Hybrid Genetic Algorithm

s1976852

March 2021

Contents

Abstract

The aim of this report is to show the results of applying a Hybrid genetic algorithm (HGA) to solve the Travelling Salesman Problem. This Algorithm was found to be more effective in the search of optimal solutions than the proposed algorithms (2-Opt and Greedy). However, the running time of the HGA was found to be significantly higher than the time of other heuristics as 2-Opt.

1 Algorithm

1.1 Introduction

Genetic algorithms (GA) are inspired in biological process of Natural selection.
In Genetic Algorithms it is usual to start with a random population of feasible solutions, then that population is ranked according to a fitness function that is problem dependant ($\frac{1}{tour\_value^2}$), the individuals with a higher fitness are more likely to breed. The old population is replaced by the new one (which has the 'genes' of the best parents), To maintain diversity and to be sure that the algorithm does not get stuck in a local minimum so easily, each offspring has a probability of mutation.
Genetic Algorithms are expected to converge[1] into the optimal solution as the best results from one generation are taken to the next one and there is still room for variability due to mutations (in case that the algorithm initially converges into a local minimum). The hybrid part of the algorithm comes from the fact that usually traditional GA take a high number of generations to converge into an optimal solution. Therefore, 2-opt heuristic was implemented inside the genetic algorithm to make the algorithm converge faster.

1.2 Motivation

The choice of this algorithm over other possible heuristics was highly motivated by curiosity. It was found interesting the fact that one could use a biological procedure to obtain a good result for a problem that has little connection with that field. The use of Genetic Algorithms to solve the TSP is really common and there were many resources online that helped to develop the HGA.

1.3 Sources

The idea of using 2-Opt in combination with the Genetic algorithm was taken from the paper "Integrating the Best 2-Opt Method to Enhance the Genetic Algorithm Execution Time in Solving the Traveler Salesman Problem"[2]. However, many other sources where use to decide the best implementation for the genetic algorithm. In the mentioned paper they substitute the mutation operator by the 2-opt optimisation. This was not done in the presented algorithm as it is believed that mutation is important to maintain diversity of the population. Instead, the 2-Opt local search was always performed to a user-defined percentage of the population.

1.4 Description

1. Initialize
   The initial population is constructed by randomly generating viable paths.

2. Selection
   A roulette wheel selection was implemented, so all the individuals of a population can be chosen to be parents of the new generation, but those with higher fitness are more likely to be chosen. Elitism was also implemented, so that it is guaranteed that the best solutions always pass from generation to generation (No mutation or optimisation is performed in this solutions and they directly pass to the new generation). $N - elite\_size$ crossovers will be needed after the selection phase is finished.

3. Crossover Operator
   The choice of a good Crossover operator is essential to develop a good Genetic Algorithm. For the proposed algorithm the Operator Chosen was a Greedy Crossover Operator. It tried to build a valid path by at each step considering the next city proposed by both parents and selecting randomly one of the two, but taking as weights for the probabilities the distance between the previously added city and the two possible choices. If

a city of one parent it moved to the next city of that parent to be considered, and if a city had already been added to the path in construction and it was found in the other parent the program would also move to the next city of that parent.

4. Mutation Operator
   for this algorithm the Mutation operator consisted in just swapping two genes (cities) of an individual if a random number was less than the probability of mutation. (This was tried for each node in each individual)

5. 2-Opt Local search
   After producing the offspring the 2-Opt Heuristic is applied to a percentage (user-defined) of the children. The selection of those which will be optimised through 2-Opt is random.

