NEUROEVOLUTION OPTIMIZATION TECHNIQUES

Name: LOH JIN XIAN

Student ID: 016763

University of Nottingham Malaysia Campus,
Department of Computer Science
Semenyih, Selangor, Malaysia

Abstract – Neuroevolution is a modern heuristic search that is based on the principle of evolution which follows the concept of genetic inheritance and Darwinian's survival of the fittest. It is widely used in solving many optimization problems. An important goal of this research is to discover and discuss better techniques to further optimize this heuristic search so it can be more efficient while solving the problem.

Keywords – Genetic algorithm, Optimization problems, Simulated Annealing, Neuroevolution, Evolutionary algorithm

I. INTRODUCTION

For the past 2 decades, Neuroevolution and Simulated Annealing (SA) have been used to solve optimization problems [1]. Both Neuroevolution and SA search a solution space throughout a sequence of iterative states. However, the underlying mechanics are different between these two metaheuristics.

Neuroevolution attempts to implement an evolutionary process that is quite similar to the human brains through an evolutionary algorithm. Given the environmental stochasticity of our real world, a supervised learning approach is not always possible. Therefore, the goal of neuroevolution is to be able to train neural networks in sequential decision tasks with sparse reinforcement information [2]. Evolutionary algorithms are problem-solving algorithms inspired by biological evolution with the idea of improving a population of individuals instead of just one individual [3].

Simulated annealing is used to find a better potential solution in the search space. SA is inspired by the annealing process of metallurgy. By having the possibility of accepting worse solutions, SA enables for a more extensive search for the global optimal solution. During the exploration of the solution space, the probability of accepting worse solutions will decrease overtime by decreasing the temperature.

Nevertheless, both Neuroevolution and SA have advantages and disadvantages. One disadvantage of Neuroevolution is that it might be trapped in a local minima due to less exploitation. As for SA, only one candidate solution is used, thus it does not build up an overall view of the search space thus less exploration. The Combination of Neuroevolution and SA is implemented to improve potential solutions' quality and reduce execution time. Besides that, some modifications in crossover heuristic and trimming population mechanic are used to optimize the neuroevolution to be more efficient.

II. METHODS

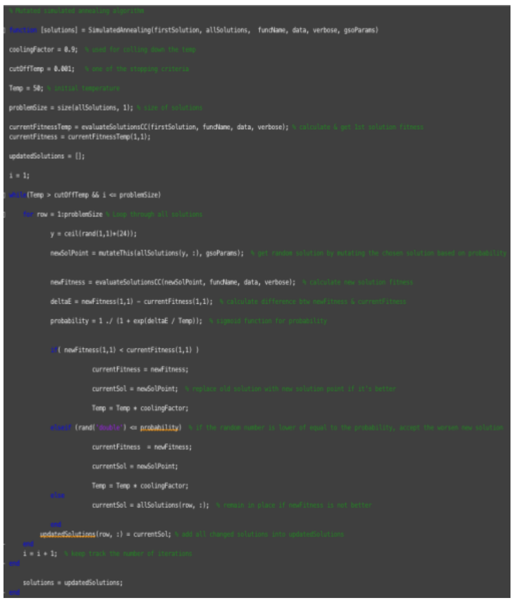
Methods used to optimize the neuroevolution are explained below:

1. **Crossover heuristic (modified)**
   In the given neuroevolution, the crossover heuristic that is performed on the parents in the initial population is known as 1-point crossover. The purpose of crossover is to generate new offsprings that help in finding new solutions. By modifying the crossover from 1-point crossover to 2-point crossover, diversification of the new population can be improved which aid in finding a solution closer to the global minima.

2. **Trimming population mechanic (modified)**
   In the default trimming population mechanic, the population are trimmed based on the constant minimum differences between the leader and followers. By changing minimum difference value to be dynamic, which is calculated by taking the average of all solutions' fitness, there is more randomness in which solutions are being trimmed.

