Anton Ridgway  
CS4623/6623  
May 5, 2014

**Introduction**

The robot navigation problem involves a robot finding a way to navigate a two-dimensional grid-based coordinate space, with obstacles mapped to some portion of the grid. We assume that obstacles can exist at any point except the initial position of the robot. To simplify calculations, I considered that the problem grids under consideration were such that they could be decomposed into smaller square grids, where the robot begins in the upper-left corner, and seeks to reach the lower-right corner. Accordingly, I only considered such square grids.

**Method**

To solve the robot navigation problem, I studied both a genetic algorithm and a simulated annealing algorithm. For the genetic algorithm, I implemented three selection operators, three crossover operators, and three mutation operators. For the simulated annealing algorithm, I made use of the three mutation operators from the GA for my perturbation operators. I compared the performance of the various combinations of operators to determine which worked the best for this problem.

To implement these methods, I first developed a C++ class for my chromosome, able to perform the different mutations I defined, evaluate its own fitness, and, if necessary, return the two-dimensional path that it represents in a chromosome-independent format. (This would allow me, if it proved relevant, to compare multiple types of chromosomes against each other.)

Basic C++ I/O operations are used to set initial parameters when the program is run, and both the genetic algorithm and the simulated annealing algorithm are defined as methods in the main source file. My terminating criteria were, for the genetic algorithm, having passed a given number of generations, and, for the simulated annealing, having passed a certain temperature value; I chose these criteria to give the user better control over how long better answers should be pursued after other terminating criteria (such as apparent stagnation) initially occur. Generally, 10,000 generations, and a 1/20 fraction of the temperature seemed sufficient.

**Chromosome and Fitness**

In defining the chromosome to be used, I sought to build from previous research by Kamran Sedighi, “Local Path-Planning of an Autonomous Mobile Robot Using a Genetic Algorithm,” which made use of a chromosome with two components—a value-based location section and two value-based switch points—and was able to navigate a wide variety of situations by switching between row-wise and column-wise navigation with a fixed number of coordinates. In the work I’ve done this semester, I expanded on this premise and made use of a chromosome that had the same number of switch points as coordinates; that is, the robot navigating can change its perspective from x-monotone to y-monotone, or vice versa, for each pair of coordinates considered.

Each chromosome is initially created by randomly assigning values to each location. Additionally, I used elitism to ensure the survival of good solutions.

The fitness function that I developed for the chromosome was:  
 where:

The fitness falls in the range [0, MAX], and its value is divided by a factor F if the chromosome is infeasible.   
WS is the weight for steps taken.  
WT is the weight for turns made in the path.  
WC is the weight for collisions.  
WE is the weight for chromosomal efficiency.  
These weights fall in [0, 1], and WS + WT + WC + WE = 1.  
WF is the weight for how far we have traveled before each collision, in penalizing collisions.  
WN is the weight of how near the collisions occur to our goal, in penalizing collisions.  
These weights also fall in [0, 1], and WF + WN = 1.  
For my initial experimentation, I used MAX = 1000, F = 2, WS = .1, WT = .2, WC = .6, WE = .1, WF = 0.5, and WN = 0.5.

We consider that the size of the map is Msize, and the size of the chromosome is Csize. Since the starting and ending points are fixed at (0,0) and (Msize-1, Msize-1), we exclude these from the chomosome’s structure, so that Csize = Msize - 2.

NS is the number of steps in the path, and NT is the number of turns.  
Smax is the maximum possible number of steps in any given path, which is equal to the number of grid spaces in the area considered, minus 1 for the beginning position.  
Tmax is the maximum possible number of turns in a path, which from observation is equal to .

C is the set of collisions in a path, and Cmax is set of all possible collisions, one for each grid position except the beginning position in the upper-left (for a totally blocked map). Using these sets are the following measures:  
  
D( ) is the distance function for a set of collisions; to penalize collisions closer to the starting point more heavily, it sums the Manhattan distance from the goal for each collision. D(Cmax) is the sum of distances for each step in the longest possible path, .  
P( ) is the exploration function for a set of collisions; to encourage exploration, it produces a sum in which, for each collision, adds the value (for the number of steps *s* taken to reach that point). P(Cmax) sums the exploration values for every possible step, which is the sum of values from -1 to 0, or .

