Application of genetic algorithms to the knapsack and travelling salesman problem

Group 1: Tu Anh Dinh - i6164898, Alain van Rijn - i6182927

1 Kapsack

1.1 Approach

1.1.1 Genotype Representation and Fitness

For the 0-1 version of the knapsack problem, a sensible representation is a list of n entries, where every entry is either a 1 if that item is carried in the knapsack and 0 otherwise. Besides that, there is an array of weights and one for rewards corresponding to each item's indices. Weights get initialised randomly uniform between 1 and 10 regardless of the problem size. Rewards are equal to the index of the item + 1. In reasonable solutions, we expect items with the higher indices and thus rewards to be present as weights are expected to be about 5.5 while rewards increase with the underlying index. The fitness value of an individual is determined as the dot product of rewards and genotype or in other words, the sum of rewards that the genotype of the individual obtains.

1.1.2 Initialisation

Our case's initialisation is done by initialising random integers 0 or 1 with equal probability and only accepting the genotype if the weight is below the threshold. Dealing with initialisation as described already puts pressure on the initial population so that all solutions are feasible. The lower the maximum weight parameter, the more solutions have to be rejected, and the more computation is done in this step by generating random numbers to throw away. A way to mitigate waste here is to tweak the probability of picking the number one or alternatively allow infeasible solutions into the population and punishing excess weight in other ways. The choice was made to stick with equal probability as initialisation is only done once per population member at the start and will likely be negligible in the run time overall.

1.1.3 Selection

The selection method evaluated for the knapsack problem is called roulette wheel selection. Individuals have a chance to be selected for the mating population in proportion to their fitness. One can think of a customised continuous roulette wheel where the pockets represent individuals and vary in size according to that individual's fitness. The roulette wheel is tuned by the temperature parameter where a lower temperature increases representation of fitter individuals in the mating pool. The mating pool is picked with replacement from the population; otherwise, there would be no selection. When looking at the chosen representation, one can observe that the dependencies between genes are weak, except that weight units can only be spent once. Therefore, items in the knapsack will exclude others. There are no esoteric combinations or building blocks that suddenly lead to great fitness, so we can hypothesise that diversity is not that important. High pressure will likely lead to faster convergence and stability afterwards.

1.1.4 Crossover

The crossover for the knapsack problem can be done in many ways since genes contribute individually to the fitness value. This work is done by using half the solution from each parent split in the middle. This could be problematic if no mutation would be done since you would only keep the same starting and ending sequences propagating. With this simple recombination scheme on the knapsack problem, one should always have a mutation to enable new solutions. We committed ourselves earlier to legal solutions only and enforce this by copying the parent in case

the proposed child exceeds the weight limit. This slows the effect of crossover, but optimal solutions will indeed not become unfeasible in this phase. Because of the legality check, mild recombination, and relatively independent genes, the need for copying is far less. The crossover rate can likely be safely set close to or at one.

1.1.5 Mutation

An individual's knapsack mutation is done by setting a gene to its alternative value with some probability. In the mutation, it is again imperative to be mindful of the legality restriction. As before, an illegal solution means the mutation will not be done . As opposed to knapsack crossover, there seems to be a greater risk of damaging good solutions since the mutation is not informed by selection beforehand. Knapsack mutation poses a direct trade-off between preservation and exploration, so the mutation rate can't be set to high without consequences.

1.2 Experiments

There are possibilities to tweak parameters as well as subroutines. Something to keep in mind is that the 0-1 knapsack problem has 2^n solutions where n is the number of unique items. Therefore, having a population size near that amount defeats the genetic algorithm's purpose as it would just be enumerating every solution with extra overhead. The following parameter settings are used:

Fixed:

- number of items: 32
- maximum sum of weights: 64
- population size: 512
- maximum generations: 10

Variable:

- mutation rate
- temperature

1.3 Results

1.4 Discussion

The intuition that the fixed point recombination would not cause instability was correct. setting the crossover rate at 1 Figure 7 gives the highest value

