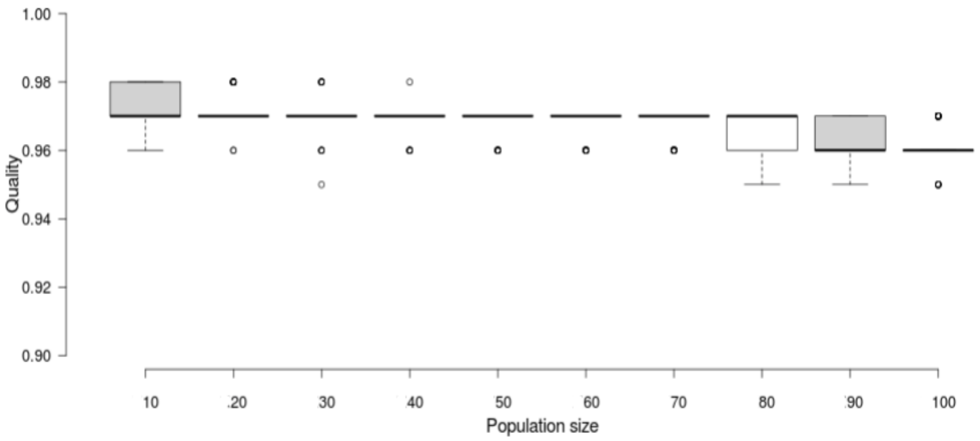
Lab 2 NISO – MaxSat Genetic Algorithm

Algorithm 1: Genetic Algorithm for the MaxSat problem

Require: Problem size $n \in \mathbb{N}$
Require: Population size $\lambda \in \mathbb{N}$ and tournament size $k \leq \lambda$
Require: Fitness function $f : \{0,1\}^n \to \mathbb{N}$
Require: Elitism proportion p where $0 \leq p \leq 1$
Require: Time limit

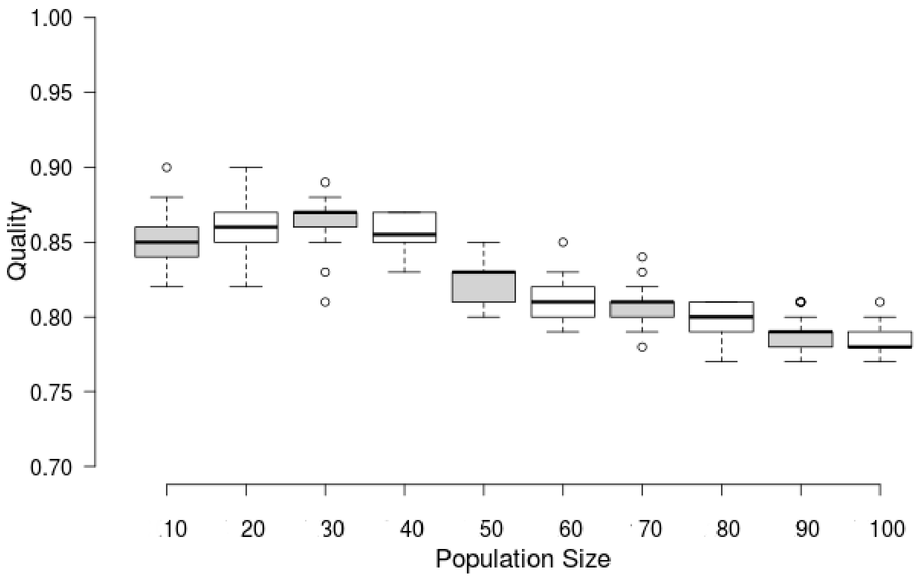
```
t = 0
for i = 1 to λ do
    P_t(i) ~ Unif({0,1}^n)
end
xbest = P_t(0)
while within time limit do
    sort P_t by fitness descending
    z = floor(p × λ)
    if z != 0 then
        for i = 0 to z do
            P_{t+1}(i) = P_t(i)
        end
    for i = z to λ do
        x = tournament_selection(P_t)
        y = tournament_selection(P_t)
        P_{t+1}(i) = crossover(mutation(x),mutation(y))
        if f(P_{t+1}(i)) ≥ f(xbest) then
            xbest = P_{t+1}(i)
    end
    P_t = P_{t+1}
    t = t + 1
end
return xbest
```

