A Genetic Algorithm Approach to Solving Optimal Projectile Trajectories in a Simulated Environment

Anthony Hills
Department of Physics, University of Surrey, Guildford, GU2 7XH, UK
E-mail address: ah00446@surrey.ac.uk, urn: 6317482

Abstract **–** The demand for solving complex, potentially chaotic, kinematics problems is present within a wide array of research, from ballistic missile testing, to predicting paths of particles passing through viscous fluids. In this regard, this paper aims to present an adaptive approach to the solution of an analytically solvable kinematics problem.

A genetic algorithm was applied to find the optimal orientation of a projectile's initial velocity vector, in order to optimize its accuracy in hitting a target. The target was stationary and placed within a simulated environment involving gravity and air resistance.

The obtained results indicate a good performance of the genetic algorithm when compared to conventional methods.

Supervisor: Dr. Maxime Delorme

I. INTRODUCTION

Genetic algorithms (G.A.) are used as optimization tools where analytical solutions are difficult to determine. G.A. apply an evolutionary approach to inductive learning, and have been successfully applied to problems that are difficult to solve using conventional techniques such as the travelling salesman problem, network routing problems, and financial marketing.

In this experiment, the G.A. begins with a random population of arrays containing the initial speeds and angles at which a projectile is launched. These values are fed into a function which rates the effectiveness of the current solution, by determining its final, relative, horizontal, Euclidean distance to a stationary target, corresponding to its resulting '*fitness*'.

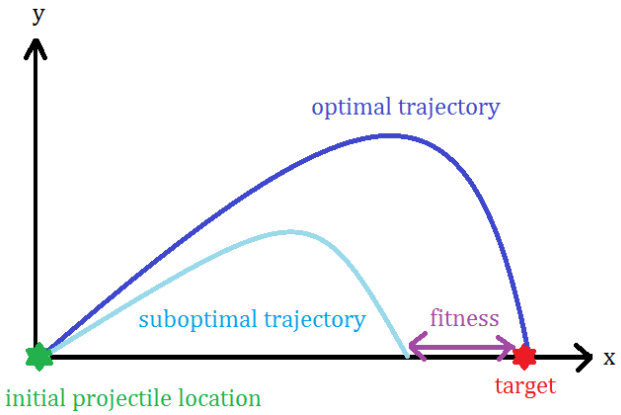

Figure 1. An illustration of the kinematics problem and the fitness function to be minimized (i.e. optimized).

Depending on the fitness of each attempt, the G.A. then applies techniques imitating biological reproduction, such as crossover and mutation, effectively breeding the successful attempts and weeding out the weaker ones. In this regard, the G.A. closely mimics the behavior exhibited by natural selection. Applying these methods to the initial population, the solutions are iteratively improved over time.

The performance of the genetic algorithm can be measured by the diversity and rate of convergence of the solutions, which depend highly on choosing the right input parameters. Maintaining diversity of solutions is important, as it improves the population's ability to adapt to a given problem.

II. THEORETICAL BACKGROUND

The programing language used to write this program was Fortran 90, and the compiler used: NAG Fortran Release 6.1.

a. Supervised genetic learning

G.A.s start with an initial population of individuals, and evaluate the fitness function of them. The fitness of the individuals is then used as training data for the algorithm to iteratively improve upon itself, supervised with the ultimate goal of optimizing each generation's overall fitness.

The final result of a supervised genetic learning session is a set of population elements that best represent the training data, in this case: a projectile trajectory that lands precisely at a user-specified target location.

b. Representation of the data in a G.A.

In a G.A. we represent the data of the problem as virtual chromosomes, individuals and populations of solutions.

Each individual is a specific solution to the optimization problem, and the population is the set of all individuals at a specific iteration of the algorithm. The full population represents a certain coverage of the parameter space, and the objective of the G.A. is to make all of these solutions evolve towards the global optimum of the fitness function. Each parameter of the problem will be represented as a *chromosome* of the individual.

This paper defines the parameters of the individuals' chromosomes to be binary: comprising of a value for the initial speed of the projectile, and another parameter consisting of the angle at which it is launched. These two parameters both form the chromosome, which in turn forms the solution. Each chromosome is a representation of the initial velocity vector of the projectile used to launch itself towards the target.

