Luke Biddell
Advanced Nature-Inspired Search & Algorithms
Lab 1: Exercise 6

Runtime vs Mutation Rate

The fixed parameters used for this test were n = 200, λ = 100 and k = 3. A k of 3 was chosen as it has a low runtime.
The varied parameter used was χ, where 0 < χ < 3, with increments of 0.125 as to avoid rounding errors storing as a double.
I set a timeout value of t = 2000 generations, giving a max run time in Figure 1 of 200000.
100 tests were done for each value of χ.

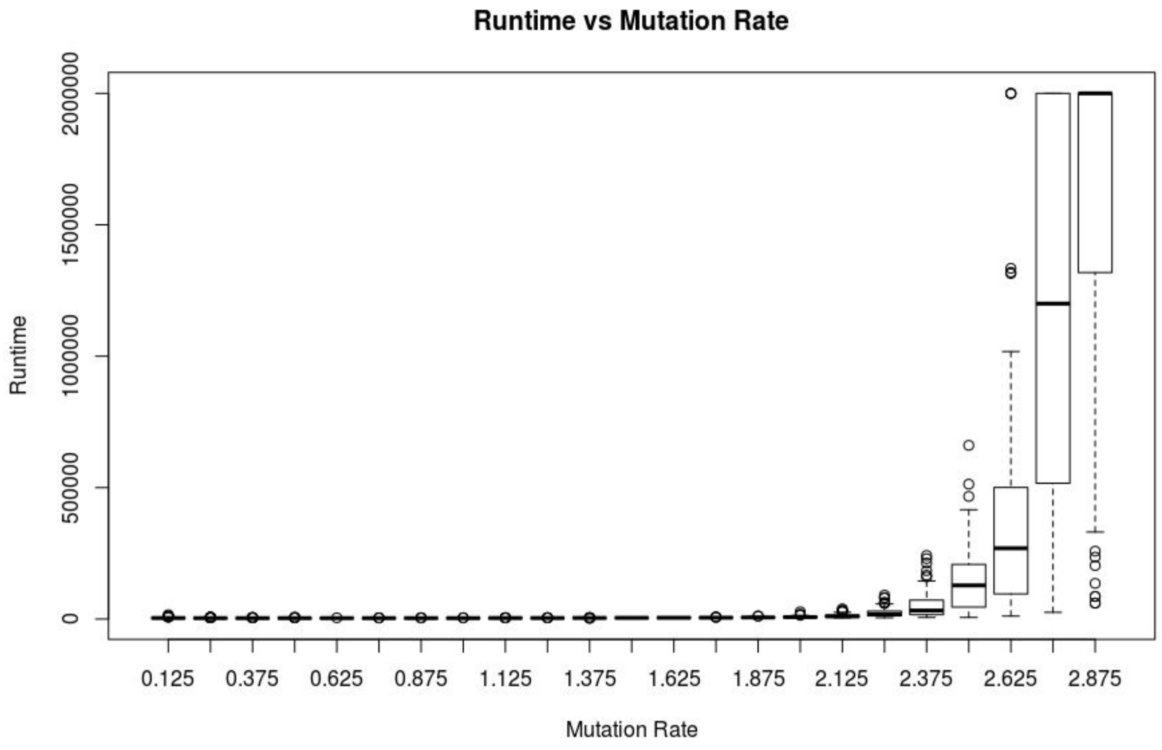

Figure 1: Boxplot showing runtime (number of generations t ✕ population size λ) vs mutation rate χ after 100 repetitions per iteration.

Figure 1 shows that the runtime is low for mutation rates up to 2.125, but from 2.25 onwards, the runtime increases rapidly. The range of the runtime also increases dramatically, producing a largely varied runtime. The graph shows that the runtime increases exponentially with respect to the mutation rate.