Population Size (lambda) Testing
WCNF file: mse17-complete-unweighted-benchmarks/kbtree/kbtree9_7_3_5_90_6.wcsp.wcnf
Parameters: n = 280, p = 0.1, chi = 0.6, time limit = 25 seconds



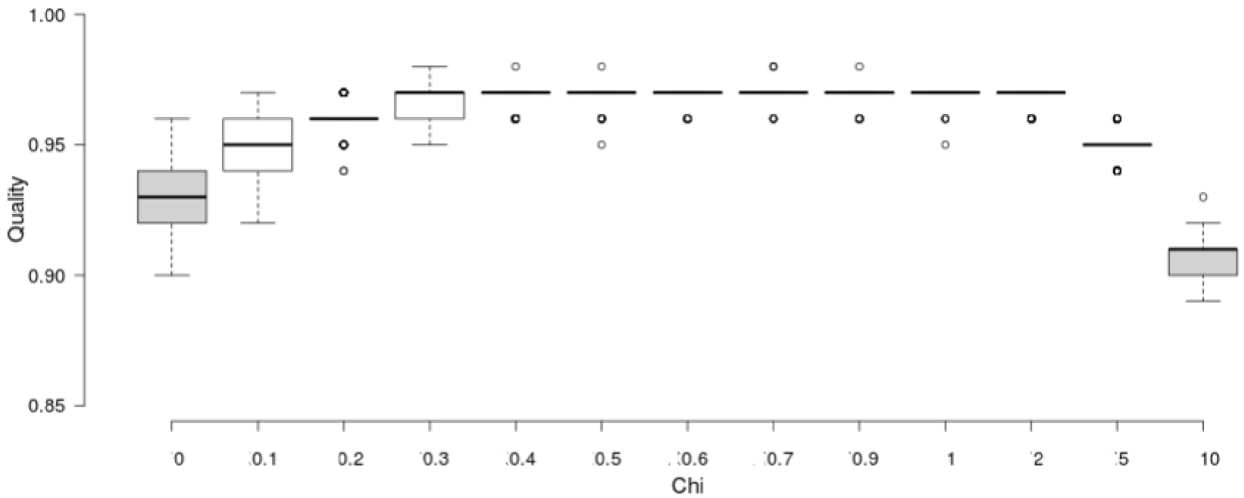
From the results using the parameters specified, we can see that there is not much difference in the quality of the solution between the different population sizes. The mean quality across each test was at least 0.97 which is near perfect satisfiability for this file, with the highest quality achieved being 0.98. Speculatively speaking, the optimum solution is believed to have 98% of clauses satisfied and so, after 25 seconds, the algorithm always converges to near this regardless of population size. Therefore, another test was done, with a much lower time limit of 5 seconds to test this hypothesis and to also possible see some variation in the results between population sizes.