3. Mutated Simulated Annealing (new heuristic)

The traditional SA is a local search heuristic that moves from solution to solution in the space of candidate solutions by applying local changes. In this new proposed version of simulated annealing which is called Mutated simulated annealing (MSA), a random solution is generated by mutating one of the solutions in the population.



```
% Mutated simulated annealing algorithm

function [solutions] = SimulatedAnnealing(firstSolution, allSolutions,  funcName, data, verbose, gsoParams)

coolingFactor = 0.9;  % used for colling down the temp

cutOffTemp = 0.001;   % one of the stopping criteria

Temp = 50; % initial temperature

problemSize = size(allSolutions, 1); % size of solutions

currentFitnessTemp = evaluateSolutionsCC(firstSolution, funcName, data, verbose); % calculate & get 1st solution fitness
currentFitness = currentFitnessTemp(1,1);

updatedSolutions = [];

i = 1;

while(Temp > cutOffTemp && i <= problemSize)

   for row = 1:problemSize % Loop through all solutions

         y = ceil(rand(1,1)*(24));

         newSolPoint = mutateThis(allSolutions(y, :), gsoParams);  % get random solution by mutating the chosen solution based on probability


         newFitness = evaluateSolutionsCC(newSolPoint, funcName, data, verbose);  % calculate new solution fitness

         deltaE = newFitness(1,1) - currentFitness(1,1);  % calculate difference btw newFitness & currentFitness

         probability = 1 ./ (1 + exp(deltaE / Temp));  % sigmoid function for probability


         if( newFitness(1,1) < currentFitness(1,1) )

                  currentFitness = newFitness;

                  currentSol = newSolPoint;  % replace old solution with new solution point if it's better

                  Temp = Temp + coolingFactor;

         elseif (rand('double') <= probability)  % if the random number is lower of equal to the probability, accept the worsen new solution

                  currentFitness  = newFitness;

                  currentSol = newSolPoint;

                  Temp = Temp * coolingFactor;
         else
                  currentSol = allSolutions(row, :);  % remain in place if newFitness is not better

         end
       updatedSolutions(row, :) = currentSol; % add all changed solutions into updatedSolutions
   end
   i = i + 1;  % keep track the number of iterations
end

   solutions = updatedSolutions;
end
```

Figure 2.0 Algorithm of Mutated Simulating Annealing (MSA)

Referring to figure 2.0, the fitness of this new solution will be calculated and compared with the fitness of the current solution in the population. If the fitness of the new solution is better, replace the old solution in the population with the new solution. The same process applied if the delta (calculated using new fitness subtracts current fitness) of MSA is lower or equal to the probability generated. If the current solution's fitness is better, then the solution will remain in the population. By applying this new heuristic into neuroevolution, we can intensify (refining) the search in the vicinity of solutions [4].