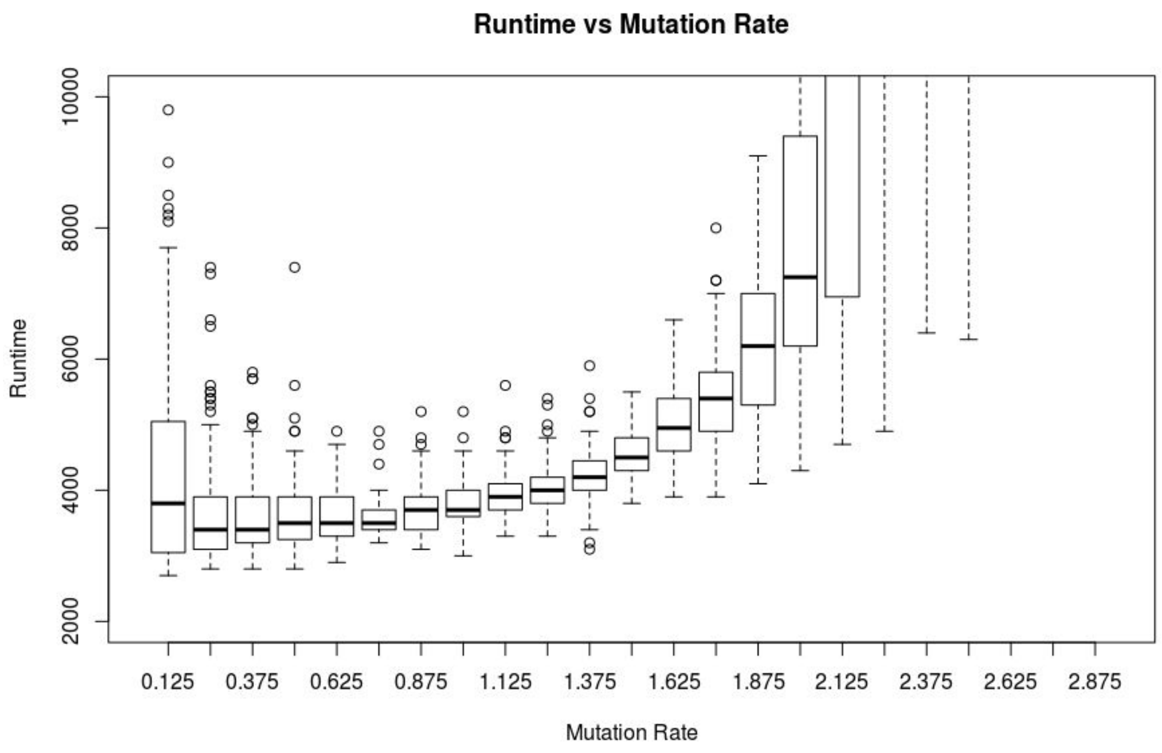

Figure 2: Boxplot showing runtime (number of generations t ✕ population size λ) vs mutation rate χ after 100 repetitions per iteration, with the y-axis values of runtime scaled to between 2000 and 10000.

Figure 2 shows us in more detail the runtime of smaller values of mutation rate. Whilst the minimum runtime generally increases in respect to the mutation rate, the smallest mutation rates, ranging from 0.125 to 0.75, are decreasing in both range of values and maximum values. From 0.75 to 2.875, they are increasing.

Whilst other mutation rates may have a lower median value, they have very high outliers of runtime. In our case, a mutation rate value of 0.75 would be the best choice, with the least variance and a low average runtime.

Runtime vs Problem Size

The fixed parameters used for this test were χ = 0.6, λ = 100 and k = 2. A k of 3 was chosen as it has a low runtime
The varied parameter used was n, where 10 < n < 200, with increments of 5.
I set a timeout value of t = 2000 generations, however no tests timed out.
100 tests were done for each value of n.

