Evolutionary Approach to the 0-1 Knapsack

Contents

1 Implementation and Methods

In this assignment, we are using a genetic algorithm to solve the simple knapsack problem. Each solution is represented by a bit string of length n. The fitness function used for our implementation of the algorithm includes constraint handling and can be written out as:

$$f(x) = \begin{cases} v^T\vec{1} + v^T x & w^T x \le c \\ (v^T\vec{1})(1 - \frac{w^T x - c}{w^T\vec{1} - c}) & w^T x > c \end{cases} \tag{1}$$

In this function, we are creating a constant to help us handle fitness and constraints by using the sum of the profits. If the constraints are not met, the constant gets scaled down by how much the solution violates the constraint. If the constraints are met, the value of the of the solution (also known as the profit) is added to our constant. The baseline mutation function is a random bit flip, and the baseline crossover function is a single-point crossover at a random point in the bit string.

When examining our algorithm and the changes in the performance as we vary parameters, we followed a few standard procedures. The first standard was our sample size. For each variable we tested, we wanted to ensure that we could have comprehensive comparisons that were statistically significant. To achieve this we pulled 30 samples for each parameter from 6 files (for testing crossover rates, mutation rates, and varying population) or 10 files (for testing modified crossover and mutation functions). For each sample, we collected data on whether the constraints were met (1 if met, 0 else), optimal difference (achieved fitness divided by optimal fitness), Calls to the fitness function, model Parameters (population size, mutation rate, etc.). The standard generation count for all our samples was 1000 generations. However, the algorithm terminated and moved onto the next text if no improvements as seen after 300 consecutive generations.

Our algorithm code, input data, testing procedures, and test results are all on Github and can be accessed by visiting `https://github.com/upadhyan/genetic-simple-knapsack`. The implementations of our GA can be found in `knapsack_functions.R`. The scripts used to test and generate are in `test_executions.Rmd`. Most of our statistical analysis and plotting scripts are in `analysis_functions.R`.

2 Mutation & Crossover Rate Examination

2.a Varying Mutation

In this section the probability of mutation is varied and subsequently tested $[0.01, 0.01]$ in steps of 0.01. The crossover rate is set for this section at a value of 0.80. By varying the mutation rate, while holding the crossover rate constant, information can be garnered to see what the optimal mutation rate should be, given the fitness call and fitness responses to different mutation rates.

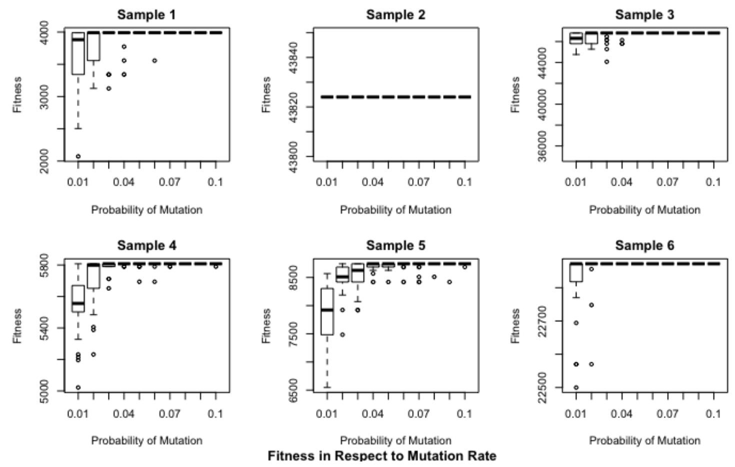


Figure 1: Fitness in Respect to Mutation Rate

Figure 1 shows how the fitness responds to the different levels of mutation probability for each sample. Here, for most of the samples the interquartile range of the fitness level decreases at the probability of mutation increases.

