AAS 14-283

LOW-THRUST TRAJECTORY OPTIMIZATION FOR ASTEROID EXPLORATION, REDIRECT, AND DEFLECTION MISSIONS

Sam Wagner*[*] and Bong Wie[†]

Recent advances in ultra-lightweight solar sails and high-power solar electric propulsion systems are stimulating a renewed interest in low-thrust interplanetary missions. Such interest includes, the Asteroid Redirect Mission (ARM), which is a robotic mission concept with the goal of returning a small (approximately 7 m diameter, 500 metric ton) asteroids, to cis-lunar space using a high-power (40-kW class) solar electric propulsion (SEP) system. This paper describes an efficient low-thrust trajectory optimization tool that employs a new hybrid optimization algorithm. The new algorithm combines evolutionary programming for global optimization and traditional calculus-of-variations based method for local optimization. The proposed optimization tool is able to solve low-thrust optimization as well as high-thrust trajectory optimization problems. The tool will be used to determine feasible asteroid redirect missions as well as low-thrust planetary defense missions.

INTRODUCTION

In this paper, we investigate the feasibility of robotically capturing and returning a near-Earth asteroid (NEA) to Earth's vicinity from a low-thrust mission design perspective. Mission design parameters will be built off of results from a feasibility study performed by the Keck Institute for Space Studies [1]. The purposes of an Asteroid Redirect Mission (ARM) are two fold. The asteroid could also be used for the purposes of astronaut activities in the vicinity of an NEA, which will be valuable for future long-term deep-space NEA missions and Mars mission. In addition to being useful as a testing ground for astronauts, natural resources found on the retrieved asteroids could be retrieved and utilized.

REFERENCE MISSION DESIGN PARAMETERS

For the study of an ARM design in this paper, several mission design drivers must be taken into account, including the required mission ΔV, total time of flight, and the mass of the returned asteroid. In addition, several assumptions are made for the study. The first is an initial mass of 15,000 kg with a C_3 value of 2.0 km^2/s^2. The low thrust outward spiral is assumed to be the same as the outward spiral in the original Keck study [1]. The next assumption is that a lunar gravity-assist can be utilized to capture the asteroid if it arrives with a maximum C_3 value of 2.0 km^2/s^2 [1].

A mission design is performed utilizing two Busek BHT-20K Hall effect thrusters. Each thruster has the maximum power allowance of 20 kW with the maximum thrust of 1.08 N per thruster. Additional parameters of this reference solar electric propulsion (SEP) system are shown in Table 1. The spacecraft is initially launched into a low-Earth orbit with an Atlas V 551 launch vehicle. The low thrust Earth-departure transfer takes approximately 2.2 years to achieve a C_3 value of 2.0 km^2/s^2. The new hybrid algorithm, outlined in later sections, is then utilized to design both the heliocentric transfer to the asteroid and the asteroid towing transfer to an Earth vicinity with an arrival C_3 of no more than 2.0 km^2/s^2. The final objective function is formulated in a way to maximize the arrival asteroid mass at the Earth arrival.

[*]Graduate Research Assistant, Asteroid Deflection Research Center, Aerospace Engineering Department, Iowa State University, Ames, IA.

[†]Vance Coffman Endowed Chair Professor, Asteroid Deflection Research Center, Aerospace Engineering Department, Iowa State University, Ames, IA.

Table 1. A 40-kW SEP System with Two Busek BHT-20K Thrusters.

Power Per Thruster	20 kW
Mass Flow Rate	40 mg/s
Maximum Thrust (each)	1.08 N
Specific Impulse	2,750 s
Thruster Efficiency	70%
Number of Thrusters	2

To narrow down the list of possible target asteroids, we only consider asteroids with an Earth close encounter of less than 0.2 AU. In addition to close encounter requirement, we consider only asteroids with a maximum allowable Earth relative velocity of 3.0 km/s. With this list, two possible mission types were considered. With the first mission type, an entire small asteroid (approximately 7 m diameter, 500 tons) is returned to the Earth. The second mission type considered returns a small piece, of approximately the same size a the first mission type, of a much larger asteroid.

With the basic list of mission requirements determined, a method or formulating the problem must be determined. In the next section, the basics of the problem formulation and objective function are outlined. The new hybrid genetic-nonlinear programming algorithm, used to perform the optimization will also be discussed prior to discussing the results of the study.

LOW-THRUST PROBLEM FORMULATION

Low-thrust trajectory optimization methods typically fall into two categories, indirect and direct methods. Indirect methods are formulated with the calculus of variations, typically by creating a two-point boundary value problem, which provide an exact solution to the problem. Indirect solutions have the main advantage of low dimensionality and extremely accurate results. The disadvantages of indirect methods are the fact that the formulation is particularly sensitive to the initial guess for both the physical states and the non-physical Lagrange multipliers (costates).