E is the efficiency index, determined by summing the value for each step in the path. This index encourages fitting as many steps from the path as possible in earlier chromosome indices; since in some cases, the chromosome can use up too many of its indices early on, not leaving it enough to solve the rest of the path successfully, this represents a way to help solve problems that require a high chromosomal efficiency more quickly. Thus, it produces larger values for paths that wait to cover more ground after consuming many indices, while it produces smaller values for paths that move as far as possible early on.

Emax is given by the least efficient path with the same number of steps as the current path; we derive its value by considering that in this worst-case, each index except the last only allows for a single step, contributing the term , while the last index is used for the remainder of the steps, contributing the term . Thus, for the total, we have .

*X* = 0 when |C| = 0, and *X* = 1 when |C| > 0. This is used both to penalize infeasible chromosomes, dividing their fitness by half, and to encourage exploration for infeasible chromosomes, by removing the number of steps taken and the number of turns from consideration until a feasible solution is found. Because the solution efficiency, as measured here, is irrelevant for feasible chromosomes, but sometimes crucial for correcting infeasible cases, it was only considered for the fitness of infeasibles.

For feasible chromosomes, the number of steps taken and the number of turns made also needs to be optimized, so these were considered for *X* = 0.

**Operators**

For my operators, I made use of the following:

* Selection I – Roulette Selection

For my roulette selection, I simply used the fractional fitness of each chromosome to perform a biased random selection for crossover. For any point in the crossover pool, a chromosome with fitness *f* from a set of chromosomes with total fitness *F* is selected with probability . Because this selection is based only on relative fitness, I expected that the best chromosomes would be more likely to dominate the population here, and cause populations to converge more quickly.

* Selection II – Rank Selection

For my rank selection algorithm, I used the relative order of the chromosomes to perform a biased random selection for crossover. At any point in the crossover pool, the ith best chromosome out of N is chosen with probability . Because this selection allowed weaker chromosomes more opportunity to reproduce than they would receive in roulette selection, I expected the greater diversity to improve the overall efficiency of the GA.

* Selection III – Tournament Selection

For my tournament selection algorithm, I compared the fitness of two chromosomes chosen with uniform randomness, and allowed the more fit of the two chromosomes to be selected with a probability *K*, where *K* is a chosen parameter. Because this selection allowed weaker chromosomes more opportunity than they would receive in roulette selection, I expected the greater diversity to improve the overall efficiency of the GA.

* Crossover I – N-Slice Crossover

For the N-slice crossover, I generated N random, unique integers, where  
N < Csize – 1, and each integer stood between 0 and Csize – 2 inclusive. This allowed each integer to stand for a random slice point in the chromosome, dividing in the space between values immediately following the associated index. Between the slice points, the child chromosomes alternate which of their two parents they inherit data from. Because it was more likely to retain consecutive, potentially-constructive sequences of digits than the more fine-grain uniform crossover, I expected the N-slice crossover to perform slightly better than the Uniform case.

* Crossover II – Limited N-Slice

For the Limited N-slice crossover, I treated feasible and infeasible chromosomes separately, in an effort to eliminate collision points as efficiently as possible. For feasible chromosomes, I performed an ordinary N-Slice crossover operation. For infeasible chromosomes, however, I first examined both parents, and identified the index P at which the first collision in the chromosome’s path occurs. Then, I generated the N slice points in the interval [0, P] (or P-1 slice points, if N > P-1). Because it was more focused on the problem points for infeasible chromosomes, I expected this method to perform better than the ordinary N-Slice operator. However, since the normal operator is able to solve issues in the chromosome back-to-front as well as front-to-back, this might also prove less efficient in the end.

* Crossover III – Uniform Crossover

For the uniform crossover, I generated each pair of chromosomes in the child pool by choosing randomly which child would inherit from which parent at every point in the chromosome. Because this chromosome was more random, it had the potential of improving the algorithm efficiency by introducing diversity into the population; yet it was less likely than N-Split to retain profitable sequences, so I expected it to be less effective.

* Mutation I – N-Point Mutation

For the n-point mutation, I simply selected n random, unique points within the chromosome and randomized their values at those points, both on the position-segment and the switch-segment. This method formed a basic baseline against which to compare the other mutation operators.