1.5 Running Time

The running time of the Hybrid Genetic Algorithm is polynomial on the size of the input (number of nodes) given that the number of individuals in each population, the limit of generations and the number of iterations in each 2-opt is defined by the user (No left until the solution has totally converged). From now on N denotes the number of individuals in a population, n denotes the number of nodes, G the number of generations and k the number of optimisations in 2-opt. The initialisation phase just involves crating N possible solutions, which are arrays of n numbers. So this phase involves generating n*N random numbers. Hence, it is bounded by $\mathcal{O}(n)$.
Then in each generation the algorithm needs to rank the generation, make the crossover, mutate and perform 2-Opt on the Offspring. Ranking involves basically sorting the individuals using the fitness so it involves calculating the fitness of all individuals $N\mathcal{O}(n) = \mathcal{O}(n)$ and sorting the N individuals, which is independent of n. Then the algorithm needs to perform the crossovers, it will perform $N - elite\_size$ crossovers and in each crossover it needs to build a new solution which will take up to $\mathcal{O}(n)$, so the total time spent in the crossovers will be bounded by $N - elite\_size\mathcal{O}(n) = \mathcal{O}(n)$. The same analysis applies for the mutation as the algorithm has to go trough all the nodes of all the children, so that part also contributes $\mathcal{O}(n)$. Then the HGA performs 2-Opt on a percentage of the children (user-defined, but at most 1), given that k is defined, each 2-Opt operation is bounded by $\mathcal{O}(n^2)$, so as the number of 2-Opt operations is defined this part will contribute $\mathcal{O}(n^2)$ each generation. All of this will be repeated each Generation G. So adding everything gives

$$\mathcal{O}(n) + G[\mathcal{O}(n) + \mathcal{O}(n) + \mathcal{O}(n) + \mathcal{O}(n^2)] = \mathcal{O}(n^2)$$

The last step is true given that G is defined and not a variable.

2 Testing

In this section some of the tests conducted in the HGA are presented. For contrast the tests were also conducted using the proposed algorithms, Greedy and the combination of SwapHeuristics ans 2-Opt. All the tests were conducted using the metric setting as they were easier to generate randomly and also there were plenty of famous sets to compare with. For the purpose of testing the parameters used for the HGA were: N = 50, EliteSize = 5, mutationRate = 0.1, G = 50,k = 10 and the percentage of childs at which the algorithm applies 2-Opt is 0.2,

2.1 Efficiency Tests

This section aims to explore the values obtained by the HGA and the other algorithms. To allow an in-depth comparison several randomly-generated graphs were considered, with values ranging from 2 nodes to 50. For this tests the value of k for the 2-Opt and Swap heuristic was set to 12, however, it is expected that varying k will generate similar results.
Firstly the values obtained by the different algorithms were obtained for the different graphs, the results shown below are represent one instance of those tests. However, similar results are obtained for any randomly generated euclidean graph.

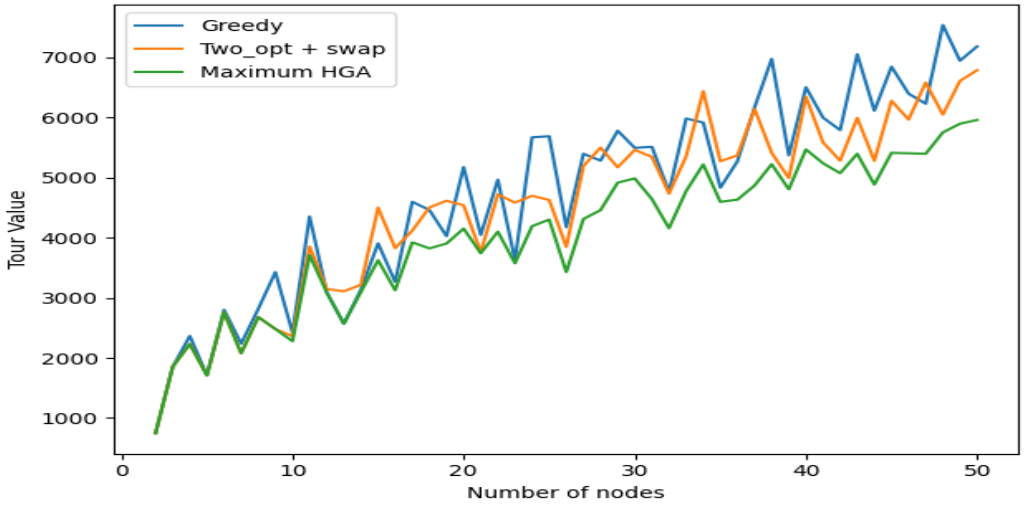

Fig. 1: This graph shows the best values obtained by each algorithm for graphs of different number of nodes, the values are not important individually but when compared between the algorithms. HGA is found to perform better in all instances, this behaviour was maintained in all the tests of this type. This graph represent one instance of those tests.

