**Optimizing TSP with GA:**

**Submitted by :**

Malka Abramovitch : 314723586

Maor Betser : 315841445

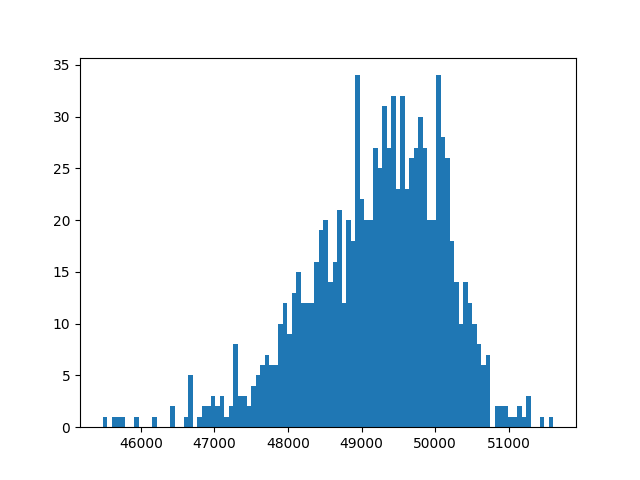
**Innovation in Genetic Algorithm (GA) Design**

We implemented a custom Genetic Algorithm (GA) called MyGA, focusing on introducing improvements to achieve better optimization results. One key innovation we implemented was retaining the best genomes from previous runs. This improvement allows the algorithm to build on previously successful solutions and continuously refine the population towards the optimal result.

**Key Innovation: Persisting the Genome Across Runs**

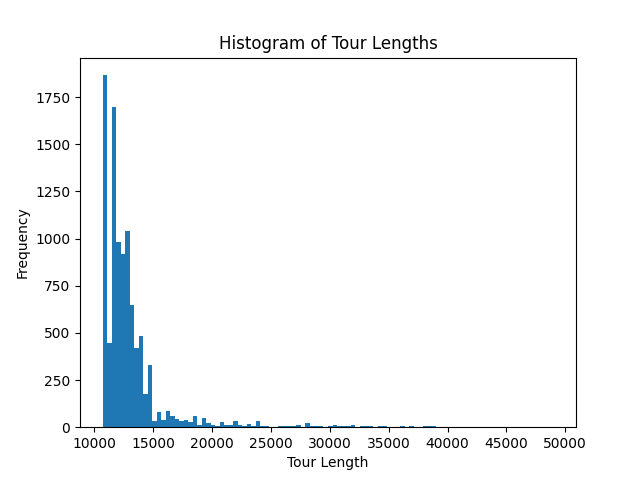
Instead of reinitializing the population for each run, we retained the genome as a class field within the MyGA implementation. This change ensures that the best solutions from previous generations remain part of the search process. By leveraging this persistence, the offspring in subsequent runs start from an improved baseline, effectively accelerating the convergence towards optimal solutions.

**Before Implementing Persistent Genome**



We can see on the X-axis the tour length and on the y-axis the amount of solutions that made it, we see we have a “Normal Distribution” around 49k, also we had to lower the maximum evals 3 orders of magnitude less.

After implementing genome as a field of the class :



We can clearly see an improvement, hovering around 10, this is with the same eval count hovering around 10\*\*4.

The parameters for both runs were with “Tournament selection” as our select method.

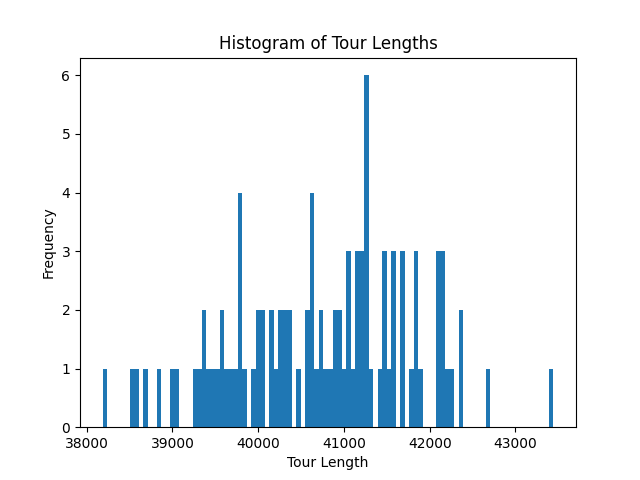
**Why we chose Tournament selection over roulette wheel?**

Our idea is the get consistently the best offspring in order to get a faster convergence,

We chose tournament selection because it guaranteed the best offsprings each time.

(We took the tournament selection from ChatGPT)

We Also tested roulette wheel but came across worse results overall



This figure is with roulette wheel, we can see that the average is around 41k, much worse than what we have achieved.

**Conclusion:**

We did not reach the optimum value, but we got close to it and fast, the best run we had was ±10k which is pretty close, and we got there with a miniscule amount of generations (this idea came to mind because lack of resources ☺ it took our computers a **LONG** time to converge to an answer).

We think our idea helped to mitigate the long run time and helped to converge a “good” solution in a timely manner ☺