* Mutation II – Limited N-Point Mutation

Here, as with the Limited N-Slice operator, I only allowed the N points to be chosen before or at the first problem point P (that is, in [0,P]) in infeasible chromosomes, and only used P+1 points if N > P+1. As before, feasible chromosomes were treated normally. I expected this method to represent an improvement on the efficiency of N-Point mutation.

* Mutation III – Gaussian N-Point Mutation

Here, in an effort to improve on the Limited N-Point mutation operator, I did not limit the scope in which the points could be chosen. Instead, I used a Gaussian distribution to determine the placement of the N mutation points probabilistically, with the mean at the first problem point P, and a standard deviation of ().

* Perturbation I – N-Point Perturbation (for Simulated Annealing)

The N-Point perturbation operator was identical to the N-Point mutation operator.

* Perturbation II – Limited N-Point Perturbation (for Simulated Annealing)

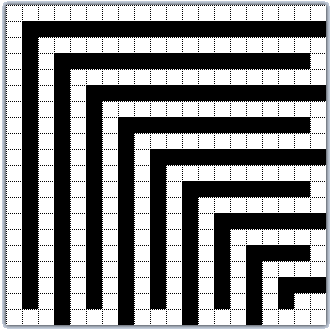
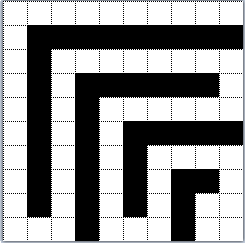
The Limited N-Point perturbation operator was identical to the Limited N-Point mutation operator.

* Perturbation II – Gaussian N-Point Perturbation (for Simulated Annealing)

The Gaussian N-Point perturbation was identical to the Gaussian N-Point mutation operator.

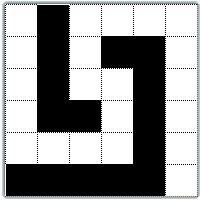
**Datasets**

I began by making use of datasets from the existing literature. In generally, these were easily solvable for the genetic algorithm.

Next, I considered a “worst-case” scenario for the chromosome structure I used, which sought to force the chromosome to switch orientations as often as possible. As it turned out, a minimum of switches were necessary for a map of size N (missing the two since beginning and ending path orientations are not required), for cases that follow the following alternating row-oriented and column-oriented edge-gap pattern:  
  


Although these cases are easily solvable for a simple backtracking algorithm, they are much more difficult for a genetic algorithm, since the gaps appear only on the edges, which tend to be extreme rather than typical cases, and generally must be discovered by mutation rather than crossover if they are not already contained (or some colliding edge-point on the opposite side) in an existing, competitive chromosome.

An additional limitation discovered in dataset design was that some simple structures could thwart the algorithm’s progress altogether by making the only feasible path one that is not possible to represent within our chromosome, since it requires backtracking too close toward the origin:



Since our chromosome must constantly approach the goal in either an x-monotone way or a y-monotone way, this structure, which must consume at least two chromosome points to reach its center, does not leave enough remaining steps to return to the uppermost row.

Structures like these do represent limitations to this chromosome formulation, but its restricted search-space scope compared to a coordinate-based approach allows it to solve most practical cases more quickly. Cases like these can often also be decomposed into smaller cells for individual analysis.

**Experiment, with Parameters Used and Results**

To test the efficiency of my various components, I used two datasets; one more natural dataset, “Searchspace10” from Godugu’s research, and one from the worst-case maps described earlier, of size 20. I began by giving each genetic algorithm configuration five trials with 10,000 generations each, with a population size of 300, a mutation rate of 0.05, and a crossover rate of 0.85. I used K = .8 for tournament selection, and N = 3 for the N-Slice crossover and N-Point mutation. These parameters seemed empirically validated by my earlier experimentation. Then, I gave the simulated annealing algorithm five trials for each perturbation function (using N = 3 for each), with the previously-successful parameters α = 0.98, β = 1.03, T0 = 1.0, and I0 = 3000; the termination temperature was 1/20 of T­0, at 0.05.

For the simulated annealing algorithms, I also tried varying the number of points used for the N-Point operations after each perturbation, choosing a uniformly random number from [2,4].