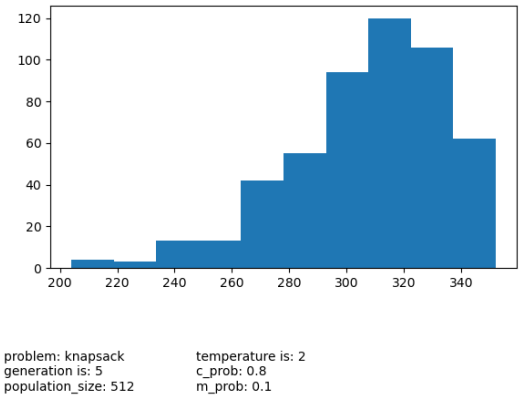


Figure 1: initial pop

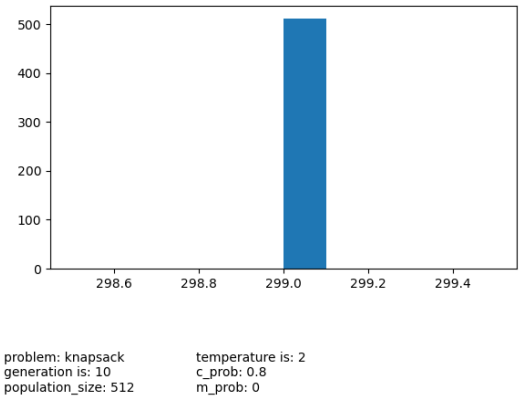


Figure 2: no mutation

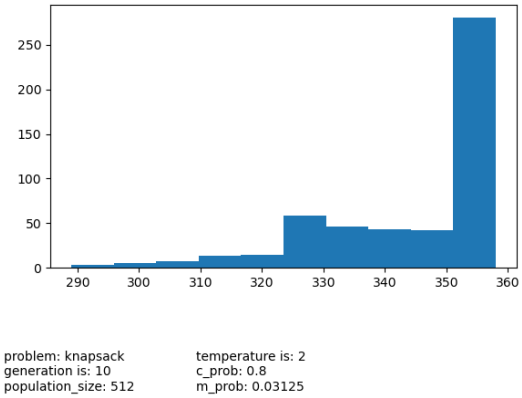
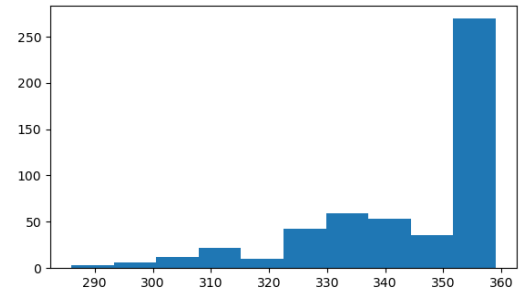


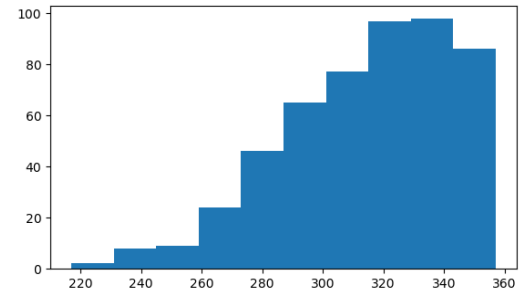
Figure 3: small mutation



problem: knapsack
generation is: 20
population_size: 512

temperature is: 2
c_prob: 0.8
m_prob: 0.03125

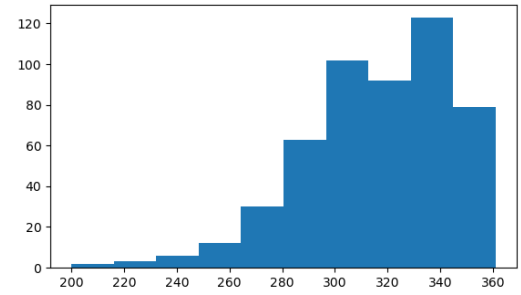
Figure 4: small mutation on last iteration



