Journal of
Emerging Investigators

Optimizing Interplanetary Travel Using a Genetic Algorithm

Anusha Murali[1], Evan Chandran[2], and Susan Seagroves[1]

[1]Bishop Brady High School, Concord, NH

[2]Phillips Exeter Academy, Exeter, NH

Summary

Humanity has long been fascinated with celestial bodies and space travel. In the current work, we investigate a novel approach to efficiently travel between planets. Specifically, we use a genetic algorithm to find a near optimal path between planets that are in a circular motion with different angular velocities under the gravitational attraction of a star. We develop mathematical expressions to find both the travel distance and the trajectory angle of the spaceship for the orbiting planets. Using derived analytical expressions and a carefully chosen bimodal fitness function, we demonstrate that our genetic algorithm rapidly converges to a near optimal solution for the shortest path between the planets. The experimental results show that our genetic algorithm converges consistently faster than an algorithm based on a unimodal fitness function. Furthermore, the experimental results indicate that our algorithm is many orders of magnitude faster than an enumeration-based technique, while providing nearly as accurate results. We envision that, in the future, the results of our work could be used to program a space probe to efficiently travel between the faraway planets of a newly discovered solar system, and to collect scientific data from deep space that could provide answers to profound questions about our universe. In addition, our results can be put into use immediately for mining asteroids for natural resources that are rare on Earth, and to eliminate space junk that poses a danger to space travel.

Received: July 22, 2018; **Accepted:** October 25, 2018; **Published:** October 28, 2018

Introduction

Humanity's fascination with space predates recorded history, long before Johannes Kepler proposed his famous laws of planetary motion in the 17th century. The period following Kepler saw numerous advances in mathematics and physics, beginning with Newton's laws of mechanics and the Universal Law of Gravitation until the dawn of the 20th century when Einstein presented his theory of special relativity. The advances in mathematics and physics during this period made possible efforts to send people to outer space and to discover the mysteries of the universe. Some of the major milestones during this period include the first human in space, the first spacewalk, the first Moon landing, the Apollo program, the space station, reusable spaceships, and the launching of space probes such as Cassini into interstellar space. Cassini's historic mission ended only a few months ago in September 2017 with its fiery plunge into Saturn (1).

The scientific and engineering communities have become increasingly interested in space travel (2-6), specifically in colonizing other planets such as Mars. Two main reasons often suggested for space exploration are finding an alternative habitat for humans in the event of a catastrophic disaster (7) and the potential for new resources such as minerals in extraterrestrial space (8).

Motivated by the ongoing curiosity of the scientific community as well as the general public's desire to travel to outer space and to possibly colonize other planets one day (3-6), in the current work, we investigated an approach to efficiently travel between planets. Specifically, we propose to use a genetic algorithm to find the shortest path to travel between planets in a solar system. This problem is analogous to the well-known Traveling Salesman Problem (TSP), which has been intensely studied by mathematicians and computer scientists during the last few decades. However, unlike the stationary cities in the classic TSP, the planets in our problem are in constant motion, each with a different angular velocity, making our problem significantly harder to solve. We hope that the results of our work could be used to program space probes such as Cassini (1) to efficiently travel between the planets of a newly discovered solar system. This would allow us to collect scientific data and uncover new knowledge, to mine asteroids for natural resources that are rare on Earth, and to eliminate space junk orbiting our planet, which poses imminent danger to space travel.

The Traveling Salesman Problem (TSP) is one of the most widely investigated problems in computer science. The problem appears quite simple when stated as follows: Given a set of *N* cities, find the shortest route that visits each of the cities exactly once and returning to the starting city. The problem can be solved relatively easily when the number of cities is small. However, as the number of cities increases, one quickly discovers that the simplicity of the problem statement is quite deceptive. The problem is often presented in computational mathematics as a graph-theory problem, where one has to determine the most efficient Hamiltonian cycle (9) through *N* cities. This is an optimization problem, which is NP-hard. This can be proved by showing that the well-known NP-complete Hamiltonian cycle problem can be

N	t_1	t_2	$(t_2/t_1)\times 100\%$	d_1 (AU)	d_2 (AU)	$(d_1-d_2)/(d_1)\times 100\%$
5	16 ms	4 ms	25	269	269	0.00
6	25 ms	26 ms	104	291	291	0.00
7	33 ms	71 ms	215	302	302	0.00
8	43 ms	614 ms	1,428	317	317	0.00
9	48 ms	9.145 s	19,052	322	322	0.00
10	50 ms	1 m 52 s	224,828	358	358	0.00
11	278 ms	21 m 19 s	460,072	399	396	0.75
12	553 ms	12 h 14 m 40 s	7,971,067	408	405	0.74

Table 1: Comparison of Results - TSSP-GA versus TSSP-BF Algorithms

reduced to TSP. The terms NP-hard and NP-complete are used in the field of algorithmic complexity theory to discuss the difficulty of solving various problems. A decision problem *p* is said to be NP-complete if *p* is in the complexity class of NP, and every problem in NP is reducible to *p* in polynomial time. A problem that satisfies the latter condition is known to be NP-hard.

