Evolutionary Algorithm for the Traveling Salesman Problem

Basil Jancso-Szabo

I. Introduction

The traveling salesman problem (TSP) is a widely studied NP-hard problem in computer science. The goal of the problem is to find the shortest path that visits all cities out of a set and returns to the starting city. Some versions of this problem extend this to include a cost of traveling between cities or other constraints, but these will not be discussed. Given a set of cities $N = \{n_1, n_2, \ldots, n_{|N|}\}$ where $n_i = (x_i, y_i)$ is the location of city i, we can then define any path as a sequence of cities that includes every city in N once. Then let a path be $E = (n_1^E, n_2^E, ..., n_{|N|}^E)$, where the path implicitly returns to the first city after the last city, and have $\mathbf{E}$ be the set of all possible such paths for a set of cities N.

For this problem we will define the length of an edge (n_j, n_k) as $l_{(n_j,n_k)} = \sqrt{(x_j - x_k)^2 + (y_j - y_k)^2}$, the euclidean distance between the start and end points. We can then define the length of a path L_E as

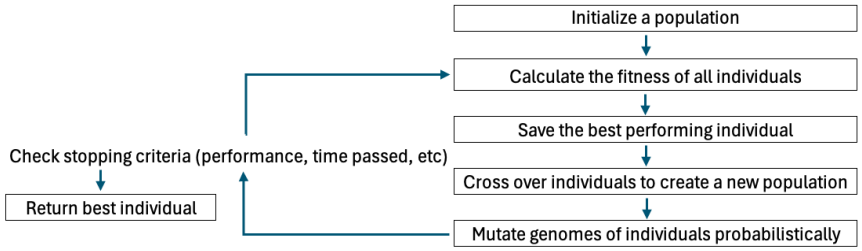
$$l(E) = l(n_1^E, n_{|N|}^E) + \sum_{i=1}^{|N|-1} l(n_i^E, n_{i+1}^E) \tag{1}$$

The goal of the TSP is to find E_{min} such that $l(E_{min}) \leq l(E)\ \forall\ E \in \mathbf{E}$, a set of edges that has the shortest possible length.

We will discuss the ch130 dataset, a TSP with 130 cities [1]. There are several approaches for finding exact solutions to this problem. A naive approach might be to sample every possible path and determine the shortest one, however, this is infeasible for a dataset of this size. Starting at any city (arbitrary since this path is a loop), considering that loops are equivalent in both directions, this results in $\frac{(130-1)!}{2} \approx 2.49\text{e}{+217}$ possible solutions to explore. There are other, more efficient approaches to this problem, such as dynamic programming, that still find the optimal solution to this problem [2]. There are also heuristic algorithms, such as ant colony optimization [3], particle swarm optimization [4], and evolutionary programming [5], that only find an approximate optimal solution, but are much faster.

This report will focus on genetic algorithms to solve the TSP. Genetic algorithms are inspired by the evolutionary process, where fit individuals are more likely to reproduce than the less fit individuals. In these algorithms, we represent possible solutions as genomes of individuals in a population. When two individuals reproduce, we combine ("crossover") their genomes to create a new individual, and possibly mutate their genome slightly to introduce diversity. A simplified overview of the flow of genetic algorithms is shown in Figure 1.

Fig. 1: Simplified overview of the evolutionary process of genetic algorithms.



By repeating this evolutionary process, we gradually improve the fitness of our populations, finding a better solutions to our problem. We will explore a specific formulation of a genetic algorithm for the TSP and compare different hyperparameters and selection criteria for individuals.

II. Methods

To formulate a genetic algorithm for the TSP, we define each individual as a path $E \in \mathbf{E}$, represented as $E = (n_1, n_2, \ldots, n_{|N|})$, a sequence of cities. Implicit in this is the return path from $n_{|N|}$ to n_1. We denote the population as $P = \{E_1, E_2, \ldots, E_s\}$, where s is the size of the population

We can now create a generic version of a genetic algorithm for the TSP as Algorithm 1. This does not necessarily generalize to all genetic algorithms for the TSP, as there is a wide range of variations on genetic algorithms [6]. Within this generic algorithm, we define termination criteria (II-A), a fitness operation (II-B), selection criteria (II-E), a crossover operation (II-C), and a mutation operation (II-D).