Additionally, due to the structure of the fitness function, numerical differences between infeasibles’ fitnesses and between feasibles’ fitnesses were very small. Thus, infeasibles generally ranged from 300 (very poor) to 350 (nearly feasible), while feasibles ranged from about 800 (unoptimized) to 830 (optimized). Once a feasible solution was found, the algorithm optimized it very quickly.

Godugu, Searchspace10:  
Optimal value found: 825.455

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Ending Fitness | Number of Generations/ Perturbations to Find | Average Unsuccessful Fitness | Average Unsuccessful  Generations | Average  Successful Generations | Success  Rate (%) |
| SGA | Selection I  Crossover I  Mutation I | **825.455**  **825.455**  **825.455**  **825.455**  **825.455** | **100**  **504**  **89**  **226**  **268** | - | - | 237.4 ± 168.109 | 100 |
|  | Selection II  Crossover I  Mutation I | **825.455**  **825.455**  **825.455**  **825.455**  **825.455** | **398**  **28**  **231**  **13**  **285** | - | - | 191 ± 166.99 | 100 |
|  | Selection III  Crossover I  Mutation I | **825.455**  **825.455**  **825.455**  **825.455**  **825.455** | **36**  **211**  **93**  **979**  **221** | - | - | 308 ± 383.22 | 100 |
|  | Selection I  Crossover II  Mutation I | **825.455**  **825.455**  **825.455**  **825.455**  **825.455** | **273**  **133**  **2154**  **637**  **183** | - | - | 676.0 ± 849.46 | 100 |
|  | Selection II  Crossover II  Mutation I | 347.445  347.445  **825.455**  **825.455**  **825.455** | 230  178  **336**  **444**  **433** | 347.445 | 204.0 ± 36.770 | 404.33 ± 59.43 | 60 |
|  | Selection III  Crossover II  Mutation I | **825.455**  **825.455**  **825.455**  **825.455**  **825.455** | **33**  **1382**  **191**  **61**  **220** | - | - | 377.4 ± 567.33 | 100 |
|  | Selection I  Crossover III  Mutation I | **825.455**  **825.455**  **825.455**  **825.455**  **825.455** | **140**  **415**  **153**  **182**  **252** | - | - | 228.4 ± 112.95 | 100 |
|  | Selection II  Crossover III  Mutation I | 347.445  **825.455**  **825.455**  **825.455**  **825.455** | 607  **18**  **89**  **69**  **40** | 347.445 | 607 | 54.0 ± 31.3156 | 80 |
|  | Selection III  Crossover III  Mutation I | **825.455**  **825.455**  **825.455**  **825.455**  **825.455** | **406**  **310**  **32**  **118**  **3852** | - | - | 943.6 ± 1632.63 | 100 |
|  | Selection I  Crossover I  Mutation II | **825.455**  **825.455**  347.445  **825.455**  **825.455** | **90**  **629**  6521  **3016**  **6365** | 347.445 | 6521 | 3324.3 ± 3053.08 |  |
|  | Selection II  Crossover I  Mutation II | **825.455**  **825.455**  **825.455**  **825.455**  **825.455** | **45**  **289**  **23**  **1514**  **23** | - | - | 378.8 ± 644.47 | 100 |
|  | Selection III  Crossover I  Mutation II | **825.455**  **825.455**  **825.455**  **825.455**  **825.455** | **50**  **65**  **81**  **189**  **293** | - | - | 135.6 ± 103.58 | 100 |
|  | Selection I  Crossover II  Mutation II | **825.455**  **825.455**  **825.455**  **825.455**  **825.455** | **1879**  **250**  **1528**  **1535**  **1446** | - | - | 1327.6 ± 624.99 | 100 |