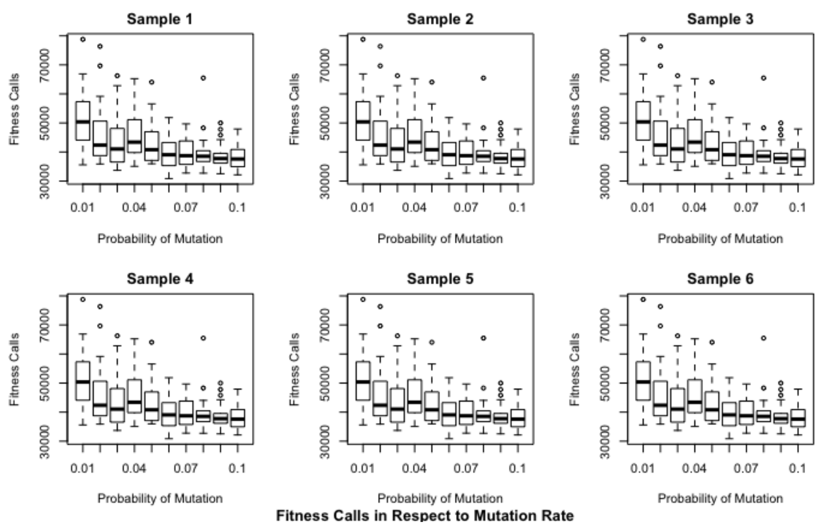


Figure 3: Fitness Calls in Respect to Mutation Rate

The fitness appears to converge around 0.02, but for some samples, there is still variation. Outliers are eliminated for the most part as the probability of mutation reaches at least 0.04. Overall, the median fitness level appears to be relatively stable across all mutation probabilities.

To confirm the hypothesis that the mean fitness of mutation probability is the same at all levels, given a generally consistent median value across levels, an ANOVA test (Figure 2) was performed. Because the box-plots were not similar across all samples, samples were blocked in this analysis. Using $\alpha = 0.01$, there is not evidence to support that at least one level's mean fitness is not equal to the others, as $p > 0.01$.

```
              Df    Sum Sq   Mean Sq F value   Pr(>F)    
block          1 1.150e+10 1.150e+10  37.891 9.21e-10 ***
pmutation      1 1.410e+07 1.410e+07   0.046    0.829    
Residuals   1797 5.456e+11 3.036e+08                     
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Figure 2: ANOVA for Fitness Response

Next, the number of fitness calls were analyzed for each level of mutation, seen in Figure 3. Each box-plot shows a different sample. All samples appear to have a similar trend, where the median and interquartile range of fitness calls decrease as the probability of mutation increases. This trend may be because the GA converges faster when there is a higher rate of mutation, thus resulting in fewer fitness calls.

Again, to provide evidence for the hypothesis that there is at least one mean fitness call value for a given mutation level that is different from the rest, an ANOVA test is run (Figure 4). The samples were blocked in this analysis. Using $\alpha = 0.01$, there is evidence to support that at least one level's mean fitness call value is not equal to the others, as $p << 0.01$.

```
              Df    Sum Sq   Mean Sq F value Pr(>F)    
block          1 2.292e+10 2.292e+10   173.3 <2e-16 ***
pmutation      1 2.146e+10 2.146e+10   162.2 <2e-16 ***
Residuals   1797 2.377e+11 1.323e+08                   
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Figure 4: ANOVA for F. Function Call Response

Finally, The proportion of the optimal value reached (optimal difference) for each mutation probability level is shown.

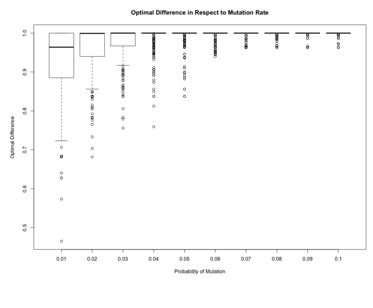
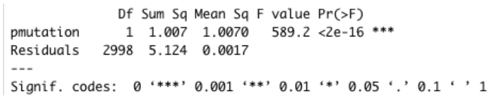


Figure 5: Prop. of Optimal Value Reached for all Samples

The median optimal difference converges to 1 quickly, but the range decreases more gradually. There appear to be a large proportion of outliers until about a mutation probability of 0.07, where the proportion of outliers between levels become more uniform.

An ANOVA test was completed to see whether there is significance in a difference of means for the mutation levels (Figure 6). As $p < 2e^{-16}$, using $\alpha = 0.01$, there is evidence to suggest at least one optimal difference mean is not equal to the others. This can be seen when the probability of mutation is small.