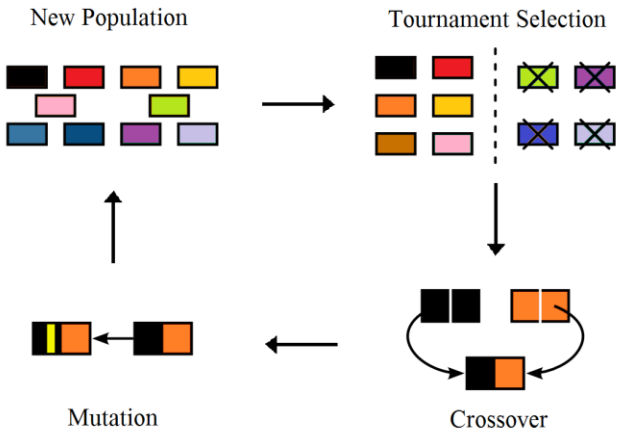

Figure 2. A representation of the flow of data, as binary chromosomes.

c. G.A. architecture

An overview of the G.A. is provided in the flow chart in figure 1, illustrating the main processes by which the G.A. takes advantage of to reach its optimized solution.

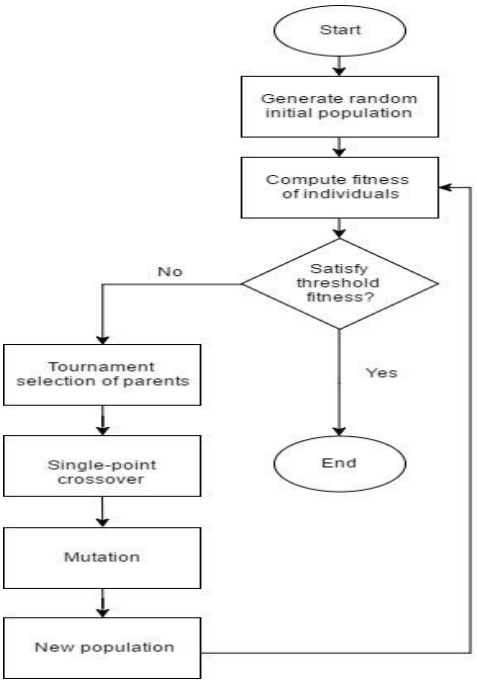

Figure 3. A flow chart diagram illustrating the main processes comprising the G.A.

The main outputs of the program are: the minimum, maximum, and mean fitness of the final population for each generation; the convergence point at which the mean fitness first exceeds a threshold fitness; and the final diversity of the population, calculated as the standard deviation of each individual's horizontal range.

d. Random initialization of population

The program begins by generating a random population of velocity vectors of initial speeds ranging from 10 ms^{-1} to 250 ms^{-1}. The angles associated to these initial speeds ranged from 0 to $\pi/2$ radians.

The population was contained within an array of 1000 individuals, each containing its corresponding initial speed and angle.

Having too large of a population size results in a long epoch time, restricting how many chances each individual has to explore its solution space. However, if the population is too small, then each generation doesn't get a good coverage of the space, reducing the efficiency of the G.A.

It was found that a population size of 1000 individuals provided a good compromise between the two extremes.

e. Fitness

The fitness of an individual was computed as its corresponding final horizontal, Euclidian, landing distance

between itself and the target after a simulation of the individual's velocity vector was launched. The target was located at a position along the x-axis (i.e. at a position of y = 0).

Concretely, the value for fitness was computed using equation 1 as follows:

$$f = |t_x - p_x| \qquad (1)$$

Where:

f = fitness value
t_x = location of fixed target in the x-axis
p_x = landing location of the projectile in the x-axis

Thus, in this application of the G.A., it can be thought of that a lower fitness corresponds to a solution that is close to hitting the target. A perfect fitness would correspond to 0, hitting the target with infinite precision. On the other hand, a large fitness would correspond to a velocity vector that would miss the target by a large distance.

During the reproduction phase of the G.A., individuals with a lower value in fitness are favored over individuals of a larger value.

f. Elitism

With each iteration of the G.A.'s generational loop, the best individual from each generation, *n*, is systematically copied to the next generation, *n* + 1, without any modification. This is *elitism* at its finest, and ensures that the best fitness will never decrease from one generation to the next.

g. Tournament style selection

Individuals in the old population are selected as parents for reproduction according to their fitness.