Algorithm 1 Generic genetic algorithm for the TSP

Input: Mutation probability p, population size s
Output: Best solution P_{best}

```
Initialize population P randomly with size s
while termination criteria not met do
    Create an empty population P_new
    Calculate fitness of individuals in P and record the best solution P_best
    for i = 1, 2, ..., s do
        Select two individuals E_j and E_k from P
        Produce offspring E' by crossing over E_j and E_k
        Mutate E' with probability p
        Add E' to P_new
    end for
    P ← P_new
end while
return P_best
```

A. Termination Criteria

We terminate the algorithm when either a best path reaches a distance of less than 6500 or when more than 10000 iterations have passed. The threshold of 6500 is used as the optimal solution for the ch130 solution is at length 6110 [1].

B. Fitness Operator

We define the fitness of individual of E_i as

$$f_{E_i} = \left(\frac{l(E_i) - l_{max}}{l_{max} - l_{min}} \right)^2 \tag{2}$$

where $l_{min} = \min(l(E_1), l(E_2), \ldots, l(E_s))$ and $l_{max} = \max(l(E_1), l(E_2), \ldots, l(E_s))$. This normalizes the fitness such that the individual with the shortest path has a score of 1, while the individual with the longest path has a fitness of 0. The squaring was added to further favor the top-performing individuals.

C. Crossover

We define a single point crossover, with compensatory mutations to maintain valid individuals. In this process, given individuals $E_i = (n_1^i, n_2^i, ..., n_{|N|}^i)$ and $E_j = (n_1^j, n_2^j, ..., n_{|N|}^j)$, we first randomly select start and end points x and y in the path of E_i , with $1 < x < y < |N|$. We then create

$$E' = (n_1^i, \ldots, n_{x-1}^i, n_x^j, n_{x+1}^j, ..., n_{y-1}^j, n_y^j, n_{y+1}^i, \ldots, n_{|N|}^i) \tag{3}$$

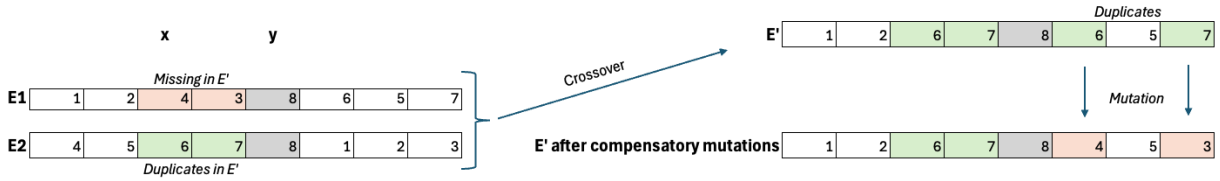


Fig. 2: Simple example of a crossover with compensatory mutations. In this example each integer corresponds to a different city, and E_1 and E_2 are two different individuals. The crossover occurs between indexes x and y, and the mutation ensures the new individual is a valid path that visits every city exactly once.

However, this may result in some cities not being visited or being visited twice. We then add in some compensatory mutations to ensure that E' is a valid path. We first find all cities in E_i that are not in E_j in the range $[x, y]$: the cities missing in E'. We then find all cities in E_j that are not in E_i in the range $[x, y]$: the duplicate cities in E'. Hence we replace all duplicate cities in E' with the missing cities in E' to create a full path. This is shown graphically in Figure 2.

D. Mutation Operator

We mutate an individual $E' = (n_1, n_2, ..., n_{|N|})$ by randomly picking a point $1 < x < |N|$, and simply make $E'' = (n_x, n_{x+1}, ..., n_{|N|}, n_1, n_2, \ldots, n_{x-1})$. This changes the ordering of cities at a single point.

E. Selecting Individuals

We try different methods for selecting individuals, discussed below.

1) ***Random****:* The individuals are selected randomly, with no effect from fitness.