Overall, lower rates of mutation appear to have a higher variability and a lower overall fitness. Hence, it is best to avoid choosing these values to be an optimal parameter. Additionally, the number of function calls decreases as the mutation rate increases. The rates with the lowest spread of points appear to be 0.08 and 0.09, which can be seen in the fitness box-plot.



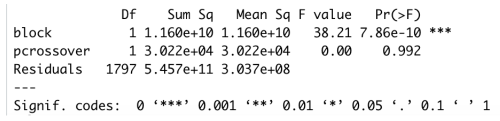
```
              Df Sum Sq Mean Sq F value Pr(>F)
pmutation      1  1.007  1.0070   589.2 <2e-16 ***
Residuals   2998  5.124  0.0017
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Figure 6: ANOVA Analysis for Optimal Proportion Reached

2.b Varying Crossover

In this section the crossover rate is varied in the GA and the mutation rate is held constant at 0.80. The crossover rate is varied [0.01, 0.91] in increments of 0.10. By varying the crossover rate, while holding the mutation rate constant, analysis can be conducted into determining what the optimal crossover rate should be, given the response variables of fitness and calling the fitness function.

When observing the fitness when varying the crossover rate for each sample, the fitness stays relatively constant for each crossover rate and that there is almost no variation or large interquartile range between populations within the sample. Because of this, there does not appear to be a great impact on fitness, no matter the crossover level. The aggregate mean for each crossover level will be evaluated using the optimal difference.



```
              Df    Sum Sq   Mean Sq F value   Pr(>F)
block          1 1.160e+10 1.160e+10   38.21 7.86e-10 ***
pcrossover     1 3.022e+04 3.022e+04    0.00    0.992
Residuals   1797 5.457e+11 3.037e+08
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Figure 7: ANOVA Analysis for Fitness Response

As hypothesized in the box-plots, there is no significant variation of the mean at any crossover rate analyzed, as $p = 0.992$, which is far greater than $\alpha = 0.01$.

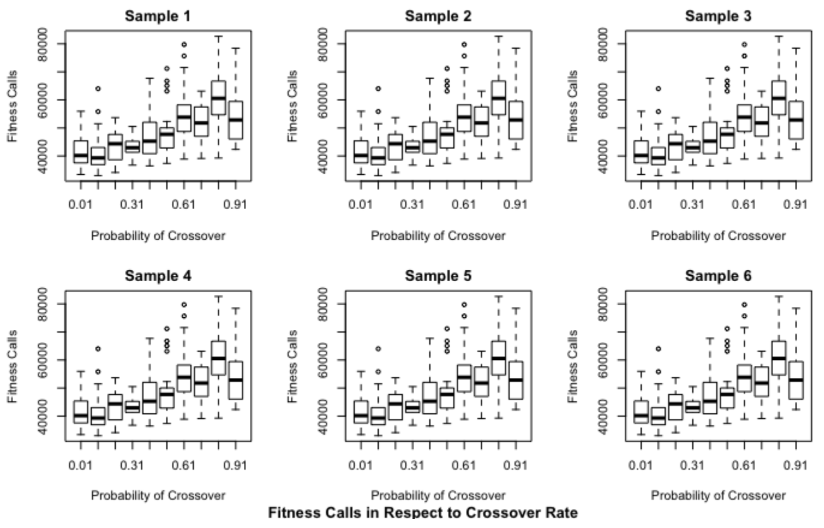
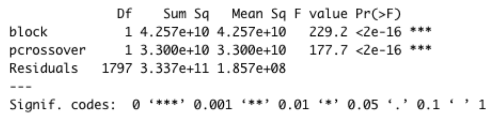