problem: knapsack
generation is: 10
population_size: 512

temperature is: 1
c_prob: 0.8
m_prob: 0.1

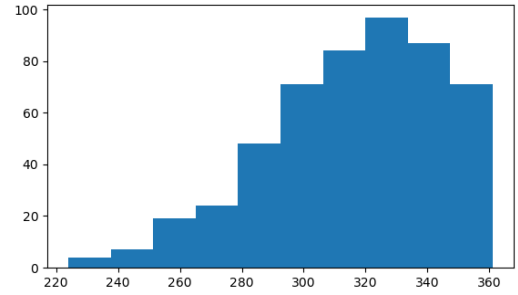
Figure 5: 10% mutation in gen 10



problem: knapsack
generation is: 20
population_size: 512

temperature is: 2
c_prob: 0.8
m_prob: 0.1

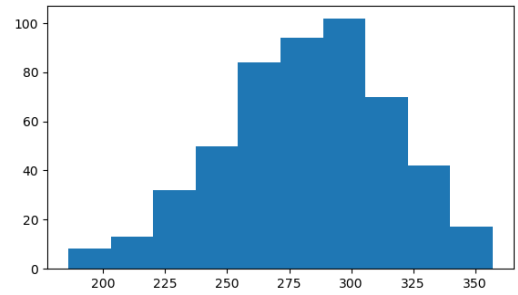
Figure 6: 10% mutation in gen 10



problem: knapsack
generation is: 10
population_size: 512

temperature is: 4
c_prob: 1
m_prob: 0.1

Figure 7: 10% mutation 100% crossover



problem: knapsack
generation is: 15
population_size: 512

temperature is: 4
c_prob: 1
m_prob: 0.2

Figure 8: 20% mutation 100% crossover

solution in generation 5, which is the fastest to reach this of all settings. A higher mutation rate Figure 8 on a relatively long genome seems to degrade too many solutions. This also conforms with the expectations of mutation. There is a $(1 - MR)^n$ chance that a solution stays the same after mutation. Improving a max weight solution takes turning off one or more genes and turning on one or more high rewarding genes. The feasibility filter does not allow temporarily having overweight solutions. With too much elitism, the solutions that don't use up the weight budget might not survive, which would cause the algorithm to get stuck without help from a crossover. Even at low temperature Figure9, the selection pressure can't cope with excessive mutation, and optima can easily be lost. As expected, a zero mutation rate Figure 2combined with fixed point recombination gives no information to the gene pool and can therefore not find better solutions unless the parts already existed in the initial population. It appeared that the mild recombination of fixed point

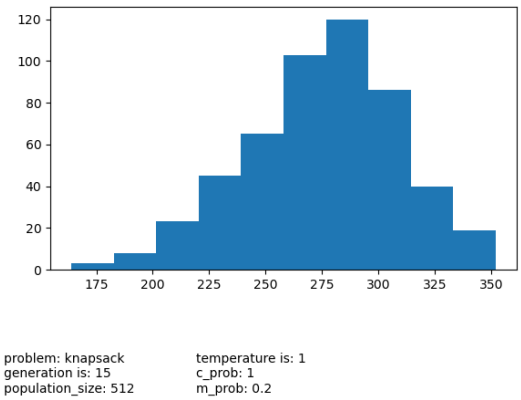


Figure 9: 20% mutation 100% crossover low temp

crossover is sufficient to improve the population as long as it is combined with a form of mutation to introduce new information. Faster convergence might be achieved with initialisation heuristics that favour a high reward to weight ratio. Additionally, the initialisation could be informed by the weight budget parameter to produce relatively more feasible starting solutions and reduce the overhead from rejection.

2 Travelling Salesman Problem

2.1 Approach

2.1.1 Genotype Representation and Fitness

The travelling salesmen solution should represent an order of visiting cities. This order is shown by a list of cities to traverse where the first and the last index are the same city and don't change during runtime. The sequence will have the number zero twice, and the numbers one till n-1 uniquely appearing in-between. Any solution is legal as long as the start and end are the same, and all cities are visited. The fitness value is the total distance travelled where lower is better.

2.1.2 Initialisation

All cities in the experimentation instance lie equidistantly on the unit circle. It is the job of the genetic algorithm to discover this. Individuals are initialised with a random order of visiting cities. As there are n! permutations possible, the probability of randomly guessing an optimal solution is much smaller compared to the 0-1 knapsack problem with the same sufficiently large genotype length.

2.1.3 Selection

In tournament selection, individuals enter a tournament of size k. The draw of k is without replacement, but subsequent drawings are taken from the whole population, placing back the k. When k equals the population size, all parents will be copies of the fittest individual in the population. K = 1 would be the equivalent of random roulette wheel selection. The advantage of tournament selection for the travelling salesmen problem is the indifference to the magnitude and sign of the fitness value. When k ¡ population size, there remains a small probability that the best individual will not be in the mating pool, and this could be mitigated with elitism. In the implementation, only the size of k is varied but use elitism in the crossover and mutation instead of at selection time.