2) ***Random with Top Performer****:* The individuals are selected randomly, but the top performer from each generation is replicated in the next generation without modification.

3) ***Fitness Based****:* The fitness of each individual is converted to a probability of being selected to reproduce by dividing their fitness by the sum of the fitness of the entire population.

4) ***Fitness Based with Top Performer****:* The probability of reproduction is calculated as before, and the top performer from each generation is replicated in the subsequent generation without modification.

5) ***Fitness Based with Varying Temperature and Top Performer****:* Inspired by [7], the randomness with which decisions were made varied over time as a temperature T. In this process, the probability of mutation is T. Further, before the fitness is converted to a probability of being selected, a random amount is added to all fitness scores. This random amount added is different for each individual, with the amount $r \sim |\mathcal{N}(0, 0.5 * (T - 0.07))|$. This means that when the temperature is high there is more randomness in the selection of individuals to reproduce.

The temperature T follows the equation

$$T(\text{iteration}) = 1 - \min\left(-0.1 \cdot e^{-\frac{\text{iteration}}{5000}} \cdot \sin\left(\frac{\text{iteration} + 800}{200}\right) + 0.9,\ 0.93\right), \tag{4}$$

with a maximum of approximately 0.246 and a minimum of 0.07. This was determined to converge to a temperature of 0.9 while oscillating between low and high probability of mutation. The top performer from each generation is also replicated in the subsequent generation without modification.

III. Results

The different selection methods for individuals are compared, and then for the base algorithm, using fitness and the top performer as in Section II-E4, we assess the algorithm for different population sizes and mutation probabilities. Full results and code are available at https://github.com/BasilJ-S/EvolutionaryTSP.

A. Selection Criteria

The trial was run with a maximum run time of 10000 iterations with 100 individuals. The probability of mutation was tuned to each scenario, with results and the probability used recorded in Table I. This shows that the Fitness Based with Varying Temperature selection method had the best performance, and the loss plot and best solution found by the algorithm are in Figure 3. Loss plots and the best solutions found by all except the completely random method are available in Appendix A.

TABLE I: Best performance achieved using different selection methods for reproduction.

Selection Method	**Probability of Mutation**	**Best Performance**
Random	0.1	41845.249
Random with Top Performer	0.1	25779.30
Fitness Based	0.05	13177.103
Fitness Based with Top Performer	0.1	7259.996
Fitness Based with Varying Temperature	varying	**7149.721**

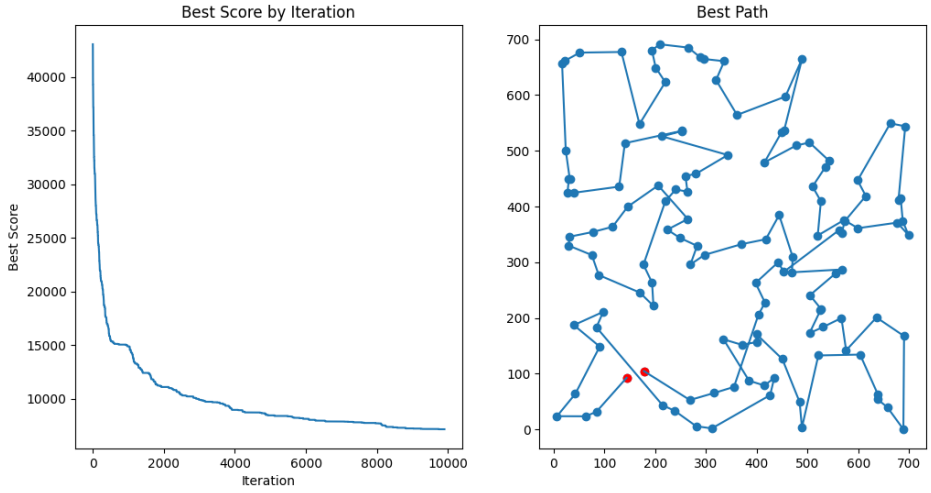


Fig. 3: Loss plot and best solution found using fitness-based selection with varying temperature and keeping the top performer between generations. The best solution found is not optimal, but creates a generally acceptable path. Red points represent the first and last city, and the score is the path length of an individual.