Figure 8: Fitness Calls in Respect to Crossover Rate

The box-plots in Figure 8 shows the number of fitness calls made at each crossover level for each sample. The trends for each sample look relatively consistent, as the median number of fitness calls increases a little from $p_{crossover} = 0.01$ to until $p_{crossover} = 0.51$, then increases at a larger rate until it peaks at $p_{crossover} = 0.81$. Additionally, the range appears to be at a minimum at $p_{crossover} = 0.31$ and $p_{crossover} = 0.51$.



```
              Df    Sum Sq   Mean Sq F value Pr(>F)
block          1 4.257e+10 4.257e+10   229.2 <2e-16 ***
pcrossover     1 3.300e+10 3.300e+10   177.7 <2e-16 ***
Residuals   1797 3.337e+11 1.857e+08
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Figure 9: ANOVA Analysis for Fitness Call Response

To confirm that the strong trends seen in the box-plot are in fact significant, an ANOVA test is run (Figure 9), with sample being used as a blocking variable and the number of fitness calls being used as the response. Because $p << 0.01$, there is evidence to suggest not the mean number of fitness calls at a given crossover rate are not the same across levels.

Finally, the optimal difference across crossover levels is evaluated. Looking at the box-plot in figure 10 (which aggregates all replications from all samples), the median optimal difference appears to be at 1 (100%) for all levels. There also appear to be several outliers for each crossover level. Based of solely the box-plot, it does not appear that there is a significant difference in the proportion of the optimal value reached for each crossover probability level.

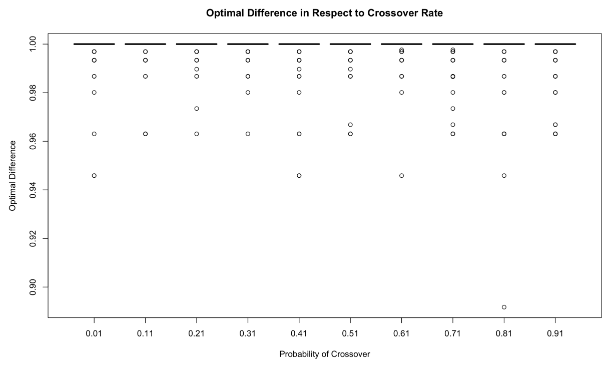
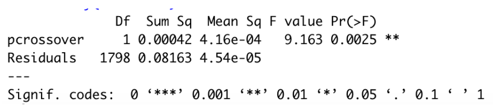


Figure 10: Proportion of Optimal Value Reached for all Samples



```
              Df  Sum Sq  Mean Sq F value Pr(>F)
pcrossover     1 0.00042 4.16e-04   9.163 0.0025 **
Residuals   1798 0.08163 4.54e-05
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Figure 11: ANOVA Analysis for Optimal Proportion Reached

When testing the hypothesis on whether the mean optimal difference value is different at at least one crossover rate, a surprising discovery is found. Contrary to the box-plot, the ANOVA test (Figure 11) states that there is a significant difference in the mean value of optimal difference for at least one crossover level, as $p = 0.0025 < \alpha$. A possible explanation for this is that the outliers could have skewed the variance of the optimal difference at each level, subsequently skewing the mean. Or, because there are 1800 data points, the data set may be more prone to finding significance in slight variations of the mean.

From all the above findings when crossover is varied, it appears that the fitness is relatively constant, no matter the crossover rate. Hence, the number of fitness calls must serve as a strong factor to determine what crossover level should be used. Here, larger crossover rates > 0.51 result in a higher number of function calls, which may hinder the GA's computational prowess and agility. A greater range of function calls are seen at $p_{crossover} = 0.01, 0.11, 0.21, 0.41$, which may hinder the overall precision of the GA when scaled to other data sets. Because of this, $p_{crossover} = 0.31, 0.51$ seem to be good candidates.

2.c Varying Mutation and Crossover

In this section, both the mutation rate and crossover rate are varied. Mutation rate is [0.01, 0.1] in steps of 0.01; crossover rate is [0.01, 0.91] in steps of 0.10.