|  | Selection II  Crossover II  Mutation II | **825.455**  347.445  **825.455**  **825.455**  **825.455** | **63**  227  **1204**  **389**  **74** | 347.445 | 227 | 432.5 ± 536.08 | 80 |
|  | Selection III  Crossover II  Mutation II | **825.455**  **825.455**  **825.455**  **825.455**  **825.455** | **148**  **710**  **3217**  **43**  **1104** | - | - | 1044.40 ± 1288.59 | 100 |
|  | Selection I  Crossover III  Mutation II | **825.455**  **825.455**  **825.455**  **825.455**  **825.455** | **95**  **292**  **545**  **141**  **305** | - | - | 275.60 ± 176.38 | 100 |
|  | Selection II  Crossover III  Mutation II | **825.455**  **825.455**  **825.455**  **825.455**  345.494 | **15**  **9103**  **27**  **2078**  4698 | 345.494 | 4698 | 2805.75 ± 4308.70 | 80 |
|  | Selection III  Crossover III  Mutation II | 347.445  **825.455**  **825.455**  347.445  **825.455** | 128  **40**  **18**  494  **58** | 347.445 | 311.0 ± 258.80 | 38.67 ± 20.03 | 60 |
|  | Selection I  Crossover I  Mutation III | **825.455**  **825.455**  **825.455**  **825.455**  **825.455** | **82**  **775**  **1207**  **61**  **150** | - | - | 455.0 ± 513.64 | 100 |
|  | Selection II  Crossover I  Mutation III | **825.455**  **825.455**  **825.455**  **825.455**  **825.455** | **28**  **29**  **17**  **11**  **179** | - | - | 52.80 ± 70.95 | 100 |
|  | Selection III  Crossover I  Mutation III | **825.455**  **825.455**  **825.455**  **825.455**  **825.455** | **86**  **44**  **3149**  **24**  **70** | - | - | 734.6 ± 1354.37 | 100 |
|  | Selection I  Crossover II  Mutation III | **825.455**  **825.455**  **825.455**  **825.455**  **825.455** | **97**  **922**  **328**  **220**  **1202** | - | - | 553.8 ± 481.35 | 100 |
|  | Selection II  Crossover II  Mutation III | 347.445  **825.455**  **825.455**  **825.455**  **825.455** | 810  **31**  **66**  **60**  **52** | 347.445 | 810.0 | 52.25 ± 15.28 | 80 |
|  | Selection III  Crossover II  Mutation III | **825.455**  **825.455**  **825.455**  **825.455**  **825.455** | **177**  **64**  **43**  **41**  **26** | - | - | 70.20 ± 61.22 | 100 |
|  | Selection I  Crossover III  Mutation III | **825.455**  **825.455**  **825.455**  **825.455**  **825.455** | **250**  **86**  **361**  **332**  **120** | - | - | 229.80 ± 123.29 | 100 |
|  | Selection II  Crossover III  Mutation III | **825.455**  **825.455**  **825.455**  **825.455**  **825.455** | **69**  **84**  **41**  **223**  **17** | - | - | 86.80 ± 80.38 | 100 |
|  | Selection III  Crossover III  Mutation III | **825.455**  **825.455**  **825.455**  **825.455**  **825.455** | **77**  **34**  **141**  **38**  **27** | - | - | 63.40 ± 47.54 | 100 |
| SA | Perturbation I  α = 0.98  β = 1.03  T0 = 1.0  I0 = 3000 | **825.455**  **825.455**  **825.455**  **825.455**  **825.455** | **6435**  **93**  **685803**  **167171**  **4826** | - | - | 172,865.60 ± 295,348.96 | 100 |
|  | Perturbation II  α = 0.98  β = 1.03  T0 = 1.0  I0 = ­3000 | **825.455**  **825.455**  **825.455**  **825.455**  **825.455** | **7056**  **1128**  **1411**  **5516**  **1220** | - | - | 3,266.20 ± 2,811.80 | 100 |
|  | Perturbation III  α = 0.98  β = 1.03  T0 = 1.0  I0 = ­3000 | **825.455**  **825.455**  **825.455**  **825.455**  **825.455** | **252796**  **36832**  **192362**  **35869**  **193545** | - | - | 142,280.80 ± 99,740.51 | 100 |