2.1.4 Crossover

With the use of the order crossover operator (Hussain et al.) A sequence or random length from one parent is recombined is fixed in place. The remaining slots from the second cut of point get the remaining cities not in the sequence in the order they appear in the other parent. The cut points are the same for both children in the crossover. Although it can be argued that having different cut points for the children yield slightly more diversity, this is probably not worth the extra computation as there will be many different cut off points in other crossovers with the same or other parents. The option is built in to enable family elitism where only the two best of four survive. This elitism seems advantageous in TSP as we can crossover aggressively without any downside to the children's fitness value.

2.1.5 Mutation

By the use of inversion, a sub tour happens in reverse (Pullan) . Just like with family elitism, the option is built-in only to accept the mutation if it is an improvement. If there is a search path that requires an inversion to a strictly worse solution to reach the global optimum, the algorithm could get stuck, because it won't accept this change. However, there is a crossover in place that changes the tour differently, so this shouldn't be a problem.

The mutation rate can be cranked up to 1 without risking losing the best solution by having this mutation elitism enabled.

2.2 Experiments

One experiment of interest was to see whether it is beneficial to have crossover rates of 1 with family elitism or relatively lower rates with or without elitism. Likewise, it will be tested for the mutation, whether a rate of 1 and only accepting the solution if it is better than the non-mutated individual. Additionally, selection pressure will be varied by adjusting the tournament size. Note that the algorithm can be terminated once a solution appears with all cities ordered as 0,1,2,...,n-1,0 or 0,n-1,n-2,...,1,0 as this is the known optimum for traversing points on a unit circle. Indeed the algorithm stops once it reaches this solution.

Fixed:

- number of cities(on the unit circle): 50
- population size: 512
- maximum generations: 100

Variable:

- mutation rate and elitism
- crossover rate and elitism
- tournament size

2.3 Results

Each generations histogram plots are created before selection. The line plots are created at termination. Termination takes place whenever either the best solution is found, or 100 generations passed.

2.4 Discussion

The results show that the fastest convergence among the experiments happen when mutation and crossover elitism are enabled, their rates at 1 and the tournament size at a quarter of the population see Figure 14. The hypothesis that the rate of crossover and mutation can be set to maximum in the presence of family and mutation elitism seems to hold. It makes sense that the mentioned parameter settings perform best on the TSP for multiple reasons. The tournament size is large but not as big as the population Figure 14; if that

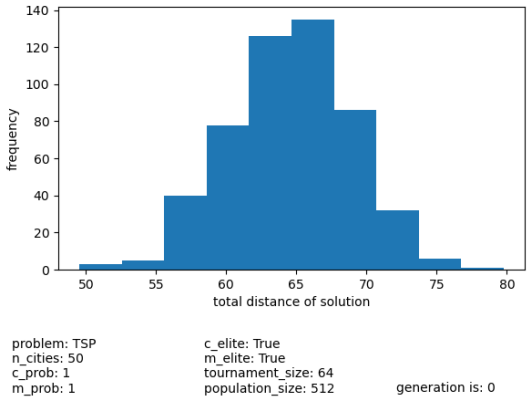
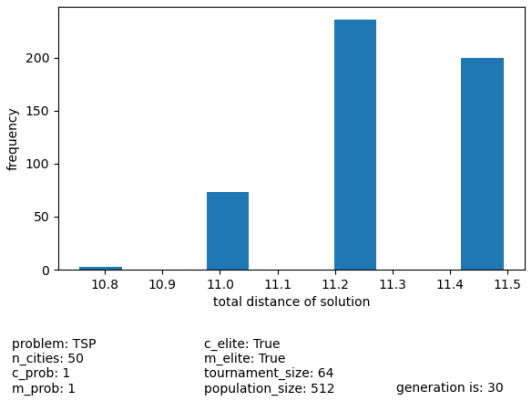
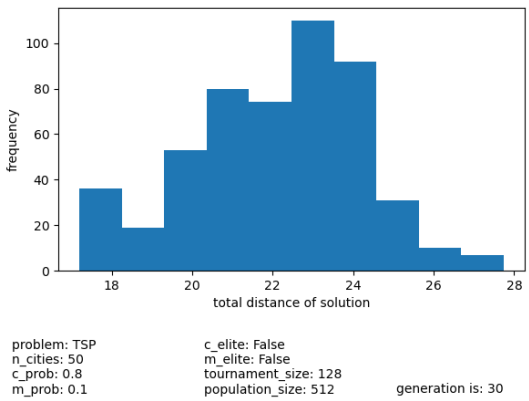