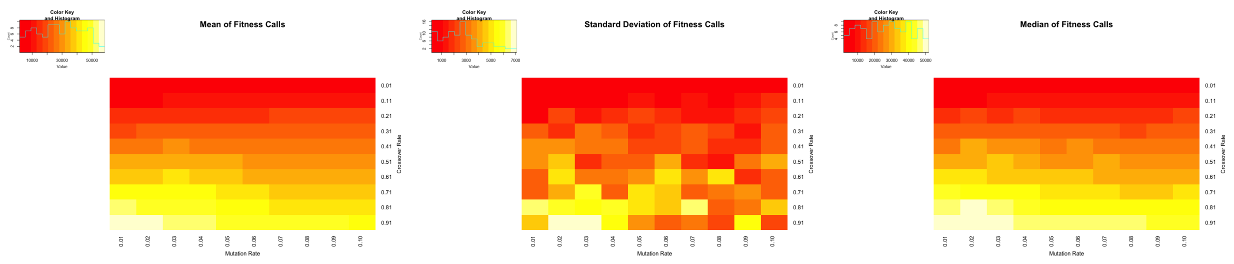


Figure 12: Fitness Calls (i) Mean (ii) Standard Deviation (iii) Median

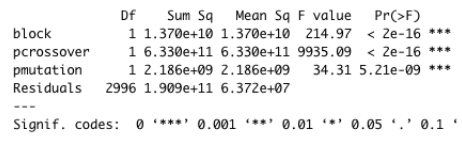
The number of fitness calls made were evaluated when both the mutation and crossover rates were varied. Figure 12 depicts heat maps for the mean, standard deviation, and median of the number of function calls made for each configuration. The desired result is to have a low mean, median, and standard deviation, as the number of function calls need to be minimized to promote overall efficiency in the GA. Moreover, it is desirable for the GA to be precise when implemented in other data sets; therefore, a small standard deviation is needed.

The mutation rate appears to have a relatively uniform mean and median number of function calls across all levels. On the other hand, the mean and median number of function calls for the crossover rate is more favorable at lower crossover rates. There is a uniform increasing gradient of function calls a low crossover rate to a high crossover rate (the more red, the lower the mean/median).

When looking at the standard deviation, a lower crossover rate yields a lower standard deviation. Again, as the mutation rate increases, the standard deviation is relatively uniform, like the mean and median. However, especially for higher crossover rates, higher mutation rates tend toward a lower standard deviation.

Overall, the heat map shows many favorable options; the only "quadrant" that does not appear to be favorable in regard to number of fitness calls is a low mutation rate with a high crossover rate.

The ANOVA test (with the sample serving a block) confirms that crossover rate and the mutation rate both have a significant impact on the number of fitness calls, as both $p << 0.01$.



```
              Df    Sum Sq   Mean Sq F value   Pr(>F)    
block          1 1.370e+10 1.370e+10  214.97  < 2e-16 ***
pcrossover     1 6.330e+11 6.330e+11 9935.09  < 2e-16 ***
pmutation      1 2.186e+09 2.186e+09   34.31 5.21e-09 ***
Residuals   2996 1.909e+11 6.372e+07                     
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Figure 13: ANOVA Analysis for Fitness Calls - Both Varied

The optimal difference when both mutation rate and crossover rates are now being analyzed to determine the "best" parameters for the GA. Here, a high mean and median optimal difference is desired to ensure the GA is reaching the optimal value consistently. A low standard deviation is desired as well to promote precision and lack of outliers when implementing the GA across other data sets.

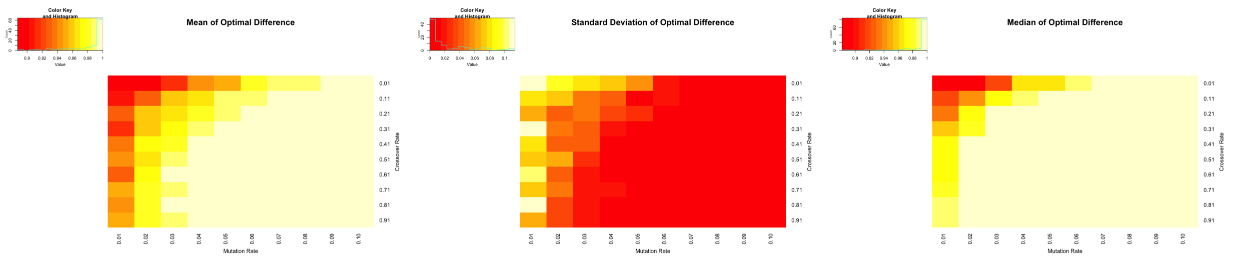
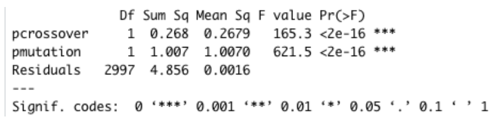


Figure 14: Optimal Difference (i) Mean (ii) Standard Deviation (iii) Median

Looking at the mean and median of the heat maps, sub-prime means and medians are located when the mutation and crossover rates are low. Hence, mid-to-high mutation rates and crossover rates are favorable. The standard deviation also expresses this, as lower standard deviations are present for mid-to-high mutation rates and crossover rates.

The ANOVA test confirms that crossover rate and the mutation rate both have a significant impact on the optimal difference, as both $p << 0.01$.