The pool of individuals must conserve some diversity to be able to mix different solutions, to ensure the program does not get stuck in local optima, and to ultimately continuously improve the average fitness until the global solution is found.

h. Single-point crossover

Once individuals are selected as parents for the next generation some of them exchange part of their genetic information to create two new individuals: their offspring.

i. Mutation

In real evolution, genetic material can be changed randomly by improper reproduction or other deformations of genes, such as by gamma radiation. In genetic algorithms, mutation is defined as a random deformation of one of the chromosomes, with a certain probability.

Due to our mutation operator, some individuals will experience a small random change in one of their chromosome. Mutation ensures that genetic diversity is preserved, and as an effect, local optima are avoided.

If the mutation rate is too high, then the G.A. risks individuals "jumping" over a solution they were already close too. However, if it is too low then they will all get stuck in local minima.

To address these potential problems, the mutation operator was defined to be large at a low generation number, and to decrease exponentially in magnitude as the algorithm approaches its global optima. This was done so that at the start of the program, the G.A. could have greater freedom in searching the solution space, producing a population of high genetic diversity. As the G.A. converges towards the solution, the mutation operator is decreased in its effect, so that the G.A. avoids wasting time oscillating about the global optima.

This was done to improve the G.A.s rapidity in converging at the solution.

Concretely, the formulae for mutation were computed using the equations as follows:

$$\Delta\theta = e^{-G/15} * (r - 0.5) \qquad (2)$$
$$\Delta v_0 = e^{-G/200} * (r - 0.5) \qquad (3)$$

Where:

$\Delta\theta$ = mutation in angle
Δv_0 = mutation in initial speed
e = exponential function
G = current generation number
r = a randomly generated real number between 1 and 0

III. NUMERICAL DETAILS

The simulated environment that the projectile was launched in involved the following parameters:

g = 9.81 ms^{-2}
F = 0.05 N

Where:

g = gravitational force
F = frictional force due to air resistance

The final diversity of the population was evaluated from the standard deviation of all the individuals within the final

population that had a solution that converged to the solution.

The formula used to evaluate the final diversity is as follows:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}} \quad (4)$$

Where:

σ = final diversity of population
n = population size
i = the i^{th} individual within the population array
x_i = the fitness of the i^{th} individual
$\bar{x}$ = the mean fitness of the entire population

The accuracy of our G.A. to determine the convergence generation for a particular set of parameters was obtained using the standard deviation of the generation of convergence, using a sample size of 20. This is further illustrated in Appendix B at the end of this paper.

The default parameters used in this G.A. are illustrated in Table 1 below:

Mutation Chance	Crossover Chance	Population Size	Tourn-ament Size	Threshold Fitness (meters)
0.33	0.33	1000	4	0.001

TABLE I. The default parameters used in the genetic algorithm.

IV. RESULTS

The results were obtained using an Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz, having 7 cores.

Figure 4 is an illustration of the trajectory of the projectile within the simulated environment, using the converged solution that the G.A. found at the 20^{th} generation, as described in Figure 5.

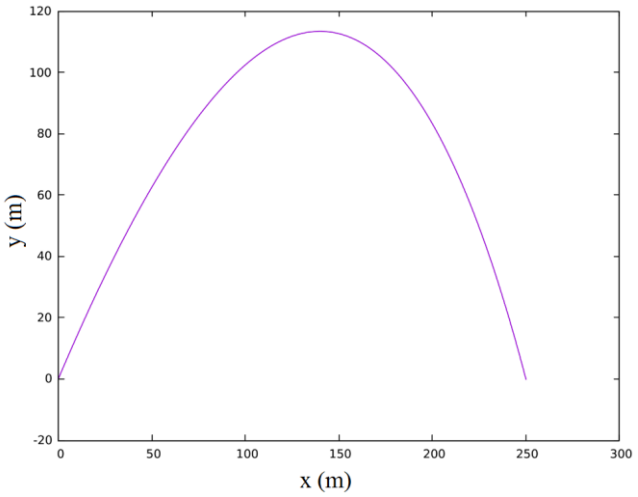

Figure 4. The trajectory evaluated using the fittest individual of the final generation, for a target location of 250 m.

Using the solution illustrated in figures 4 and 5, the G.A. converged at an angle of θ = 0.96421 radians, and an initial velocity of v_0 = 57.491238 ms^{-1}. Using this solution, the projectile landed at 250.000632 m along the x-axis. This corresponds to a distance that was 0.0006 m away from the target. This result is within the specified threshold fitness.