In the Moving-Target TSP, suggested by Helvig (10), and its variants (11-13), the cities are all given a constant initial linear velocity and therefore are no longer stationary. The goal of the Moving-Target TSP and its variants is the same as that of the standard TSP, which is to find the shortest-possible route visiting every city. However, our current problem, called the traveling spaceship problem (TSSP), involves planets in circular orbits with different angular velocities, making it significantly harder than these previous works. We provide a novel near-optimal solution for TSSP by employing a genetic algorithm that uses a carefully chosen bimodal fitness function. Even though genetic algorithms have been used in space technology for various applications, including the design of the famous NASA antenna (14), there has been no work so far that would solve the problem of finding a near-optimal travel path for efficient interplanetary travel.

Our experiments confirm the hypothesis that the bimodal fitness function, which forms the heart of our genetic algorithm, outperforms an algorithm based on a traditional unimodal fitness function in identifying a near-optimal solution for the shortest path, both in terms of speed and accuracy. The experiments further confirm the hypothesis that the convergence rate of our algorithm is many orders of magnitude faster than an algorithm based on enumeration, while providing nearly accurate results.

Results

In order to test our hypothesis, we compared the results of the Java implementations of our genetic algorithm, TSSP-GA, and an enumeration-based brute-force algorithm, TSSP-BF. We further compared the convergence rate of our genetic algorithm to an otherwise-identical algorithm based on a unimodal fitness function. Our experiments can be divided into four broad categories as follows.

Time Taken to Find the Shortest Path

By varying the number of planets in the solar system from N = 5 to N = 12, we found the times to obtain the shortest path for TSSP-GA (t_1) and TSSP-BF (t_2). We also found the shortest distances from both TSSP-GA (d_1) and TSSP-BF (d_2). We computed the percentage run-time performance improvement of TSSP-GA over

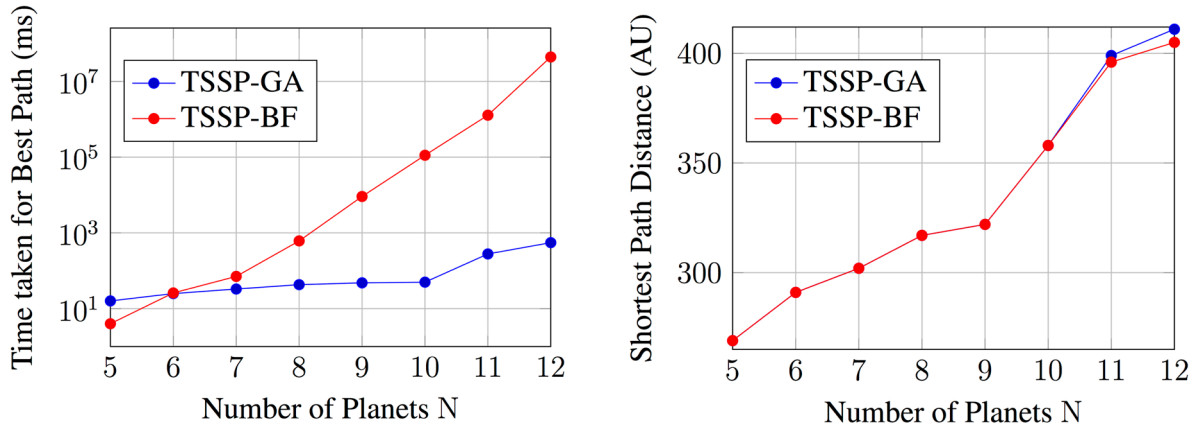

Figure 1: Number of planets versus time taken for best path and the shortest path distance. Left: We varied the number of planets and measured the time to find the shortest path using TSSP-BF and the near-optimal path using TSSP-GA. **Right:** We varied the number of planets and found the best distance returned by TSSP-BF and the near-optimal distance returned by TSSP-GA.

