CT421 Project 1: Evolutionary Search

Colm O'Riordan

January 2024

1 Part A: Initial search landscapes

1.1 One-max problem

Consider the simple case of evolving a string that contains all 1s in every location. Let the length of the strings be 30. The initial population should be randomly created. Use standard mutation and one-point crossover. The fitness of a solution is the number of `1`s in the string.

Plot the average fitness of the population versus the generations passed.
This exercise is to show the operation of a genetic algorithm. We know the optimal solution in advance. This represents a very easy search landscape and is included to illustrate the operation of a genetic algorithm.
When plotting the average fitness, you don't need to write code to do the plotting; feel free to output the values to a text/csv file and use excel, gnuplot or other to plot the results.

1.2 Evolving to a target string

This is the same as before but instead of evolving to a string of `1`s, the population will evolve to find some target string.

Define a target string (a sequence of 1s and 0s) and adopt the same approach as before with an appropriate fitness function (number of matching values). Plot the average fitness as before.

1.3 Deceptive Landscape

Modify the fitness function from the one-max problem, such that the fitness function is equal to the number of 1s in the string for all cases except when there are no 1s present. In this case, the fitness should be $2\times$ `(length of the solution)`. Plot the average fitness over time.

This problem represents a deceptive problem and should be very hard for the GA to find the optimal.

1.4 Submission guidelines for part (a)

The following should be submitted:

- Code - link to github repository of code and plots. There should also be a description of the code structure (1 page maximum).
- A short report outlining:
    1. description of your representation; fitness function; selection, crossover, mutation, and any other operations.
    2. Plots of the performance of the genetic algorithm.
    3. Description of your results
    4. If submitting as part of a pair, a description of your contribution.
- Make sure you cite any sources you have used.

2 Part (b): Bin-packing Problem

Note: Most of your code - selection, mutation, crossover, over all structure can be re-used with some minor changes. The main aspects to concentrate on are: fitness function, representation, analysis of results.

The Bin_packing_problem on wikipedia gives a high-level description of the bin-packing problem. There are many applications of the bin-packing problem and many extensions of the problem that have varied applications. It is a combinatorial optimisation problem where the goal is to efficiently pack a set of items into the minimum of bins.

In this question, you are provided with several problem instances - items weights, bins and their capacity. Your developed GA should be able to tackle these problems and give a solution.

Typically, you have a set of items each with a weight (or size or some other values. The goal is to pack these items into a set of bins each with a fixed capacity. The goal is to use as few bins as possible to pack the items.

You should submit:

- A description of your representation of your solutions
- A description of your fitness function and operators together with any justifications of your choice
- Any insights into the problem landscape.
- Include code in the same repository as part (a).