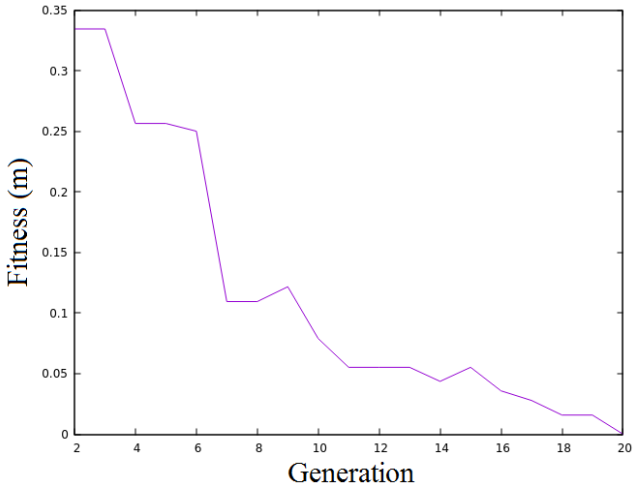

Figure 5. The distance to the target (i.e. fitness) in meters, using the fittest individual of each generation.

Figure 5 illustrates a convergence at the solution at the 20^{th} generation (Figure 5), indicating a fairly smooth reduction in fitness as it approaches the solution.

The results illustrated in Figures 4 and 5 were obtained using the default parameters (discussed in Table 1) for the G.A.

Below in figures 6 and 7 are results obtained by varying the crossover and mutation probabilities, and measuring the number of generations required for the G.A. to converge at a solution. These results illustrate an attempt at tuning the G.A. parameters for crossover probability and mutation probability.

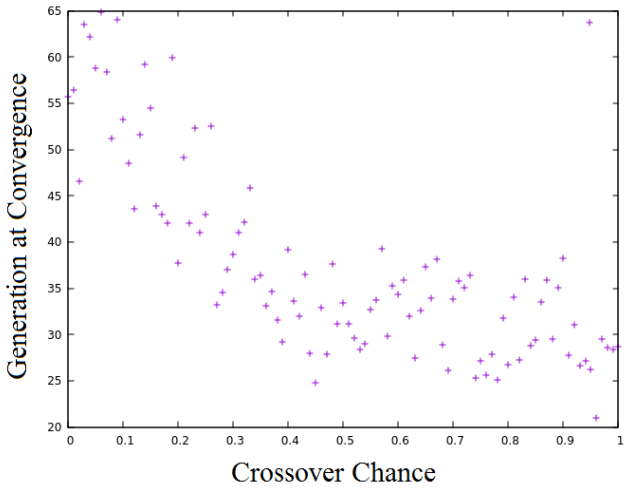

Figure 6. Number of generations until solution converged at, against crossover probability.

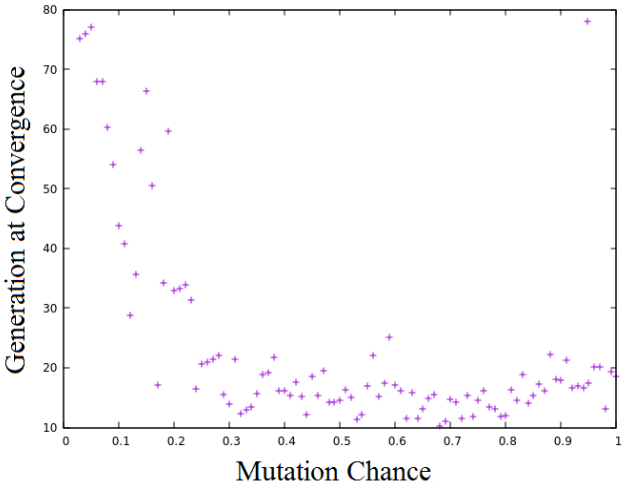

Figure 7. Number of generations until solution converged at, against mutation probability.

V. SUMMARY AND CONCLUSIONS

It was found that the default parameters of 33% for both mutation probability and crossover chance, with a population size of 1000 individuals proved to be suitable in solving this kinematics problem; converging at a solution at the 20th generation, landing at an accuracy within the specified fitness, for a target located at 250 m. This version incorporated all subroutines mentioned, including elitism.

