JOURNAL OF SPACECRAFT AND ROCKETS
Vol. 40, No. 6, November–December 2003

Genetic Algorithm and Calculus of Variations-Based Trajectory Optimization Technique

Adam Wuerl*
University of Washington, Seattle, Washington 98195
and
Tim Crain† and Ellen Braden‡
NASA Johnson Space Center, Houston, Texas 77058

A genetic algorithm is used cooperatively with the Davidon–Fletcher–Powell penalty function method and the calculus of variations to optimize low-thrust, Mars-to-Earth trajectories for the Mars Sample Return Mission. The return trajectory is chosen thrust-coast-thrust a priori, has a fixed time of flight, and is subject to initial and final position and velocity equality constraints. The global search properties of the genetic algorithm combine with the local search capabilities of the calculus of variations to produce solutions that are superior to those generated with the calculus of variations alone, and these solutions are obtained more quickly and require less user interaction than previously possible. The genetic algorithm is not hampered by ill-behaved gradients and is relatively insensitive to problems with a small radius of convergence, allowing it to optimize trajectories for which solutions had not yet been obtained. The use of the calculus of variations within the genetic algorithm optimization routine increased the precision of the final solutions to levels uncommon for a genetic algorithm.

Nomenclature

$\boldsymbol{C}_p$	=	position constraint weighting vector in heliocentric Cartesian coordinates, m^{-1}
$\boldsymbol{C}_v$	=	velocity constraint weighting vector in heliocentric Cartesian coordinates, s/m
F_i	=	absolute fitness of ith genome, dimensionless
f_i	=	normalized fitness of ith genome, dimensionless
$\boldsymbol{G}_p$	=	difference between actual and desired final position vectors, m
$\boldsymbol{G}_v$	=	difference between actual and desired final velocity vectors, m/s
H	=	Hamiltonian matrix, dimensionless
m_{initial}	=	initial mass of the spacecraft at the Martian sphere of influence, kg
$m_{\text{propellant}}$	=	mass of propellant used during heliocentric transfer, kg
n	=	number of genomes in the current generation, dimensionless
P	=	penalty function value, dimensionless

Introduction

TECHNIQUES for optimizing low-thrust spacecraft trajectory problems have become increasingly important as more missions are designed around high specific impulse, low-thrust propulsion systems. Both direct and indirect routines have been used to optimize low-thrust spacecraft trajectories; however, for some scenarios, convergence to the optimal solution is time-consuming, tedious, and sometimes not even possible. Direct methods that solve for controls to optimize the objective function directly, often via a gradient-based search, suffer from two major drawbacks. First, because search direction is ultimately driven by the local value of the gradient vector, the solution can converge on local, rather than global, minima, resulting in a final solution that is not globally optimal and cannot be further optimized. Second, the optimal solution often has a small radius of convergence, requiring that the guesses for the initial parameters be close to the optimal answer.[1] Indirect methods, such as the calculus of variations, obtain optimal results by solving for the costates of a related boundary value problem and not for the controls directly. Although indirect methods are generally more likely to find a true, rather than local, optimum, both direct and indirect approaches share many of the same drawbacks, most notably a small radius of convergence.[2]

Therefore, it is vital to choose initial parameter values intelligently; failure to do so will either dramatically increase the required computation or preclude obtaining a solution entirely. When indirect methods are used, where the optimization parameters are generally not related to the trajectory in an intuitive or straightforward manner, there may not be knowledge of the parameter bounds or their sensitivity. A common strategy to improve initial parameter selection uses previously optimized parameter values from a similar problem as an initial guess. If no closely related solutions exist, initial values are found by optimization of an entire series of intermediate problems relating the new scenario to one with a known solution, a procedure known as homotopy analysis.[3] Unfortunately, this method is time consuming, does not always succeed, and previously optimized solutions are not always available.

More recently, the genetic algorithm (GA) has emerged as an alternative optimization technique that avoids these shortcomings because it does not attempt to extract information from complicated or ill-behaved gradients. It is also less hampered by discontinuities and irregularities in the solution space.[1,4] GAs do not utilize initial guesses; they are able to search for the global minimum without knowing its approximate location by statistically sampling the parameter space for the optimal solution. Although GAs also require the user to set initial parameters such as population size and convergence criteria, these parameters are less difficult to determine because the results of previous research, an understanding of statistical analysis, and even intuition are often enough to enable an intelligent selection. If the initial population is large enough, this method is virtually guaranteed to be accurate, but if the GA is not combined with a local search algorithm, a common drawback is insufficient precision in the final solution.[1,2,4,5] Crain et al.[6] have addressed this problem by using a GA in series with a gradient-based optimization technique, feeding a GA's results to a recursive

Received 16 August 2002; revision received 7 March 2003; accepted for publication 14 March 2003.

