Advanced Nature Inspired Search and Optimization Lab 1

Brendan Hart - 1560038

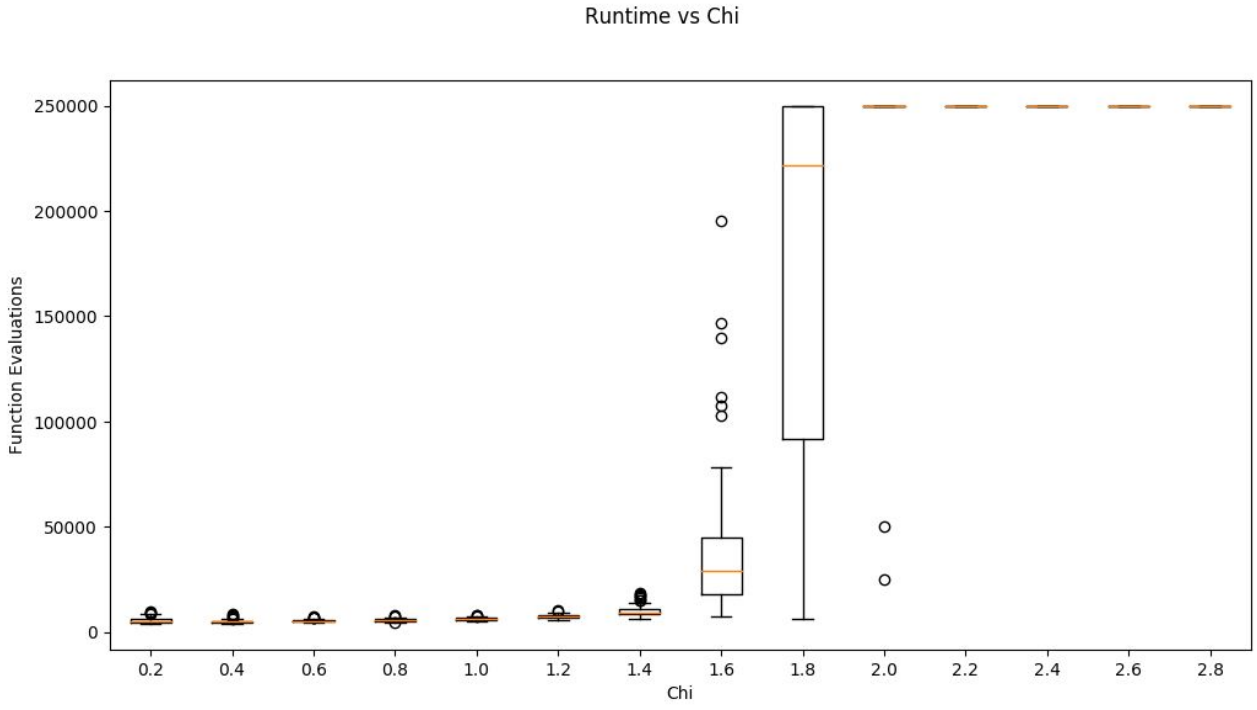
This document will investigate the effects on the runtime of a genetic algorithm under a variety of parameters. The runtime was defined as the number of function evaluations, which is equal to $population\ size * generations$. The parameters of the algorithm were:

- n : the problem size
- k : the tournament size
- λ : the population size
- χ : Used to calculate the mutation rate with the formula: mutation rate = χ / n).

The genetic algorithm will use bit strings, with uniform crossover, tournament selection and mutation by randomly mutating each bit in a bit string with a probability of the chosen mutation rate. The fitness function used was equal to the number of 1s present in the bit string, meaning the optimal solution was a bit string consisting of length n consisting of n 1s. For all experiments, the experiment is ended when the most recent generation contains a member with maximum fitness or the 2500 generation timeout is reached.