```
             Df Sum Sq Mean Sq F value Pr(>F)    
pcrossover    1  0.268  0.2679   165.3 <2e-16 ***
pmutation     1  1.007  1.0070   621.5 <2e-16 ***
Residuals  2997  4.856  0.0016                   
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Figure 15: ANOVA Analysis for Optimal Proportion Reached - Both Varied

In summary, when analyzing the impact of only varying the mutation rate, it was determined that either 0.08 or 0.09 would be ideal candidates for the best mutation parameters, as they both had high fitness, and a low number of fitness calls with a small range. When analyzing the impact of varying only the crossover rate, 0.31 and 0.51 rose to be ideal candidates because of their high level of fitness, and low amount of range in results coupled with a generally lower number of fitness calls (although not the lowest). Here, a trade off was made to prioritize precision over median number of fitness calls. In the end, it was decided to use 0.08 as the "best" mutation parameter and 0.51 as the "best" crossover parameter. Here, the standard deviation for the number of fitness calls is at a minimum (compared to the other choices), the standard deviation for the optimal difference is at a minimum, and the mean and median for the optimal difference is maximized at 1. The only area where a slight concession was made was in the mean and median number of function calls, where it was not minimized. This means that the efficiency of the GA may be compromised a little, as it may take longer for the GA to run.

3 Population Size Examination

3.a Optimal Population Size

After calculating the best mutation and crossover rate, we wanted to explore how the size of the population may impact performance. During our procedures, we attempted to control for number of fitness calls as much as we could.

A random sample of population sizes were taken and tested to determine how well the genetic algorithm could come to reaching the optimal value. Looking at the plot, small population sizes less frequently reached the optimal value (denoted by 1.00 on the y-axis). In addition to a lower mean optimal difference, the variance is high between the results of different trials when the populations are low. Compared to larger population sizes (e.g., 150) that have a small variance, smaller population sizes (e.g., 40) have optimal differences between roughly 0.84 and 1.00. As the population size increases, the optimal difference converges to 1.00, meaning the optimal value is reached. The genetic algorithm appears to converge to the optimal solution (optimal difference is 1.00) when the population size is at least 135. Because of this, a population size of 135 will be used when implementing different mutation and crossover functions.

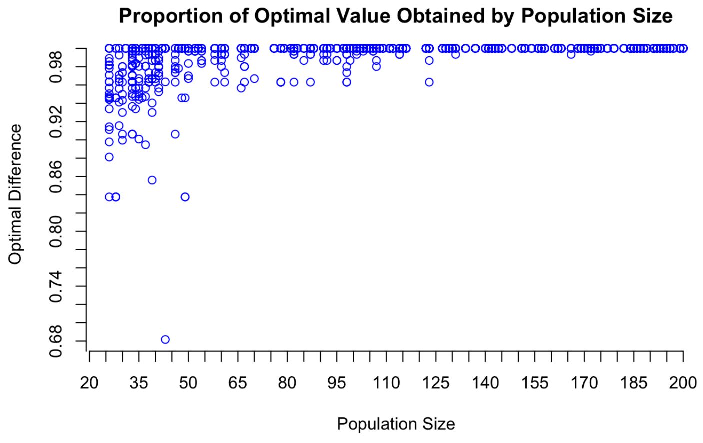


Figure 16: Optimal Difference of Varied Population Sizes

3.b Random Solution Generation

To examine the effect of *evolution*, we tested creating a large set of individuals and seeing whether or not we got similar results to the evolutionary process. To ensure fairness in our comparison, we set the number of individuals equivalent to the number of fitness calls done on a baseline run.

