Interplanetary Trajectory Optimization using a Genetic Algorithm

Abby Weeks

Aerospace Engineering Dept
Pennsylvania State University
State College, PA 16801

Abstract

Minimizing the cost of a space mission is a major concern in the space industry. Sending satellites on interplanetary trajectories is risky and expensive. Trajectory optimization for such missions has been developed through classical methods of optimization. However, the application of Genetic Algorithms has become increasingly popular. In this project I formulate a preliminary mission design for an interplanetary trajectory taking a satellite from Earth to Jupiter via a gravity assist at Mars, using a genetic algorithm to optimize the trajectory based on the assumption of a constant low level thrust. The main objective of this optimization is to minimize the time of flight. By minimizing the time of flight the risk of damage to the satellite during the course of the mission is reduced as well as the cost of fuel. The Simple Genetic Algorithm implemented in this application gives results which could be used as a first guess in a more localized exploration.

1 INTRODUCTION

1.1 INTERPLANETARY TRAJECTORIES

Interplanetary missions are complicated due to the dynamics of our solar system. As a spacecraft travels through our solar system it may encounter several celestial bodies, and be influenced by their gravitational fields. In order to simplify the development of a mission, it is common to analyze a series of two-body problems. The patched conic method is used to analyze a mission where a spacecraft encounters more than one celestial body. A spacecraft launching from Earth will start within the sphere of influence (SOI) of Earth. As the spacecraft exits Earth's SOI it enters the Sun's SOI until it enters the SOI of another celestial body. While the spacecraft is inside the SOI of a celestial body such as the Earth or Sun the effects of all other celestial bodies can be ignored (as a first approximation) and the dynamics can be modeled as a two-body gravitational system. By patching together a sequence of such two-body problems a complete trajectory can be determined.

A trajectory such as that of one from Earth to Mars may be represented by a combination of the following three two-body problems:

? Spacecraft and Earth

? Spacecraft and Sun

? Spacecraft and Mars

We can take advantage of the effects of a spacecraft passing near a celestial body that is not its destination body. This is known as a planetary flyby or gravity assist. A planetary flyby or gravity assist is useful to obtain a velocity change without expending propellant. This technique is of great interest to mission designers due to the "free" velocity change induced by the gravitational pull of the flyby planet. By flying a spacecraft within a close proximity of a planet we can use the gravitational field of the planet to either speed up the spacecraft or slow it down.

Most interplanetary missions make use of a low-thrust propulsion system. In this project a constant low-level thrust is implemented meaning the thrusters are always on. For this reason the reduction of the duration of the mission becomes the key to reducing the cost of propellant. Essentially the shorter the mission, the less propellant required. In terms of the low-thrust problem formulation, a control profile that minimizes the propellant consumption while satisfying certain boundary conditions is to be determined. This control profile is given by a thrust direction, ?, assuming constant thrust magnitude T. The trajectory is broken up into discrete segments for which the direction of thrust or ? must be determined. By determining these angles the objective is to minimizes the time of flight and therefore the cost of propellant. Figure one contains the relationship of the flight path angle to the position of the spacecraft (s/c).

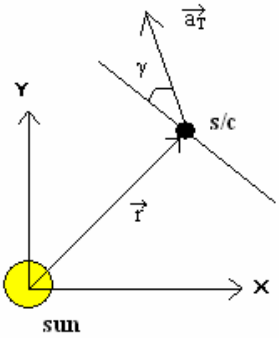

Figure 1: The direction of thrust given by a_T is defined by the flight path angle ?.

