A Leveled Mentorship Approach to Genetic Algorithms for the Travelling Salesman Problem

Akil Hosang
akilhosang@gmail.com
Florida Institute of Technology
Melbourne, Florida, USA

, ,

1 INTRODUCTION

Artificial Intelligence is predicated on the idea of creating something we consider intelligent. One good idea to create an intelligent model is to analyze a system or process we already consider to be so. One place we see this is in neural networks that take inspiration from the neural pathways of the brain. Another instance of this is in the process of evolution which has led to the concept of genetic algorithms.

1.1 Genetic Algorithms

In this algorithm, hypotheses are typically categorized by bit strings to symbolize features and expressions. These hypotheses go through stages of selection, crossover, and mutation akin to biological evolution which leads to a new generation that repeats the process until a goal is reached. Selection can be thought of similarly to natural selection where agents that perform better are more likely to survive whereas weaker, less fit agents have a small chance to survive. The selection process typically involves using the fitness of each agent or hypothesis to decide how likely it is to move on to the next generation. Crossover in biological terms is similar to procreation between two "parents" which involves the crossing over of DNA. In genetic algorithms, this involves taking two hypotheses, crossing over or swapping sections of their bit strings creating two children that would be then added to the population for the next generation. The simplest method to choose parents is a random selection of pairs from the population but making a more informed selection of pairs for crossover could lead to convergence faster.

1.2 Traveling Salesman Problem (TSP)

The Traveling Salesman Problem proposes the question, given a list of cities, their positions, and the distance between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the city you started at? This problem is an NP-hard problem which is an important classification in computer science. Genetic algorithms have been proposed to solve this problem by framing each individual or chromosome as a list of cities where each gene represents a city and the order of the genes or cities in the list represents the order in which the traveling salesman visits the city. In this implementation of this algorithm, a minor alteration is made where the starting city is always given by the first city in the dataset and each individual always starts and ends with the origin city.

In this paper, I am implementing a new method of selection as well as an additional method of parent selection for the crossover operator to improve convergence speed while keeping diversity. This method will then be compared against the "book method" referring to a standard approach to genetic algorithms and the "tournament method" which uses a modified version of the book method.

2 RELATED WORK

Multiple methods have been considered and analyzed as viable selection methods. In this work, three methods were considered and two were used to compare against the experimental method.

2.1 Roulette Wheel Selection

What we will consider as the generic version of the algorithm, referred to as the "book method" later further in this experiment, covered by Tom Mitchel is known as the Roulette Wheel Selection (RWS). In this implementation, $(1-r)p$ individuals are added to P_s where the probability of each individual's selection is a ratio of its fitness to the total fitness of the population given by $Pr(h_i) = \frac{Fitness(h_i)}{\sum_{j=1}^{p} Fitness(h_j)}$ where p is the size of the population to be included and r is the fraction of the population to be replaced by crossover. The crossover method then probabilistically selects $\frac{r*p}{2}$ pairs of hypotheses to add to P_s then finally the mutation operation is applied by the mutation rate m. One notable difference between this method and other mentioned methods is the individuals from the selection phase are put in the next generation and the crossover method chooses parents from the entire population rather than the individuals from the group of individuals from the selection method. The book method selection phase can be seen as the "greediest" of the mentioned methods as it is biased to pick the highest fitness hypotheses and so it can more often fall into a local minimum.[3]

2.2 Tournament Selection

The Tournament Selection algorithm involves randomly pairing hypotheses and comparing their fitness values where the "winner" is put into a winner's group and the "loser" is eliminated. Similarly to the book method, $(1-r)p$ individuals are added to P_s by $argmax_h[Fitness(h_A), Fitness(h_B)]$. Therefore, the fitter individual moves on to the next 'round' while the other is disqualified. This process is repeated until the number of winners is equal to the desired number of parents. This method, referred to as the tourn method in this experiment, may seem greedy at first as it always tries to choose the best hypothesis but given the hypotheses are chosen at random there is the scenario where the two worst hypotheses are chosen to be compared against each other which adds diversity the population but still does not guarantee it.[4]

2.3 Linear Rank Selection

Finally, Linear Rank Selection, which is a variant of RWS, is based on the rank of individuals rather than their fitness so the individual's probability of being chosen is increased given a higher rank. Based on an individual's rank, each individual i has the probability of being selected by $Pr(i) = \frac{rank(i)}{n*(n-1)}$.[1]

3 APPROACH

My proposed approach can be described as a Leveled Mentor Selection which uses the RWS.