Parameters: n = 280, p = 0.1, chi = 0.6, time limit = 5 seconds



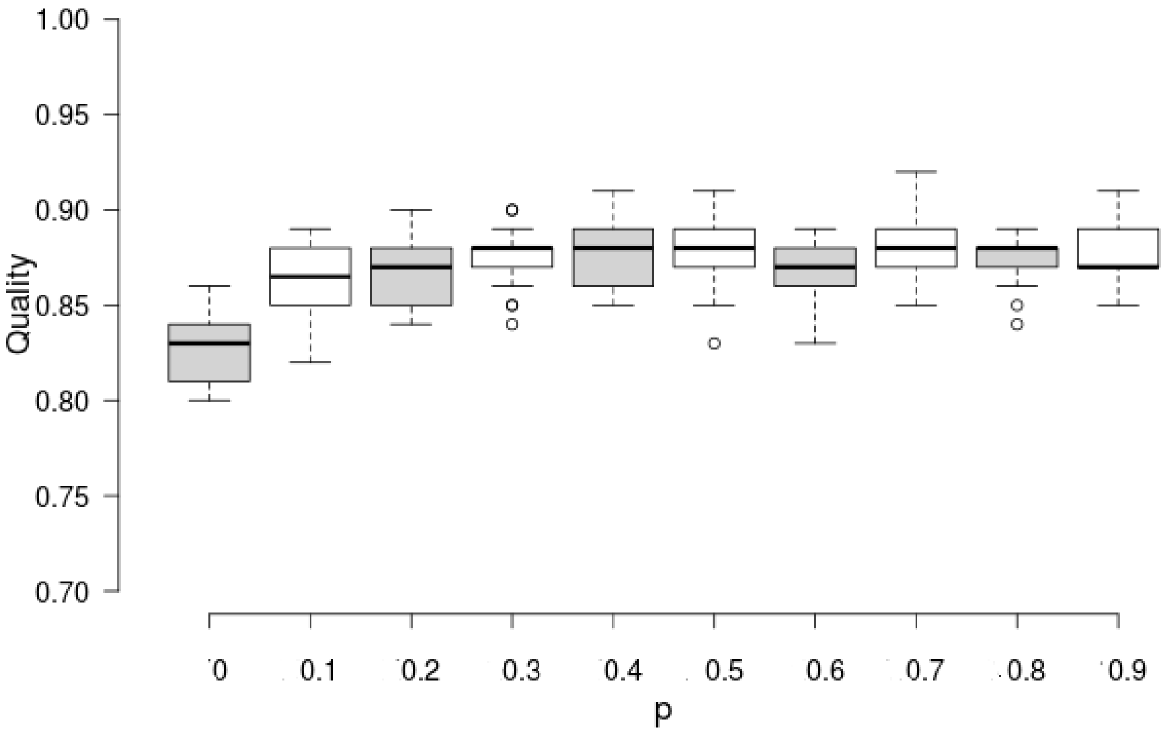
It appears from these results that given a much stricter time limit, the population size does affect the quality of the solution produced. Having a smaller population size seems to produce higher quality results possibly due to the fact there are less evaluations being performed for each generation and so converges to the local optima quicker time wise.

<u>Mutation Rate (Chi) Testing</u>
WCNF file: mse17-complete-unweighted-benchmarks/kbtree/kbtree9_7_3_5_90_6.wcsp.wcnf
Parameters: n = 280, lambda = 50, p = 0.1, time limit = 25 seconds



From these results, we can see that having a value of chi between 0.3 and 2 produced the best results. With no mutation at all, the best solution found was not as good as the values in the best range. This is because with no mutation rate, exploration of the search space is limited and fails at finding the local optimum. As we increase the mutation rate slightly, the algorithm finds what seems to be the optimum solution for this particular case. As the mutation rate increases to 5 and above, the algorithm starts turning into a random walk and the quality of the solution begins to suffer.

Elitism Percentage (p) Testing
WCNF file: mse17-complete-unweighted-benchmarks/kbtree/kbtree9_7_3_5_90_6.wcsp.wcnf
Parameters: n = 280, lambda = 30, chi = 0.6, time limit = 5 seconds



From these results, we can see that adding elitism does help the quality slightly. When p=0 (elitism is not applied) we can see a lower difference in the mean quality compared to the other tests when elitism is applied. Adding in elitism benefits performance by not needing to create new members and also evaluating them. All values of elitism above 0.1 seems to perform the same as each other. A reason as to why this could be is because as p gets bigger, we save time computing new members in the population so is able to make multiple generations very quickly, converging at the same rate as it would if p was lower.