Figure 10: initial population





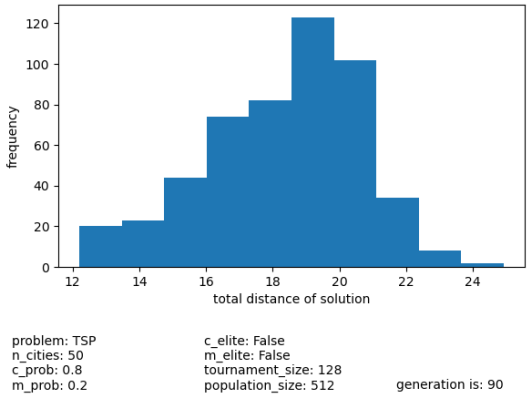
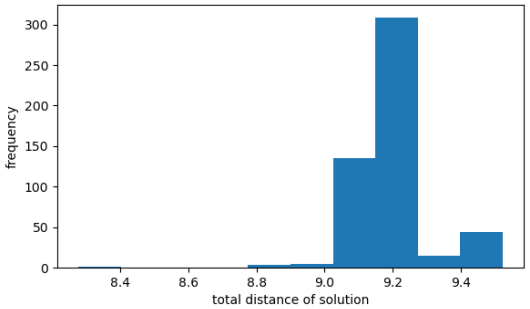


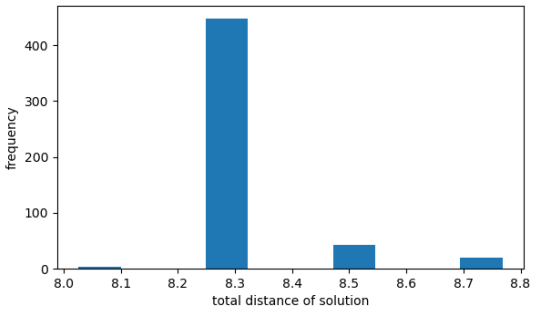
Figure 11: no elitism crossover and mutation