2 MODELING THE TRAJECTORY

2.1 EQUATIONS OF MOTION

The motion of the spacecraft is governed by equations (1)-(4). This group of differential equations is solved by numerical integration using the Runge-Kutta variable step size integrator implemented in Matlab as ODE45.

$$\frac{dx}{dt} = Vx \qquad (1)$$

$$\frac{dVx}{dt} = -\frac{(\mu_{\odot} * x)}{r^3} - a_T * \sin(\theta-\gamma) + a_{G,x} \qquad (2)$$

$$\frac{dy}{dt} = Vy \qquad (3)$$

$$\frac{dVy}{dt} = -\frac{(\mu_{\odot} * x)}{r^3} - a_T * \cos(\theta-\gamma) + a_{G,y} \qquad (4)$$

All of the components of these equations of motion are listed below:

$\mu_{\odot}$ = 1.327e11 km^3/s^2, gravitational constant of sun

a_T = 2e-7 km/s^2, thrust acceleration

$a_{G,x}$ = the gravitational acceleration in the x-direction due to mars

$a_{G,y}$ = gravitational acceleration in the y-direction due to mars

r = the position of the spacecraft

? = flight path angle, to be optimized

? = angular position of the spacecraft

3 PROBLEM FORMULATION

3.1 OBJECTIVE

The objective of this optimization was to reduce the time-of-flight and, as a result, the propellant cost. The mission was divided into two legs, the first leg taking place from Earth to Mars and the second from Mars to Jupiter. Each leg of the complete trajectory was divided up into 10 segments, an arbitrary number. It is assumed that the larger the number of segments, the more precise the solution will be. The Genetic Algorithm used will be responsible for determining the optimal thrust direction or flight path angle ? at the begining of each time segment and time-of-flight for both legs of the mission.

3.2 CONSTRAINTS

Several boundary conditions are placed on the system and they are implemented as constraints. The optimization of this system is dependent on the trajectory solved for by the Genetic Algorithm. In order to ensure that the trajectory is feasible a penalty method was implemented to optimize the objective.

My objective is to minimize the TOF and the penalties to this minimization are on the position and velocity of the spacecraft at mars and at Jupiter. The fitness function which is evaluated with respect to the objective and penalties is given as follows:

fitness = 1.0^5 – TOF/year – (w2*P1) – (w1*P2) – (w3* P3) – (w2* P4) – (w2*P5)

Where TOF is the time-of-flight, w1,w2,and w3 are weighting factors with values 1, 1,000, and 10,000 respectively and P1-P5 are the penalties used in this optimization that take the form:

The first two penalties are applied as the s/c targets Mars in the first leg of the mission.

P1 = (radial position of s/c – radial position of mars)^2

P2 = (radial velocity of s/c)^2

The final three penalties are applied as the s/c targets Jupiter in the second leg of the mission.

P3 = (radial position of s/c – radial position of Jupiter)^2

P4 = (radial velocity of s/c)^2

P5 = (angular velocity of s/c – angular velocity of Jupiter)^2

The genetic algorithm used in this project acts as a maximizer. It must have positive values for all fitness values. In order to turn this algorithm into a minimizer an offset value must be used. This was implemented as the GA was trying to maximize the fitness function. I offset this action by applying a value of 10^5 from which the TOF and penalties once normalized are subtracted from. Thus by minimizing the TOF and the penalties incurred it would work to maximize the fitness function, with the

maximum value of fitness possible valued at 10^5. In order to keep the fitness value positive I used an IF statement, which took any negative value of fitness and reset it to 0.1. If a negative value of fitness occured the solution would be very bad and thus the 0.1 value should eliminate it from the next generation.

4 SIMPLE GENETIC ALGORITHM

4.1 GENETIC ALGORITHMS

The optimization technique implemented in this paper is the simple genetic algorithm. A genetic algorithm is a stochastic global search algorithm which was developed first by Holland in 1975 and further developed by his student and now a leading expert in the field of GAs, David E. Goldberg in the 1980's.

GAs belong to a class of methods called Evolutionary Algorithms (EA) that are inspired by the processes of natural selection. EAs evolve a population of individuals through selection, mating and mutation. This evolution is done by a stochastic process that selects the more adaptive individuals and then mates them to produce better and better solutions. Mutation is also applied to random individuals to introduce diversity into the population and to prevent early convergence to local optima.