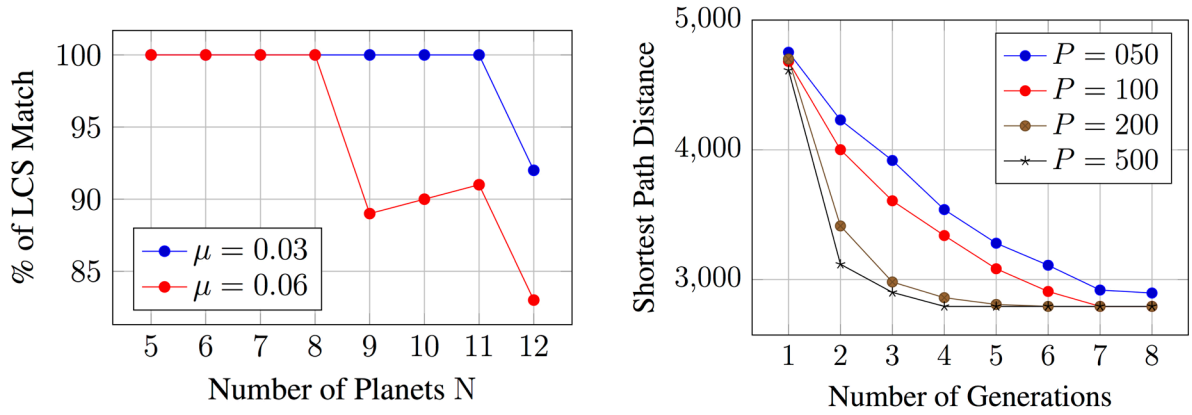

Figure 2: Percentage of LCS matching at different mutation rates and Number of generations to achieve near-optimal distance. Left: We varied the number of planets and measured the percent of LCS matching at two different mutation rates (μ= 0.03 and μ = 0.06). **Right:** We plotted the near-optimal distance returned by TSSP-GA against the number of generations for the initial population sizes of 50, 100, 200 and 500.

that of TSSP-BF as well as the percentage difference of the path distance obtained from TSSP-GA to the true shortest distance obtained from TSSP-BF, obtained by enumerating all N! paths (**Table 1**).

The time to obtain the shortest path from TSSP-GA increases only slightly as the number of planets in the solar system increases (**Figure 1**, **left**). In other words, the time to identify a near-optimal interplanetary path using our genetic algorithm is growing slowly with the number of planets. However, TSSP-BF shows an exponential increase in run time with the number of planets in the solar system (**Figure 1**). This is expected as there are N! possibilities to be analyzed. The optimal path produced by TSSP-GA is only marginally longer than the shortest possible path when N = 12, while the shortest paths identified by both techniques are identical for $N \leq 10$ (**Figure 1**, **right**). We could not obtain the shortest distance using the brute force algorithm for N > 12, as extrapolation shows that the program will run at least 30 days for N = 13 on our machine. However, even with a hypothetical solar system with N = 100, our genetic algorithm was able to return its best path in approximately 2 seconds. The power of our TSSP algorithm is obvious.

Longest Common Substring Match

A higher mutation rate is associated with a decrease in the percent of longest common substring (LCS) match (**Figure 2**, **left**). The LCS indicates the longest common segment of the paths returned by TSSP-GA and TSSP-BF. Our experiments therefore show that selecting an appropriate mutation rate is essential for increasing the percentage of LCS match between TSSP-GA and TSSP-BF. By varying the mutation rate from μ = 0.005 to μ = 0.100 at an increment of 0.001, we found that our algorithm performs optimally at μ = 0.030.

The Initial Population Size and the Number of Generations to Converge

In the next set of experiments using TSSP-GA, using N = 25, we varied the size of the initial population P and recorded the number of generations required to find the shortest path. We found that as the size of the initial population is increased, the rate of convergence of the algorithm also increased. **Figure 2** (**right**) shows the relationship between the initial population size and the number of generations required to identify the shortest path distance. For example, when the population size was P = 50, TSSP-GA needed at least 8 generations to identify the shortest path distance. On the other hand, with a population size of 500, even though the program took longer to finish, it only needed 4 generations to identify the shortest path distance. This demonstrates that an optimal initial population size containing sufficient genetic diversity is essential for convergence in fewer generations.

Bimodal Fitness Function

In the final set of experiments, we sought to test the hypothesis that our bimodal fitness function would outperform an algorithm employing a unimodal fitness function. This is done by computing a fitness score based on the inverse of the total segment distances. The average fitness score always increases between successive generations. We consider the population to be converged when the difference between two consecutive fitness scores is less than a predetermined tolerance value. Using an initial population size of 200, we demonstrated that convergence was achieved in a mere 8 generations using our bimodal fitness function, whereas it took 22 generations to obtain similar results with the unimodal fitness function (**Figure 3**, **left**). Finally, we plotted the fitness scores returned at the end of the eighth generation as we varied the size of initial population from 50 to 500 (**Figure 3**, **right**). This showed us that our bimodal fitness function was able to achieve convergence with an initial population size of only 200, whereas the use of the unimodal fitness function required an initial population size of 500 to achieve convergence for an identical solar system.