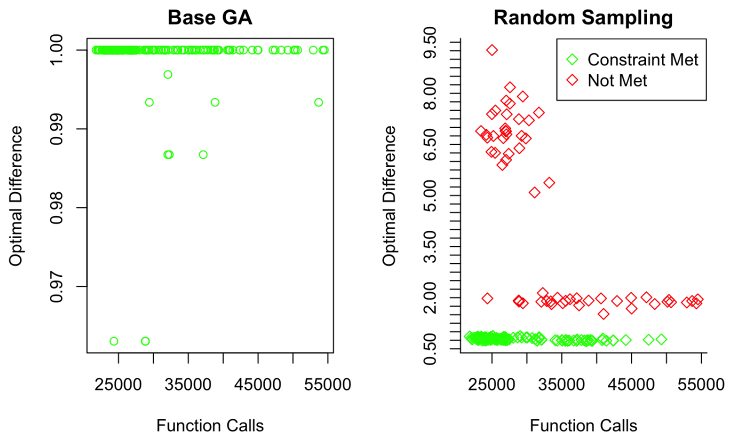


Figure 17: Optimal Difference between Base GA and Random Sampling

Figure 17 shows, after setting the number of function calls equal, sampling the solution space randomly without iterating through multiple generations does not allow for as many individuals to reach a feasible optimal value. The solutions generated were either within the feasible space but sub-optimal, or were not in the feasible space.

4 Custom Mutation and Crossover Function

4.a Weighted Mutation

While the rest of the tests were run using uniform random mutation, we wanted to explore the possibility of utilizing a ”weighted” mutation where the probability of a bit changing is linked to its weight-to-value proportion. We wanted to ensure that high-value bits did not leave or enter a solutions easily, and defined the probability of of bit b_i being switched as

$$p(b_i) = \frac{w_i/v_i}{\sum_{j=1}^{n} w_j/v_j}$$

With this definition, a high value item will find it hard to enter a knapsack solution, but once it enters, it will find it hard to leave. The mutation function then uses this probability definition to choose a bit to flip. To ensure optimal performance, we utilized the best population size, mutation rate, and crossover rate we discovered from previous testing (135, .08, and .51 respectively)

4.a.1 Fitness Calls

Summary statistics were run for the base mutation and weight mutation fitness calls. The base GA mutation function on average called the fitness function an average of 30,758 times and a median of 28,648. A difference of about 2,000 calls between the median and mean suggest a non-normal distribution and skewing in the data. The weighted mutation function on average calls the fitness function 45,070 times, and a median of 45,070. Here, the median and mean are much more aligned, but appears to be significantly higher than the base GA mutation function.

To confirm this, a two-sample t-test is conducted to test the mean number of fitness function calls for each mutation function. Using $\alpha = 0.01$, the hypothesis that the two means are equivalent is overwhelmingly rejected, as $p < 2.2e^{-16}$. The 99% confidence interval for this analysis of means is $[-16125.68, -12409.69]$. Hence, there is evidence that the GA using the weighted mutation calls is not as computationally profitable as the base GA in terms of of mean number of fitness function calls.

4.a.2 Optimal Difference

Summary statistics were also compiled for the optimal difference values for both the base GA mutation function and the weighted mutation function. Both the base and the weighted mutation functions had a median optimal difference of 1.00. Furthermore, the mean values between the base and the weighted mutation functions were similar as well, as they were 0.999 and 0.9982, respectively. This means that on average, an individual's optimal value was roughly 99.9% of the true optimal value.

A two-sample t-test is conducted to see if there is statistical significance between the mean optimal differences for the base and weighted mutation functions. Again, using $\alpha = 0.01$, there is not a significant difference between the means of the base and weighted mutation functions, as the p-value is 0.06954. The 99% confidence interval includes zero and is $[-0.00035, 0.00193]$.