Direct methods parameterize the low-thrust trajectory problem. This formulation results in a much larger design space, requiring large-scale nonlinear programming techniques to be used. While direct methods require a sufficiently accurate initial guess, the nonphysical Lagrange multiplies are not utilized, thus only an initial guess for physical parameters are used. The main drawback of direct methods is that they produce results with limited accuracy. In this paper, a new hybrid algorithm is presented to robustly calculate optimal low-thrust trajectories using a direct transcription model [2–4]. The problem of finding sufficiently accurate initial guesses is solved by utilizing a genetic algorithm to determine random initial guesses and perform the global optimization. The initial guesses for each member of population, for the genetic algorithm, are then used as the initial guess for a non-linear programming (NLP) solver. The variables for each member of population are then updated with the results of the NLP solver. The typical genetic operators, crossover, mutation, reproduction, and survival of the fittest operators are then used to create a new population on the genetic operator. The process is repeated until the genetic algorithm converges on a solution.

A Sims-Flanagan Low-Thrust Model

Figure 1 illustrates a low-thrust trajectory model described by Sims and Flanagan [2]. For each leg of the mission the low thrust trajectory arcs are modeled as a series of small impulsive ΔV maneuvers connected by conic 2-body arcs. Each leg of the trajectory is broken into N segments, with an impulse ΔV applied at the mid-point of each section.

The ΔV for each segment is not allowed to exceed a maximum magnitude, denoted by ΔV_{max-i}. This maximum impulsive ΔV is determined so that it cannot exceed the capability of the solar electric thruster at full power of two Busek BHT-20K Hall effect thrusters. The maximum ΔV for each segment can then be determined as

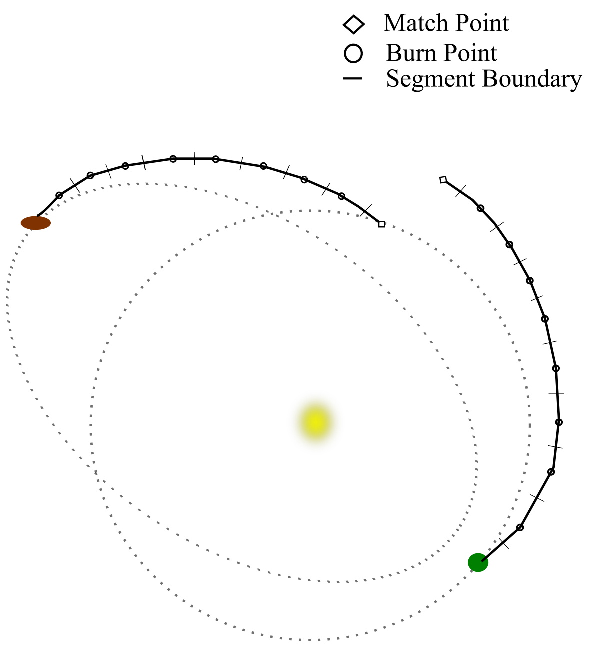

Figure 1. An Impulsive $\triangle$V Low-Thrust Trajectory Model by Sims and Flanagan [2].

$$\Delta V_{max,i} = \frac{T_{max}}{\dot{m}} \ln \frac{m_i}{m_i - \dot{m}\Delta t} \tag{1}$$

where the maximum time for each segment, Δt, is defined as

$$\Delta t = \frac{t_{leg}}{N} \tag{2}$$

In low-thrust optimization problems, the objective is often to maximize the final spacecraft mass. It is therefore necessary to update the spacecraft mass after each ΔV. For this purpose, Tsiolkovsky's rocket equation is expressed as

$$m_{i+1} = m_i e^{-\frac{-\Delta V_i}{g_0 I_{sp}}} \tag{3}$$

For each leg mission, the trajectory is propagated, via solutions to Kepler's problem, forward from the starting point and backward from the ending point to a match point. The forward and backward propagated half legs are required to meet at the match point, which is ensured in the optimization algorithm as

$$\mathbf{Z}_{fp} - \mathbf{Z}_{bp} = [\Delta r_x, \Delta r_y, \Delta r_z, \Delta V_x, \Delta V_y, \Delta V_z]^T \tag{4}$$

The Earth departure velocity is determined as a function of the Earth escape C_3, provided by the lunar gravity assist, and the departure ascension and declination angles. These two angles provide the direction of the Earth-system departure.

$$\mathbf{V}_{dep} = \sqrt{c_3} \begin{bmatrix} \cos\alpha \cos\beta \\ \sin\alpha \cos\beta \\ \sin\beta \end{bmatrix} \tag{5}$$

Sphere point picking is used as the method to determine the direction of each impulse maneuver. The direction of the ΔV is determined by two parameters u and ν, which range from 0 to 1. The two spherical pointing angles are determined from the u and ν decision variables as

$$\theta = u(2\pi) \tag{6}$$

$$\phi = 2\nu - 1 \tag{7}$$

A throttle parameter, ϵ, which also ranges from 0 to 1, is used to determined the magnitude of the ΔV applied at each burn point. This allows a ΔV range from 0 up to the maximum allowable for each segment as

$$\Delta V_i = \epsilon \Delta V_{max,i} \tag{8}$$

The final ΔV vector is calculated from the two spherical angles and ΔV magnitude, as follows:

$$\mathbf{\Delta V} = \Delta V \begin{bmatrix} \cos\theta \sin\phi \\ \sin\theta \sin\phi \\ \cos\phi \end{bmatrix} \tag{9}$$

ARM Design Problem Formulation

With the groundwork formulated for the Sims-Flanagan low-thrust problem, the next step is to develop a cost function for the hybrid GA-NLP solver. The hybrid solver is used to determine the initial launch date, time-of-flights, departure variables, and the 3 spherical burn variables for each ΔV. The Earth arrival velocity vector is determined in the same fashion as the Earth departure velocity.