The power of our bimodal fitness function is self-evident. In addition, the results support the hypothesis

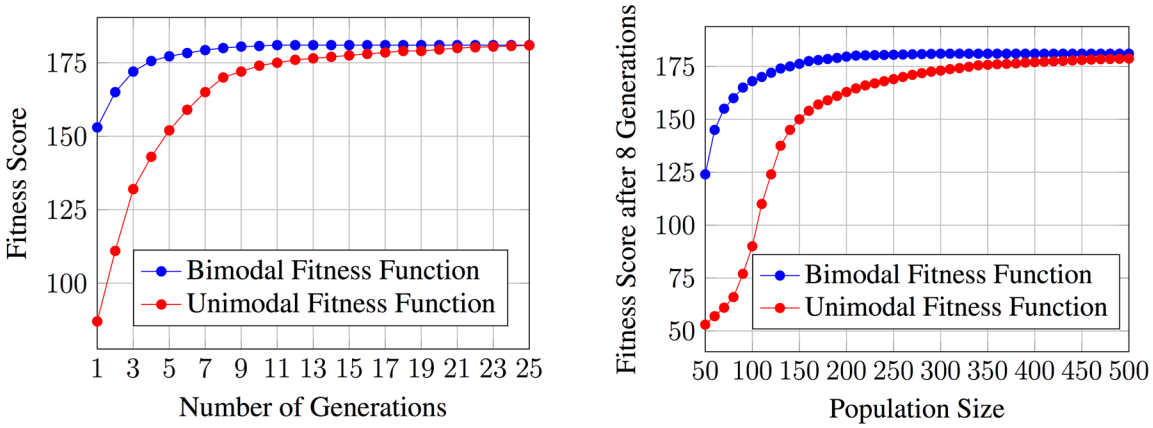

Figure 3: Bimodal and unimodal fitness function comparison. Left: Bimodal fitness function resulted in a faster convergence than the unimodal fitness function. **Right**: Fitness score after eight generations is higher with the bimodal fitness function than with the unimodal fitness function with small initial population sizes. As the initial population size is increased, the difference in fitness score between the two types of fitness function decreases.

that our genetic algorithm is able to identify a near-optimal path for interplanetary travel significantly faster than an enumeration-based technique. The results are many orders of magnitude faster for a solar system with more than nine planets. In addition, the optimal path found by our genetic algorithm is identical to the best possible route for $N \leq 10$ and is only marginally different than the best possible route for larger N found by TSSP-BF. Therefore, we hope that our work will form the basis for finding an optimal interplanetary travel path of a newly discovered solar system in a futuristic world.

Discussion

Let S be a solar system with N planets in circular orbits around a star. The traveling spaceship problem is to find the shortest path at any random point in time, starting from any one of the *N* planets, visiting each planet only once, and returning to the initial planet. The spaceship is assumed to be moving at constant velocity and cannot "wait" on a planet.

For a solar system with three planets, there are only 6 possible paths. **Figure 4** shows two possible paths for such a solar system. However, as in the case of the classic TSP, the number of possible paths in TSSP quickly explodes as the number of planets increases. For example, for our own solar system, there are 9! = 362,880 possible paths, and for a hypothetical solar system with 20 planets, the number of possibilities is indeed an "astronomical" 20!, which is larger than 2.43 × 10^{18}. In addition, even though the classic TSP is NP-hard, it only deals with stationary cities. The Moving-Target TSP introduced by Helvig *et. al.* (10), determines the minimum time required to reach a small set of moving targets, all moving with a constant velocity, but confined to a straight line. It is not difficult to see that our problem is significantly more difficult than both the classic TSP and the Moving-Target TSP, as the planets are all in motion with *different angular velocities*, which requires that the trajectory angle of the spaceship be determined dynamically based on the origin and the target planet.

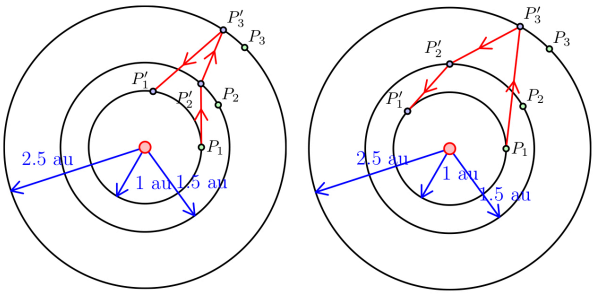

Figure 4: Two possible travel paths in a three-planet solar system. Left: The spaceship departs from planet P_1 and meets planet P_2 at $P_{2'}$, continues toward planet P_3, and meets it at $P_{3'}$. Finally, it returns to planet P_1 and meets it at $P_{1'}$, making a connected path, $P_1P_{2'}P_{3'}P_{1'}$. **Right**: The spaceship departs from planet P1 and meets planet P_3 at $P_{3'}$, continues toward planet P_2, and meets it at $P_{2'}$. Finally, it returns to planet P_1 and meets it at $P_{1'}$, making another connected path, $P_1P_{3'}P_{2'}P_{1'}$.