In these trials, the combinations that seemed to perform the best consistently were:

* Rank Selection, N-Slice Crossover, and Gaussian N-Point Perturbation:
  + 52.80 ± 70.95 generations
* Tournament Selection, Uniform Crossover, and Gaussian N-Point Perturbation:
  + 63.40 ± 47.54 generations
* Tournament Selection, Limited N-Slice Crossover, and Gaussian N-Point Perturbation:
  + 70.20 ± 61.22 generations

In general, methods that employed Rank Selection, Limited N-Slice Crossover, or Limited N-Point Mutation tended to produce relatively worse results.

Of the simulated annealing algorithms, the Limited N-Point perturbation performed the most efficiently by a large margin.

In working with a size-20 worst-case graph, I altered my algorithm parameters slightly, to optimize performance as much as possible. Here, it often proved more effective to use 2 mutation points instead of 3, and it seemed prudent in many cases to raise the generation limit to a 20,000 maximum. Finally, I rebalanced the WF and WN weights to at WF = 0.6 and WN = 0.4 to encourage exploring more to help the algorithms find their way around the corners, by favoring exploration over nearness to the goal. The other parameters remained the same as before.

Worst-case configuration, size 20:   
Optimal value found: **825.376**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Ending  Fitness | Number of Generations/ Perturbations to Find | Average Unsuccessful Fitness | Average Unsuccessful  Generations | Average  Successful Generations | Success Rate  (%) |
| SGA | Selection I  Crossover I  Mutation I | **825.376**  **825.376**  **825.376**  **825.376**  **825.376** | **3629**  **6787**  **10401**  **4242**  **4318** | - | - | 5875.4 ± 2804.10 | 100 |
|  | Selection II  Crossover I  Mutation I | 348.721  **825.376**  348.653  348.788  **825.376** | 1372  **829**  1976  4421  **2234** | 348.72 ± 0.07 | 2589.67 ± 1614.48 | 1531.50 ± 993.49 | 40 |
|  | Selection III  Crossover I  Mutation I | **825.376**  **825.376**  **825.376**  348.381  348.162 | **3787**  **2909**  **1230**  1307  425 | 348.27 ± 0.15 | 866.00 ± 623.67 | 2642.00 ± 1299.24 | 60 |
|  | Selection I  Crossover II  Mutation I | 348.788  **825.376**  **825.376**  **825.376**  349.068 | 10175  **11918**  **7057**  **5639**  8730 | 348.93 ± 0.20 | 9452.50 ± 1021.77 | 8204.67 ± 3293.07 | 60 |
|  | Selection II  Crossover II  Mutation I | 347.94  349.083  349.068  **825.376**  347.876 | 5809  3207  3590  **6032**  5715 | 348.49 ± 0.67 | 4580.25 ± 1374.03 | 6032 | 20 |
|  | Selection III  Crossover II  Mutation I | 349.068  348.721  347.909  348.788  348.665 | 4259  4547  1191  3322  16617 | 348.63 ± 0.43 | 5987.2 ± 6086.06 | - | 0 |
|  | Selection I  Crossover III  Mutation I | 348.878  348.881  349.034  **825.376**  **825.376** | 1007  6630  5005  **3949**  **1588** | 348.93 ± 0.09 | 4214.00 ± 2893.75 | 2768.50 ± 1669.48 | 20 |
|  | Selection II  Crossover III  Mutation I | 349.231  349.231  **825.376**  349.147  **825.376** | 3932  3614  **18504**  1527  **1700** | 349.20 ± 0.05 | 3024.33 ± 1306.44 | 10102.00 ± 11882.22 | 40 |
|  | Selection III  Crossover III  Mutation I | **825.376**  **825.376**  **825.376**  348.878  **825.376** | **4941**  **6601**  **5164**  3062  **2157** | 348.878 | 3062 | 4715.75 ± 1857.69 | 80 |
|  | Selection I  Crossover I  Mutation II | 347.486  344.734  345.773  347.412  345.352 | 8179  19631  10470  9284  14050 | 346.15 ± 1.24 | 12322.80 ± 4643.04 | - | 0 |
|  | Selection II  Crossover I  Mutation II | 346.711  344.819  345.708  347.887  346.809 | 15049  14497  4114  8058  6419 | 286.39 ± 133.99 | 9627.40 ± 4905.64 | - | 0 |