B. Hyperparameters

Hyperparameters were tested for the Fitness Based with Top Performer selection criteria from Section II-E3. A grid search was done with mutation probabilities $\{0.01, 0.05, 0.1, 0.2, 0.3, 0.5, 0.7, 0.9, 1\}$ and population sizes $\{10, 20, 40, 50, 100, 300, 500\}$. All combinations were run for 8000 iterations, stopping only if all individuals received the same path length, indicating that there is no diversity left in the population. These results show that the best performance was achieved with a population size of 100 and a mutation probability of 0.1. The performance of the different models is shown in Figure 4.

IV. Discussion

The model with randomness varying with temperature outperformed all other models, demonstrating the effectiveness of varying temperatures. Qualitatively, this appeared to allow the population to explore more later in the algorithm once the initial benefits of the existing population had been fully utilized.

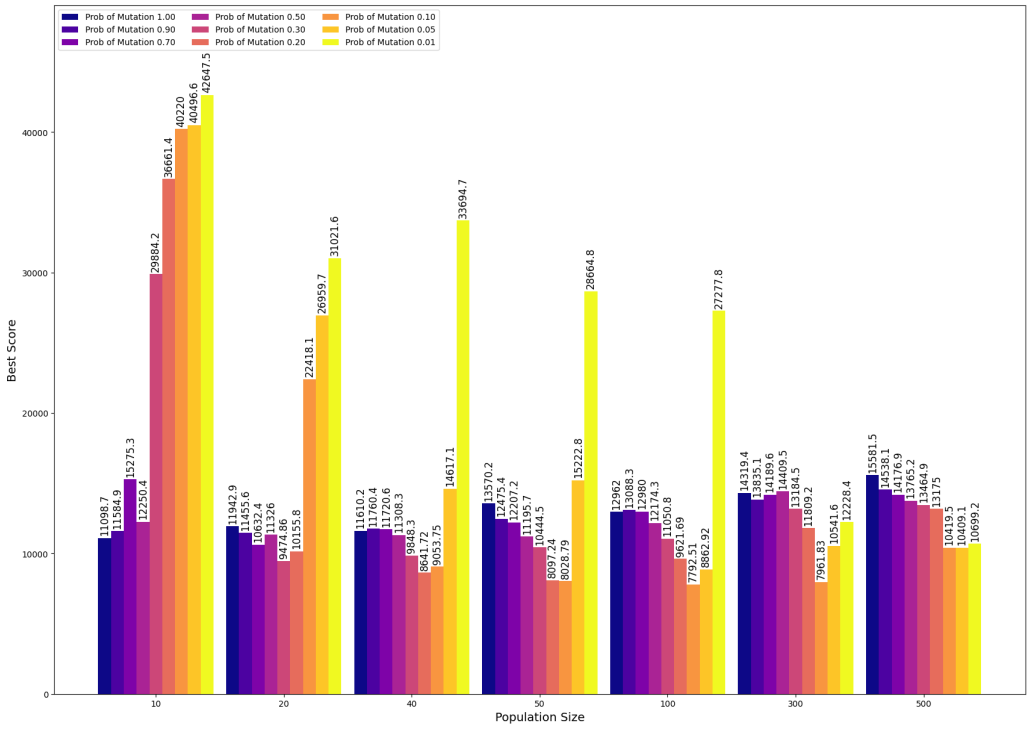


Fig. 4: Comparison between population size, probability of mutation, and best path length after 8000 iterations. This shows that as the population becomes larger, there is a higher acceptable probability of mutation. Also evident is the poor performance of low probabilities of mutation, explored further in Appendix B.

Further, it is interesting to note the substantial increase in performance when the best performing individual is held over between generations. This intuitively makes sense, as it allows the algorithm to better exploit minor improvements that may be lost otherwise. This also demonstrates the potential benefit of adding in other methods to recognize and prioritize beneficial mutations and crossovers.

