -2 and 2. The graph (Figure 5) shows a clear curve where those values achieve the minimum fitness values.

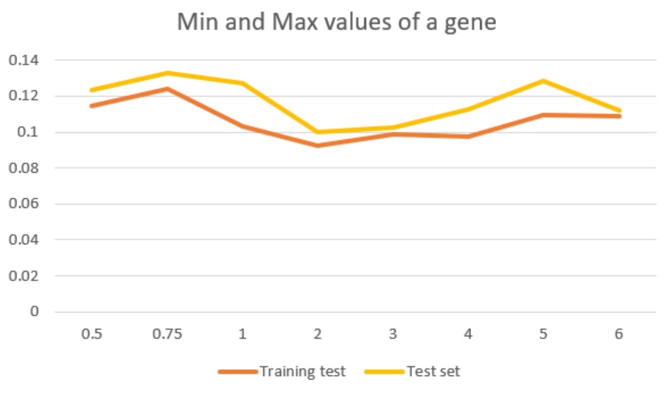

Figure 5: Min-Max Gene VS Average fitness scores.

4 CONCLUSION

The extensive search allowed to find the best combination of parameters that produced the fittest generation. However, during testing, the parameters were tested at the same time, in the same script, with the same values and not one by one after they were tested, which means that the results displayed above do not show the best combination of parameters. After all the operators and parameters were combined, the results achieved were the following:

Final model	Training set fitness	Test set fitness
	0.015143976803579293	0.2679186570204928

The parameters used to produce these were:

- Operators: random initialisation, Roulette selection, Uniform crossover, Swap mutation (with 0.75 mutation rate and 0.9 mutation change), worst member replacement and both sawtooth and hill climber as diversity operators
- Neural Network parameters: ReLU activation function, 15 hidden nodes and a gene values between 2 and -2.
- Population size of 50 or 150

In conclusion, an effective Evolutionary Algorithm was developed to train the weights of a Neural Network landing lunar rockets safely. The operators used were evaluated and tested to find out which ones had a higher effect on changing the fitness values as well as which ones produced the fitter generations. It was suggested that the crossover method depends highly on the selection method chosen, as it decides the amount of selection pressure, and in the contrary, the replacement method was critical to moving the evolution forward and a good approach was critical to finding the best solutions.

In some cases the results confirmed the theories produced in the literature review, such as the Roulette selection method begin the optimum one, while in others, it demonstrated alternative ideas, like for instance the random initialisation producing better results than the best of 2 method.

4.1 Future Work

Future work could look into developing an Island Model [10] or a Simulated Annealing approach to increase diversity [11]. These methodologies could find new ways of escaping local optima and have higher chances to finding the perfect balance so that the model would no overfit to the training set.

A grid search could have been implemented to simply provide a list of parameters of the Neural Network to be tested and find the best combination without the need of an Evolutionary Algorithm. However, this process is more computationally expensive and using Evolutionary Algorithms together with ANNs can accelerate the learning process as it has been shown in previous reviews. A grid search function could then be used to test all the combinations of the hyper parameters of the EA together with the NN to be tested with normal, boundary and abnormal values and have it test every single possible combination programmatically, reducing the time spent in parameter tuning.

REFERENCES

[1] Rosshairy Abd Rahman, Razamin Ramli, Zainoddin Jamari, and Ku Ruhana Ku-Mahamud. 2016. Evolutionary algorithm with roulette-tournament selection for solving aquaculture diet formulation. *Mathematical Problems in Engineering* 2016 (2016).

[2] Alexandru Agapie. 2001. Theoretical analysis of mutation-adaptive evolutionary algorithms. *Evolutionary Computation* 9, 2 (2001), 127–146.

[3] Edmund K Burke, James P Newall, and Rupert F Weare. 1998. Initialization strategies and diversity in evolutionary timetabling. *Evolutionary computation* 6, 1 (1998), 81–103.

[4] Erick Cantú-Paz. 2001. Migration policies, selection pressure, and parallel evolutionary algorithms. *Journal of heuristics* 7, 4 (2001), 311–334.

[5] Junliang Cao, Junjun Cao, Zheng Zeng, Baoheng Yao, and Lian Lian. 2017. Toward optimal rendezvous of multiple underwater gliders: 3D path planning with combined sawtooth and spiral motion. *Journal of Intelligent & Robotic Systems* 85, 1 (2017), 189–206.

[6] Tianshi Chen, Ke Tang, Guoliang Chen, and Xin Yao. 2012. A large population size can be unhelpful in evolutionary algorithms. *Theoretical Computer Science* 436 (2012), 54–70.

[7] Keith L Downing. 2015. *Intelligence emerging: adaptivity and search in evolving neural systems.* MIT Press.

[8] Ashish Ghosh, Shigeyoshi Tsutsui, and Hideo Tanaka. 1996. Individual aging in genetic algorithms. *Proceedings of the Australian and New Zealand Conference on Intelligent Information Systems,* 276 – 279. https://doi.org/10.1109/ANZIIS.1996.573957

[9] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics.* 315–323.

[10] Raquel Hernández Gómez, Carlos A Coello Coello, and Enrique Alba. 2020. A Parallel Island Model for Hypervolume-Based Many-Objective Optimization. In *High-Performance Simulation-Based Optimization.* Springer, 247–273.

[11] Ameet V Joshi. 2020. Evolutionary Algorithms. In *Machine Learning and Artificial Intelligence.* Springer, 99–106.

[12] Tae Kyun Kim. 2015. T test as a parametric statistic. *Korean journal of anesthesiology* 68, 6 (2015), 540.

[13] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. 2017. Self-normalizing neural networks. In *Advances in neural information processing systems.* 971–980.

[14] Usama Mehboob, Junaid Qadir, Salman Ali, and Athanasios Vasilakos. 2016. Genetic algorithms in wireless networking: techniques, applications, and issues. *Soft Computing* 20, 6 (2016), 2467–2501.

[15] Seyedali Mirjalili. 2019. Genetic algorithm. In *Evolutionary algorithms and neural networks.* Springer, 43–55.

[16] William M Spears and Kenneth D De Jong. 1995. *On the virtues of parameterized uniform crossover.* Technical Report. Naval Research Lab Washington DC.