The desired initial and final desired state variables for each leg of the mission are defined by the problem as functions of launch and final arrival times. In additional to the launch date, the initial departure state is also taken to be a function of launch C_3, as well as the launch ascension and declination, α and β, respectively.

$$\mathbf{Z}_{Earth-Dep} = \mathbf{f}\left(t_{Earth-Dep}, C_3, \alpha, \beta\right) \tag{10}$$

$$\mathbf{Z}_{Ast-Arr} = \mathbf{f}\left(t_{Ast-Arr}\right) \tag{11}$$

Because the proposed solver is an unconstrained minimization solver, all nonlinear constraints must be handled directly by the cost function. The two match point radius and velocity magnitudes are calculated for use in the final objective function as

$$r_i = |\mathbf{r}_{fp} - \mathbf{r}_{bp}| \tag{12}$$

$$v_i = |\mathbf{v}_{fp} - \mathbf{v}_{bp}| \tag{13}$$

The final objective function is constructed to maximize the final mass, as well as to ensure the match point for each leg is met, as

$$J_{dep} = w_1 \ln \frac{m_0}{m_{ast-arr}} + w_2 \ln \frac{m_{ast-dep}}{m_{Earth-arr}} + w_3\left(r_1 + r_2\right) + w_3\left(v_1 + v_2\right) \tag{14}$$

where the weights w_i are used to ensure each of the elements added to the cost function have approximately the same order of magnitude. Using weights that ensure all the values are in canonical units works well for the hybrid GA-NLP solver. Details of the hybrid solver used to optimize the low thrust asteroid redirect mission are discussed in the next section.

DEVELOPMENT OF THE HYBRID GENETIC-NLP ALGORITHM (GA-NLP)

The hybrid Genetic-Nonlinear Programming algorithm has been shown to work well with complex optimization problems, requiring no prior knowledge or approximate solution to the problem. The algorithm will be utilized to find optimal solution to high-thrust, Cassini type mission, and it will be shown to work well for low-thrust asteroid redirect missions.

Genetic Algorithm

The heart of a genetic algorithm is the stochastic simulation of natural selection, reproduction, and mutations found in nature. In these simulations genetic operators are used to 'evolve' an initial population, through genetic operators, in order to determine a best fitness design [5]. The purpose of this section is to develop and implement a genetic algorithm for the multiple gravity assist (MGA), MGA-DSM, and other complex mission design problems.

A genetic algorithm (GA) is a stochastic optimization method based on the principles of evolution. Genetic algorithms perform a probabilistic search by evolving a randomly chosen initial population. The population is just a series of sets of variables that are evaluated by a cost function, in this case the cost functions previously developed. The advantage of using evolutionary methods over traditional optimization methods is that no initial solution is necessary. This helps ensure, but does not guarantee, that solutions are not confined to some locally optimal solution. Genetic algorithms also perform well in very complex nonlinear design spaces. Despite all the advantages, evolutionary algorithms also have a downside. Almost always, they require a greater number of cost function evaluations than traditional calculus based methods, increasing the computational requirements. Additionally evolutionary algorithms do not make use of gradients, so there is no proof of convergence. It should also be noted that genetic algorithms were developed for constrained minimization problems and may not always perform well for unconstrained minimization.

The basic genetic operators are, selection, reproduction, crossover, mutation, and elitism. The algorithm developed in this section can utilize a number of different selection, reproduction, crossover, and mutation methods. Each of the methods will be discussed in detail in later sections.

In a real valued genetic algorithm (as implemented for this study), each variable is represented by either it's integer or real number value. Each real and integer variable is referred to as a gene. A complete set of variables that define a solution to the problem are then concatenated to form a complete chromosome. One chromosome correspond to one member of the entire population, which can number in the hundreds of thousands for some problems. This real valued approach has advantages over the traditional binary representation of variables because the real valued variables are not discretized to a set resolution. For the GA individual input variables include upper and lower bounds, size of the population, and number of generations the population is to be run out, and various other parameters to control the flow and output of the final optimization routine.

Then first step in the evolutionary process is to use a uniform random number generator to generate the initial population, ensure a random somewhat evenly dispersed population. Each chromosome is generated using the integer and real valued user supplied upper and lower bounds. The process is then repeated until the entire population has been filled.

From this point each member of the population is assigned a fitness value, via a user supplied cost function. The genetic operators are then used to generate a new population from the initial parent population. This is the crux of how genetic algorithms are able to evolve to more fit populations. This process continues until the genetic algorithm is told to exit and output the final population. Common stopping conditions include stopping after a certain number of iterations, monitoring when the best fit solution stop changing, or monitoring when the average cost function value approaches the population minimum. For this study the stopping condition is always running the genetic algorithm out by a user specified number of generations. The sequence of operations of the core operations of the genetic algorithm is illustrated in Figure 2. One secondary operator, elitism, is used in the genetic algorithm as well. The elitism mechanism ensures that the best fit solution(s) are not lost from generation to generations. A simple elitism operator is used, in which the