Space Travel Equations: Change in Angle of the Destination Planet

In this and the following sections, we derive two important equations that will help us to plan the trajectory of the spaceship for interplanetary travel. Our genetic algorithm, implemented in Java, depends on the results of these equations in order to determine the coordinates of the planets at different points in time during the journey of the spaceship, as well as the trajectory angles for the journeys.

Two of our principal simplifying assumptions are that the spaceship travels at a constant speed, and that the spaceship travels in a straight line between planets. In addition, we assume that all of the objects underconsideration are found on the same plane and the planets are in circular motion. The first two assumptions are most realistic when the spaceship travels faster than the fastest planet, for then the central star has a relatively small effect on the spaceship's velocity vector. We will therefore focus on this case, which will have an additional benefit described at the end of this section. Therefore, given that the initial coordinates of planet *A* and planet *B*

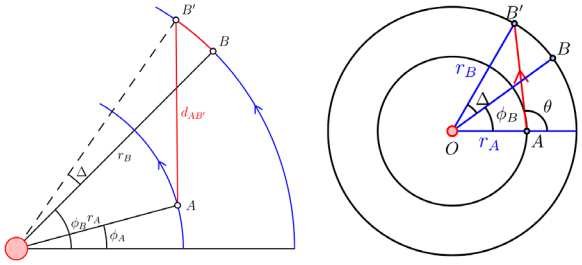

Figure 5: Computation of travel distance and launch angle. Left: The spaceship travels from planet *A* and arrives at the destination planet at *B'*. The distance traveled by the spaceship is $d_{AB'}$. **Right:** The spaceship needs to be launched at angle Θ as shown in the diagram in order to meet the destination planet at *B'*.

are (r_A, θ_A) and (r_B, θ_B) respectively, we would like to determine how to launch our spaceship from planet *A* so that it would meet planet *B* as the spaceship arrives at the orbit of planet *B*. This can be done by finding the angle Δ through which planet *B* moves from the time when the spaceship leaves planet *A* to when the spaceship arrives at planet *B* (**Figure 5**).

Let us consider a planet in a circular orbit of radius *r* around a star of mass *M*. From Newton's law of universal gravitation, ignoring the gravitational effect of other celestial bodies, the gravitational acceleration of the planet is, $a = \omega^2 r = \frac{GM}{r^2}$, where ω is the angular velocity of the planet. Therefore, we find, $\omega = \sqrt{\frac{GM}{r^3}}$. Hence, the polar coordinates of the planet at time t can be given as, $(r, \omega t + \phi) = \left(r, t\sqrt{\frac{GM}{r^3}} + \phi\right)$, where ω is the initial angle of the planet at t = 0. Representing the positions of the planets using the polar coordinates given by the above equation, we will now find the angle Δ through which the destination planet moves from the time when the spaceship leaves the first planet to when the spaceship arrives at the destination planet. Let us consider the journey of our spaceship from planet $A = (r_A, \Phi_B)$ to planet $B = (r_B, \Phi_B)$. Let Δ be the change in planet *B*'s angle from when the spaceship leaves planet *A* to when the spaceship arrives at planet *B*. Let *B*' be planet *B*'s new location when the spaceship arrives. Then, using the cosine rule, the distance from *A* to *B*' is,

$$d_{AB'} = \sqrt{r_A^2 + r_B^2 - 2r_A r_B \cos(\phi_B - \phi_A + \Delta)}.$$

Since the time for the spaceship to travel the above distance, $d_{AB'}$ is equal to the time for planet *B* to orbit through the angle Δ, letting the speed of the spaceship be υ, we find that

$$\frac{d_{AB'}}{v} = \frac{\Delta}{\omega_B} = \frac{\Delta}{\sqrt{GM/r_B^3}}.$$

Substituting for $d_{AB'}$ into the above equation and simplifying, we obtain,

$$\Delta^2\left(\frac{v^2 r_B^3}{GM}\right) + 2r_A r_B \cos(\Delta + \phi_B - \phi_A) - (r_A^2 + r_B^2) = 0.$$

The above equation has both a Δ^2 and a cos(Δ) term, and consequently does not have a clear closed-form solution. We use the Newton-Raphson method to obtain a numerical solution to the change in angle using the above equation. In general, the above equation has multiple solutions for Δ. When the spaceship is significantly slower than the target planet, the target planet can orbit the star multiple times. Adjusting the launch angle of the spaceship can alter the number of times the target planet orbits the star before the spaceship reaches it. However, since we impose the condition that the spaceship always travels faster than the fastest planet (which is the one closest to the star), the spaceship will intercept the target planet in a unique orbit, and therefore, assuming $r_B > r_A$, we guarantee a single unique solution to the above equation.