|  | Selection III  Crossover I  Mutation II | 348.513  346.582  **825.376**  348.933  348.999 | 6905  8341  **4636**  9495  9306 | 348.26 ± 1.14 | 8511.75 ± 1184.40 | 4636 | 20 |
|  | Selection I  Crossover II  Mutation II | 335.821  347.388  334.119  349.231  **825.376** | 10172  15882  2246  16677  **14640** | 341.64 ± 7.77 | 11244.25 ± 6661.87 | 14640 | 20 |
|  | Selection II  Crossover II  Mutation II | 330.078  322.55  334.544  329.185  339.812 | 805  15729  308  3346  1566 | 331.23 ± 6.43 | 4350.80 ± 6464.23 | - | 0 |
|  | Selection III  Crossover II  Mutation II | 349.231  348.71  344.848  349.231  349.231 | 9835  8244  10581  17302  1989 | 348.25 ± 1.92 | 9590.2 ± 54.60 | - | 0 |
|  | Selection I  Crossover III  Mutation II | 347.682  **825.376**  346.817  348.602  348.422 | 15602  **19049**  7886  18731  9908 | 347.88 ± 0.81 | 13031.75 ± 5010.79 | 19049 | 20 |
|  | Selection II  Crossover III  Mutation II | 346.717  347.328  347.296  348.352  345.179 | 461  6742  6956  709  5077 | 346.97 ± 1.16 | 3989.00 ± 3192.61 | - | 0 |
|  | Selection III  Crossover III  Mutation II | 349.192  348.319  **825.376 825.376**  348.484 | 15368  8180  **8433**  **10698**  453 | 348.67 ± 0.46 | 8000.33 ± 7459.12 | 9565.5 ± 1601.60 | 40 |
|  | Selection I  Crossover I  Mutation III | **825.376**  **825.376**  343.679  348.202  349.231 | **12957**  **18066**  8326  17453  17050 | 347.037 ± 2.95 | 14276.33 ± 5157.08 | 15511.50 ± 3612.61 | 40 |
|  | Selection II  Crossover I  Mutation III | 349.068  **825.376**  **825.376**  349.231  **825.376** | 2462  **539**  **440**  8323  **814** | 349.15 ± 0.12 | 5392.50 ± 4144.35 | 597.67 ± 193.78 | 60 |
|  | Selection III  Crossover I  Mutation III | **825.376**  **825.376**  **825.376**  **825.376**  **825.376** | **1876**  **688**  **593**  **1396**  **2217** | - | - | 1354 ± 714.44 | 100 |
|  | Selection I  Crossover II  Mutation III | 346.827  346.925  344.949  347.357  349.065 | 18795  9417  18868  16363  18151 | 347.025 ± 1.47 | 16318.80 ± 3988.22 | - | 0 |
|  | Selection II  Crossover II  Mutation III | **825.376**  **825.376**  **825.376**  **825.376**  **825.376** | **3765**  **2086**  **418**  **2229**  **1030** | - | - | 1905.60 ± 1282.00 | 100 |
|  | Selection III  Crossover II  Mutation III | **825.376**  **825.376**  348.583  **825.376**  **825.376** | **1271**  **913**  1583  **18808**  **563** | 348.583 | 1583 | 5388.75 ± 8950.83 | 80 |
|  | Selection I  Crossover III  Mutation III | 349.049  349.231  346.865  348.209  347.93 | 5291  2695  746  15807  19881 | 348.257 ± 0.95 | 8884.00 ± 8460.24 | - | 0 |
|  | Selection II  Crossover III  Mutation III | 348.63  348.582  348.216  349.034  349.068 | 1820  10294  1297  842  2192 | 348.706 ± 0.35351 | 3289.00 ± 3949.28 | - | 0 |
|  | Selection III  Crossover III  Mutation III | 349.231  **825.376**  **825.376**  348.463  **825.376** | 928  **1201**  **774**  1573  **510** | 348.847 ± 0.543 | 1250.50 ± 456.08 | 828.33 ± 348.69 | 60 |
| SA | Perturbation I  α = 0.98  β = 1.03  T0 = 1.0  I0 = 3000 | **825.376**  **825.376**  **825.376**  **825.376**  **825.376** | **1611959**  **696421**  **1036073**  **1516599**  **1246098** | - | - | 1,221,430.00 ± 370,890.08 | 100 |
|  | Perturbation II  α = 0.98  β = 1.03  T0 = 1.0  I0 = ­3000 | 349.169  **825.376**  349.033  349.181  349.22 | 4264948  **7910951**  4027478  1723622  4745897 | 349.151 ± 0.081 | 3,690,486.25 ± 1344868.84 | 7,910,951 | 20 |
|  | Perturbation III  α = 0.98  β = 1.03  T0 = 1.0  I0 = ­3000 | 304.509  314.875  299.079  305.537  321.338 | 7464800  7534585  1573  7829197  2248379 | 309.068 ± 8.907 | 5,015,706.80 ± 3,642,052.06 | - | 0 |