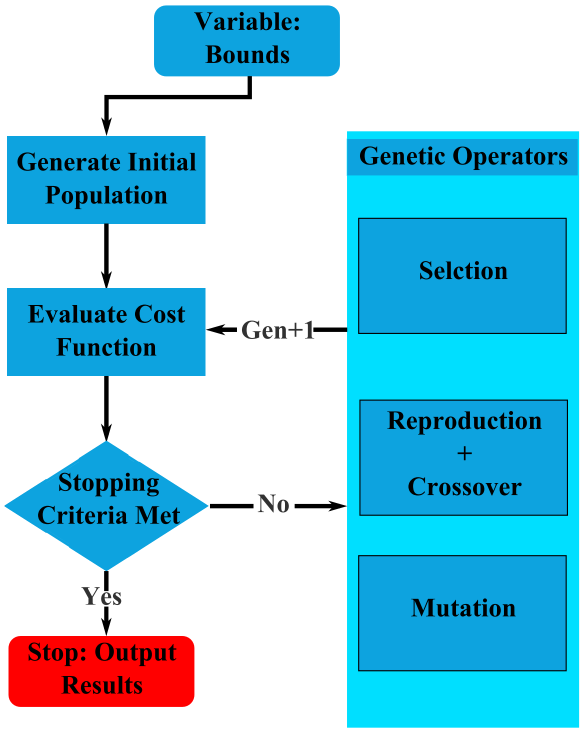

Figure 2. Flow Chart for the Simple Real- and Integer-Valued Genetic Algorithm.

top two solutions from the parent population are directly inserted into the next generation.

Selection Types In genetic algorithms, selection operator, also the first genetic operator, is used to ensure that the best fit solutions are chosen to pass on their genes through the reproduction and crossover operators. The selection operator is essentially the operator that represents the survival of the fittest principle. A total of two selection types have been implemented in this genetic algorithm one based solely on a roulette selection method, and another that combines roulette and tournament selection. These two operators are by no means the only possible selection methods, but that are some of the simplest to use and prove to work well for the mission design problems.

Roulette selection is a fitness proportionate selection method in which better fit individuals are more likely to be chosen for reproduction and crossover. There are four fitness types that are used in the roulette selection method described in this section. The ultimate goal is to get a normalized fitness that ranges from 0 to 1. More fit members of the population will have a higher normalized fitness value, thus they will be more likely to be chosen for further genetic operations.

The first fitness type is the raw fitness, $r(i)$, which is the fitness values provided directly from the cost function for each population member. In this case i represents the chromosome/population number. The raw fitness is then standardized, depending on whether the problem is being minimized and maximized. If the problem is a minimization problem, the standardized fitness is the same as the raw fitness as

$$s(i) = r(i) \tag{15}$$

However, if the problem is a maximization problem, the standardized fitness becomes

$$s(i) = r_{max} - r(i) \tag{16}$$

Now the fitness is adjusted so that individual fitnesses lie between 0 and 1. The adjusted fitness is larger for better individuals. The advantage of this optional step is that small differences in the most fit individuals, those that approach an adjusted of 0, are exaggerated. This has a larger benefit when the population improves by emphasizing small differences in individuals. This effect is most exaggerated in problems where the best solution has a cost function near 0, [6, 7] as follows:

$$a(i) = \frac{1}{1 + s(i)} \tag{17}$$

Now that the adjusted fitness values lie between 0 and 1 the next step is to normalize the whole population, so the sum of each normalized fitness value is 1. Ultimately, this is necessary because individuals are chosen through the use of random numbers, that also in 0 and 1 range, as

$$n(i) = \frac{a(i)}{\sum_{j}^{n} a(j)} \tag{18}$$

The actual selection of individuals, based on normalized fitness, is done in the following manner. Firstly the normalized fitnesses are sorted from largest to smallest. Next a random number is generated. This random number will decide which individual is then selected. The actual selection of an individual is done by a summation of the normalized fitness values. The individual that causes the summation value to be greater than the random number is chosen to move along through the rest of the genetic operators. This process is repeated and both of the individuals then go through the reproduction or crossover operators. Thus, more fit individuals are more likely to be passed on to the next generation.

Tournament selection is a greedy over selection method that also utilized the roulette selection method. Two individual are chosen using roulette selection and then compete to see which individual is allowed to pass on its genetic material. This is done through the direct comparison of their normalized fitness function. The individual with the best fitness is chosen as the one that gets to pass through to the reproduction and crossover operations. The process is repeated twice, resulting in two individuals who's genetic material can be passed on.

This selection method has a distinct advantage in that it drives down the raw fitness and average of the fitness function in fewer generations than pure roulette selection. With all greedy over selection methods there is a risk that genetic diversity, which could help out the population in later generations, will be lost. Therefore tournament selection should be used with care by the user. In the orbital problems discussed in this paper, tournament selection does appear to work well.

Reproduction and Crossover Operator Types While the selection operator determines which population members get the privilege of reproducing and passing on their genetic material to future generations, the crossover and reproduction operators decided what to do with that genetic material. In the algorithm utilized in this paper the user must supply the probability in which an individual will under go either reproduction or crossover. These probabilities must total up to 1.0 (100%), with values typically being close to 0.9 and 0.1 for crossover and reproduction respectively. It should also noted that both operations require two parent and produce two offspring.