4.b Favorite Child Crossover

We also wanted to explore changing the crossover function and what impact that would have on convergence rates. The custom crossover function developed is called ”Favorite Child Crossover.” In this crossover method, we take 2 parents and use them to produce 2 children. The first child (the favorite child) has a high probability to receive traits from the more fit parent, and the second child has a higher probability to receive traits from the less fit parent. For the first child:

$$p(b_i^{c_1} = b_i^{p_1}) = \frac{f(p_1)}{f(p_1) + f(p_2)} \qquad p(b_i^{c_1} = b_i^{p_2}) = \frac{f(p_2)}{f(p_1) + f(p_2)}$$

Where p_1 and p_2 are parent one and two respectively, b_i^x represents the ith bit in the bitstring of individual x. For the second child, the probability is flipped (the probability of a bit coming from parent one is equal to the fitness of parent two over the total fitness and vice versa). The idea behind this is to try and balance exploration vs. exploitation. By allowing one child to have a chance to inherit mostly ”good” traits, the algorithm may find a solution faster, but could cause pre-mature convergence. Therefore we can balance this out by having a second child explore what could happen along the evolutionary path of the inferior parent. To ensure optimal performance, we utilized the best population size, mutation rate, and crossover rate we discovered from previous testing (135, .08, and .51 respectively).

4.b.1 Fitness Calls

Both the mean and the median are smaller when the GA uses the base crossover function, as the mean and median for the base crossover function are 30,758 and 28,648, respectively, and the mean and median for the Favorite Child crossover function are 34,550 and 32,632, respectively. There is some overlap between the 25^{th} and 75^{th} percentile for both crossover functions. The base crossover function has an interquartile range of [25149, 34533] and the Favorite Child crossover has an interquartile range of [29272, 37492].

Even though there is overlap in the interquartiel range, there is no evidence to suggest mean differences between the two crossover functions equal zero, as the p-value, when evaluated at $\alpha = 0.01$ is $2.703e^{-10}$. The 99% confidence interval, $[-5317.324, -2266.850]$ confirms this, and suggests that the the base crossover function calls the fitness function a few thousand times less with a population of 135.

4.b.2 Optimal Difference

Finally, the optimal difference between the base crossover and Favorite Child crossover functions are evaluated. The base crossover has a mean and median optimal difference rate of 1.00 and 0.999, respectively. The minimum optimal difference an population produced was 0.9631. The Favorite Child crossover has a median optimal difference of 1.00 and a mean of 0.9997, with a minimum value of also 0.9631. As the two means are close in value, a two-sample t-test is conducted to see whether the Favorite Child crossover has a significantly higher mean optimal difference.

Using $\alpha = 0.01$, there is not evidence to suggest the difference in means is not equal to zero, as $p = 0.02463$. Hence, the null hypothesis of equal means cannot be rejected. The 99% confidence interval contains zero, as it is $[-0.00159, 0.000109]$.

5 Integer Based Representation Consideration

Using an integer-based representation for solving our knapsack problem has a few benefits. The first one is potentially reducing space needed. Since we are representing a solution as a set $S \subseteq \{1, 2, ..., n\}$, each solution will not need to take up a full n spaces in in our memory, and instead may only needs $m \leq n$ spaces, where m is the number of items in the solution. This is in comparison to an integer representation, which needs n spaces in memory regardless of how many items the solution has. Additionally, an integer based representation is more understandable to humans. Displaying a bit string of n items is not a user-friendly way to display information, but a list of items used is. A potential use case

One detriment of utilizing an integer-based representation over a binary representation is the increased run-time and complexity of performing cross-over and mutation. The simplest mutation for a knapsack problem is adding and removing items randomly, and adding an item is as simple as accessing a random point in the representation array (an $O(1)$ operation) and switching the bit. This implementation also means that the program doesn't need to make a decision on whether or not to add or remove items, it just bit flips. In comparison, an integer representation cannot add and remove items using the same operation, it would need to define separate operations for doing each. Additionally, these operations would either involve searching through the array to identify whether or not an item is in the knapsack ($O(n)$ operation), or would require a second set of unused items the function could reference. Similarly, integer representations complicate crossover operations. Knowing what items a child needs to inherit requires more heuristics and operations than something simple like a single-point crossover or choosing specific bits.