**Crossover Options:**

**“crossHalf”:** takes the coordinates of both parents and sorts all the values. Then it separates in half so that the first half is one offspring, and the second half is the other offspring; **“crossSinglePoint”:** this mutation picks one random crossover point in the combination of the parents and then takes the first part of the first parent combined with the second part of the second parent for the first offspring. For the other offspring is the other way around; **“crossCycle”:** ensure that each offspring gets genes from both parents in a cyclic manner. Once a cycle is complete switch to the other parent to fill the remaining positions in the offspring. The second offspring is the same logic but taking the other parent as initial reference; **“crossGeometricSemantic”:** each coordinate of the offspring is a weighted average of the corresponding coordinates of the parents given by random weights that add up to one.

**Selection Options:**

**“Tournament Selection”:** randomly selects two individuals from the population and chooses the one with the highest fitness to perform mutation. **“Roulette Wheel Selection”:** it chooses individuals with a probability directly proportional to fitness, so that better individuals have a higher chance of being picked.

**Mutation Options:**

**“Position with conflict for random”:** for the selected individuals it identifies conflict positions, removing it and adding a random position, for both coordinates. **“Shift Coordinate on Position with Conflict”:** also gets a position that has conflict but only shifts one of the coordinates (randomly). **“Individual for Random”:** it doesn’t perform any of the previous mutations and only ensures the selected individual has the correct number of elements by adding random positions.

**Looking for the best model:**

Because there are many parameters and function options for several steps of the algorithm, we have to search independently in different sections of the possible search space. Now this is not the optimal or most accurate way to obtain the absolute best model but given the computational complexity and time we have for this project, we find this to be a good approximation. Each dot corresponding to each board size is the average time of 30 runs for that specific combination. To make the graphs possible to read we will join the points of different board sizes with lines, but this of course is not an accurate representation of the problem.

First, we look for the best combination of functions for crossover, mutation and selection (using population size of 200). Then we optimize for the size of the initial population and if using elitism is beneficial or not. After this we have the “best model” and will run it more thoroughly to get the final results. We then analyse how the initial positions that are set randomly impact these results.

**Finding the best combination of functions:**

Because there are many different types of functions for “selection”, “mutation” and “crossover”, and some of them take a long time to run. In order to compare them all we will run by sections so that we progressively see which of these combinations gives better results, also just looking at low values of n (size of board). As the best combination becomes more apparent, we will look at higher values for this parameter. The most accurate way would be to just directly compare all possible combinations, but as this is not computationally feasible, we decided to take this approach. In figure (a) we have all the selection and mutation functions but just the first two crossover functions. It is notable that the “tournament selection” selection has the lines above “Roulette Wheel Selection”, meaning it is performing worse. We only use Roulette from now onwards. For mutation, “Individual For Random” is performing the worse for both selection functions, so we will only keep the other two. Now we take this information and add the other crossover functions that weren’t used on the first run. Looking at figure (b), for the same selection and mutation, the crossovers “crossHalf” and “crossSinglePoint” have better performances than “crossCycle” and “crossGeometricSemantic”, so will only keep the first two. Now that we have less combinations we will run for higher board sizes to make a better assessment of what the best combination of functions is. Looking at figure (c) we will take the lowest line as the best combination, assuming that it would maintains bellow the other ones for higher board sizes. So that is “Roulette Wheel Selection” for selection function, “Shift Coordinate on Position with Conflict” for mutation function and “crossHalf” for crossover function.

**Optimizing population size and elitism:**

We follow the same logic as before and run the best combination for different population sizes and elitism option to compare the results. When elitism is “True”, the fraction of the best population kept in order of best fitness is always 10%. Looking at the results (figure (d)), the best model changes depending on board size. Because for low sizes the solving speeds are fast but start to increase significantly for higher sizes, it is preferable to choose a model that is the best one for higher board sizes, as that is what will save the most amount of time in the long run. With that, population size of 100 and elitism being enabled seem to be the better option. Now that we have all the optimized parameters, we can run the “best” model.

**Results and associated fluctuations:**

Running until n=9, we can see that the growth in time is exponential as would be expected (figure (e)). We can now do some analysis on how the initial positions, that are set randomly, affect the time it takes to solve the positions. For that we maintain all the parameters constant and obtain the distribution of the results for each run via boxplot. Using the parameters of the best model, in figure (f) we follow this logic for different board sizes. The horizontal line of 0 indicates the average. There are differences higher than a factor of 2, which is very significant. The problem is seen across different sizes equally for small values. This might indicate that we should have used more than 30 runs for each point for the graphs above to get more consistent results and that the performance of the algorithm is very dependent on a random factor, which is not a great sign.

**Annex:**

Uma imagem com texto, captura de ecrã, file, Paralelo

Descrição gerada automaticamente

Figure (a): First comparison to get the best combination of functions.

Uma imagem com texto, captura de ecrã, file, Paralelo

Descrição gerada automaticamente

Figure (b): Second comparison to get the best combination of functions.

Uma imagem com texto, file, diagrama, captura de ecrã

Descrição gerada automaticamente

Figure (c): Getting the set of functions with best performance for low (n).

Uma imagem com texto, captura de ecrã, diagrama, file

Descrição gerada automaticamente

Figure (d): Different population sizes and elitism option.

Uma imagem com texto, file, diagrama, Gráfico

Descrição gerada automaticamente

Figure (e): Results for the best model.

Uma imagem com diagrama, texto, Esquema, file

Descrição gerada automaticamente

Figure (f): Variation of results on different runs.