The simplest of these two operators is reproduction. In traditional genetic algorithm terms this means that the operation is a fitness-proportionate process in which individuals are allowed to directly pass to the next generation with a probability proportionate to their fitness values. Reproduction occurs when the two parents are passed directly from the parent generation into the next generation. As with every other critical procedure in the genetic algorithm, whether the parents undergoes reproduction or crossover is chosen by a random process with the probabilities of each as the input. In the implemented genetic algorithm a random number is generated, if the random number is greater than the user given crossover probability the parents undergoes reproduction. If not, the a crossover operation is applied. This, of course, is the reason that the two probabilities must total up to 100%. Without reproduction (and also mutations) genetic algorithms would be no more useful than a random search.

The crossover process is a process in which biological reproduction is used to allow new individuals, with new and unique genetics, to be created. The purpose of each crossover type is to promote genetic diversity

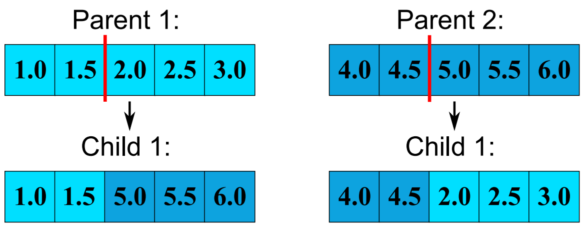

Figure 3. Simple illustration of the single point crossover with the 3rd variable chosen as the crossover point.

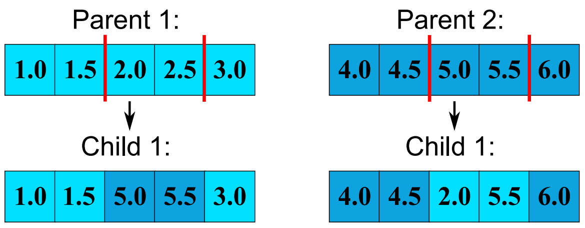

Figure 4. Simple Illustration of the Double Point Crossover with the 3rd and 5th Variables Chosen as the Crossover Points.

and expand, in a controlled way, the search of the design space. Unlike the reproduction process, crossovers allow new points in the design space to be searched. The crossover operation does this by producing two offspring from two parents that contain genetic material from both parents. A total of 6 distinct crossover types have been implemented for use with the genetic algorithm described in this paper. The user supplies which crossover type should be used, making the final genetic algorithm suitable for many different optimization problems. For the mission design problems discussed in this paper the double point or arithmetic crossover has proven to work very well.

Single Point Crossover The first crossover operator implemented in the genetic algorithm is single point crossover. In the single point crossover operator one variable is randomly chosen to be the crossover point. The first child is created by copying everything from the first parent prior to the crossover point and then everything from the second parent after the crossover point. The single point crossover process is shown graphically in Fig. 3.

Double Point Crossover Double point crossover is the very similar to the single point crossover operator. In this case two crossover points are randomly chosen. After the two points are chosen two new individuals are formed, as shown by Fig. 4, by swapping the variables between the two crossover points.

Uniform Crossover Uniform crossover has been shown to be a very effective method for promoting genetic diversity, and in turn the discovery of new useful chromosomes [5, 8]. In uniform crossover each variable in the chromosome is a crossover point. The two offspring are generated by a series of virtual coin flips. The virtual coin flip is used to decide which offspring gets the genetic material from the separate parent individuals. Graphically this process is similar to both the single and double point crossover methods.

Arithmetic The arithmetic crossover operator, sometimes referred to as the whole arithmetic operator as implemented in the algorithm, linearly combines the chromosomes from the two parents. The two offspring are created according to Eqs. (19) and (20). In this case the variable α is chosen to be between 0 and 1. It is possible to use other bounds for α as long as the algorithm checks to ensure none of the resulting variable changes are not out of bounds as

$$C_1 = \alpha P_1 + (1-\alpha) P_2 \tag{19}$$

$$C_2 = (1-\alpha) P_1 + \alpha P_2 \tag{20}$$

A simple illustration of the arithmetic operator can be seen in Fig. (5). This illustration shows a chromo-

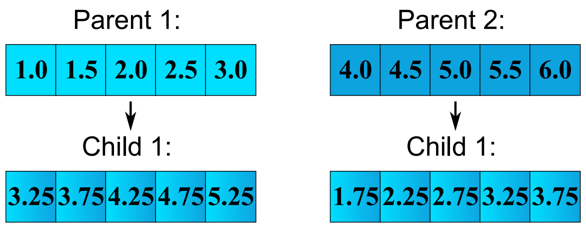

Figure 5. Simple Illustration of the Arithmetic Crossover Operator with $\alpha = 0.25$.

some consisting of only real valued variables, but the method can be easily extended to include integer value variables as well.

Heuristic Approach The heuristic crossover operator determines a search direction from the two parent members. This crossover method is similar to the arithmetic operator, but extend the better parent in the direction from the worse to the better parent. The two offspring are created as

$$C_1 = P_{best} + r \cdot (P_{best} - P_{worst}) \tag{21}$$

$$C_2 = P_{best} \tag{22}$$

where the variable r is a random number between 0 and 1.

