GA Report
Vladimir Cuc
Riley Sweeting

SUBMISSION #2 - UPDATE

The main issue with our original genetic algorithm was its extremely slow computation of about 3 seconds per generation. To improve this, we changed the genotype representation so that each candidate has exactly 20 genes instead of varying lengths. Fitness calculation and route plotting stop early once the time limit is exceeded, eliminating unnecessary computations. This allowed us to implement NumPy more effectively with a rectangular, homogeneous population. Additionally, we updated the parent and survivor selection functions to return fitness scores, avoiding redundant fitness calculations. Other improvements included chronologically reordering methods for clarity, condensing and optimizing them with NumPy, and fixing printed displays to show correct fitness scores. These changes improved the runtime significantly, with the entire program now completing in about 3 seconds.

ORIGINAL SUBMISSION

We used a modified integer representation for our genotype, where each ride string represents an integer that can appear multiple times within an individual solution. Our representation also allows chromosomes to have differing numbers of genes to allow for diversity in the number of rides per itinerary.

During population initialization, every time a gene is added to a chromosome, we check if the chromosome exceeds the time limit constraint. Genes stop being added once the time limit is exceeded. The population is constantly kept at 50 individuals for our GA implementation.

After initialization, the fitness scores are computed for the population. As required, the fitness of a candidate solution is the total amount of fun experienced within the time limit, with the constraint that every time a ride is repeated in the itinerary, it's fun score is halved. This discourages the GA from selecting the same ride multiple times.

We implement a tournament selection in our GA. We extract two subsets from the population with replacement, in which each subset contains the same number of individuals as the population. The sets of candidates are pitted against each other

1v1, with the higher fitness candidate winning and being reinserted back into the population. Our selection method essentially eliminates weaker individuals in the population by replacing them with fitter individuals as determined by the tournament.

During the recombination stage, we implement a modified 3 point crossover. Since the chromosomes are of differing gene lengths, we limit the maximum crossover point to be the number of genes ‘n’ of the smaller chromosome. Crossover proceeds as normal up until ‘n’, after which the tail (if any) of the longer parent is copied to both children. Both children have the same number of genes as the longer parent, and the same exact genes as the longer parent from gene ‘n’ to the last gene of the longer parent. If the offspring exceed the time limit, the last rides are removed until the time limit is satisfied.

For our mutation strategy, we employed two distinct methods to maximize mutation efficiency. The first is an *inversion mutation*, chosen to retain potentially effective paths while allowing exploration of neighboring routes. This method inverts a segment within the path, balancing the preservation of successful subpaths with the discovery of new configurations. The second approach is a *random mutation*, where any ride in the path has a chance of being replaced with a completely new ride value drawn from the set of possible ride numbers. We set the total mutation rate to 0.1, divided equally between the two methods. This combination aims to encourage diversity in genotypes by introducing random variations while keeping intact segments that have shown effectiveness.

During mutation, some plans may exceed the allocated time limit, so we trim these plans as necessary to stay within the allowed duration. Conversely, some plans might end up well below the maximum time, leaving room for additional rides. To optimize these underutilized plans, we add random rides to the end of the genotype until the plan is as close as possible to the time limit. This approach ensures that each plan remains feasible while maximizing its potential to include more rides within the allowed timeframe.

During our replacement, we have a design parameter representing the proportion of elite parents that should compose the updated population. An ‘elite’ value of 0.2 means that after survivor selection, the new population should be 20% of the most elite parents and 80% of the fittest offspring.

The most challenging aspect of the implementation was the inability to use numpy for the majority of the methods. Our representation includes chromosomes of

varying gene lengths, and numpy requires a homogenous rectangular shape, meaning a consistent number of features for a data frame. We used built-in list methods, which are less efficient and slower than numpy data manipulation methods.

Intuitively, the optimal magic kingdom itinerary solution should be a circular route around the park, which minimizes the walking distance between each ride. We were surprised to see solutions that appeared to jump between rides with large distances between them. However, when considering that the goal of the GA is to optimize the total fun experienced, and that wait/ride times vary, it is reasonable for the GA to be biased towards the most fun and shortest rides, which explains the diversity of the routes.

Our GA took an average of approx. 2.5 seconds to compute each generation. The maximum fitness per generation takes multiple generations to increase, often upwards of 20 generations. This seems slow, and a potential solution would be to increase the mutation rate design parameter to encourage the GA to create more diverse offspring.

Solution with Maximum Fitness of 107 at Generation 300


```
Here's the plan...
 0 00:00:00 Entrance
 1 01:12:41 Seven Dwarfs Mine Train                      + 8 = 8
 2 01:38:05 Tomorrowland Transit Authority PeopleMover + 9 = 17
 3 02:27:50 Peter Pan's Flight                           + 8 = 25
 4 03:27:20 Space Mountain                               + 9 = 34
 5 04:21:32 TRON Lightcycle Run                          +10 = 44
 6 04:53:22 Astro Orbiter                                + 5 = 49
 7 05:25:40 The Hall of Presidents                       +10 = 59
 8 06:00:36 Pirates of the Caribbean                     + 7 = 66
 9 06:29:45 Monsters Inc. Laugh Floor                    + 7 = 73
10 06:56:33 Walt Disney's Carousel of Progress           + 6 = 79
11 07:22:46 Tomorrowland Speedway                        + 4 = 83
12 07:50:58 Walt Disney's Carousel of Progress           + 6 = 89
13 08:43:18 Haunted Mansion                              + 9 = 98
14 09:19:09 It's a Small World                           + 5 = 103
15 09:47:51 Mickey's PhilharMagic                        + 7 = 110
16 09:55:58 Entrance                                     + 0 = 110
```

**** NOTE: When computing the fitness of an individual, we halved the fun score for a ride each time it is included in the itinerary. Since ‘`Walt Disney's Carousel of Progress`’ is included twice, the second appearance has a halved fun score of ‘3’.**