3.1 Fitness Function

The fitness of each individual is given by the total cost for each path taken where a path between two cities is the Euclidian distance between the two cities and so the fitness of an individual is given by $Fitness(h)$. It is important to note that lower scores are preferable and so when applying the Roulette Wheel selection in any of the methods, the fitness is calculated by $\frac{1}{Fitness(h)}$

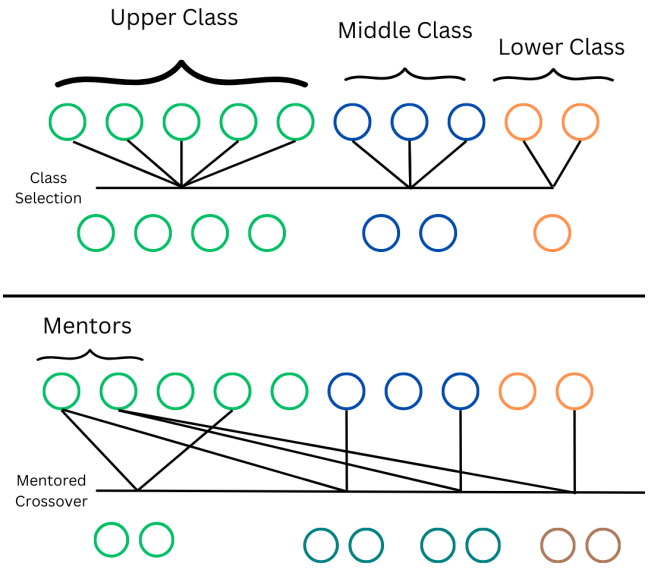

Figure 1: A diagram showing the class selection method and mentored crossover method. The long horizontal lines are only to point to the name of the operation/process being done

3.2 Leveled Selection

The initial population is evaluated and separated into levels we can call the upper, middle, and lower levels. The sizing of each level is proportionally set where we prioritize giving larger size to the higher levels so an example proportional split for upper, middle, and lower class is 0.6, 0.3, 0.1 respectively. After this split is made, the RWS is applied to each level until we have the required number of individuals for each level. In other words, given the 3 splits, we select a group of individuals only from the upper class, then select individuals only from the middle, and the same for the lower. After each of these operations are complete the resulting groups are added to P_s which represent the group of parents for crossover. In other words, given the selection proportion $selprop$, each level has $(P * selprop) * levelprop_v$ individuals where $v \in upper, middle, lower$.

Algorithm 1 Leveled Mentor Approach

```
Initialize population P by randomly generating hypotheses
Sort population P by Fitness(h)

while currGeneration < numGenerations do

    Leveled Selection:
    Create 3 lists one for each level
    Upper Level List: Top x individuals where x is the upper level size
    Middle Level List: Individuals between x and y where y is (x + middle level size)
    Lower Level List: Bottom z individuals where z is the lower level size
    for Each level do
        for 1 to level selection size do
            Probabilistically select a value from levelList and add it to P_s
        end for
    end for

    Regular Crossover:
    Probabilistically select (crossProp*popSize)/2 pairs of individuals from P
    For each pair, apply crossover operation to produce pair of offspring and add offspring to P_s

    Mentored Crossover:
    Select top m individuals based on Fitness(h) to be in mentorsList
    while mentoredOffspring < mentorSize do
        parent1: next value from mentorsList (if at end of list, start list from beginning)
        parent2: random individual from P that is not in mentorsList
        Apply crossover operator to parent1 and parent2 and add offspring to mentoredOffspring
    end while
    Add mentoredOffspring to P_s
end while
Return highest fitness individual of P_s
```

3.3 Regular Crossover

For crossover, crossprop * P2 pairs of individuals are selected from P_s using RWS to probabilistically select the parents and for each parent pair we produce a pair of offspring by applying the crossover

operator that will be discussed further. The resulting offspring are then added to P_s.

3.4 Mentored Crossover

A final group of individuals is created by a mentorship approach where the top m individuals from P are used as mentors. These mentors are cycled through using one individual as $parent1$ and then randomly choosing another parent from P as $parent2$. Given both parents, the crossover operator is applied and the result is a pair of offspring. The next parent1 is chosen by moving to the next mentor in the list of chosen mentors and if the last $parent1$ was the last in the mentor list, it is cycled back to the start of the list. Then $parent2$ is chosen randomly again and this is done the required amount of offspring is met and the list of "mentored individuals" are added to P_s.