For these trials, many of the combinations were not able to find a feasible path in any case; of these, some converged early and stagnated, unable to find the path to proceed—these produced their best guesses in a low number of generations but only solved the problem infrequently. This suggests that the algorithm may have still been considering other options, but did not retain enough alternative solution strains to find a way to solve a sticking point. The combination of Rank Selection, Uniform Crossover, and Limited N-Point Mutation exhibited this sort of behavior.

Others continued improving steadily, but did so too slowly to finish in the number of generations needed—these too solved the problem infrequently, but took a larger number of generations to do so, and generally produced answers that were closer to feasible. This suggests that the components were able to solve problems, but were inefficient in doing so because their scope was either too restrictive or too broad. The combination of Roulette Selection, Limited N-Point Crossover, and Gaussian N-Point Mutation, for example, showed this behavior.

Still others either solved the problem or failed quickly; the combination of Tournament Selection, Uniform Crossover, and Gaussian N-Point Mutation showed this behavior. This behavior suggests that perhaps the chromosome pool converged too soon to the best solution so-far, and did not retain enough diversity to solve its problems. These combinations might prove competitive compared to other strategies if they were completely restarted once stagnation occurred, but this is an additional inefficiency that may be unnecessary.

Overall, the most successful operator combinations proved to be:

* Tournament Selection, N-Slice Crossover, and Gaussian N-Point Mutation
  + 1354 ± 714.44 generations
* Rank Selection, Limited N-Slice Crossover, and Gaussian N-Point Mutation
  + 1905.60 ± 1282.00 generations
* Roulette Selection, N-Slice Crossover, and N-Point Mutation
  + 5875.4 ± 2804.10 generations

For simulated annealing, the N-Point Perturbation performed the best of all (at 1,221,430 average perturbations), and while the Limited N-Point steadily evolved nearly feasible chromosomes in every case, it was not able to find a feasible solution in the time given.

**Conclusions**

In keeping with our prior hypotheses, Tournament Selection generally performed much better than Roulette Selection; however, Rank Selection generally performed more poorly. Each of the crossovers seemed to perform decently, though surprisingly, Uniform Crossover often performed better than N-Slice Crossover, and more consistently than Limited N-Slice. Of the mutation methods, Limited N-Point Mutation tended to perform worse than the others, while Gaussian N-Point Mutation often proved the best.

From the data that I collected, it seems that the combination of Tournament Selection and the Gaussian N-Slice crossover proved the most reliable, while the crossover varied depending on the dataset at hand. The N-Slice Crossover certainly performed the best in testing with worst-case scenarios, but each proved effective for a more general case. N-Point Perturbation performed the most reliably for simulated annealing, though Limited N-Point Perturbation was extremely effective in some cases. In general, simulated annealing also seemed able to solve datasets more quickly than GA configurations, suggesting that they are the method of choice for this solution method.

**Future Work**

Since stagnation was a major problem in dealing with the worst-case scenario, it might prove fruitful to investigate using some kind of kick function to jump-start more extreme exploration when it seems prudent.

A possible alternative mutation operator that I have not yet investigated might involve inserting or deleting random values at indices in the chromosome, moving the values at surrounding indices outward or inward accordingly; often stagnated solutions in the worst-case testing were caused by chromosomes that expended their coordinate-points too soon, and they might hit on a solution if indices that waste time are removed, or a new point is introduced.

Additionally, it might prove helpful to develop Alternating Limited N-Point operators, which switch between only considering indices before the first collision point, and only considering those after.

**References**

Godugu, Jagruthi. “Development of a Benchmark for Robot Path Planning”. 2004.

Sedighi, Kamran. “Local Path-Planning of an Autonomous Mobile Robot Using a Genetic Algorithm.” 2004.