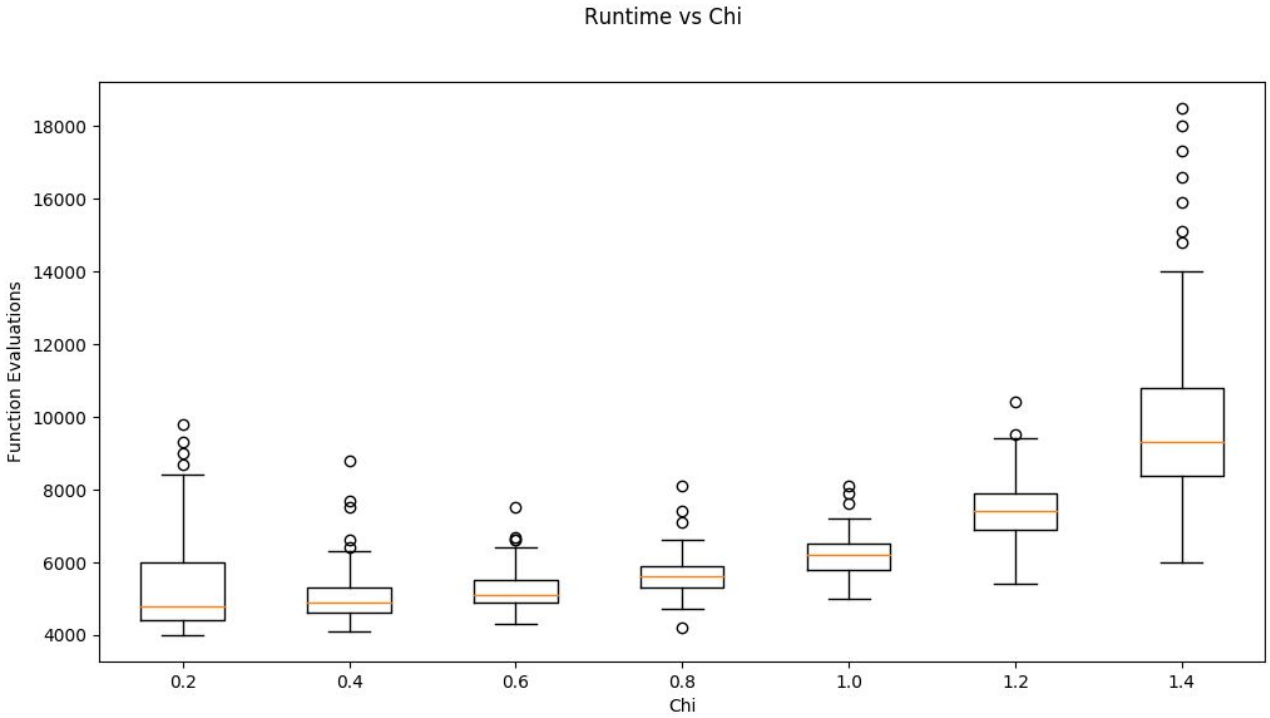
Runtime vs χ

χ was varied between 0.2 and 2.8 inclusive, with intervals of 0.2. The other parameters were held constant at n = 200, λ = 100, k = 2. The value for k means binary tournament selection was used, which allows for the fit members of the population to be chosen whilst allowing for diversity in the following generation.



The above plot shows the effect of varying χ on the runtime. It can be seen that from χ = 2.0, the algorithm does not find the optimal solution within the timeout. We can also see that χ = 1.8, whilst

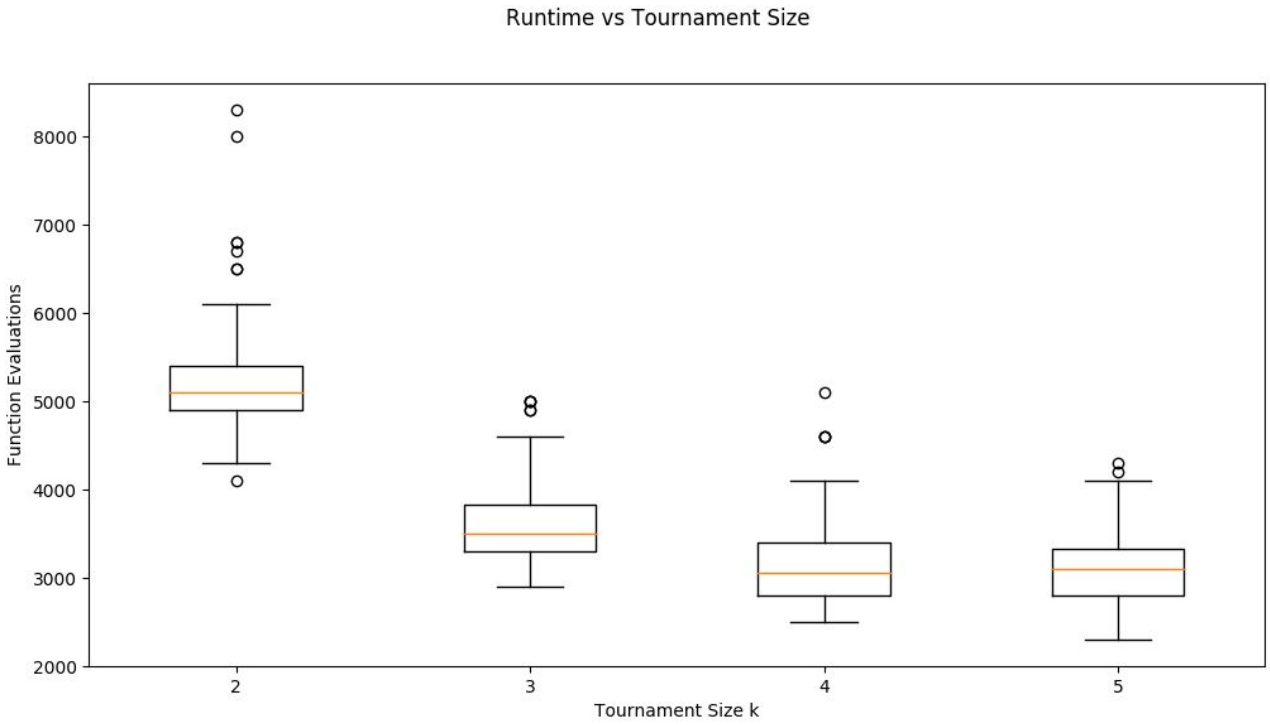
completing on average within the timeout, takes drastically longer than χ = 1.6 to complete. This suggests that high mutation rates can reduce the algorithms chances of finding the optimal solution, potentially because solutions close to the optimal solution have multiple bits mutated when only one may have needed changing.