Mutation Types The last genetic operator, which the newly generated population must pass through, is the mutation operator. The probability that a mutation will occur and the desired type of mutation are the last two user inputs. The probability that a mutation will occur should be small, typically less than 0.05 (5%). When the mutation probability is set too high the genetic algorithm will start to resemble a simple random search. If the user doesn't wish to make use of the mutation operator, a probability of 0 can be entered as well.

Allowing the genetic algorithm to utilize a mutation operator has several advantages. Mutations are used to help maintain genetic diversity in a population as it ages. Often times after many generations the population can lose genetic diversity and stagnate at a local minimum. The mutation operator helps to prevent this from happening. By introducing (or in some cases reintroducing) changes in a chromosome or individual gene it is possible for better more fit individuals to appear. The mutation process changes the value of one randomly selected gene (i.e. variable) in the chromosome. In some situations changes to an individual variable can greatly improve (or alternatively worsen) the individuals fitness.

As with the crossover operator several separate types of mutations have been implemented. In practice, certain mutation types may work best for each separate problem types. A total of 3 mutation types have been implemented in the hybrid algorithm, which can all be applied to both integer and real valued variables.

Uniform Mutation The mutation operator replaces the value of a randomly chosen gene with a uniform random value between the specified variable's upper and lower bounds. The advantage of this operator is that it allows the population's genetic diversity to be maintained by allowing any value within the user specified range. Similar non-uniform mutation can be utilized as well, but have not been implemented in this algorithm.

Boundary Mutation With boundary mutations, a single, randomly selected, variable is changed to either the variables upper or lower bounds. The choice to use either the upper or lower bound is done through a virtual coin flip. This method best promotes genetic diversity when used in combination with either the arithmetic or heuristic crossover operators.

Sliding The sliding mutation operator is similar in performance to uniform mutations. A randomly selected gene is multiplied by a randomly selected value between 0.8 and 1.2. In essence this operator allows the gene to be slightly mutated $\pm 20\%$. The operator then checks to ensure the neither of the variable bounds have been violated, if a violation occurs the variable is set to the closest bound.

NON-LINEAR PROGRAMMING SOLVER

Evolutionary algorithms, particularly genetic algorithms, are well suited for global optimization. However, when genetic algorithms are utilized for MGA-DSM missions with multiple (3+) gravity-assists and low thrust missions they often won't converge on the best global solution. On the other hand non-linear programming (NLP) solver software will often times only converge to locally optimal solutions. By modifying the genetic algorithm to utilize an NLP solver during the cost function evaluation stage, to find at least locally optimum solutions, the hybrid GA-NLP algorithm is able to efficiently solve complex problems. The hybrid algorithm also allows the population size and number of generations run to be reduced significantly over the standard genetic algorithm.

Utilizing an NLP solver during the cost function evaluation in the hybrid algorithm has its downside. Namely the algorithm can only optimize functions that are continuous an at least twice differentiable (near the local optimum). For this reason the NLP solver doesn't iterate upon integer variables (in mission design problems usually planetary gravity-assist orders, arrival planets, etc), which would often introduce large discontinuities in total required mission ΔV. Except for the planetary gravity-assist order both high an low thrust mission design problems are good candidates for NLP solvers. Thus, both types of mission design problems are ideal candidate for the hybrid GA-NLP solver

UNCMIN Based Non-linear Programming Solver

The NLP solver implemented in the hybrid algorithm is based off the UNCMIN optimization algorithm (originally written in Fortran 77) introduced in [9,10]. The UNCMIN algorithm is classified as a quasi-newton algorithm based on steepest descent methods. This NLP solver requires only the cost function values, but does allow for user supplied gradient and hessians. The NLP algorithm implemented in the hybrid algorithm is a modern (Fortran 90/95) version of the original UNCMIN algorithm. Some changes have been made, to ensure the algorithm can ultimately be easily translated to run on modern GPUs utilizing PGI's CUDA Fortran compiler.

The NLP algorithm is an extremely modular system of algorithms capable of multiple step selection strategies and multiple derivative calculation techniques (1^{st} and 2^{nd} order). Derivatives can be calculated analytically, finite difference (forward and central difference), and the Hessian can be calculated with a BFGS approximation. For the MGA-DSM problem gradients were calculated with forward finite difference, while the Hessians were calculated with a BFGS approximation. Calculating the Hessian via finite difference is a very computationally intensive process, so the BFGS Hessian approximation is used. Near locally minimum solution the Hessian approximation gives sufficiently accurate results while minimizing the NLP solver run time.

HYBRID ALGORITHM IMPLEMENTATION

The final hybrid GA-NLP algorithm is able to achieve significant performance increases, in terms of computational cost and run time, over the standard GA implementation. This is achieved by optimizing each individual population member with the NLP solver, rather than simply evaluating the cost function and relying on the GA for the entire optimization process. Through this process we are able to utilize the strengths associated with both genetic algorithms and non-linear programming solvers. The genetic algorithm functions as the method for global optimization, while the NLP solver initially ensures such global optimization.

ARM DESIGN RESULTS

For our study, it is assumed that the asteroid return mass will be 500 metric tons. The first two asteroid presented (2008 HU_4 and 2006 RH_{120}) are small asteroids that can be towed by the spacecraft to the Earth vicinity. The last two asteroid, 2000 SG_{344} and 2004 EO_{20}, have diameters much larger than 7 m. In this case it is assumed that an approximately 500 metric tons boulder would be collected and return to the Earth.