*Student, Department of Aeronautics and Astronautics. Member AIAA.
†Engineer, EG5/Advanced Mission Design. Member AIAA.
‡Engineer, EG5/Advanced Mission Design.

quadratic programming (RQP) module to increase solution precision efficiently.

The technique presented in this paper is an extension of the serial GA–RQP technique that more closely couples the different search methods, combining a GA with the Davidon–Fletcher–Powell (DFP) penalty function and calculus of variations (COV) method to optimize low-thrust, Mars-to-Earth trajectories. The DFP–COV method and the GA work cooperatively throughout the optimization process, optionally providing mutual and active feedback as they sample the parameter space seeking the global minimum. After the approximate location of the global optimum is found, the DFP–COV technique is used to further refine the results of the cooperative effort. The cooperative technique works around the GA's inherent precision limitation and the COV's sensitivity to initial guesses to obtain quickly and reliably optimal solutions superior to those obtained with other methods.

A similar GA and COV-based technique was implemented by Hartmann et al.[7] to find families of Pareto optimal solutions, or solutions to problems with a set of competing objectives. The authors' success at finding families of Pareto optimal solutions to the Earth–Mars transfer problem, with final mass and time of flight as competing objectives, supports the use of GAs in trajectory optimization problems. The research presented in this paper complements previous efforts, but differs by optimizing a single performance parameter, solving the return Mars–Earth transfer, and implementing a Lamarckian learning strategy not studied by the Hartmann et al.[7] Pareto GA.

Mars Sample Return Mission

The results presented in this paper support the Mars Sample Return Mission (MSRM), described in detail by Vadali et al.[8] A chemical propulsion system will be used to land a spacecraft on the surface of Mars to collect samples. After ascent to low-Mars orbit (LMO), a solar electric propulsion (SEP) system will perform the Earth return, which consists of three major segments: a spiral from LMO to the Martian sphere of influence (SOI), a heliocentric transfer from the Martian SOI to the Earth's SOI, and a spiral from the Earth's SOI to low Earth orbit. The technique presented in this paper has been used to optimize the heliocentric leg of this journey. The initial heliocentric spacecraft position and velocity are those of Mars, obtained from an ephemeris model.[9] The heliocentric transfer is modeled with two-body dynamics, and a fixed-step, fourth-order Runge–Kutta integrator is used to forward propagate the equations of motion. The spacecraft's target position and velocity are equal to Earth's position and velocity. Details of the SEP system are shown in Table 1.

Past MSRM Optimization Methods

Past efforts to optimize the MSRM return trajectory were accomplished using the DFP penalty function and COV method.[10] The heliocentric return trajectory was modeled as three arcs, chosen thrust-coast-thrust a priori, and described by 38 parameters; there was no switching function to determine the thrust sequence. Two of the parameters defined the duration of the first thrust arc and duration of the coast arc. Both of the thrust arcs were partitioned into three equal-duration segments, each of which was described in the COV problem by a six-element costate vector of Lagrange multipliers, which provided the remaining 36 parameters. Additional thrust and coast arcs could be added, potentially to improve performance, at the expense of additional duration and costate parameters. The state vector was required to be continuous at arc boundaries, but the costate vector was not. Similarly, the spacecraft trajectory was subject to initial and final position and velocity constraints, as described earlier, but transversality (boundary) constraints were not imposed on the costates. The DFP method was used to determine the values of the 38 parameters that minimize a penalty, rather than objective, function, which enables the constrained minimization problem to be simplified by reexpression of the constraints as terms in the penalty function, shown as Eq. (1):

Table 1 SEP system details

Parameter	Value
Specific impulse, s	3800
Thruster input power, kW	3.00
Thruster efficiency	0.68

$$P = \frac{m_{\text{propellant}}}{m_{\text{initial}}} + \frac{1}{2}\left[\left(\sum_{i=1}^{n} C_{p,i} G_{p,i}^2 + \sum_{i=1}^{n} C_{v,i} G_{v,i}^2\right)\right] \quad (1)$$

The first term in the penalty function is the desired performance index, the propellant mass fraction, defined as the required propellant mass divided by the initial mass of the spacecraft. The propellant mass is an explicit function of the two time parameters and the user-defined time of flight (TOF) because the propulsion system has a constant specific impulse. The vehicle's final position and velocity are obtained from the 38 parameters by forward integration of the state and costate differential equations.

The second term in the penalty function is the penalty term, which quantifies constraints violation. The vectors $\boldsymbol{C}_p$ and $\boldsymbol{C}_v$ can be adjusted to enhance or diminish the influence of the penalty terms.