III. RESULTS

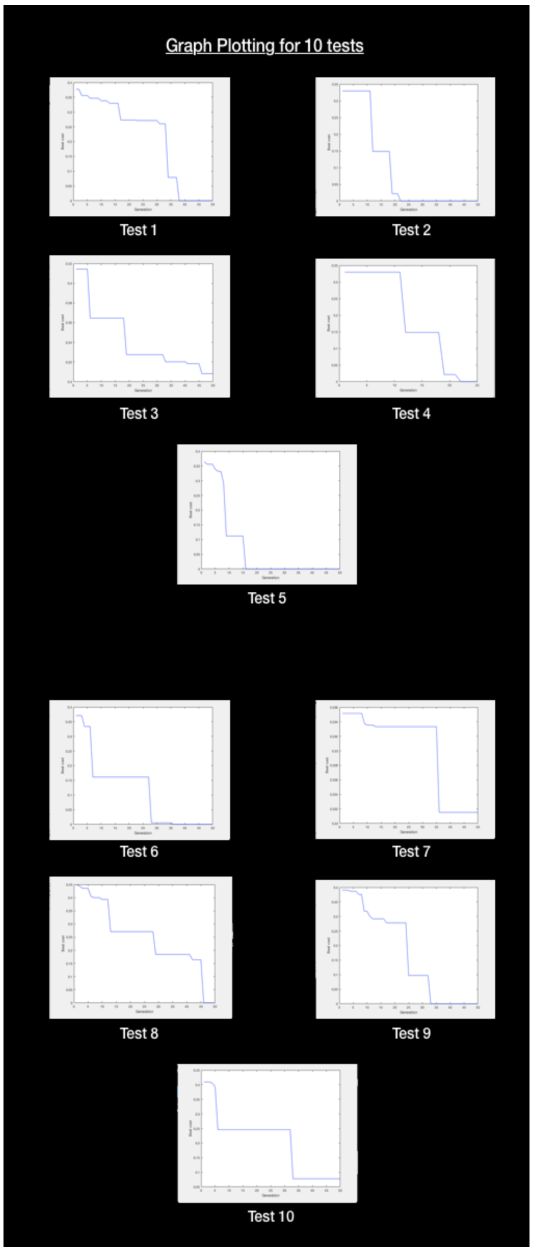


Figure 3.0 Graph plotting for 10 tests

As seen in figure 3.0, the newly optimized neuroevolution is able to achieve global minima at a considerable amount of execution time. To visualize the data better, all data collected are inserted into the boxplot as seen below:

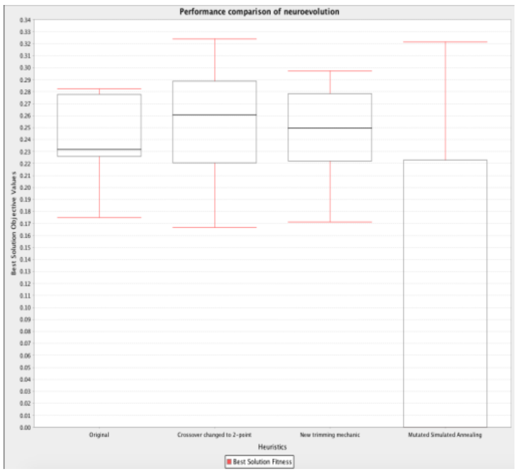


Figure 3.1 Performance comparison between different neuroevolution

The boxplot is generated by inserting the data collected with each new neuroevolution. Each new change made to the neuroevolution are built on top of the previous changes. Hence, the last boxplot on the right shows a substantial improvement compared to the first boxplot.

IV. DISCUSSION

The main objective of this research is to optimize the neuroevolution to be capable of obtaining a solution that is nearer to the global minima as much as possible. Therefore, the 3 changes made as discussed in 'Methods' section above have achieved the objective. Nevertheless, there is always more refinements that can be done to the algorithms used in exploration and exploitation in the search space.

For example, heuristics that affect the diversification of a population can be optimized by changing the way an offspring is created. Referring to figure 3.1, the boxplot for changed crossover performs worse compared to the original boxplot because the design of heuristics should be based on the problems that we are solving. That is why understanding the underlying concept and working of Neural Diversity Machine (NDM) is crucial in perfecting the heuristic search.

Other than that, the method used to select a random new solution in simulated annealing can also be adjusted to find a nearer neighbour to the current solutions in the neighbourhood instead of using randomness which improve the consistency of the data. In the end, both consistency and accuracy of the heuristic search should be prioritised equally.

V. REFERENCES

[1] Y. R. Elhaddad, “Combined Simulated Annealing and Genetic Algorithm to Solve Optimization Problems,” 2012, p. 4.

[2] R. Miikkulainen, “Neuroevolution,” *Encyclopedia of Machine Learning, New York 2010. Springer.* p. 6, 2010.

[3] V. Hoekstra, “An overview of neuroevolution techniques,” VU university in Amsterdam, 2011.

[4] H. E. Lehtihet, “What is the difference between ‘exploration’ vs. ‘exploitation’, ‘intensification’ vs. ‘diversification’ and ‘global search’ vs. ‘local search’?,” 2014. [Online]. Available: https://www.researchgate.net/post/What_is_the_difference_between_exploration_vs_exploitation_intensification_vs_diversification_and_global_search_vs_local_search. [Accessed: 25-Apr-2018].