The above graph shows the first 7 values used for χ, with the y-axis scale such that the box plots are more visible and comparable. The trend seems to show that increasing χ, for this problem, increases the runtime.

Runtime vs k

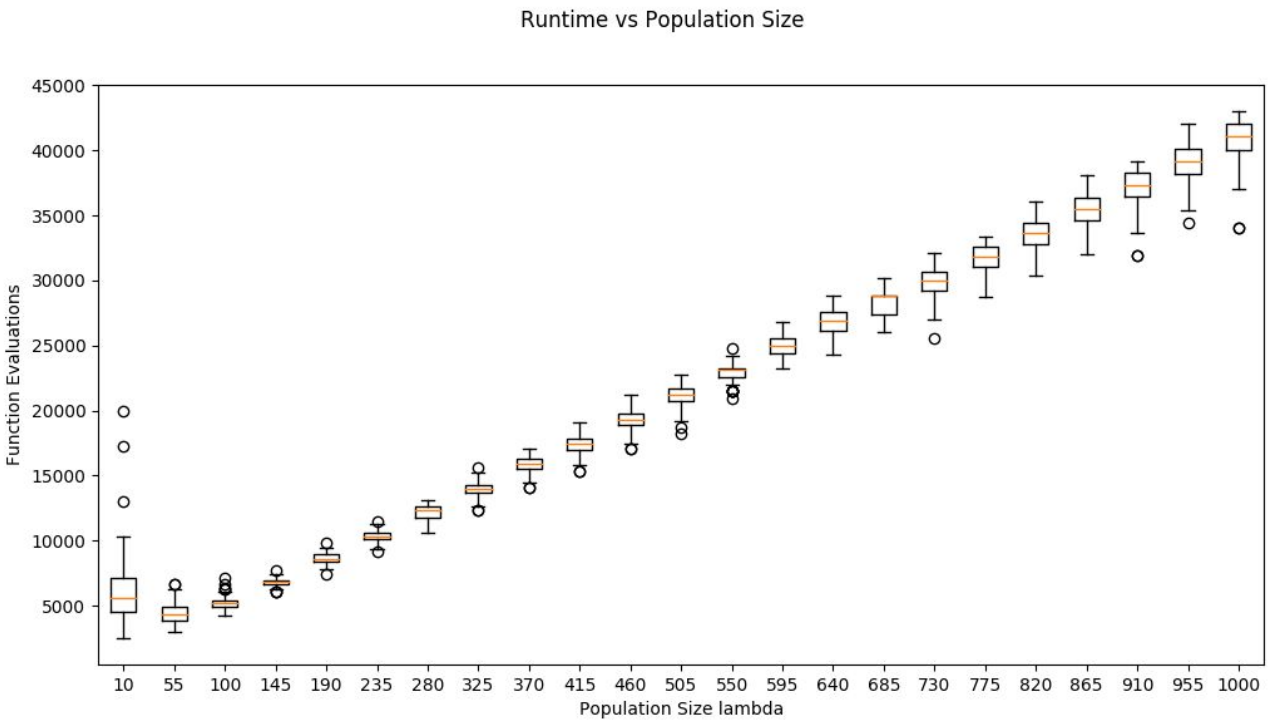
k was varied between 2 and 5 inclusive, with intervals of 1. The other parameters were held constant at n = 200, λ = 100, χ = 0.6. n was chosen to be 200 as to minimise the probability that the initial population will contain the optimal solution and hence provide a true view of the behaviour of the mutation rate.