The DFP method assumes that the function to be minimized is quadratic and perturbs the initial parameter values conjugate with respect to H, which approaches the matrix of second partials near the optimum. This technique has been found to achieve faster convergence than other unconstrained minimization methods, such as sectioning or steepest descents, but it is slower and less prone to succeed when the quadratic approximation is inaccurate or the initial values are far from the optimal values. A more rigorous discussion of the DFP penalty function method is provided by Johnson.[10] The DFP penalty function and COV optimization technique just described is referred to in this paper as the DFP–COV method, and it yields promising results. However, the MSRM scenarios have a small radius of convergence, which has made it tedious and time-consuming to obtain results.

GA

GAs optimize problems through a mechanism similar to evolution by randomly sampling the parameter space to create an initial population of potentially optimal solutions (genomes), which are bred toward the optimal solution by encouraging the best-performing solutions to pass their information to future generations. This process iterates, with each generation becoming progressively more optimal, until one condition in a set of convergence criteria is met. A more detailed explanation of the GA used in this research is explained hereafter; for a generalized discussion of GAs, the reader is referred to the literature.[11,12]

The GA was introduced to the MSRM optimization routine as a method to decrease the required analysis time by providing a systematic and intelligent process for the identification of initial costate vectors to the DFP–COV program. Primary objectives were to minimize human oversight (maximize autonomy) and to obtain solutions to problems that could not be solved with DFP–COV alone. Pursuant with these goals, the two programs were closely coupled to reduce the need for human interaction and duplicate computation. The GA used the penalty function, Eq. (1), as its performance index, which eliminated the necessity to remodel, and recompute, the problem dynamics. Additionally, use of the penalty function allowed for a more direct comparison of the GA's results to those previously produced by the DFP–COV method alone. Reduced human interaction was achieved when various learning strategies were enabled and by the judicious selection of convergence criteria, which strike a balance between early and late termination, which lead to unnecessary oversight and extraneous calculation, respectively.

The GA used in this research was originally formulated to maximize, not minimize, the performance index, which required one minor modification to the penalty function. The GA's performance index F was defined to be the reciprocal of the penalty function, Eq. (1). An attempt to maximize the reciprocal of P was equivalent

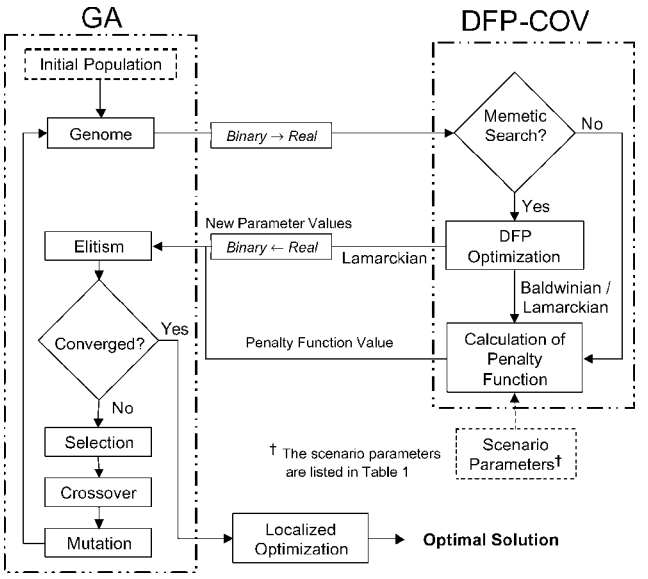

Fig. 1 Organization of the GA and its relationship with the DFP–COV program.

to the minimization P because selection was based on normalized values. (See the Selection section.)

The GA used in this research also required that the 38 real-valued parameters be represented as binary numbers. Between 13 and 15 bits were used to represent each parameter, based on its bounds and the required precision, which were determined by the analysis of previous solutions. The bits representing each parameter were concatenated to construct a genome, a 518-bit binary string that constituted a potential solution. A group of genomes is a set of potential solutions and is known as a generation.

Initial Generation

The operation of the GA and its interaction with the DFP–COV program is shown in Fig. 1. An initial generation was populated with a set of randomly created genomes that is large enough to ensure genetic diversity. Too small a population would have difficulty converging on the optimal solution because the genomes would not accurately represent the entire set of possible solutions. However, an overly large population would unnecessarily increase the GA's computational requirements. The optimal population size depends on the problem being optimized, but for the MSRM problems being solved, a 380-genome population was found to provide solutions as optimal as those found with the DFP–COV. (See the Results section.) Potential future work could study the impact of population size on performance.

Penalty Function Evaluation

After the initial generation was created, the 380 genomes were linearly mapped to 38-parameter sets of real-valued numbers and passed to the DFP–COV program. When the GA and DFP–COV program were used cooperatively (Learning Strategies section), the output from the DFP–COV program was used to determine appropriate parameter bounds. If the DFP optimizer tried to force a parameter past one of its bounds, the user was notified so that the bound could be expanded. Likewise, the user was informed if the magnitude of the DFP method's perturbation was below the binary precision limit of the GA. This allowed the user to shrink the bounds or increase the number of bits to increase precision.