problem: TSP
n_cities: 50
c_prob: 1
m_prob: 1

c_elite: True
m_elite: True
tournament_size: 128
population_size: 512

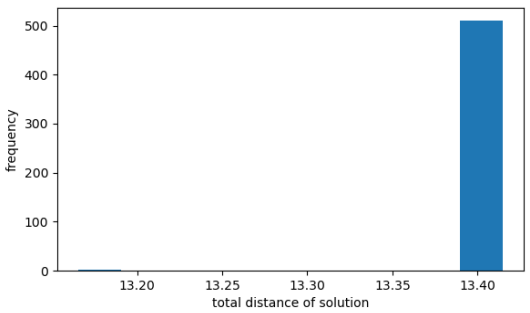
generation is: 30



problem: TSP
n_cities: 50
c_prob: 1
m_prob: 1

c_elite: True
m_elite: True
tournament_size: 256
population_size: 512

generation is: 30

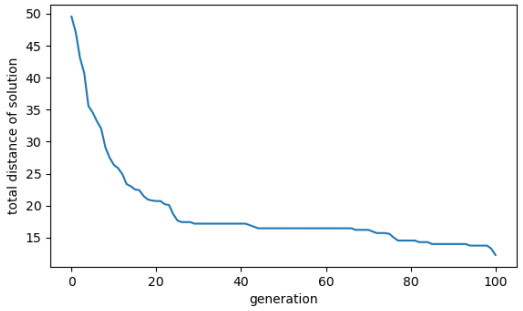


problem: TSP
n_cities: 50
c_prob: 1
m_prob: 1

c_elite: True
m_elite: True
tournament_size: 512
population_size: 512

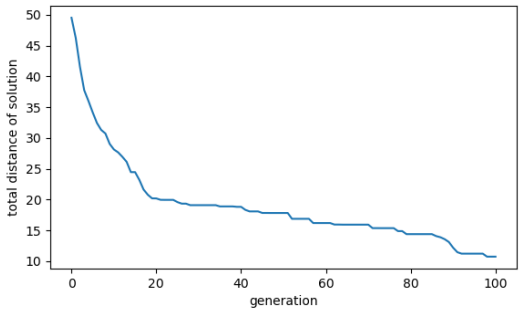
generation is: 30

Figure 12: different temperatures with family and mutation elitism



problem = TSP
n_cities: 50
c_prob: 0.8
m_prob: 0.1

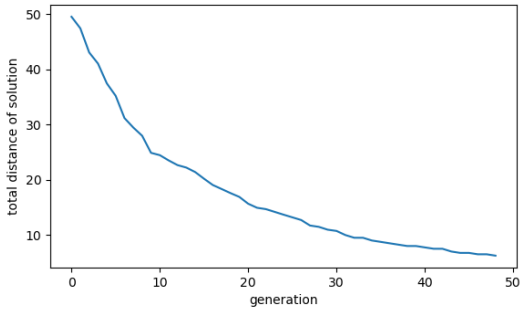
c_elite: False
m_elite: False
tournament_size: 128
population_size: 512



problem = TSP
n_cities: 50
c_prob: 0.8
m_prob: 0.2

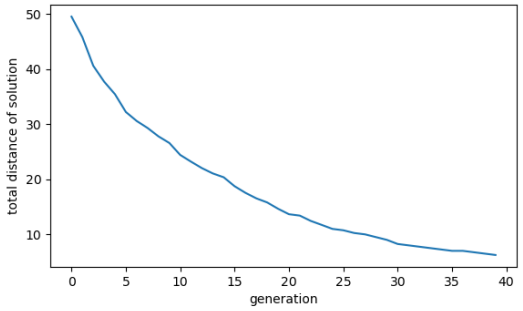
c_elite: False
m_elite: False
tournament_size: 128
population_size: 512

Figure 13: fitness progressions no elitism



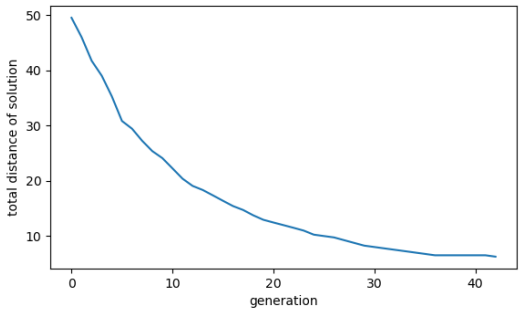
problem = TSP
n_cities: 50
c_prob: 1
m_prob: 1

c_elite: True
m_elite: True
tournament_size: 64
population_size: 512



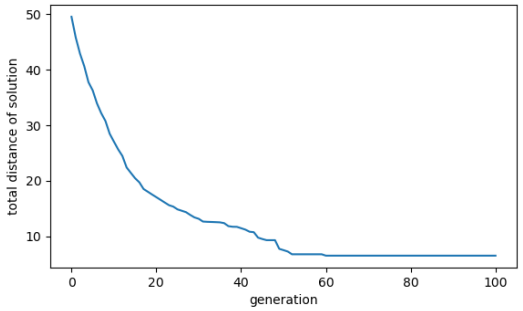
problem = TSP
n_cities: 50
c_prob: 1
m_prob: 1

c_elite: True
m_elite: True
tournament_size: 128
population_size: 512



problem = TSP
n_cities: 50
c_prob: 1
m_prob: 1

c_elite: True
m_elite: True
tournament_size: 256
population_size: 512



problem = TSP
n_cities: 50
c_prob: 1
m_prob: 1

c_elite: True
m_elite: True
tournament_size: 512
population_size: 512

Figure 14: fitness progressions with elitism

where the case, the crossover operator would be rendered ineffective. When the tournament size is n/2, there is very little diversity, as the histograms show in Figure 12. With full mutation, the algorithm has the best chance to find beneficial inversions. Since it will discard any non-beneficial inversion mutation, there is no risk of obtaining a worse solution; this is visible in the non-increasing best distance in each iteration. That the approach works so well on the unit circle instance is likely helped by the fact that there are no locally optimal connections that are not globally optimal. If a city has found its closest neighbours to connect with, the algorithm does not need to break this connection to find something better.

References

Abid Hussain, Yousaf Shad Muhammad, M. Nauman Sajid, Ijaz Hussain, Alaa Mohamd Shoukry, and Showkat Gani. Genetic algorithm for traveling salesman problem with modified cycle crossover operator. 2017:1–7.

W. Pullan. Adapting the genetic algorithm to the travelling salesman problem. In *The 2003 Congress on Evolutionary Computation, 2003. CEC '03.*, volume 2, pages 1029–1035. IEEE.