Another important feature that can be tested on randomly generated euclidean graphs is the time taken by all the algorithms. The number of nodes explored were the same that in the Efficiency tests. The results are shown in the graph below.

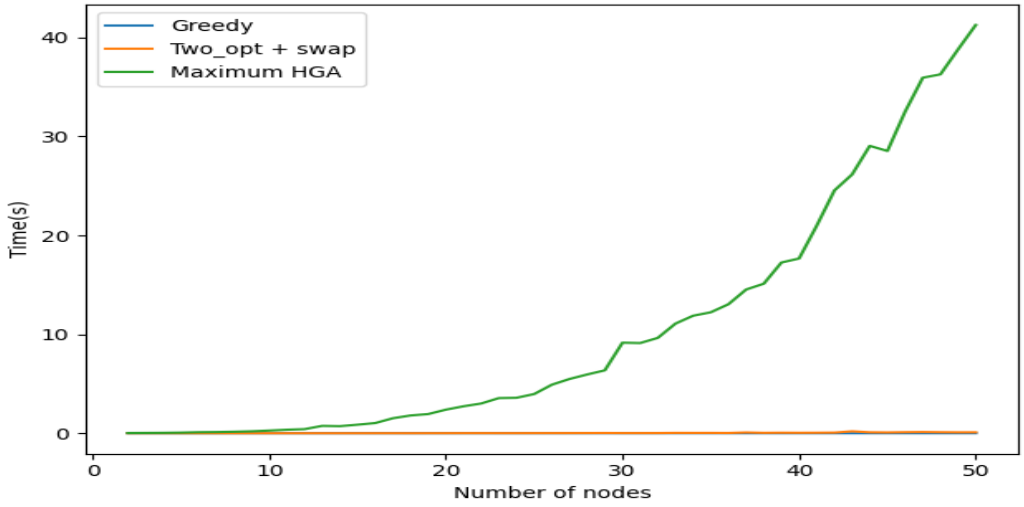

Fig. 2: It is easy to see that the running time cost of the HGA is much greater than any of the two other alternatives. The individual time values are not really important(As they will vary from one computer to another), and one should focus on the comparison between the different algorithms.

2.2 Efficiency on Known Graphs

It is also important to test how close the algorithms are of the optimal solution, therefore our algorithms were tested against some known graphs. The Graphs used in these tests were 0liver30, att48, st70, pr76, KD100, lin105, xqf131.[1]

[1]The optimal tours were found on TSPLIB, however the actual optimal values will be different in our case as the function that we used to calculate the distance is slightly different from the one used in TSPLIB, this does not change the fact that the optimum tour