The DFP–COV program evaluated the value of the penalty function for each genome via one of two methods. In the first method, the COV program used the durations of the trajectory arcs and the costate vectors to compute the spacecraft trajectory and evaluate the resulting penalty function value. The DFP method was not used to optimize the parameters before the penalty function value was returned to the GA program.

In the second method, the DFP–COV optimization routine was used cooperatively with the GA. As in the first method, the COV program evaluated the penalty function, but the DFP program also executed several times in an attempt to partially optimize the potential solution. The number of DFP iterations was fixed for all genomes (typically 5–15 iterations) at 5–15% of the average number of iterations required to optimize a trajectory fully, given a good initial parameter set and H matrix. A good initial parameter set requires no further homotopy analysis or reinitialization of the H matrix to converge and shows the largest improvement in the penalty function value during the first 5–15 iterations because H is close to the optimal direction of search. If the initial guesses are not good, the first several iterations are spent on the reinitialization of H, with little improvement in the penalty function. Therefore, rapid improvement in the penalty function during a small number of iterations can be used as an evaluation of the quality of the genome's ability to converge to the optimum and not just its proximity to the optimum. The user has the option of returning only the improved penalty function value or the penalty function value and the perturbed parameters.

Learning Strategies

The choice to partially optimize each genome is represented by the memetic search decision gate in Fig. 1. The noncooperative option is similar to the method used by Crain et al.[6] and does not use the DFP optimization routine, which reduces computation time; therefore, an attempt is always made to use this method before penalty function feedback is implemented. The second method employs the DFP- and GA-based methods working cooperatively as a memetic algorithm, genetic search coupled with a localized optimization program.

One cooperative option returns the partially optimized penalty function values, but not their corresponding improved parameter values, to evaluate the genome's potential to be an optimal solution. This is measured by its ability to achieve a large decrease in penalty function value over a small number of iterations. The genomes are not actually changed, and, therefore, the original genetic information remains intact. However, because the penalty function is the sole factor influencing selection, genomes that improve quickly during partial optimization attain higher fitness. This method allows learning ability to influence natural selection without directly altering genetic information and is, therefore, analogous to the biological mechanism described by Baldwin[13,14] and the learning strategy implemented by Hartmann et al.[7]

The second cooperative option returns the perturbed parameter values and the corresponding updated penalty function value to the GA, overwriting the original genome. Because any improvement in a genome's penalty function value represents learning and adaptation, the genome replacement option represents a situation where learned traits become heritable to later generations, a theory first proposed for biological systems by Lamarck in the early 1800s.[15,16] Direct alteration of the gene pool via a Lamarckian strategy allows problems to converge much more quickly; however, this is not necessarily positive because a lack of genetic diversity can lead to premature convergence and insufficient evaluation of the parameter space. This research suggests that premature convergence has not been an issue because the solutions compare favorably to previous results (see the Results section), but future researchers should be cognizant of various learning strategies and the potential for early convergence.

Selection

Following the process outlined in Fig. 1, after the penalty function values (and possibly new parameter values) are returned to the GA, the next step is selection. (Elitism and convergence are not enforced on the initial generation; see the Elitism and Convergence Criteria section.) Selection determines the genomes that will be allowed to pass their genetic information onto future generations. The selection method is a Roulette wheel, where the probability of a particular genome being selected for procreation is determined by its

normalized fitness, defined by Eq. (2):

$$f_i = F_i \Big/ \sum_{i=1}^{n} F_i \tag{2}$$

The use of a normalized fitness distribution linearizes the effect of evaluation of the reciprocal of the penalty function, such that if two genomes' penalty function values differ by a factor of two, the genome with the better performance has twice the probability of being selected.

Crossover and Mutation

Once parents are selected, their binary (genetic) data is crossed-over to create a child genome. There is a 70% chance of crossover occurring in each bit string, such that crossover could occur a maximum of 38 times during the creation of one child genome, once within each parameter's bit string. Figure 2 is an example of crossover between the bit strings corresponding to the first parameter of each parent. The crossover location, denoted by the solid black line, is chosen at random. The bits from the first parent, up to the crossover point, are copied into the bit string of the child. The remaining bits are copied from the second parent. If crossover does not occur for a certain parameter's bit string, the bit string from the parent with the highest fitness is passed to the child without change. The selection and crossover process repeats to create n children, which form the next generation. Parents that have been selected for procreation are not removed from the selection pool, and so a single genome can be the parent of multiple children.

After a new generation of children has been created, there is a low probability, on the order of 1%, that each new bit will be mutated (1 changed to 0 or vice versa); this process is also shown in Fig. 2. Mutation prevents genetic diversity from being weeded out prematurely and increases the chances of a global minimum that has a small radius of convergence being found. The values of the GA operating parameters used to generate the reported data, such as generation size and crossover probability, are summarized in Table 2.