However, it is evident through the analysis of the effect of hyperparameters that there must be a balance between the chance of mutation and the population size. Explored further in Figure 8, we can see that when mutations were limited in small populations, the population would often lose all diversity, resulting in early convergence to a local minimum. It is possible that this also occurs for larger populations, but is simply delayed by the larger size.

It is interesting that the algorithms still achieved good performance when offspring are guaranteed to mutate. This could be explained by the replication of the top performing individual, which may compensate for this and benefit from the added exploration. This could also indicate that the mutation used is insufficient for introducing substantial changes in an individual.

V. Conclusion and Future Work

Overall, we demonstrate the effectiveness and efficiency of genetic algorithms for tasks like the TSP. All fitness-based variants of the evolutionary algorithm proposed were able to quickly achieve good performance, with the top performing temperature based algorithm achieving a near optimal solution and a mostly sensible path.

Future work on this can explore ways to maintain genetic diversity in the algorithm. This could be achieved by using different crossovers, fitness operators, or mutation operators. Other techniques could also be explored, such as introducing new individuals later in the algorithm. Another potentially interesting method would involve training multiple models partially before and combining their populations. This may extend the initial burst in performance gain seen.

Future work could also integrate more recent developments in genetic algorithms, as this work was completed explicitly without consulting the existing literature on genetic algorithms for the TSP.

REFERENCES

[1] *ch130*. URL: http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/tsp/ (visited on 12/11/2024).

[2] Richard Bellman. "Dynamic Programming Treatment of the Travelling Salesman Problem". In: *Journal of the ACM* 9.1 (Jan. 1962), pp. 61–63. ISSN: 0004-5411, 1557-735X. DOI: 10.1145/321105.321111. URL: https://dl.acm.org/doi/10.1145/321105.321111 (visited on 12/11/2024).

[3] Thomas ST Utzle and Marco Dorigo. "ACO algorithms for the traveling salesman problem." In: *Evolutionary algorithms in engineering and computer science 4* (1999), pp. 163–183.

[4] X. H. Shi et al. "Particle swarm optimization-based algorithms for TSP and generalized TSP". In: *Information Processing Letters* 103.5 (Aug. 31, 2007), pp. 169–176. ISSN: 0020-0190. DOI: 10.1016/j.ipl.2007.03.010. URL: https://www.sciencedirect.com/science/article/pii/S0020019007000804 (visited on 12/11/2024).

[5] David B. Fogel. "Applying Evolutionary Programming to Selected Traveling Salesman Problems". In: *Cybernetics and Systems* (Jan. 1, 1993). Publisher: Taylor & Francis Group. DOI: 10.1080/01969729308961697. URL: https://www.tandfonline.com/doi/abs/10.1080/01969729308961697 (visited on 12/11/2024).

[6] Thomas Bäck and Hans-Paul Schwefel. "An Overview of Evolutionary Algorithms for Parameter Optimization". In: *Evolutionary Computation* 1.1 (Mar. 1993), pp. 1–23. ISSN: 1063-6560, 1530-9304. DOI: 10.1162/evco.1993.1.1.1. URL: https://direct.mit.edu/evco/article/1/1/1-23/1092 (visited on 12/11/2024).

[7] Peter J. M. van Laarhoven and Emile H. L. Aarts. "Simulated annealing". In: *Simulated Annealing: Theory and Applications*. Ed. by Peter J. M. van Laarhoven and Emile H. L. Aarts. Dordrecht: Springer Netherlands, 1987, pp. 7–15. ISBN: 978-94-015-7744-1. DOI: 10.1007/978-94-015-7744-1_2. URL: https://doi.org/10.1007/978-94-015-7744-1_2 (visited on 12/11/2024).

APPENDIX A
LOSS PLOTS

In this appendix we show further loss plots from Section III-A. Figure 5 shows the results from the algorithm picking individuals randomly as in Section II-E2, Figure 6 shows the results from the algorithm using fitness-based selection as in Section II-E3, and Figure 7 shows the algorithm that adds the carries the top performing individual of a generation to the next generation as in Section II-E4.