GAs are traditionally binary coded algorithms, and will be implemented as such in this study. GAs are different from more traditional optimization techniques because they search from a population of points rather than a single point. They also use payoff information based on an objective function defined by the user rather than derivatives or other secondary knowledge. GAs are blind, meaning that they only require the payoff value to evolve the solution, and thus they are virtually blind to the mechanics of the system being optimized. They also use random choice as a tool toward searching regions with likely improvement.

An individual is a string containing one set of values of the unknowns in the problem solution. The unknowns in this paper consist of a set of flight path angles, ?, and a time-of-flight for each leg of the mission (leg 1 = Earth to Mars, leg 2 = Mars To Jupiter.) Each unknown is represented as a binary string using 36 bits. The unknown strings are concatenated to form a single string, which is manipulated by the GA.

The fitness of an individual is a measure of how well the values contained in the corresponding set of unknowns solve the problem. Some individuals will have small time-of-flight, but the flight path angles will lead to severe constraint violations; other individuals will have longer time-of-flight, but the flight path angles will result in a trajectory that more nearly meets the constraints. Finding the proper combination of flight path angles and time-of-flight will lead to the optimal solution. To evaluate the fitness of each individual in a population, the governing equations of motion are integrated, using the discretized flight path angles (and interpolation to find intermediate values) and time-of-flight for each trajectory segment.

Once this fitness is evaluated the genetic algorithms uses three main operators to innovate the system. The three operators are: Selection, Mating, Mutation. These operators will be explained in section 4.2.

4.2 CLASSICAL SIMPLE GENETIC ALGORITHM IN MATLAB

The equations of motion were developed in Matlab, therefore a genetic algorithm that was written in Matlab was implemented. This Simple Genetic Algorithm was written by Andrew Potvin of Mathworks in 1993. The GA was taken from [Goldberg, 1989.] This GA allowed for both real and binary representation. However, in this project binary representation was uesd. The SGA implements all three genetic operators: selection, mating, and mutation.

4.2.1 Selection

The selection operator is applied first to choose strings from the population to be reproduced in the next generation. This first generation is comprised of the selected strings and they are referred to as parent strings. These strings are selected based on their fitness value. Once selection has taken place the chosen strings are carried to the next step.

This genetic algorithm makes use of a proportionate selection method known as the roulette wheel. As desired with this type of selection strings with higher values of fitness have a greater chance of contributing an offspring to the next generation. A biased roulette wheel is created, with each current string in the population having a slot sized in proportion to its fitness on the wheel. In order to select or reproduce, the wheel is simply spun. Each string it lands on will be reproduced in the next generation. The spinning of the wheel continues until the population is filled. The concept being that the individuals with larger portion of the wheel will be selected more times than the individuals with a smaller portion of the wheel. The generation gap, G=1 meaning that the entire population is replaced. The size of the population is also held constant. Once a string is selected for reproduction it is then entered into the next generation for the mating and mutation operators to take action.

4.2.2 Mating

In the next step the population is subject to the mating operator. Mating, or crossover, is the process of first selecting random pairs of strings from the population to be mated. In binary coding a cut is made in the binary string and the strings are crossed. The latter half of the parent strings are switched resulting in two new strings referred to as child strings.

Single point crossover was utilized to mate the binary strings. Meaning that given two parent strings x and y and a crossover point which is selected randomly by choosing an integer k ~ u(1, L-1), where L is the length of the string.

$(x_1 \ldots\ldots x_k \;\; x_{k+1} \ldots x_L) \rightarrow (x_1 \ldots x_k \;\; y_{k+1} \ldots .. y_L)$

$(y_1 \ldots\ldots y_k \;\; y_{k+1} \ldots .. y_L) \rightarrow (y_1 \ldots .. y_k \;\; x_{k+1} \ldots x_L)$

The strings are cut at integer k and the tails of the strings are switched between the parent strings to produce two new candidates or child strings.