Space Travel Equations: Trajectory Angle of the Spaceship

Now that we have derived an expression for the change in angle Δ of planet *B* during the time the spaceship travels from planet *A* to meet planet *B* at location *B'*, we will proceed to find the trajectory angle θ of the spaceship when it is launched from planet *A*. The trajectory angle depends on the velocity of the spaceship as well as the angular velocity of the destination planet. It does not depend on the angular velocity of the origin planet. Using polar coordinates, the following proposition gives the launch angle θ for the spaceship when it departs from planet *A* to meet planet *B* at *B'*.

Proposition 1. *Given two planets A and B with coordinates* $(r_A, 0)$ *and* (r_B, Φ_B) *respectively at time* $t = 0$, *and* $r_B > r_A$, *the launch angle* θ *of the spaceship with respect to the positive x-axis is given by,*

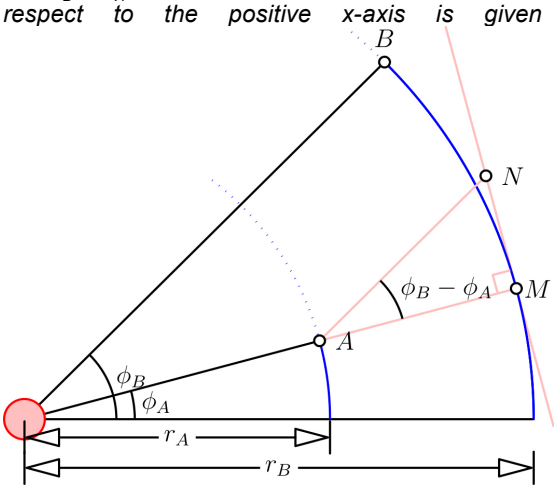

Figure 6: Fitness function f_1. The segment MN is tangent to the path of planet *B*. The distance |MN| is approximately proportional to the distance that the spaceship needs to travel in the direction vertical to line AM.

$$\tan\theta = \frac{r_B \sin(\Delta+\phi_B)}{r_B \cos(\Delta+\phi_B) - r_A},$$

where Δ *is the angle through which planet B rotated from B to B' during its journey.*

Proof. Using the Law of Sine on triangle *AOB'* in **Figure 5** (**right**),

$$\frac{\sin(180^\circ-\theta)}{r_B} = \frac{\sin(\theta-\Delta-\phi_B)}{r_A}, \quad \text{or}$$

$$\frac{\sin\theta}{r_B} = \frac{\sin\theta\cos(\Delta+\phi_B) - \cos\theta\sin(\Delta+\phi_B)}{r_A}$$

Simplifying, we find,

$$\tan\theta = \frac{r_B \sin(\Delta+\phi_B)}{r_B \cos(\Delta+\phi_B) - r_A}.$$

Future Work

Our work provides a near-optimal solution to the traveling spaceship problem in a hypothetical mathematical model of a solar system. However, it did not factor in the gravitational influence of the star and the planets on the spaceship. Our model also assumed that all the planets are in perfect circular orbits around the star. Therefore, it will be important in a future work to account for both the gravitational force of the various celestial bodies on the spaceship and the elliptical orbits of the planets. As an alternative to our TSSP-GA, we also plan to investigate the use of Covariance Matrix Adaptation Evolution Strategy (CMA-ES) (15, 16), which employs stochastic searches to more rapidly converge to the global optimum. In an unfinished related work, we investigated how to employ genetic algorithms to optimize the timing and direction of the fuel firing in order to efficiently navigate the spaceship. This investigation led us to believe that there may exist an optimal route that may not be the shortest but is likely to cost the least amount of fuel, as such a route would minimize the change of spaceship trajectories. This is a much harder problem to solve, but could be very important for optimizing space travel.

Methods

Genetic algorithms are used in computer science to iteratively improve the solution, by using a combination of randomization, selection, recombination and mutation in tandem with an appropriately designed fitness function. These algorithms mimic the process of evolution - a process of biological wonder, elegance and beauty - that has proven itself by giving rise to millions of successful distinct species on our planet.

A genetic algorithm typically begins with an initial population consisting of a set of a few individuals. The population evolves via a series of three major steps, namely selection, crossover, and mutation. The fitness

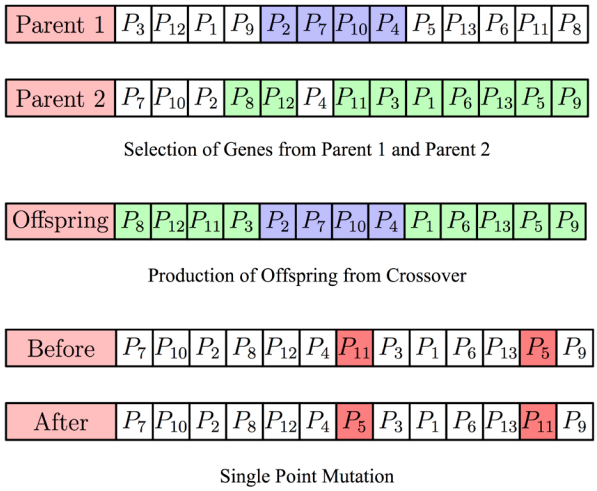

Figure 7: Overview of the TSSP-GA algorithm. The top two diagrams show that gene segment $P_2P_7P_{10}P_4$, is randomly selected from parent 1, and the remaining genes from parent 2 are used in place of the remaining genes of parent 1. The third diagram shows the resulting offspring from this crossover. The bottom two diagrams illustrate a single point mutation, where genes P_{11} and P_5 swap their positions in the chromosome.