Elitism and Convergence Criteria

After mutation, the fitness distribution of the new generation is calculated, but before it is allowed to have its own offspring, elitism is enforced, and the convergence criteria are evaluated. Elitism ensures that the best solution is never lost. If the genome with the largest value of F in the new generation (the new generation's elite genome) does not outperform the preceding generation's elite genome, then the old elite genome is copied over the worst performing member of the new generation.

The convergence criteria, and typical values for the operating parameters, are shown in Table 3. The GA most commonly converged because either the elitism or percent improvement criterion was met, which were included as a means of measuring when the GA was no longer effectively improving genomes. Because the elite genome ultimately became the final solution, if it had not changed in a number of generations, the GA was halted. The elitism criterion most often produced a genome within the DFP–COV method's radius of convergence, which made further GA iteration unnecessary. On occasions when the elitism criterion produced premature convergence, the GA was restarted and allowed to continue until halted by another criterion.

The percent improvement criterion provided a measure of each generation's improvement and halted operation when the best performing genomes, as a group, were not improving from generation to generation. When the percent improvement criterion was enforced, a solution could usually be obtained by application of the DFP–COV method to the elite genome. The penalty function criterion was included to terminate the program at a penalty function value that would indicate close proximity to an optimal solution, based on penalty function values for scenarios optimized earlier.

Localized Optimization

After convergence, the DFP–COV program performed additional calculations to refine the GA's solution and more precisely define the optimal trajectory. The trajectory optimization technique consisted of the GA and DFP–COV program working together to find the approximate location of the global minimum, which was further refined by the DFP–COV program to determine a precise solution. It is not possible to prove that the final solutions obtained are true global minimums, but the results can be compared against those obtained with different optimization routines, especially the DFP–COV method, to show that they are superior or at least equally optimal solutions. (See the Results section.)

GA convergence typically occurred in fewer than 20 generations, remarkably rapid given that a GA with similar genome length and population size might be expected to create hundreds of generations before converging.[11,12] This potentially premature convergence was intentional; the GA's convergence criteria were tuned to halt the genetic search when it was estimated that the elite genome was within the radius of convergence of the best solution. It is possible, even likely, that a substantially increasing population size and number of GA iterations before convergence could obtain superior solutions,

Table 2 Typical GA operating parameters

Parameter	Value
Number of optimization parameters	38
Genome length, bits	570
Population size, genomes	380
Mutation probability	0.01
Crossover probability	0.7
Multipoint crossover	Yes
Elitism	Yes

[**1 1 0 1 1 1** | 0 0 1 0 1 0 1, …] Parent 1
[0 0 1 0 1 0 | **1 0 0 1 1 1 0**, …] Parent 2
[1 1 0 1 **1** | 1 0 0 1 1 1 0, …] Child
[1 1 0 1 **0** 1 1 0 0 1 1 1 0, …] Mutated Child

Fig. 2 Example of crossover between the sixth and seventh bits and mutation of the fifth bit.

Table 3 GA convergence criteria and typical variable values

Criterion			Number of genomes evaluated	Dependent on earlier generation(s)
Name	Convergence condition	Typical value		
Elitism	The elite genome has not changed in X generations.	$X = 4$	1	Yes
Penalty function value	The penalty function value of the elite genome is below the cutoff threshold of X.	$X = 5$	1	No
Percent improvement	The summed penalty function values for the top $X\%$ of the population has not improved by more than $Y\%$.	$X = 40\%$ $Y = 0\%$	$X\%$ of population	Yes

but the primary goal of this research was to create a rapid trajectory optimization tool that produced solutions at least as good as those obtained with only the DFP–COV method. Further optimization, at the cost of additional computing time, was not a priority, although it would provide an interesting area of future research. Likewise, GA precision could be enhanced by increasing the number of bits used to represent each parameter or by narrowing the parameter bounds. However, once acceptable initial guesses for the DFP–COV program had been found, it was more computationally efficient to halt the GA and proceed with optimization of only the elite genome.

Results

A number of Mars Sample Return scenarios had been optimized by the use of the DFP–COV method only. These scenarios were optimized again by the use of the GA for comparison purposes. The cooperative GA and DFP–COV tool (named RAPTOR) met the goals of providing optimal solutions, especially to problems without solutions, while also minimizing human oversight.

Finding a single solution with the DFP–COV method is a lengthy process that can occupy an experienced user from a day to well over a week, depending on the complexity of the problem and how closely related it is to previously solved scenarios. Even a change as small as lengthening the TOF from 460 to 530 days, while all other aspects of the problem remain constant, could require several homotopy steps where, in addition to the 530-day case, solutions for the 475-, 490-, and 515-day cases might also be solved. Each case would involve numerous restarts of the optimization program, where input files would be edited by hand to update the initial guesses parameters.