In our G.A., selection is clearly an important genetic operator. However, our results remain inconclusive regarding the importance of crossover versus mutation. Mutation seems to be only necessary to ensure that potential solutions are not lost, while crossover in a largely uniform population (i.e. a low genetic diversity) only appears to propagate innovations originally found by mutation. In a non-uniform population, crossover seems to be nearly equivalent to a very large mutation. Our results indicate that the G.A. rapidly locates solutions for crossover and mutation rates of above 30%.

For future work, it may be worthwhile to combine the G.A. with other metaheuristics and machine learning algorithms, such as the "hill climbing" algorithm.[15] Alternating the G.A. with other metaheuristics to form a hybridized algorithm could further improve the robustness of this technique when applied to kinematics problems of even larger search spaces.

ACKNOWLEDGEMENTS

I would like to thank my supervising academic, Dr. Maxime Delorme who provided insight and expertise that greatly assisted this research.

I would also like to thank Dr. Richard Sear, and show my gratitude to Professor Mark Gieles, for their comments on an earlier version of the manuscript.

Any errors are my own and should not tarnish the reputations of these esteemed persons.

APPENDIX A

Here is a copy of the code that was written to obtain the final results in the G.A.

https://github.com/INASIC/genetic_algorithm.f90

[1] Eiben, A. E. et al (1994). "Genetic algorithms with multi-parent recombination". PPSN III: Proceedings of the International Conference on Evolutionary Computation. The Third Conference on Parallel Problem Solving from Nature: 78–87. ISBN 3-540-58484-6.

[2] Ting, Chuan-Kang (2005). "On the Mean Convergence Time of Multiparent Genetic Algorithms without Selection". Advances in Artificial Life: 403–412. ISBN 978-3-540-28848-0.

[3] Akbari, Ziarati (2010). "A multilevel evolutionary algorithm for optimizing numerical functions" IJIEC 2 (2011): 419–43

[4] [G. Harik. Learning linkage to efficiently solve problems of bounded difficulty using genetic algorithms. PhD thesis, Dept. Computer Science, University of Michigan, Ann Arbour, 1997

[5] Goldberg, David E. (1989). Genetic Algorithms in Search Optimization and Machine Learning. Addison Wesley. P.41 ISBN0-201-15767-5.

[6] Wolpert, D.H., Macready, W.G., 1995. No Free Lunch Theorems for Optimisation. Santa Fe Institute, SFI-TR-05-010, Santa Fe.

[7] Doctoral Dissertation, University of Michigan, Dissertation Abstracts International 28(12) Cavicchio: Adaptive search using simulated evolution, unpublished doctoral dissertation, University of Michigan, Ann Arbor

[8] Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning, 1989, Addison-Wesley Publishing Co.

[9] Holland: Adaptation in Natural and Artificial Systems, Ann Arbor, Mi: University of Michigan Press, 1975

[10] De Jong: An Analysis of the behavior of a class of genetic adaptive systems, Doctoral Dissertation, University of Michigan, Dissertation Abstracts International 36(10)

[11] S. Baluja, “Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning,” Carnegie Mellon Univ., Pittsburgh, PA, Tech. Rep. CMUCS-94-163, 1994.

[12] M. Hohfeld and G. Rudolph, “Toward a theory of population-based ¨ incremental learning,” in Proc. 4th Int. Conf. Evolutionary Computation, T. Back, Ed. Piscataway, NJ: IEEE Press, 1997, pp. 1–5.

[13] D. Thierens, and D. E. Goldberg, “Mixing in genetic algorithms,” in Proc. 5th Int. Conf. Genetic Algorithms, S. Forrest, Ed. San Mateo, CA: Morgan Kaufmann, 1993, pp. 38–45.

[14] G. Syswerda, “Simulated crossover in genetic algorithms,” in Foundations of Genetic Algorithms 2, L. D. Whitley, Ed. San Mateo, CA: Morgan Kaufmann, 1993, pp. 239–255.

[15] Kurt A. Hacker, John Eddy and Kemper E. Lewis, 2002, “Efficient Global optimization using Hybrid Genetic Algorithms”, American Institute of Aeronautics and Astronautics, Inc

[16] Milenic Mitchell, john H. Holland, 1994,”When will a genetic algorithm outperform Hill climbing”