of each individual is determined, and the fittest individuals or the "most optimal solutions", are selected for amongst the population. The process thereby produces offspring that are even further improved from the original selection using chromosomal crossovers. Splices of the fittest individuals (chromosomes) are recombined, creating a more "evolved" offspring. Finally, a small percentage of random mutations are introduced in the genes, which entails flipping bits of some individuals in order to add more diversity to the gene pool and to prevent premature convergence of the algorithm. Two prerequisites must be met in order for a genetic algorithm to succeed. The first is that there must be a method to encode viable solutions to the problem. The second is that there must be a method to determine the degree of validity of such a solution. In other words, one must not only be able to determine a solution, but also must be able to determine the level of correctness of this solution. The gene pool of our initial population is assumed to contain sufficient genetic diversity.

Using the mathematical derivations from the Discussion section, we implemented a genetic algorithm to solve TSSP using Java. Given that the classic TSP is an NP-complete problem (17), our intuition tells us that the Traveling Spaceship Problem is likely to be NP-complete. We omitted the proof of this statement as it is beyond the scope of this paper. We then provided an overview of the major components of our genetic algorithm solving the Traveling Spaceship Problem. A number of experimental results from the Java implementation of our algorithm (TSSP-GA) in comparison to an enumeration-based brute-force algorithm (TSSP-BF) were discussed earlier. We note that the TSSP-BF algorithm compares the travel distances of every possible *N*! paths in an *N*-planet solar system in order to identify the best path.

Overview of the Implementation

The major modules of our algorithm are (1) initialization, (2) selection, (3) crossover, and (4) mutation. During initialization, we create a "small" population consisting of random paths connecting every planet exactly once. The individuals in our populations are distinct paths connecting the planets. For example, in a solar system with 12 planets, there are 12! = 479,001,600 unique individuals, but our population only contains a tiny fraction of these.

Bimodal Fitness Function

The key to the success of our genetic algorithm is a novel bi-modal fitness function that forms the heart of our algorithm. This fitness function allows us to rapidly determine the suitability of an individual (path) to be selected for reproduction. Unlike the traditional genetic algorithms in the literature, our fitness function consists of two functions, where the first function, f_1, is used to rapidly converge to a population with superior characteristics, and the second function, f_2, is used to further iteratively refine the population. In **Figure 6**, consider the planets A and B at coordinates (r_A, Φ_A) and (r_B, Φ_B) at an arbitrary time t. We note that the distance $MN = (r_B - r_A)\tan(\Phi_B - \Phi_A)$ is proportional to the distance the spaceship needs to travel in the tangential direction in order to arrive at planet *B*, assuming that the planets are stationary. Therefore, we use the fitness function in the first step to be $f_1 = \sum \frac{1}{|MN|} = \sum \frac{\cot(\phi_B - \phi_A)}{r_B - r_A}$, and aim to maximize it. The first function, f_1, described above, which is much easier to compute than the true travel distance, enables us to achieve partial convergence rapidly. When successive populations are no longer evolving, we use the inverse of the path distance as the second fitness function, $f_2 = \sum \frac{1}{d_i}$ to further evolve and refine the individuals. The higher the fitness value, the better the probability for an individual to be selected for reproduction.

Crossover

Crossover is the mechanism that enables a population to increase its genetic diversity and consequently is a very important component of our algorithm. Our implementation of crossover is as follows: one of the two parents is selected with equal probability, and then a segment of consecutive connected planets is selected randomly. We form the offspring using this segment of the selected parent. Then we go through the second parent's path (chromosome) and copy its genes to the offspring, making sure that the genes forming the segment from the first parent are not copied again. For example, **Figure 7** (**top**) shows two paths selected as parents in a solar system with 13 planets. Assume that a random contiguous segment (highlighted in blue in **Figure 7** (**top**)) of the first parent is selected for crossover. Now the remaining genes (highlighted in green in **Figure 7** (**top**)) are identified from the second parent and are selected for crossover with those genes previously selected from the first parent. The offspring is then formed by first applying the selected segment (highlighted in blue) from the first parent and then filling up the remaining slots using the other genes (highlighted in green) from the second parent in order (**Figure 7 middle**).