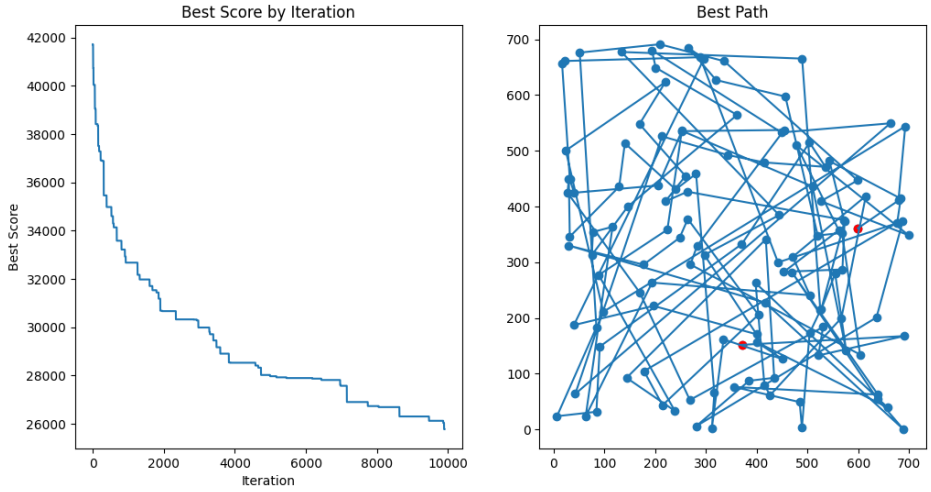


Fig. 5: Loss plot and best solution found using random selection, but saving the best performing individual between generations. Interestingly, this still shows some convergence towards better solutions through this single mechanism, but does not reach a good path. Red points represent the first and last city, and the score is the path length of an individual.

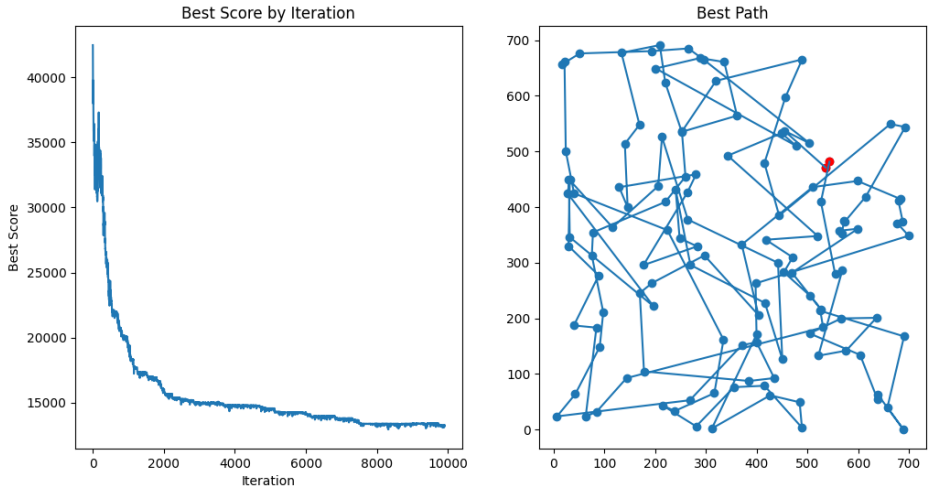


Fig. 6: Loss plot and best solution found using fitness-based selection. This converges substantially better than with random selection, removing many clearly sub-optimal paths. Red points represent the first and last city, and the score is the path length of an individual.