3.5 Alternative Operator Implementation

Typical implementations of genetic algorithms have individuals in the form of binary strings and so the usual way of implementing crossover and mutation involves swapping segments of the string and flipping the bit string from 0 to 1 or vice versa but given the TSP, individuals are encoded by their city number rather than bit strings and so this poses issues for crossover and mutation.

3.5.1 Partially Mapped Crossover. If the crossover operator is applied by swapping segments of two individuals, there may exist duplicate cities in a path that renders the individual invalid. To avoid this, Partially Mapped Crossover (PMX) is used which involves two parents s and t. PMX selects a start and end point and the genes/cities between this start and end point will be swapped. Starting with a copy of s and t the genes between the start and end point are swapped. The subpaths between the start and end points are saved as the mapping sections. After the swap between the copies of s and t, we look at every other position from start to end of ss beside the mapped section. The city from the parent s is placed as the value at the same position in the copy of s. If that city already exists in the mapped section in the copy, we find the index i of that city in the mapped section of s and replace it with the city at that same index i in the mapped section of t. We then check again if that new city exists in the mapped section of s and this process is repeated until a city is found not to exist in the mapped section of s then the same is done for the copy of string t.[2] [5]

3.5.2 Insertion Mutation. Given the typical binary string that genetic algorithms use, to mutate an individual, a random bit is flipped but given that in this problem an individual is made up of cities, these cannot be flipped. Instead, Insertion Mutation is used where a city is chosen at random, removed from its position, and then inserted into a random position beside the one it was just taken from.[2]

This method guarantees some measure of diversity as there will be a representative of each division of fitness but keeps the poorly performing individual size small in our population.

4 EMPERICAL EVALUATION

4.1 Evaluation Criteria

In this project three methods were employed, the book method, tournament method, and leveled mentor method. To evaluate this project's approach, the main metrics observed were the best fitness for each generation, average fitness across the population for each generation, and a measure of diversity across the population in each generation. The diversity score is calculated as the average hamming distance between the best individual and all other individuals, given by H_{avg}, then divided by the size of the population-1, therefore, $\frac{H_{avg}}{P-1}$.

4.2 Experimental Data and Procedures

This experiment was conducted using 3 TSP problem sets from the VLSI datasets given by the University of Bonn. These problems ranged in size from 131 cities up to 744,710 cities. In this experiment, the three datasets used were xqf131, xqg237, and pma343 each containing 131, 237, and 343 respectively. The cost of travel between cities is given by the Euclidian distance rounded to the nearest whole number. Each of these datasets also contained the optimal tour found and so each method compared its solution to the optimal tour given.

For each method, the genetic algorithm started with a random population of 1024 individuals. Then the size of the population was set to 300 individuals, with a crossover proportion of 0.7, a mutation rate of 0.2, and each algorithm was run for 500 generations. An important point to note is for experimental consistency, each method used the same initial configuration of 1024 individuals since that initial population was randomly generated. Without doing this, there existed the risk of biased results due to some methods having a better initial configuration. Specifically for the leveled mentor method, the size of the regular crossover output and mentored individuals was half of the crossover proportion, and the level split for each experiment was set to 0.6, 0.3, and 0.1 for upper, middle, and lower class respectively.

4.3 Results

From Figure 2, it is significantly noticeable the difference in the clarity of paths. It is intuitive that the fewer paths cross each other the better as this means the tour is taking a sub-optimal route passing a close city and opting to visit a further city typically.

One major goal of this method was to maintain diversity but from the diversity score algorithm, it shows that the mentored approach consistently produces lower scores which is not preferable as the higher the score the more the fittest individual varies from the other individuals. Though, this could be a consequence of the algorithm used to score the diversity given that as the algorithm defines better individuals, there is a lower ceiling for one individual to differ from another.

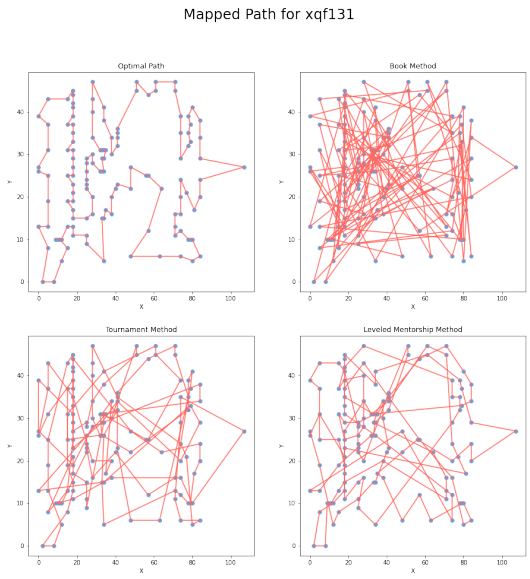

Figure 2: A representation of the paths each method develops after learning on the xqf131 dataset compared to the optimal path