Mutation

Mutations in nature are generally detrimental to organisms. However, mutations occasionally help a population by adding some much-needed genetic variation, thereby producing a more successful offspring. Our algorithm implements a single-point mutation by swapping two planets in the path. In the example shown in **Figure 7** (**bottom**), planet P_{11} is swapped with planet P_5.

Both the crossover and the mutation modules operate on those individuals returned by the selection module, which makes its decision based on our bimodal fitness function. The process is repeated until the shortest path length between successive generations is no longer changing.

References

1. "NASA's Cassini Spacecraft Ends Its Historic Exploration of Saturn." *NASA Solar System News*, 15 Sept. 2017, saturn.jpl.nasa.gov/news/3121/nasas-cassini-spacecraft-ends-its-historic-exploration-of-saturn. Accessed 15 Sept. 2017.
2. Bednar, Scott. "NASA 360 - The Future of Human Space Exploration." *NASA*, NASA, 12 Mar. 2015, www.nasa.gov/content/nasa-360-the-future-of-human-space-exploration. Accessed 21 Aug. 2017.
3. Chang, Kenneth. "Elon Musk's Plan: Get Humans to Mars, and Beyond." *The New York Times*, The New York Times, 27 Sept. 2016, www.nytimes.com/2016/09/28/science/elon-musk-spacex-mars-exploration.html?mcubz=0&_r=0. Accessed 22 Aug. 2017.
4. Anthony, Sebastian. "Space Giants Join Forces to Battle SpaceX: This Is How Cheap Space Travel Begins." *ExtremeTech*, 16 June 2014, www.extremetech.com/extreme/184434-space-giants-join-forces-to-battle-spacex-this-is-how-cheap-space-travel-begins. Accessed 22 Aug. 2017.
5. Simberg, Rand. "Elon Musk on SpaceX's Reusable Rocket Plans." *Popular Mechanics*, Popular Mechanics, 15 Feb. 2018, www.popularmechanics.com/space/rockets/a7446/elon-musk-on-spacexs-reusable-rocket-plans-6653023/. Accessed 22 Aug. 2017.
6. Weaver, Matthew. "'Welcome Back, Baby': Elon Musk Celebrates SpaceX Rocket Launch – and Landing." *The Guardian*, Guardian News and Media, 22 Dec. 2015, www.theguardian.com/science/2015/dec/22/welcome-back-baby-elon-musk-celebrates-spacex-rocket-launch-and-landing. Accessed 10 Aug. 2017.
7. Editor, Roger Highfield Science. "Colonies in Space

May Be Only Hope, Says Hawking." *The Telegraph*, Telegraph Media Group, 16 Oct. 2001, www.telegraph.co.uk/news/uknews/1359562/Colonies-in-space-may-be-only-hope-says-Hawking.html. Accessed 20 Aug. 2017.
8. Zuppero, A. "Discovery of Abundant, Accessible Hydrocarbons Nearly Everywhere in the Solar System." *Engineering, Construction, and Operations in Space V*, 1996, doi:10.1061/40177(207)107.
9. Skiena, Steven. I*mplementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica*. Addison-Wesley, 1991.
10. Helvig, C.S., et al. "The Moving-Target Traveling Salesman Problem." *Journal of Algorithms*, vol. 49, no. 1, 2003, pp. 153–174., doi:10.1016/s0196-6774(03)00075-0.
11. Applegate, David, et al. "Implementing the Dantzig-Fulkerson-Johnson Algorithm for Large Traveling Salesman Problems." *Mathematical Programming*, vol. 97, no. 1, 2003, pp. 91–153., doi:10.1007/s10107-003-0440-4.
12. Stieber, Anke, et al. "The Multiple Traveling Salesmen Problem with Moving Targets." *Optimization Letters*, vol. 9, no. 8, 2014, pp. 1569–1583., doi:10.1007/s11590-014-0835-6.
13. Jindal, Pawan, et al. "Dynamic Version of Traveling Salesman Problem." *International Journal of Computer Applications* (0975 – 8887), vol. 19, no. 1, Apr. 2011.
14. *Automated Antenna Design with Evolutionary Algorithms*. ti.arc.nasa.gov/m/pub-archive/1244h/1244 (Hornby).pdf. Accessed 20 Aug. 2017.
15. Jastrebski, G.A., and D.V. Arnold. "Improving Evolution Strategies through Active Covariance Matrix Adaptation." *2006 IEEE International Conference on Evolutionary Computation*, doi:10.1109/cec.2006.1688662.
16. Hansen, Nikolaus. "The CMA Evolution Strategy: A Comparing Review." *Towards a New Evolutionary Computation Studies in Fuzziness and Soft Computing*, pp. 75–102., doi:10.1007/11007937_4.
17. Cormen, Thomas H., et al. *Introduction to Algorithms*, 3rd ed., The MIT Press, 2009.