Individuals that are chosen for reproduction are mated at random. Mating produces two offspring so as to maintain a constant population size. In this project crossover occurs with a probability, Pc = 0.8.

4.2.3 Mutation

After mating another innovating operator, mutation, is applied to the current population. In binary coding, mutation is simply a bit flip from 0 to 1 and vice versa. This operator helps to keep diversity in the population. In genetic algorithms mutation usually occurs with a lower probability allowing selection and mating to dominate the evolution.

Jump mutation was utilized to mutate the binary strings. Jump mutation randomly selects a bit in the string and flips it with probability Pm. I choose to use Pm=1/population. The designation of this probability is intended to avoid disruption and allow for mating to be the primary operator used to evolve the system. When working with smaller populations the probability of mutation will be greater, thus maintaining diversity and preventing early convergence.

4.2.4 Elitism

Elitism is the method of ensuring that the best solution stays in the next generation. Due to the stochastic nature of genetic algorithms it is not always guaranteed that the best solution will make it into the next generation. By invoking elitism this is no longer a worry. Elitism can help lead to a quicker convergence. However, in problems which converge very quickly, elitism may reduce the exploration of the algorithm.

4.2.5 Termination

After all three of the main operators have been used a new generation of parent strings is created. This new generation is then evaluated for fitness and moved through the genetic operators, this process is iterated until the termination criterion is met.

The termination criterion is dependent either on the maximum number of generations which can be chosen by the user, or a convergence threshold. Both of these termination criteria were used in this project. The maximum number of generations was set to 200. However all of the runs terminated under the convergence threshold. The convergence threshold terminated dependent on the number of generations passed. If the run had greater than five generations then convergence threshold was measured using the variation of the current best fitness and the best fitness from 5 generations ago being less than a set tolerance. And if the run was under 5 generations but higher than 1 generation the threshold criterion used the variation in the mean fitness.

The pseudo-codes for the termination criterion are shown below:

IF generation>5,

IF (best current fitness – best fitness from generation - 5 / the best current fitness)< terminate)

AND

(best current fitness > best current fitness from generation - 5)

THEN STOP

ELSEIF generation>1,

IF (current mean fitness-mean fitness from generation – 1)/current mean fitness<terminate)

AND

(current mean fitness > mean fitness from generation -1)

THEN STOP

4.3 PROCEDURE

Each leg of the trajectory is broken down into 10 segments, for which the direction of the thrust is being optmized. The genetic algorithm will first randomly populate 11 decision variables, consisting of 10 gamma values and 1 time of flight for each leg of the mission. The number of gamma values in this vector will govern the precision of the solution By increasing the number of segments in each leg of the trajectory the solution will be more precise. The trade-off being that as the number of segments in each leg increases the more computation is necessary. Since each individual in the vector of gammas must be populated and optimized throughout the run of the genetic algorithm. I have made an arbitrary decision to use 10 segments, however this is definitely a possible way to optimize the solution by experimenting with the optimal number of segments, and discovering where the computational time trade-off becomes too expensive.

These gamma and TOF values are passed into the equations of motion and their fitness values are determined. Based on the fitness, selection occurs and then mating and mutation operators evolve the current population. That population is injected back into the equations of motion and the fitness value is once again evaluated. This process continues until the termination criterion is met.

5 RESULTS AND ANALYSIS

5.1 POPULATION-RELIABILITY RELATIONSHIP

In this project I formulated a set of equations that would model the motion of a spacecraft. I separated the problem into two legs. The first leg of the mission took the s/c from Earth to Mars. The second leg of the mission took the s/c from Mars to Jupiter. This separation of the problem was implemented by allowing for two integrations to take place. The first integration modeled the mission from Earth to Mars and the final solutions of that integration were used as the initialization for the second integration modeling the mission from Mars to Jupiter.

I broke the problem down this way in order to insure that the s/c would fly-by Mars and thereby perform a gravity assist at Mars. I used the first leg of the mission to gain preliminary results as to the reliability of the genetic algorithm.