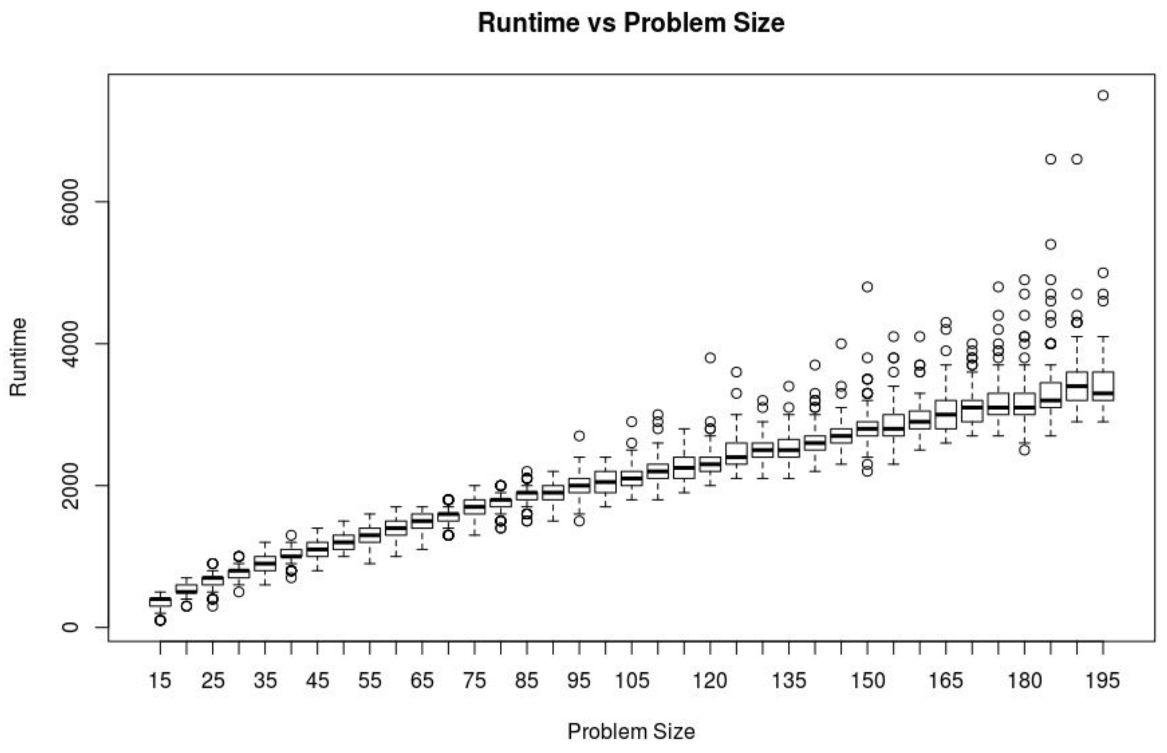

Figure 3: Boxplot showing runtime (number of generations t ✕ population size λ) vs problem size n after 100 repetitions per iteration.

The median runtime generally increases linearly with problem size. However, the larger the problem size, there are more outliers of increasingly large runtime, which would occasionally cause much longer runtimes. There are no large outliers for a shorter runtime.

Runtime vs Population Size

The fixed parameters used for this test were n = 200, χ = 0.6 and k = 2. A k of 3 was chosen as it has a low runtime.
The varied parameter used was λ, where 10 ≤ λ ≤ 1000, with increments of 25.
I set a timeout value of t = 2000 generations, however no tests timed out.
100 tests were done for each value of λ.