The objective of this study is to determine an algorithm that can be utilized for automated searches of asteroids. The cost function is formulated in such a way as to automate the choice of decision variables

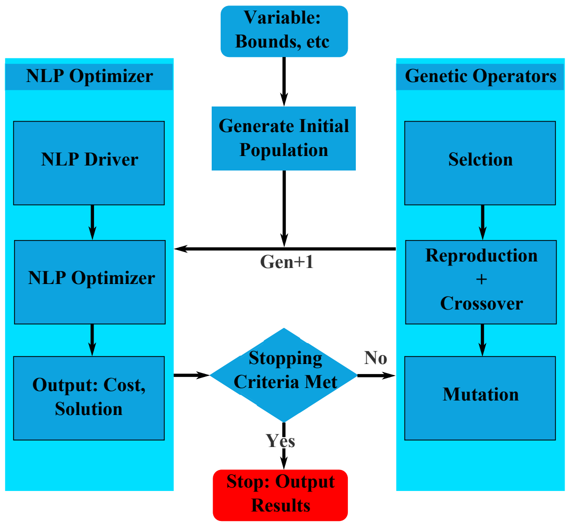

Figure 6. Flow Chart for Hybrid GA-NLP Algorithm.

Table 2. Initial Earth-Departure Spacecraft Parameters.

Thrust Power, kW	40
Specific Impulse, s	2,750
Number of Thrusters	2
Efficiency	70%
S/C Dry Mass, kg	5,500
S/C Mass at Earth Departure, kg	15,000

such as launch dates, times of flights, and all other mission parameters. With this approach asteroid redirect mission can be calculated for any asteroid. The results presented here are the best missions found in our search.

The first asteroid examined is asteroid 2008 HU_4, which is the reference asteroid considered for the original Keck study [1]. The initial Earth departure spacecraft parameters for all of the mission are shown in Table 2. For the asteroid redirect mission, the spacecraft dry mass is assume to be 5,500 kg, with 9,500 kg of xenon propellant.

The mission to 2008 HU_4 has earliest launch date out of 4 candidate asteroids, as well as the lowest Heliocentric ΔV for the transfer to the asteroid, at approximately 160 m/s. However, with an optimal launch date of May 24th, 2016 the mission may not be feasible. It should be noted, as with the Keck study, that the launch date can be pushed back with minimal impact to the final mass returned to the Earth. This is due to the efficiency of the solar electric propulsion system and the low mass, relative to the return tow, during the initial heliocentric phase. The optimal return mass found for asteroid 2008 HU_4 is approximately 509 metric tons, leaving 3,500 kg of xenon propellant left for Earth proximity operations. The details of this mission design can be found in Table 3.

The second asteroid redirect mission found is to asteroid 2006 RH_{120}. This mission has a much shorter total flight time, of approximately 6.6. In addition the Earth return transfer only requires a ΔV of 160 m/s. The low return ΔV enables a total return mass of approximately 511 metric tons, with over 5700 kg of xenon left for Earth proximity operations. Further details of the 2006 RH_{120} redirect mission can be found in Table 4.

Two candidates for the alternative mission type, in which pieces of a larger asteroid, are return to the Earth

Table 3. ARM Design Summary for Asteroid 2008 HU_4.

Transfer to NEA	
Earth Departure Date	24-May-16
Escape C_3, km^2/s^2	2
Flight Time, days	277.8
Heliocentric ΔV, km/s	0.161
Asteroid Arrival Mass, kg	14,910.47
Asteroid Arrival Date	25-Feb-17
Asteroid Mass (Estimated), kg	500,000
Transfer to Earth Vicinity	
Asteroid Stay Time	293.1
Departure Date	15-Dec-17
Departure Mass, kg	514,910
Flight Time, days	3,408.2
Heliocentric ΔV, km/s	0.311
Arrival Mass, kg	509,000
Arrival C_3, km^2/s^2	0.95
Earth Arrival Date	16-Apr-27
Total Xenon Used, kg	**6,000**
Remaining Propellant, kg	**3,500**
Total ΔV, km/s	**0.473**
Total Flight Time, yrs	**10.9**

are considered below. The details for asteroids 2000 SG_{344} and 2004 EO_{20} are provided in Tables 5 and 6 respectively. The mission to asteroid 2000 SG_{344} has an Earth departure date of Feb. 5th, 2023, a total flight time of approximately 6.5 years, and a return mass of 507.7 metric tons. This mission has the lowest propellant mass left for Earth proximity at 2200 kg. The second mission to asteroid 2004 EO_{20} has the lowest total flight time of 5.9 years with a returned mass of 508 metric tons.

Table 4. ARM Design Summary for Asteroid 2006 RH$_{120}$.