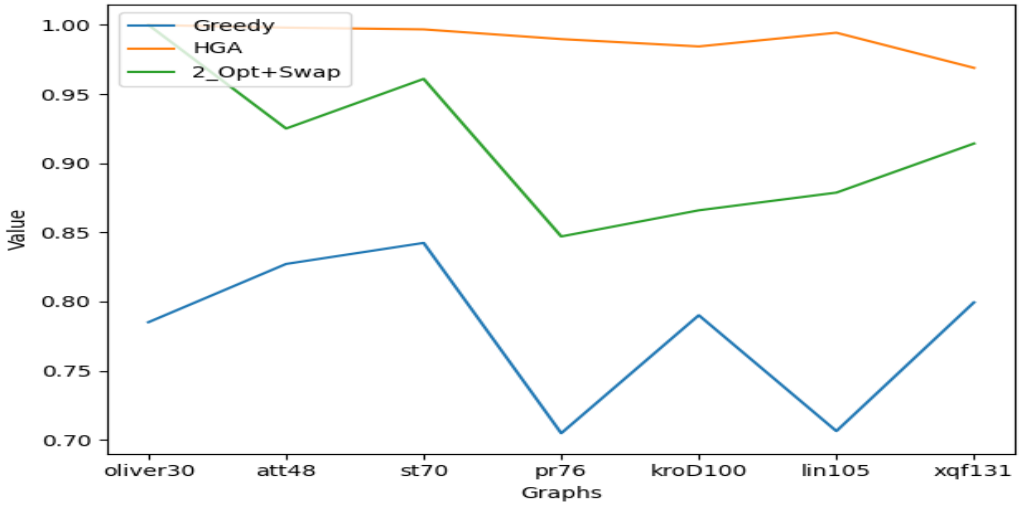

Fig. 3: This graphs shows $\frac{OptimumValue}{tourValue}$ vs the graphs, the closer to one the more optimum the solution is. It can be seen that the HGA finds values close to the optimal solution (and even the optimal solution in some instances), a better performance could be obtained by adding more individuals to the initial population or by increasing the number of generations. However, this would have a great impact on the running time.

2.3 Effect of the Euclidean Window

The effect of expanding the window for the Euclidean TSP was explored by calculating the best tour with the different algorithms for 30 nodes and varying the window. The results are shown below

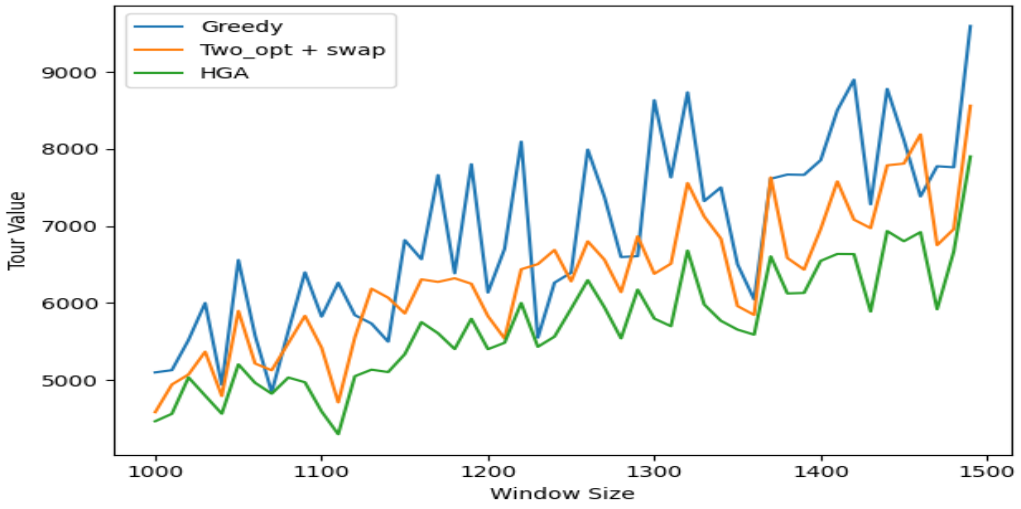

Fig. 4: This graph shows the best values obtained by each algorithm for graphs of 30 nodes, the difference between each graph is the range of values considered for the coordinates of each node ("The window"). It can be seen that now great effect is detected by increasing the size of the window as the HGA is the best in all instances and Greedy is usually the worst.

This ends testing and the proper report the appendix is optional and it does not include anything relevant for the testing or the description of the algorithm.

given is actually the optimum

A Images of Obtained tours

This is left just as they are pretty.