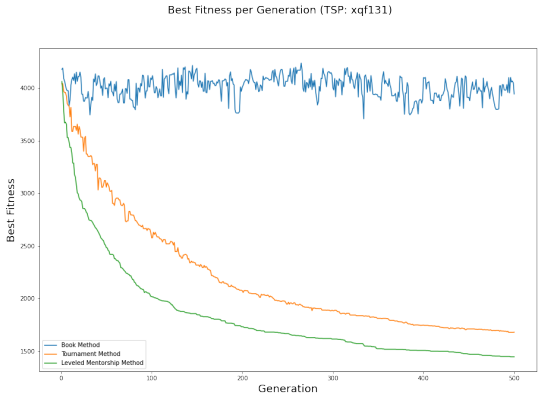

Figure 3: The fitness of the best individual of each generation from each method

Table 1: Evaluation Metrics of 3 Methods over 3 Iterations for xqf131 dataset
BF: Best Fitness, AF: Average Fitness, DS: Diversity Score

	Book			Tourn			Mentor		
	BF	AS	DF	BS	AS	DS	BS	AS	DS
1	3955	4515	284	1620	1658	202	1153	1320	79
2	4094	4632	283	1519	1563	74	1365	1488	66
3	3944	4597	281	1679	1724	180	1445	1610	113

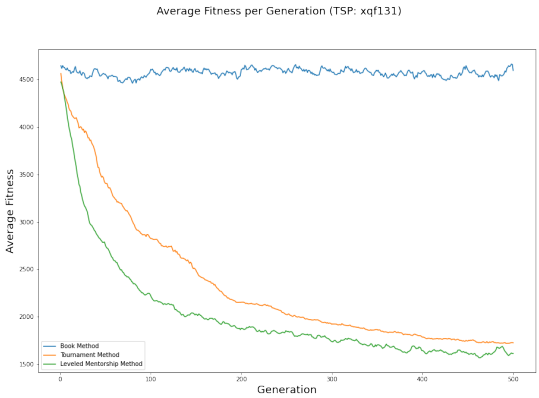

Figure 4: The average fitness of all individuals generated by each method on each generation

Table 2: Evaluation Metrics of 3 Methods over 3 Iterations for xqg237 dataset
BF: Best Fitness, AF: Average Fitness, DS: Diversity Score

	Book			Tourn			Mentor		
	BF	AS	DF	BS	AS	DS	BS	AS	DS
1	3955	4515	284	1620	1658	202	1153	1320	79
2	4094	4632	283	1519	1563	74	1365	1488	66
3	3944	4597	281	1679	1724	180	1445	1610	113

5 CONCLUSION

In this paper, 3 methods for solving the TSP were offered where each was evaluated on their fitness over generations quickness to converge, and diversity within the population. After evaluating these metrics over the given graphs, it was shown that the proposed method of a leveled mentorship approach performs slightly better than the tournament selection method over multiple iterations which shows promising signs for an alternate way to look at selecting parents in genetic algorithms.

REFERENCES

[1] Khalid Jebari and Mohammed Madiafi. 2013. Selection methods for genetic algorithms. *International Journal of Emerging Sciences* 3, 4 (2013), 333–344.

[2] Pedro Larranaga, Cindy M. H. Kuijpers, Roberto H. Murga, Inaki Inza, and Sejla Dizdarevic. 1999. Genetic algorithms for the travelling salesman problem: A review of representations and operators. *Artificial intelligence review* 13 (1999), 129–170.

[3] Tom Michael Mitchell et al. 1997. *Machine learning*. Vol. 1. McGraw-hill New York. 250–252 pages.

[4] Noraini Mohd Razali, John Geraghty, et al. 2011. Genetic algorithm performance with different selection strategies in solving TSP. In *Proceedings of the world congress on engineering*, Vol. 2. International Association of Engineers Hong Kong, China, 1–6.

[5] Göktürk Üçoluk. 2002. Genetic algorithm solution of the TSP avoiding special crossover and mutation. *Intelligent Automation & Soft Computing* 8, 3 (2002), 265–272.