Transfer to NEA	
Earth Departure Date	14-Apr-22
Escape C_3, km^2/s^2	2
Flight Time, days	1,380.3
Heliocentric ΔV, km/s	1.316
Asteroid Arrival Mass, kg	14,258.61
Asteroid Arrival Date	23-Jan-26
Asteroid Mass (Estimated), kg	500,000
Transfer to Earth Vicinity	
Asteroid Stay Time	442.7
Departure Date	11-Apr-27
Departure Mass, kg	514,286
Flight Time, days	599.1
Heliocentric ΔV, km/s	0.160
Arrival Mass (kg)	511,253
Arrival C_3, km^2/s^2	0.43
Earth Arrival Date	30-Nov-28
Total Xenon Used, kg	**3,747**
Remaining Propellant, kg	**5,753**
Total ΔV, km/s	**1.475**
Total Flight Time, yrs	**6.6**

Table 5. ARM Design Summary for Asteroid 2000 SG$_{344}$.

Transfer to NEA	
Earth Departure Date	5-Feb-23
Escape C_3, km^2/s^2	2
Flight Time, days	768.6
Heliocentric ΔV, km/s	1.600
Asteroid Arrival Mass, kg	14,135.70
Asteroid Arrival Date	15-Mar-25
Asteroid Mass (Estimates), kg	500,000

Transfer to Earth Vicinity	
Asteroid Stay Time	155.3
Departure Date	17-Aug-25
Departure Mass, kg	514,136
Flight Time (days)	1,461.0
Heliocentric ΔV, km/s	0.311
Arrival Mass, kg	507,700
Arrival C_3, km^2/s^2	0.92
Earth Arrival Date	17-Aug-29
Total Xenon Used, kg	**7,300**
Remaining Propellant, kg	**2,200**
Total ΔV, km/s	**1.912**
Total Flight Time, yrs	**6.5**

Table 6. **ARM Design Summary for Asteroid 2004 EO$_{20}$.**

Transfer to NEA	
Earth Departure Date	18-May-23
Escape C_3, km^2/s^2	2
Flight Time (days)	444.0
Heliocentric ΔV, km/s	2.690
Asteroid Arrival Mass, kg	13,577.70
Asteroid Arrival Date	4-Aug-24
Asteroid Mass (Estimated), kg	500,000

Transfer to Earth Vicinity	
Asteroid Stay Time	352.1
Departure Date	22-Jul-25
Departure Mass, kg	51,578
Flight Time (days)	1,347.0
Heliocentric ΔV, km/s	0.287
Arrival Mass, kg	508,132
Arrival C_3, km/s	1.46
Earth Arrival Date	30-Mar-29
Total Xenon Used , kg	**6,868**
Remaining Propellant, kg	**2,632**
Total ΔV, km/s	**2.974**
Total Flight Time, yrs	**5.9**

CONCLUSION

A low-thrust mission trajectory design method has been implemented in conjunction with the new hybrid GA-NLP algorithm to determine feasible asteroid redirect trajectories. The algorithm has been developed to automate the search for possible target asteroids. Several possible target asteroids have been analyzed and presented in this paper. It has been shown that a 500 ton asteroid can be retrieved with a 15 ton spacecraft. With typical trajectories, several tons of propellant are left for earth proximity operations, which could be utilized to establishing a stable orbit and/or for station keeping maneuvers. In a future study, a constrained nonlinear solver will be added to the hybrid algorithm, which will enforce the match point through nonlinear constraints instead of penalty functions. In addition, indirect method will be implemented, allowing for high-fidelity searches to be automated and optimized.

ACKNOWLEDGMENT

This research has been supported in part by NASA's Iowa Space Grant Consortium and a NASA Innovative Advanced Concepts (NIAC) Phase 2 project.

REFERENCES

[1] Keck Institute for Space Studies, "Asteroid Retrieval Reasibility Study," tech. rep., Keck Institute for Space Studies, California Institute of Technology, Jet Propulsion Laboratory, April 2012.

[2] Sims, J.A. and Flanagan, S.N., "Preliminary Design of Low-Thrust Interplanetary Missions," *AAS/AIAA Astrodynamics Specialist Conference*, No. AAS 99-338, 1999.

[3] C. H. Yam, D. Di Lorenzo, and D. Izzo, "Constrained global optimization of low-thrust interplanetary trajectories," *Evolutionary Computation (CEC), 2010 IEEE Congress on*, July 2010, pp. 1–7, 10.1109/CEC.2010.5586019.

[4] Englander, Jacob A., *Automated Trajectory Planning for Multiple Flyby Interplanetary Missions*. PhD thesis, University Of Illinois, Aug. 2012.

[5] Vavrina, Matthew, *A Hybrid Genetic Algorithm Approach to Global Low-thrust Trajectory Optimization*. PhD thesis, Purdue University, 2010.

[6] Goldberg, David, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, First ed., 1989.

[7] Koza, John, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, Massachusetts: The MIT Press, First ed., 1992.

[8] Syswerda, Gilbert, "Uniform Crossover in Genetic Algorithms," *Proceedings of the 3rd International Conference on Genetic Algorithms*, San Francisco, CA, USA, Morgan Kaufmann Publishers Inc., 1989, pp. 2–.

[9] Schnabel, Robert and Koontz, John and Weiss, Barry, "A Modular System of Algorithms for Unconstrained Minimization," *University of Colorado at Boulder: Department of Computer Science*, 1985.

[10] Kahaner, David and Moler, Cleve and Nash, Stephen, *Numerical Methods and Software*. Englewood Cliffs, New Jersey 07632: Prentice Hall Series in Computational Mathematics, 2nd ed., 1989.