The above plot indicates that as the tournament size increases, the runtime decreases. The reason for this could be that the algorithm is more likely to choose on of the best members of the population for crossover. In this trivial problem, a high tournament size may find the optimal solution quicker. For other problems, this may not be the case, as an increase in tournament size would mean the best solutions become selected for crossover more, resulting in less diversity in the population. This can result in the algorithm choosing a locally good solution, but possibly not the global optimum.

Runtime vs λ

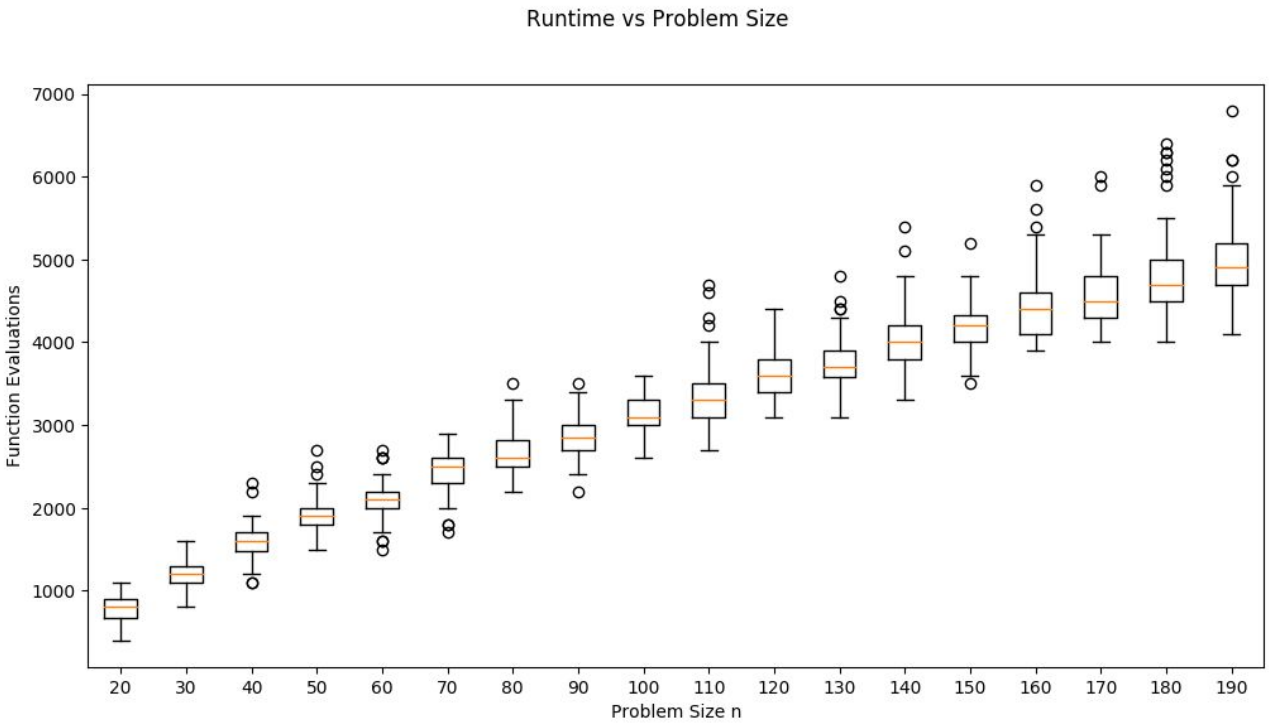
λ was varied between 10 and 1000 inclusive, with intervals of 45. The other parameters were held constant at n = 200, k = 2, χ = 0.6. k was again chosen to be 2 for the same reasons.



It can be seen that as the population size increases, so does the runtime. This is due to each generation carrying out a number of function evaluations which are equal to the population size, making the number of function evaluations proportional to the population size. The obvious outlier for population size is when the population size (λ) = 10. This may be due to the population size being too small that it fails to represent a large enough variety of solutions.

Runtime vs n

n was varied between 20 and 190 inclusive, with intervals of 10. The other parameters were held constant at λ = 100, k = 2, χ = 0.6. λ was chosen to be a value large enough to contain a variety of the solution space whilst keeping the runtime reasonable. k was again chosen to be 2.



There is a clear positive trend in the graph above. This leads to the suggestion that as the problem size increases, the runtime of the genetic algorithm also increases. This would follow our intuition, as it can be interpreted as a harder problem takes longer to solve.