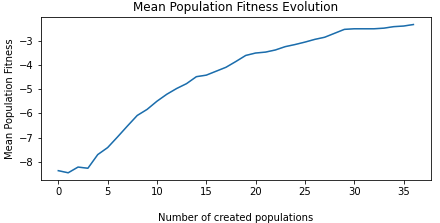
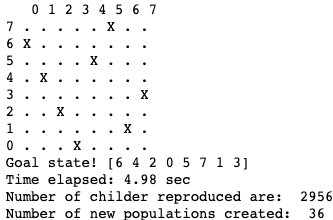
Task 3: Report

Basic GA: `Genetic\_Algorithm\_8` implements (Russell and Norvig, 2009) pseudocode. Firstly, I adopted EightQueensState () class I from FoAI and update the \_\_str\_\_ method to print the chess board. Then created a new class, `Genetic\_Algorithm\_8`. I repurpose the cost method in EightQueensState by having it as **- (cost)** and adopted to be the fitness function where GA is maximizing fitness function and is targeting to reach an optimal solution that has [0] as its fitness state. GA class has an attribute **verbose** that give an insight into the behind scene of the class. I took several design changes along the way, all listed in Table 1.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Parameter** | **Basic GA** | **Enhanced GA** | | | | |
| **Parameter Search Space** | | | **Best Parameter Set** | **Best Parameter Set** |
| **Problem** | 8 Queen | 8 Queen | 12 Queen | | 8 Queen | 12 Queen |
| **Initialization** | Random | | | | | |
| **Population size** | 80 | 80-160 | 120-240 | | 120-240 | 240 |
| **Parent Selection** | Fitness Proportional Selection (FPS) | Fitness Proportional Selection (FPS) *OR* Tournament Selection Algorithm | | | Tournament Selection Algorithm | Tournament Selection Algorithm |
| **Tournament k Selection** | N.A | [2,4] | | | 4 | 2 |
| **Mutation** | Random resetting | Swap *OR* Random resetting | | | Swap | Swap |
| **Mutation probability** | 0.05 | [0-0.8] | | | 0.7 | 0.6 |
| **Crossover** | One-point Crossover | | | | | |
| **Crossover Probability** | 0.5 | [0.7-1] | | | 0.99 | 0.99 |
| **Number of offspring** | 80 | 80-160 | | 120-240 | 80 | 240 |
| **Survival selection** | fitness-based method | | | | | |
| **Termination condition** | time constraint (31 sec) | fitness improvement remains under a threshold value for 10 sec *AND* time constraint (61 sec) | | | | |

Table 1 GA Design Choices



Chart, histogram

Description automatically generatedChart, histogram

Description automatically generatedWhen running basic GA, I reach a goal state in 4.98 sec after 36 rounds of populations. To validate that Basic GA is getting closer to an optimal solution, I used **Mean Population Fitness (MPF)** as an indicator of convergence. MPF is the average fitness of all individuals in a certain population. When MPF is logged and plotted against number of created populations, a sharp increase in MPF is visible on Figure 1. Genetic\_Algorithm\_8 performance was evaluated during multiple runs and plot answers to figure out the algorithm 's capability, see Figure 2 . In 100 runs of the 8 queen puzzle, Basic GA resolves the puzzle in 83 round out of 100. It came up with 83 solutions, 52 of them are unique. The distribution for time elapsed to reach a solution is right-skewed. I note that the majority of solutions took less than 10 sec with around 7 sec median time. Conversely, when Basic GA is run for N-queen problem (n>8), GA reaches the termination condition without finding an optimal solution. Thus, more improvement is necessary to enable GA to resolve N-Queen puzzle.

Figure Results of 100 runs on Basic GA Implementation (8-queen puzzle)

Figure 1 a solution for 8 Queen Puzzle

Figure Results of 100 runs on Enhanced GA Implementation (8-queen puzzle)