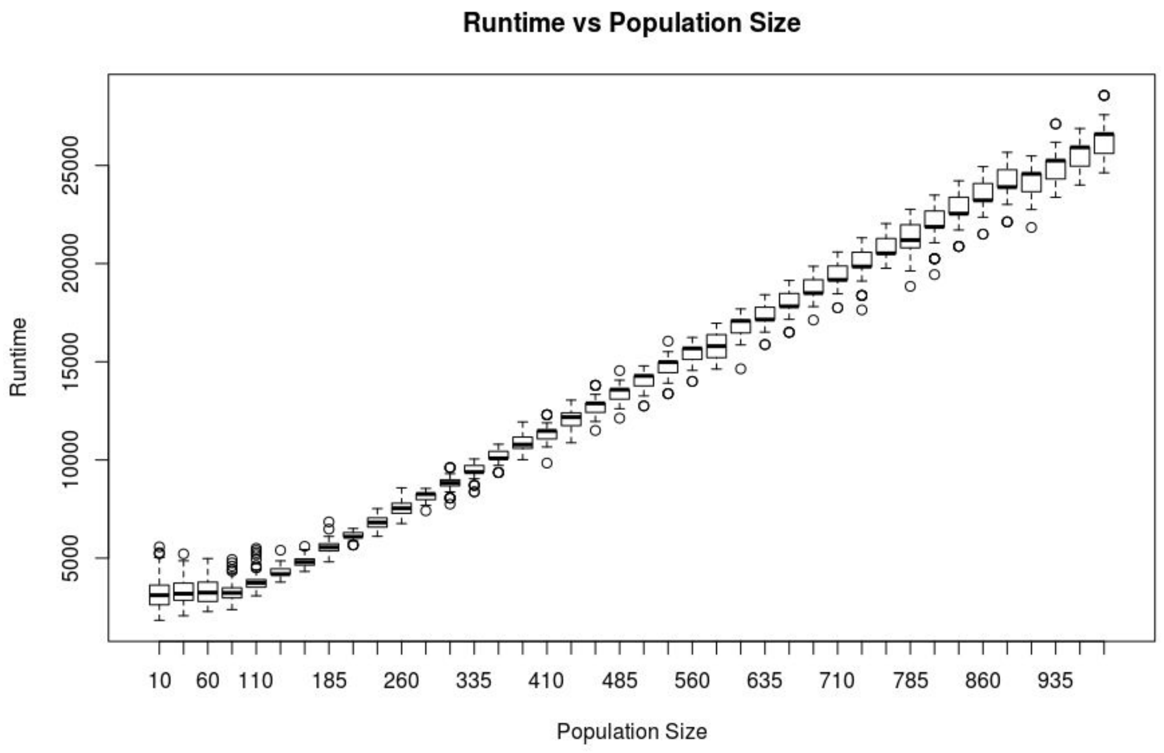

Figure 4: Boxplot showing runtime (number of generations t ✕ population size λ) vs population size λ after 100 repetitions per iteration.

As seen in figure 4, from 85 onwards, the population size increases linearly with the runtime. As population size increases, the range of run times for each one generally increases.

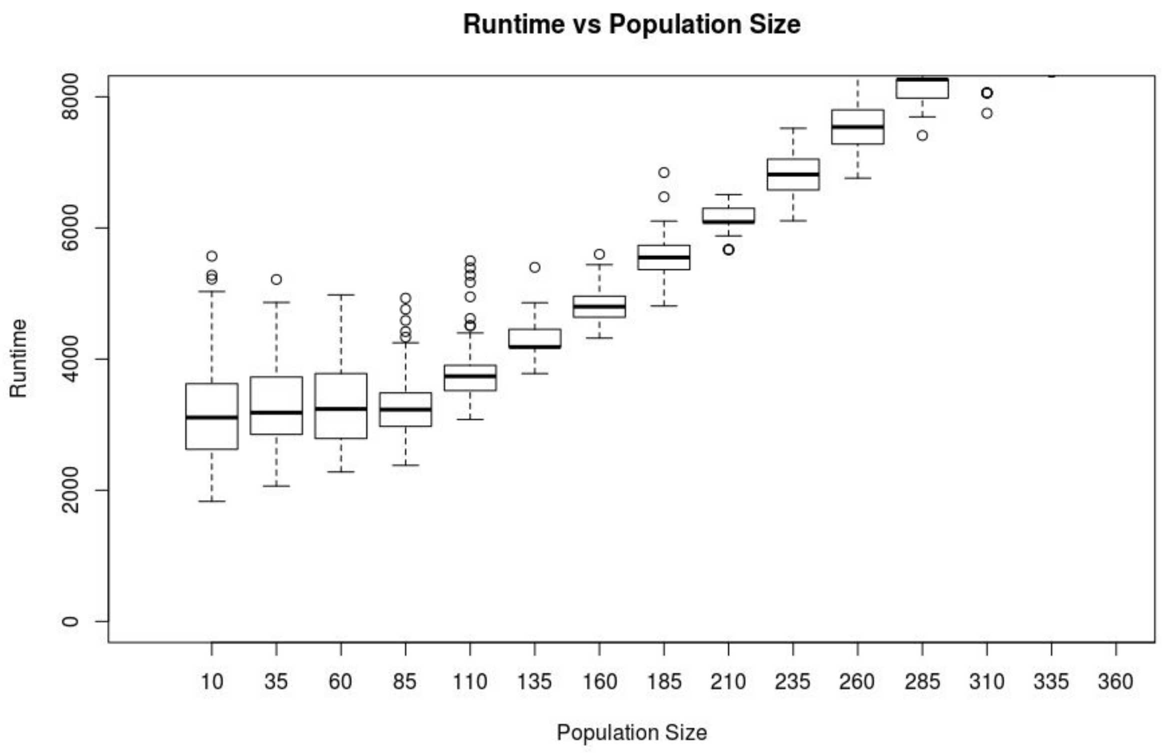

Figure 5: Boxplot showing runtime (number of generations t ✕ population size λ) vs population size n after 100 repetitions per iteration, with the y-axis values of runtime scaled to between 0 and 8000 and the x-axis values of population size scaled between 10 and 360

.