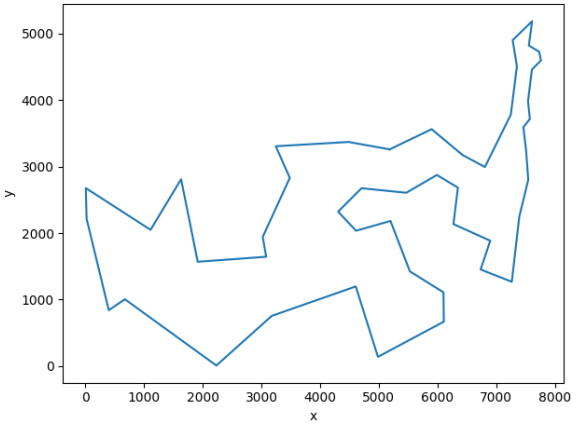

Fig. 5: Tour obtained for att48 with HGA

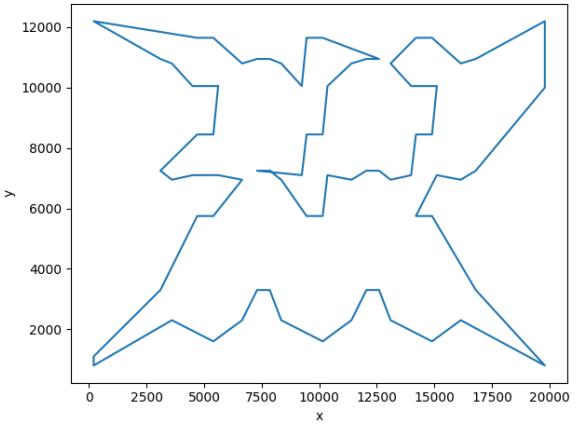

Fig. 6: Tour obtained for pr76 with HGA

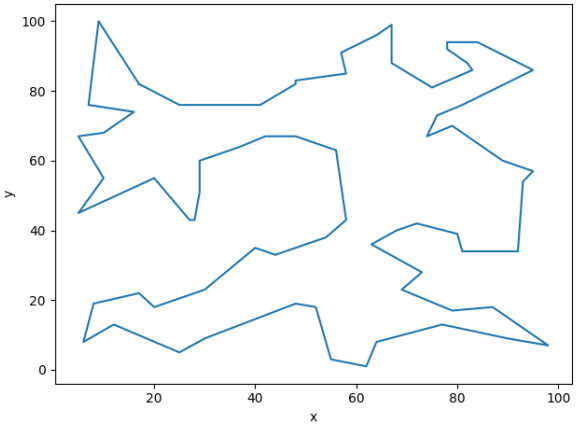

Fig. 7: Tour obtained for st70 with HGA

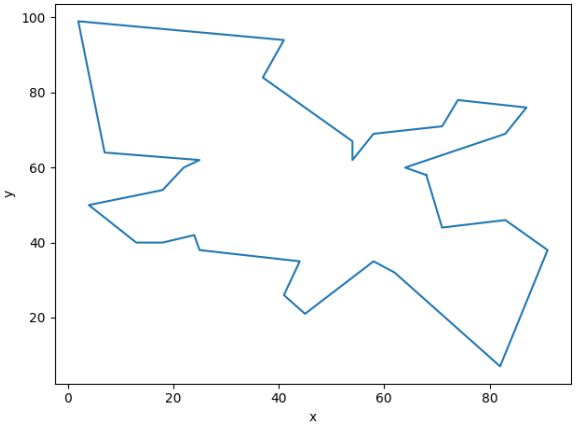

Fig. 8: Tour obtained for oliver30 with HGA

References

[1] Dana Bani-Hani. Genetic algorithm (GA): A Simple and Intuitive Guide. `https://towardsdatascience.com/genetic-algorithm-a-simple-and-intuitive-guide-51c04cc1f9ed`. Accessed: 14/03/2021.

[2] Sara Sabba and Salim Chikhi. Integrating the best 2-opt method to enhance the genetic algorithm execution time in solving the traveler salesman problem. In Wojciech Zamojski, Jacek Mazurkiewicz, Jarosław Sugier, Tomasz Walkowiak, and Janusz Kacprzyk, editors, *Complex Systems and Dependability*, pages 195–208, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.