In order to evaluate the population-reliability relationship I limited the calculation to just the first leg of the mission. I ran this calculation for 6 different population sizes ranging from 6-160 to see if the size of the population affected the solution. This analysis was run without elitism first. The results from this analysis are seen in Table 1.

Table 1: Population Analysis, Non-Elitist

Pop	Time (sec)	Func. Evals.	Gen.	Max	Std Dev.
6	23	42	7	99962.8	88.510
10	239	490	49	99998.9	66.451
20	165	360	18	99999	58.794
40	631	1320	33	99999	59.817
80	80	160	2	99995.1	2.404
160	529	1120	7	99997.4	1.827

As you can see in Table 1, the population size was doubled starting at 6 and increasing to 160. The maximum values of the fitness function where found at population 20 and 40. However, the fitness values for all 6 of the population runs were close. The standard deviation of the maximum fitness decreased as the population size was increased. This leads me to believe that the algorithm becomes more reliable as the population increases.

Performing this same population analysis with elitism gave very different results. These results are given in Table 2. The number of generations needed to reach termination was drastically reduced. However, it can be seen that the maximum fitness is not found in Table 2, but is found in Table 1. This could be a result of pre-convergence. By saving the best solution the algorithm converges quickly, leading to less exploration. These results however do all produce a good first guess that could be used in a different optimization method.

Table 2: Population Analysis, Eltist

Pop	Time (sec)	Func. Evals.	Gen.	Max	Std Dev.
6	20	42	7	99963.9	.416
10	98	200	20	99992.6	14.588
20	56	120	6	99997.2	.980
40	76	160	4	99994.9	3.578
80	233	480	6	99998.1	1.185
160	453	960	6	99997	.582

5.2 ELITIST TRAJECTORY PLOTS

Figure 2 shows a run of the first leg of the mission ran without elitism and with population size 160 implementing Pc=0.8 and Pm = 1/ Population. With the green orbit representing Earth's solar orbit, the blue orbit representing Mars' solar orbit and the red representing the s/c trajectory. All of the runs converged and gave a reasonable result. This plot shows a great result where the s/c trajectory, shown in red takes the s/c directly to the orbit of Mars.

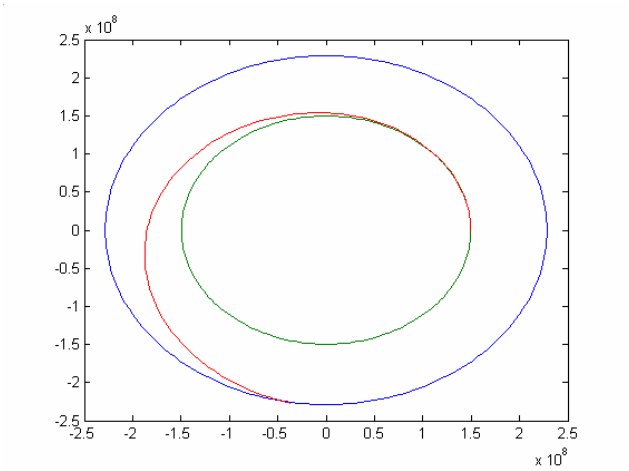

Figure 2: Trajectory from Earth to Mars

Figure 3 shows a run of both legs of the mission. The inner green orbit representing Earth's solar orbit, the blue representing Mars' solar orbit, the red representing Jupiter's solar orbit and the light blue representing the trajectory of the s/c. This run was produced using a population of 160 individuals. I also implemented the Pc =0.8 and Pm = 1/Population.