RAPTOR eliminated the hand editing process because the GA interfaced with the DFP–COV program to update the parameter values. Instead of having to monitor program progress and continuously restart the solver, the user could define the problem and then perform other tasks or leave RAPTOR to run overnight, which allowed a 24-h task to be completed in one 24-h day as opposed to three 8-h days. Each of the scenarios described in this section were autonomously solved with RAPTOR in less than 24 h. The decrease in convergence time allowed the tool to be used as a method to survey different spacecraft configurations and departure parameters.

A number of possible return scenarios are described in Table 4. Each scenario was optimized with the DFP–COV, and then again with RAPTOR. There is one exception, noted in Table 4 and discussed later, for which the DFP–COV method could not find a solution. Two different LMO spacecraft initial masses were analyzed at a variety of departure dates and TOF from Mars to Earth. For the smaller 400-kg spacecraft, a single two-thruster scenario was also defined, which served only to double the mass flow rate and thrust.

Genetic search and both memetic learning strategies were used to obtain results in Table 4. The 7 May 2013 and the 430-day 16 July 2013 scenarios were solved by the use of the Lamarckian and Baldwinian learning strategies, respectively. The remaining scenarios were initially solved without partial optimization, although the implemention of either of the parameter feedback techniques yielded the same solutions. The ability of the Lamarckian learning strategy to obtain a solution to the heretofore unsolved 430-day 16 July 2013 mission is a notable example of the benefit of this memetic search technique.

The results obtained with RAPTOR show larger final masses delivered to the Earth's SOI, confirming that the GA is able to find optimal solutions of the same quality as the DFP–COV method. For the two-thruster scenario, RAPTOR used the Baldwinian learning strategy to obtain a nearly 20% increase in final mass. The 7 May 2013 scenario was the first two-thruster case to be solved; therefore, the initial COV costate vectors had to be obtained via homotopy analysis from a one-thruster solution. Starting from a parameter set optimized for a one-thruster spacecraft prevented the DFP optimizer from obtaining the more optimal trajectory found by RAPTOR. The two-thruster solution illustrates the strength of RAPTOR's independence from initial guesses and freedom from the influence of earlier solutions, which in this case resulted in superior optimization.

Figure 3 shows the improvement of the elite genome's penalty function value for the 460-day mission departing Mars on 16 July 2013. The dramatic improvement in the first two generations (the initial generation is counted as zero) is typical and occurs as the GA quickly converges from randomized parameter values to values closer to the optimal solution. Improvement continued, albeit less quickly, through the 13th generation, when the percent improvement criterion halted the GA. Figure 4 shows the final trajectories

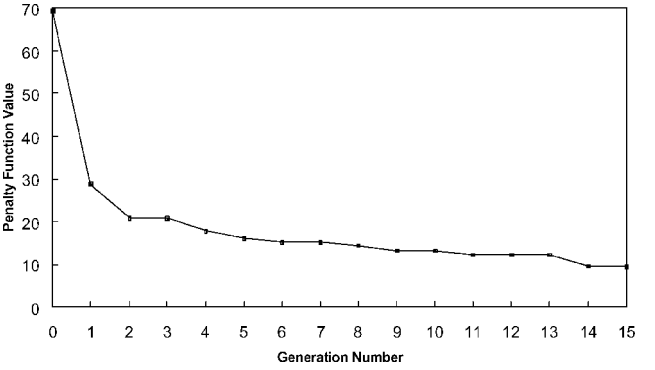

Fig. 3 Improvement of elite genome for the 460-day, 16 July 2013 scenario.

Table 4 Comparison of final spacecraft mass obtained with RAPTOR vs the DFP–COV method for various scenarios

Case						
Initial mass in LMO, kg	Number of thrusters	Departure date[a]	TOF,[b] days	Final mass, kg[c] RAPTOR	Final mass, kg[c] DEF–COV	Improvement,[d] %
600	1	07/16/2013	490	465.30	453.99	2.49
600	1	07/16/2013	460	468.27	461.19	1.54
600	1	07/16/2013	430	466.84	no convergence	——
400	2	05/07/2013	530	279.17	232.92	19.86
400	1	06/11/2013	470	311.58	309.45	0.69
400	1	07/11/2013	430	301.07	279.05	7.89
400	1	07/11/2013	460	310.01	307.35	0.87
400	1	09/09/2013	460	299.16	296.36	0.94
400	1	09/09/2013	430	303.04	301.37	0.56
400	1	10/09/2013	430	297.19	293.83	1.14

[a]Departure date is the beginning of heliocentric transfer.
[b]TOF is the duration of heliocentric transfer.
[c]Final mass is the mass of spacecraft at the end of the heliocentric transfer.
[d]Improvement is the (RAPTOR final mass minus the DFP–COV final mass times 100% over the DFP–COV final mass.

