Introduction

The purpose of conducting these experiments is to better understand and acquire a 'feel for' the local search evolutionary algorithm. The objective was to answer the questions of *how* each binary encoding and mutation operator affects the result of the algorithm (the fitness), and *why*. Following this are the analysis of the results of my C++ implementation for each combination of encoding/operator.

Methods

Note: Results are a few example runs formatted as such:

- 1st column = Number of evaluations to reach best fitness
- 2nd column = Number of improvements made to reach best fitness
- 3rd column = Best fitness reached

These are followed by the mean and standard deviation for each column for 1000 runs of the method, and the number of unique optima found for the method.

Plain encoding – Random Jump (0 0)

This performed surprisingly well. My hypothesis is that the large number of evaluations allowed random jumping to fully utilize its strength of the entire search space being its local neighborhood.

```
1455 11 0.992480        col Mean    Stddev
4293 9 0.987311         1 4820.22   2873.37
9866 9 0.998121         2    9.66      2.71
3594 16 0.993120        3    0.99      0.01
671 7 0.992932          Num unique optima found: 302
```

Grey encoding – Random Jump (1 0)

Under Grey encoding, random jump performed essentially the same, as the encoding did not significantly affect the purely random nature of the mutation.

```
6033 14 0.997094        col Mean    Stddev
3903 11 0.991802        1 4923.25   2944.82
4335 9 0.995880         2    9.79      2.95
3794 9 0.995652         3    0.99      0.01
3757 15 0.970609        Num unique optima found: 296
```

Plain encoding – Bit Flip (0 1)

Random bit flipping also performed very well over a large number of evaluations, like random jumping. However, the average number of improvements it made was higher and had greater variation than that of random jumping, as well as ~25% more unique local optima. Since it changes a single bit, the neighborhood is only the possible values resulting from a single changed bit, not the entire search space like random jumping. Therefore, it was able to "hit" more local optima on its way to best fitness.

```
121 7 0.990038          col Mean    Stddev
2432 10 0.998693        1 4826.79   2877.22
2337 12 0.998540        2   14.7       5.26
6352 10 0.997855        3    0.99      0.01
8972 15 0.987535        Num unique optima found: 407
```

Grey encoding – Bit Flip (1 1)

The number of improvements and evaluations made on average was slightly higher than for plain encoding, with slightly less local optima found. This is a result of the Grey encoding affecting the range of the neighborhood, making it easier to skip over local optima, thus taking slightly longer to reach best fit.

```
1001 12 0.989701       col Mean    Stddev
8141 16 0.983858       1 5152.28   2855.5
4818 11 0.998540       2   15.62      5.64
8191 20 0.961796       3    0.99      0.01
2037 11 0.970537       Num unique optima found: 397
```

Plain encoding – Single-Dimension Increment/Decrement (0 2)

This mutation had terrible fitness performance under plain encoding. Due to its limited neighborhood, it was getting easily stuck on local maxima with no way out, which is why there were so many local optima found throughout the runs. (997 out of 1000 runs)

```
7369 81 0.014650       col Mean    Stddev
3956 59 0.012469       1 4898.45   3461.13
2910 56 0.016069       2   96.59     70.4
501 60 0.021563        3    0.09      0.17
968 18 0.064953        Num unique optima found: 997
```

Grey encoding – Single-Dimension Increment/Decrement (1 2)

Under Grey encoding, however, it performed much better. While the size of the neighborhood remained the same, the range of it was greater. The greater mobility afforded by this increased range allowed it to converge much faster, due to not getting stuck on local optima “ruts” as often.

```
5034 23 0.023071       col Mean    Stddev
5445 24 0.015314       1 4876.86   3229.38
2102 18 0.192070       2   23.04     12.58
2038 18 0.017801       3    0.13      0.22
5459 61 0.116256       Num unique optima found: 849
```

Conclusions

Grey encoding seemed to only significantly affect the single-dimension increment/decrement operator. My conclusion was that how much it affected an operator was dependent on how large of a neighborhood the operator had, and how random the operator was. If it had a neighborhood the size of the search space, in the case of random jump, then it hardly affected it, while it significantly affected the small neighborhood of SD Inc/Dec.

For the operators, the neighborhood size mattered, as the operator with the largest neighborhood (random jump) converged on an optimal solution in the smallest number of improvements, while the smallest (sdIncDec) took the most.