Graded Assignment 1 - Genetic Algorithms

For this assignment, you will need to provide a Jupyter notebook with a working Python implementation of Genetic Algorithms (GAs) that can solve the 8 Queens problem, an extended and optimised version of this implementation, and a 3-page report explaining and contextualising your work, submitted as a .pdf file.

This assignment will be marked out 100. 30 marks will be awarded for the basic implementation, 30 marks will be awarded for extensions and optimisations to the basic code, and 40 marks will be awarded based on the quality of your report.

To get started, you might want to use the provided notebook with skeleton code from the 8 Queens Notebook in the Foundations of Artificial Intelligence unit, but you are also welcome to write your notebook from scratch.

Task 1 (30%): Solve the 8 Queens problem using a genetic algorithms approach. You may use the pseudocode from Russell and Norvig provided in the course content, or an alternate source, provided it is properly cited.

Full marks for this second will be awarded for any submission which is able to find a solution to the puzzle in 30 seconds or less, from a random population. Your code should clearly demonstrate that a genetic algorithms approach has been used, through print statements, appropriate variable/class/function names, and/or code comments. As an output, you need to show a solution, i.e. an arrangement of n-queens (n=8) on the n-by-n board.

Please note you will not be penalised if it occasionally solves the puzzle above 30 seconds. Partial marks are available for limited functionality. No Python packages that provide off-the-shelf implementation of GAs or substantial aspects of GAs should be used, basic packages such as numpy should be perfectly sufficient for implementation.

Task 2 (30%): Adapt and extend your solution for a wider class of problems and/or optimise your code to improve performance over the basic approach. For the extended/optimised version, code should be written separately from the basic code, so that the improvements are clear. This section of the assignment can be considered an optional extension, and high marks for this section will require work of exceptional quality and might require spending time on this assignment beyond the provided estimates.

There are no explicit guidelines for extensions, and these could be implemented in multiple ways (e.g., n-Queens for n≠8 (e.g. n=4 to n=20), showing all unique solutions on the n-by-n board, magic squares, extensions to non-standard boards/non-standard pieces, etc.). For higher marks, we will be looking for extremely sophisticated and surprising extensions. For reference, you could see various methods discussed on the Wikipedia page of the Eight queens puzzle.

For the optimisations, you could take inspiration from external literature, for example implementing variations in mutation, crossover, fitness functions, elitism, and approaches for hyper-parameter tuning. and could demonstrate effective results in terms of time and space complexity. The marks for this part will be awarded based on how successful and far-reaching these extensions/optimisations are. Please note for optimisations, algorithmic improvements should be prioritised rather than the use of non-GA specific Python techniques or packages.

Task 3 (40%): Write a 3-page report detailing the algorithmic and implementational choices you have made and justify them using the course content and external literature. You should discuss both your basic implementation and the extended/optimised version. You should also compare GA with one evolutionary-based optimisation algorithm of your choice, discussing any real-world application of GA, where its implementation has led to significant improvements being observed.

For high marks in this section, you will need to demonstrate a very sophisticated command of the theoretical underpinnings of GAs, and extensive engagement with a range of external literature, and might require spending time on this assignment beyond the provided estimates.