However, from a population size of 10 to 85, as seen in figure 5, the median values are very similar, with the smallest population sizes having a slightly bigger range of values. So any values in this range would be a good population size n to use in an algorithm in our case.

Runtime vs Tournament Size

The fixed parameters used for this test were χ = 0.6, λ = 100 and n = 200. An n of 200 was chosen as that was what was required in the previous tests.
The varied parameter used was k, where 2 ≤ k ≤ 5, with increments of 1.
I set a timeout value of t = 2000 generations, however no tests timed out.
100 tests were done for each value of k.

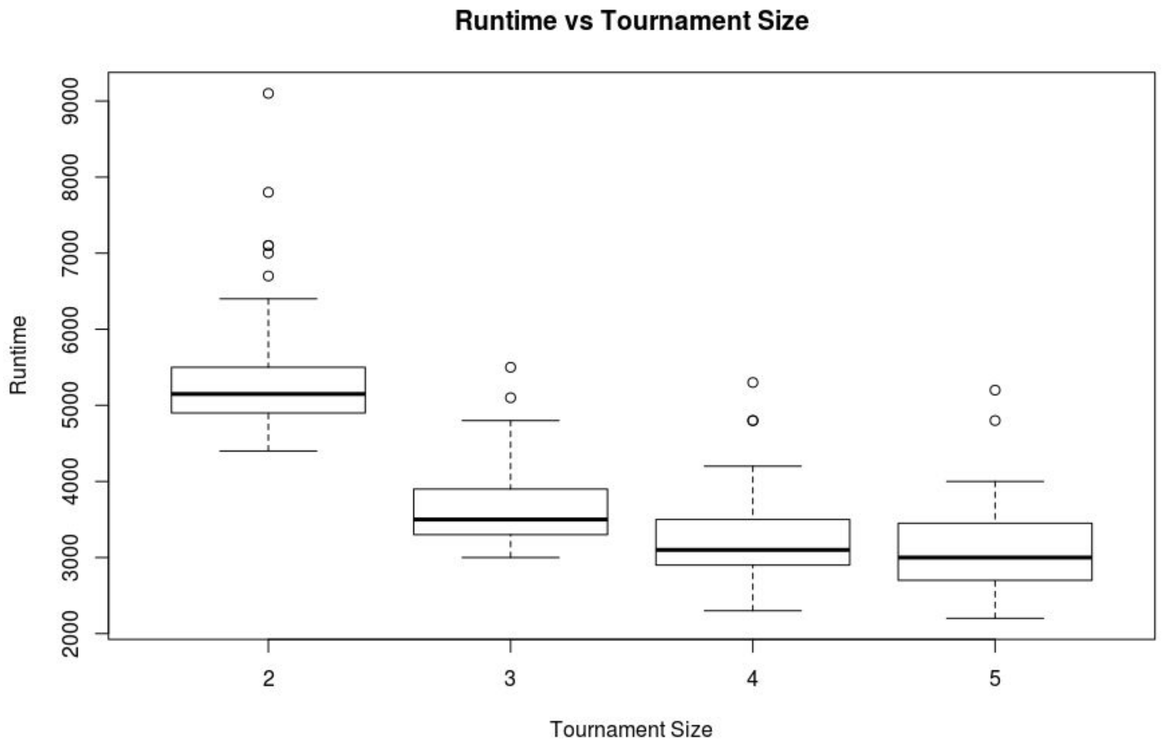

Figure 6: Boxplot showing runtime (number of generations t ✕ population size λ) vs tournament size k after 100 repetitions per iteration.

From this graph, we can see that a tournament of size 5 has the lowest values of runtime, so would be the best choice to use in an algorithm in this situation. A tournament size of 2 would be the worst, with a much higher median and very high outlier values of runtime. As tournament size increases, runtime decreases, at least between values of 2 to 5.