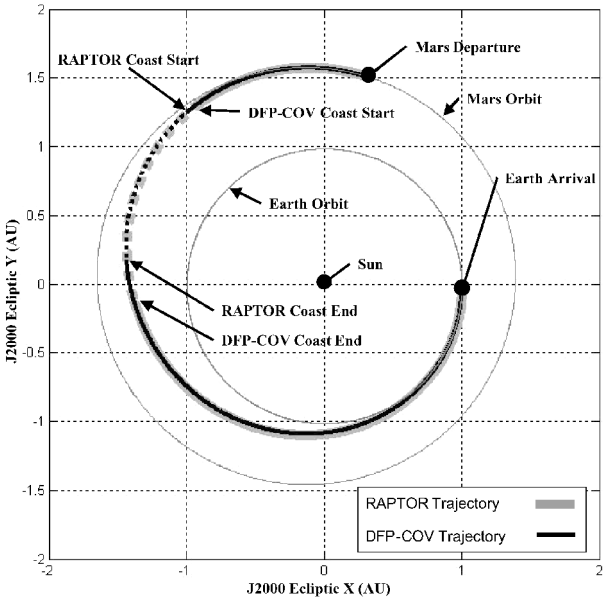

Fig. 4 Comparison of RAPTOR and DFP–COV method only optimized trajectories for the 460-day, 16 July 2013 scenario.

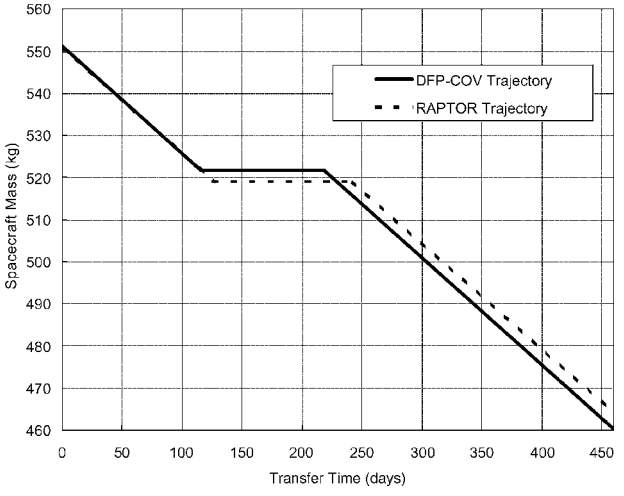

Fig. 5 Comparison of spacecraft mass profiles for RAPTOR and DFP–COV method only trajectories for 460-day, 16 July 2013 scenario.

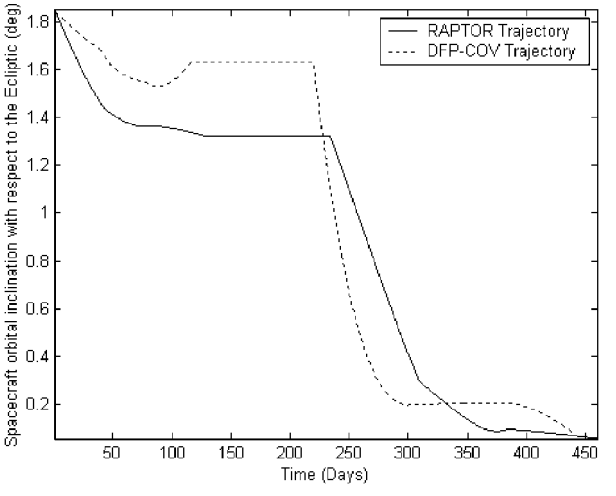

Fig. 6 Comparison of RAPTOR and DFP–COV method only inclination for the 460-day, 16 July 2013 scenario.

for the same scenario obtained with RAPTOR and the DFP–COV method only. The in-plane trajectories are almost identical, with the exception of the slightly longer coast arc in the trajectory obtained by RAPTOR, which resulted in a smaller propellant mass and larger payload delivered to Earth's SOI (Fig. 5). A more pronounced difference is shown in Fig. 6, which compares the orbital inclination of both trajectories. For the RAPTOR trajectory, a larger portion of the required inclination change occurred during the first thrust arc, which is more efficient because the spacecraft's velocity is smaller as it leaves Mars. The more efficient plane change found by the GA results in a longer coast arc and, thus, a higher spacecraft mass returned to Earth. Figures 3–6 are typical results for the scenarios presented in Table 4.