# Enhanced GA: I implement the 5 improvement to basic GA, all listed in Table 1: **1)** - Allow the algorithm to handle N queen problems by relaxing the termination condition of time constraint (from 31 to 61 sec) also introducing new condition that monitor if fitness improvement remains under a threshold value for 10 sec. **2)** - Tournament selection algorithm: this is a method for parent selection that select randomly a k individual from population then pick the fittest pair for mutation. I implement the pseudocode in Fig. 5.3. in (Eiben and Smith, 2015). FPS suffers a problem of premature convergence. Outstanding individuals take over the entire population very quickly. This tends to focus the search process, and makes it less likely that the algorithm will thoroughly search the space of possible solutions, where better solutions may exist. **3)** - When fitness values are all very close together, there is almost no selection pressure. Therefore, later in a run, when some convergence has taken place and the worst individuals are gone, it is observed that the mean population fitness only increases very slowly. I introduce a condition that bypass fitness-based method for survivor selection when new population is identical to existing population for 4 consecutive rounds. This would choose randomly from the merged pool of existing population and children. **4)** - Swap mutation allow to maintain permutations intact. Two positions (genes) in the chromosome are selected at random and their allele values swapped**. 5)** - Enhance diversity by removing redundant children. The diversity of a population is a measure of the number of different solutions present. I remove offspring whose genotypes are identical to individual in population. Figure 4 illustrates a valid solution for 12-queen puzzle, the solution was reached in 32 sec and 181 round of generations. I run the enhanced GA 100 times to resolve 8-queen problem, Figure 3 shows that although success rate is similar for enhanced and basic GA (82%, 83% respectively), *average* time elapsed to solution *has reduced significantly* with the enhanced model (4 sec) and distribution mode around 2 sec.

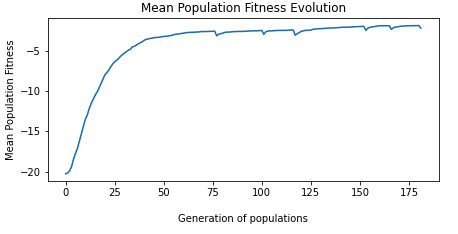
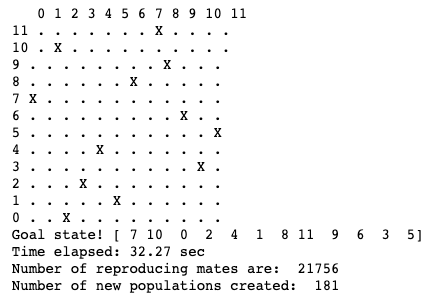


Figure a solution for 12 Queen Puzzle

A known deficiency in my approach is not using parameter control. The set of algorithm parameters is static/ deterministic through the run, although it has been proven that a different value of parameters might be optimal at different stages of the evolutionary process, (Mosayebi and Sodhi, 2020).

The parameters tuning problem depends on the problem to be solved (e.g., 12 queens puzzle), and the utility function that defines how algorithm quality is measured. Because of the stochastic nature of EAs, a good estimation of performance requires multiple runs (10 run) on the same problem with the same parameter values and some statistical aggregation of the measures defined for single runs. Doing this for the four measures gives us the performance metrics commonly used: • **Mean Best Fitness (MBF):** For each run, I log the fitness of the best individual at termination. The MBF is the average of these values over all runs. **• Average number of Evaluations to a Solution (AES)** is an algorithm efficiency indicator. **• Success Rate (SR):** percentage of runs where optimal solution is reached. **• Average Time Elapsed to Solution (ATES):** although time-related indicators depend on the specific hardware, operating system, compiler, network load, and so on, it is another a good indicator for algorithm efficiency. Metrics are *normalized* and summed per parameter combination. I add negative sign to AES and ATES to ensure selecting parameter set that generates optimal solution in the lowest number of evaluations to solution and the least possible time.

The best parameters set, Table 1, are picked up and plugged in a 100 run. Result summary is illustrated in Figure 5. 57% success rate was reported. Solutions are found in less than 60 sec, on average less than 40 sec.

# Chart, histogram Description automatically generatedBibliography:

1. Eiben, A. and Smith, J.E., 2015. *Introduction to Evolutionary Computing* [Online]. 2nd ed. ‎ Springer. Available from: https://link.springer.com/book/10.1007/978-3-662-44874-8 [Accessed 6 November 2022].
2. Mosayebi, M. and Sodhi, M., 2020. Tuning genetic algorithm parameters using design of experiments. *Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion* [Online], GECCO ’20: Genetic and Evolutionary Computation Conference, Cancún Mexico. Cancún Mexico: ACM, pp.1937–1944. Available from: https://doi.org/10.1145/3377929.3398136 [Accessed 6 November 2022].
3. Chart

   Description automatically generatedRussell, S. and Norvig, P., 2021. 4.1.4 Genetic algorithms. *Artificial Intelligence: A Modern Approach, 4th*

Figure 5 Results of 100 runs on Improved GA Implementation (12-queen puzzle)