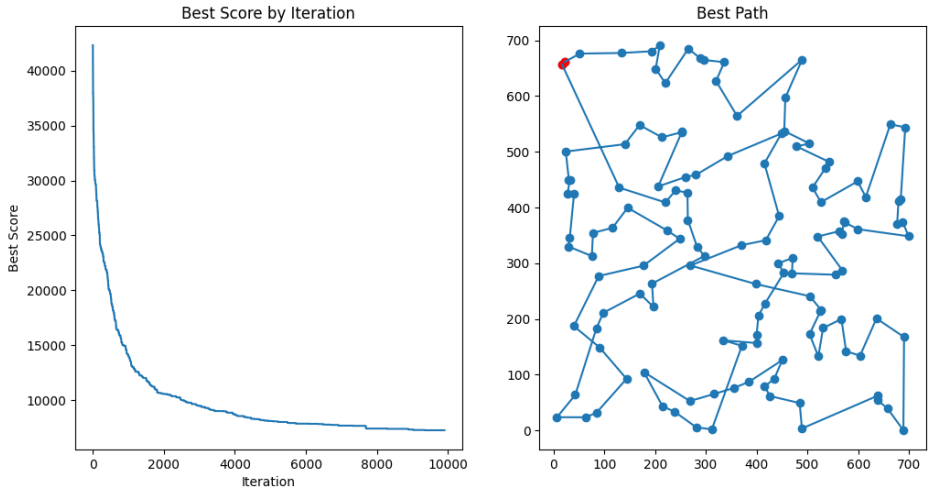


Fig. 7: Loss plot and best solution found using fitness based selection while keeping the top performer between generations. This shows a substantial performance increase over using fitness-based selection alone. Red points represent the first and last city, and the score is the path length of an individual.

APPENDIX B
HYPERPARAMETER ASSESSMENT

In Figure 8 we show how many trials each combination survived before there was no diversity left in the population. This shows how a low probability of mutation significantly increases the probability of losing diversity in a population.

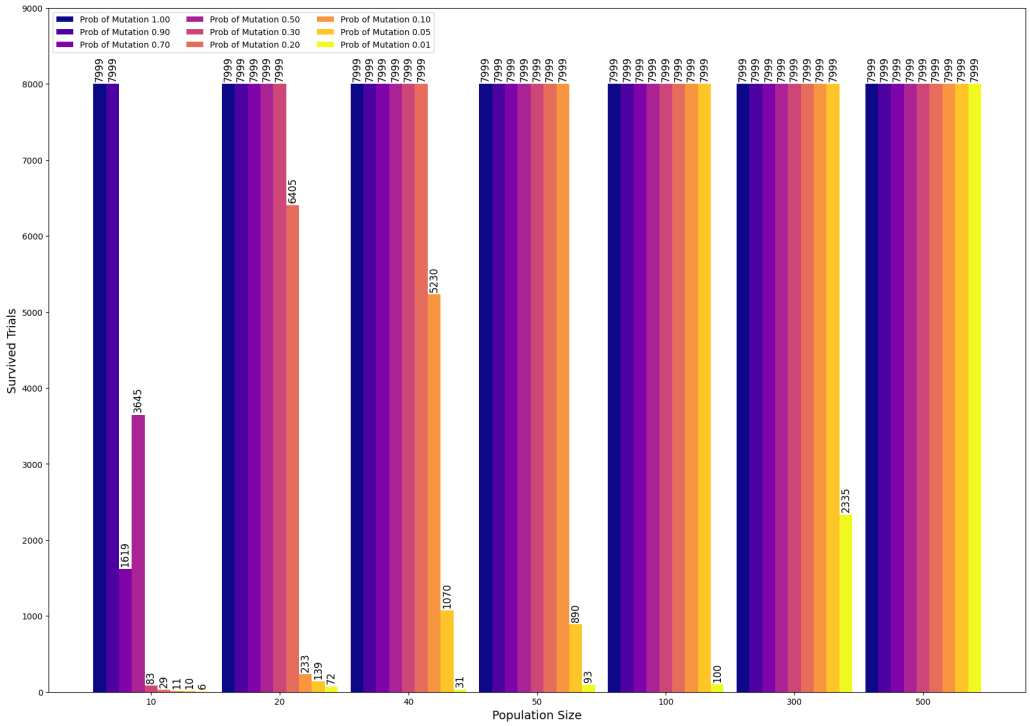


Fig. 8: Comparison between population size, probability of mutation, and number of iterations completed before all diversity is lost in a population; the value in balancing the mutation probability and the population size is evident.