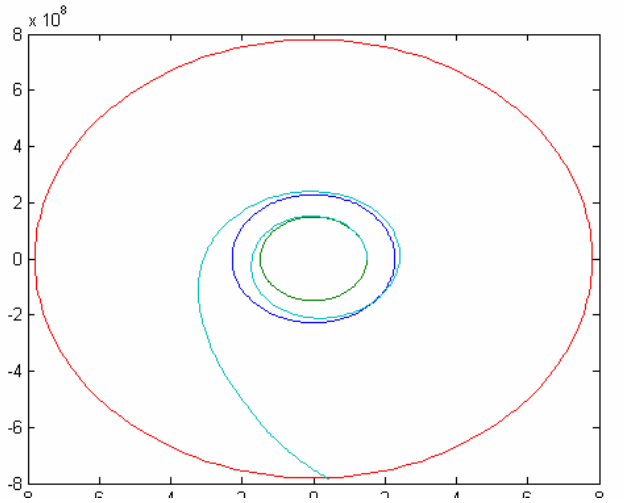

Figure 3: Trajectory from Earth to Jupiter via a Gravity Assist at Mars

Mars encounter occurs after approximately one revolution of the sun and one sees a very small change in the trajectory's curvature due to the gravity-assist. The final constraints (reaching Jupiter's orbit with zero radial velocity and arriving tangent to Jupiter's path) were not met. This is a consequence of using penalty methods, which work to drive a solution toward meeting the constraints, but do not guarantee exact compliance. Nevertheless, this solution could serve as a useful first approximation for use in a different type of optimizer (e.g., one that requires a first "guess" that is close to satisfying all the constraints).

5.3 RANDOM SEED ANALYSIS

The prior analysis used the same random seed to populate the individuals. In order to determine if the algorithm was reliable over different random seeds I performed a random seed analysis, using 25 random number seeds for population size 20, 40 and 80. This analysis was done on the first leg of the mission to keep computation time to a minimum. All of the random number seeds lead to convergence.

Table 2: Random Seed Analysis

Population	Standard Deviation of the # of generations	Standard Deviation of fitness	Maximum Fitness
20	3.216	15.50	99999.1
40	2.737	1.737	99999.1
80	3.897	1.565	99999.1

I looked at these three different populations to see if the larger population proved to be more reliable. As you can see in Table 2 the standard deviation in the # of generations between population size 20, 40 and 80 are very similar. It is also seen however that the standard deviation in the maximum fitness achieved with population 20 is greater than that of population size 40 and only decreases as the population is increased to 80on 80. All three populations reach a maximum fitness of 9999.1. The standard deviation of the fitness however shows that the larger population is more reliable.

6 CONCLUSION

From the analysis performed I can say that the algorithm is reliable in that it converged on every run. Looking at the random seed analysis the number of generations it took to converge did vary. The time of flight corresponding to the best fitness was not always the smallest; this is due to the trade off between the objective and the penalties.

This algorithm however did give good results for preliminary mission design. Figures 2 and 3 show that the trajectory is on target for a gravity assist at Mars and a final destination of Jupiter. These results could be used as a first guess in a different optimization technique.

7 FUTURE WORK

After looking at the results obtained by this classical simple genetic algorithm, I feel that future work could be done by implementing the equations of motion in a more modern genetic algorithm that has proven more reliable.

Also there are several ways to make this project more complex and therefore productive:

1.) Use real coding to maintain accuracy

2.) Solve for arrival and dep arture time based on the actual positioning of the planets.

3.) Create a region of approach at mars so as not to exactly target Mars and crash the s/c. (I was fairly lucky this did not occur)

4.) Look at the actual affects of the gravity assist in comparison to missions that do not utilize gravity assist.

5.) Optimizing the weighting factors for the penalty functions.

6.) Optimizing the number of segments to be made in each leg of the mission.

7.) Optimizing the genetic Parameters

Plenty of work remains to improve this project. Much of these suggestions could not be implemented due to time constraints.

8 REFERENCES

Goldberg, David E. (1989) *Genetic Algorithms in Search, Optimization, and Machine learning*, Addison Wesley Longman Inc.

Prussing, J.E. and Conway (1993), B.A. Orbital Mechanics, Oxford University Press Inc., p 120-137.