Conclusions

A GA has been used in conjunction with the DFP penalty function method and the calculus of variations to optimize low-thrust, interplanetary trajectories for the MSRM. The reliance of the DFP–COV method on earlier solutions and its sensitivity to the quality of the initial guesses has been eliminated by relying on the GA to search the parameter space to find the location of the globally optimal solution. The DFP–COV method is used to refine the parameter set found by the GA, improving the precision of the final answer beyond what would be possible by the use of the GA alone. The genetic search was further enhanced with learning strategies enabled by the use of active feedback from the DFP–COV program, which increased the likelihood of an optimal solution being found. Active feedback also allowed the cooperative GA/DFP–COV technique to optimize scenarios that could not be optimized with DFP–COV alone. A number of MSRM mission scenarios were analyzed with only the DFP–COV method and then again with the DFP–COV method and GA working in conjunction. When the results were compared, the cooperative GA/DFP–COV method obtained comparable or superior solutions in substantially less time. Overall, the new GA and COV-based method has increased the speed and efficiency of the optimization routine, provided more optimal results, and allowed some scenarios to be optimized for which solutions had not previously been obtained.

References

[1] Dewell, L., and Menon, P., "Low-Thrust Orbit Transfer Optimization Using Genetic Search," *AIAA Guidance, Navigation, and Control Conference and Exhibit*, AIAA, Reston, VA, 1999, pp. 1109–1111.

[2] Betts, J. T., "Survey of Numerical Methods for Trajectory Optimization," *Journal of Guidance, Control, and Dynamics*, Vol. 21, No. 2, 1998, pp. 193–205.

[3] Bulirsch, R., Montrone, F., and Pesche, H., "Abort Landing in the Presence of a Windshear as a Minimax Optimal Control Problem, Part 2: Multiple Shooting and Homotopy," *Journal of Optimization Theory and Applications*, Vol. 70, Aug. 1991, pp. 223–254.

[4] Rauwolf, G., and Coverstone-Carroll, V., "Near-Optimal Low-Thrust Orbit Transfers Generated by a Genetic Algorithm," *Journal of Spacecraft and Rockets*, Vol. 33, No. 6, 1996, pp. 859–861.

[5] Coverstone-Carroll, V., "Near-Optimal Low-Thrust Trajectories via Microgenetic Algorithms," *Journal of Guidance, Control, and Dynamics*, Vol. 20, No. 1, 1996, pp. 196–198.

[6] Crain, T. P., Bishop, R. H., Fowler, W. T., and Rock, K., "Interplanetary Flyby Mission Optimization Using a Hybrid Global/Local Search Method," *Journal of Spacecraft and Rockets*, Vol. 37, No. 4, 2000, pp. 468–473.

[7] Hartmann, J. W., Coverstone-Carroll, V. L., and Williams, S. N., "Optimal Interplanetary Spacecraft Trajectories via a Pareto Genetic Algorithm," *Journal of the Astronautical Sciences*, Vol. 46, No. 3, 1998, pp. 267–282.

[8] Vadali, S., Aroonwilairut, K., and Braden, E., "A Hybrid Trajectory Optimization Technique for the Mars Sample Return Mission," American Astronautical Society/AIAA, Paper AAS 01-466, July 2001.

[9] Van Flandern, T. C., and Pulkkinen, K. F., "Low-Precision Formulea for Planetary Positions," *Astrophysical Journal Supplement Series*, Vol. 43, No. 1, 1979, pp. 391–411.

[10] Johnson, I., "The Davidon–Fletcher–Powell Penalty Function Method: A Generalized Iterative Technique for Solving Parameter Optimization Problems," NASA TN D-8251, May 1976.

[11]Goldberg, D. E., *Genetic Algorithms in Search, Optimization, and Machine Learning*, 1st ed., Addison Wesley Longman, Reading, MA, 1989, pp. 200–230.

[12]Goldberg, D. E., *The Design of Innovation: Lessons from and for Competent Genetic Algorithms*, 1st ed., Kluwer Academic, Boston, 2002, Chaps. 4–12.

[13]Baldwin, J. M., "A New Factor in Evolution," *American Naturalist*, Vol. 30, No. 354, 1896, pp. 441–451.

[14]Baldwin, J. M., "A New Factor in Evolution (continued)," *American Naturalist*, Vol. 30, No. 355, 1896, pp. 536–553.

[15]Lamarck, J. B., *Zoological Philosophy: An Exposition with Regard to the Natural History of Animals*, Translated by H. Elliot, 1st ed., Univ. of Chicago Press, Chicago, 1984, Chap. 7.

[16]Lamarck, J. B., "Of the Influence of the Environment on the Activities and Habits of Animals, and the Influence of the Activities and Habits of These Living Bodies in Modifying Their Organisation and Structure," *Adaptive Individuals in Evolving Populations: Models and Algorithms*, edited by R. K. Belew and M. Mitchell, Vol. 1, Santa Fe Institute Studies in the Sciences of Complexity, Addison Wesley Longman, New York, 1996, Chap. 4.

C. A. Kluever
Associate Editor