Optimal Space Trajectory Design: A Heuristic-Based Approach

C. R. Bessette*
Pennsylvania State University
University Park, PA 16802

D. B. Spencer, Ph.D.†
Pennsylvania State University
University Park, PA 16802

Abstract

While it is widely known that the Hohmann transfer is the optimal trajectory in terms of Δv, it is certainly non-optimal in terms of time of flight (TOF). As a result, the aim of this paper is, using Lambert's Algorithm, to determine those orbital trajectories that require low Δv's, but also require small TOFs. However, there are two restrictions placed on these transfer trajectories: that the transfer arcs are less than 180^O, and their rendezvous time (TOF+Time of Launch) is less than that of the Hohmann case. In the two cases examined (Low-Earth Orbit (LEO) to LEO and LEO to Geosynchronous Earth Orbit (GEO)), the evolutionary algorithm (EA) was able to obtain solutions which were marginally more expensive in Δv, yet much lower in terms of rendezvous time. For the LEO to LEO case, the trajectory would require an additional 50% more Δv; but it would required a rendezvous time that is 68% that of the Hohmann case. For the LEO to GEO case, the trajectory would require 45% more fuel, but the savings in terms of rendezvous time would be 22%. In addition, three different heuristics were used in the analysis of this problem: Particle Swarm Optimization (PSO), Differential Evolution (DE), and Covariance Matrix Adapted Evolutionary Strategies (CMA-ES). After a comprehensive comparison of the aforementioned algorithms, CMA-ES had the quickest convergence, yet also had a poor reliability record based upon random seed analyses. PSO converged the second quickest as compared to the other two algorithms, and it had a perfect reliability record. As a result, PSO was deemed the best algorithm to solve this problem due to its quick convergence, and its excellent reliability.

1 INTRODUCTION

1.1 ORBITAL TRANSFERS

This paper deals with the concept of minimizing the amount of fuel required to rendezvous with another spacecraft, while also obtaining trajectories much quicker than Hohmann transfers in terms of rendezvous time. This paper considers rendezvous with both spacecraft residing inside Earth's sphere of influence. For the dynamics of the system, a 2-body, perturbation-free gravitational model was assumed. Equation 1 describes the model:

$$\ddot{\vec{r}} = -\frac{\mu_\oplus \vec{r}}{r^3} \quad (1)$$

While it is no secret that the Hohmann Transfer is the optimal impulsive transfer, it is a long-duration transfer with a 180^O transfer arc. Equation 2 describes the time of flight (TOF) using a Hohmann Transfer,

$$t_{Hohmann,1} = \pi\sqrt{\frac{6678^3}{\mu_\oplus}} = 52\text{min.} \qquad t_{Hohmann,2} = \pi\sqrt{\frac{44240^3}{\mu_\oplus}} = 317\text{min.} \quad (2)$$

Consequently, for the Hohmann transfer, the target vehicle, at rendezvous, must be exactly 180^O from the chaser vehicle, upon departure. If a time-critical rendezvous needs to be done, mission planners must wait for the opportunity where the above configuration occurs – more about this phasing time will be discussed later in detail. In summary, the Hohmann Transfer is a long transfer, and necessary conditions must be met for it to be of use.

As a result, Lambert's Algorithm was used. This algorithm was developed in Prussing and Conway[1], but the f and g series used to solve for the required Δv's came

* Graduate Student, Aerospace Engineering
† Assistant Professor of Aerospace Engineering

from Vallado[2]. This algorithm requires the position vector of the chaser vehicle at departure, the position vector of the target vehicle at arrival, and the required TOF. The algorithm outputs the velocity vector required to leave the initial orbit, and the velocity vector required to enter the final orbit. It is important to note that a singularity exists in the f and g series – and that singularity exists for transfer orbits of 180^O – i.e. the Hohmann transfer. So consequently, Lambert's Algorithm will not return a low Δv for Hohmann transfers, thus excluding them as possible solutions for the Evolutionary Algorithms (EA).

It is important to note that the metric measuring fuel consumption is Δv. This stands for the total change in velocity required for the transfer. It is inherent that the required amount of fuel is directly proportional to the Δv.

2 PROBLEM FORMULATION

2.1 PROBLEM SETUP

In this paper, two cases will be analyzed. The first has the chaser vehicle initially in a circular, 20^O inclined orbit, with a 300 km altitude. The target vehicle, meanwhile, is in a circular, 1600 km altitude, 30^O inclined orbit. At the epoch time, each of these vehicles share an argument of latitude of 0^O. For convenience, this case will be referred to as the Low-Earth Orbit (LEO) to LEO case.

The second case, has the chaser vehicle in the same orbit as the first case; but the target vehicle this time is in Geosynchronous Earth Orbit (GEO) (a=44240 km). As specified in the first case, the initial argument of latitude for each of vehicle in this case is 0^O. Similar to the first case, this one will be referred to as the LEO-to-GEO case. Figure 1 describes the radius and velocity vectors used for each case:

$$r_{c,1,2} = \left\{6678\hat{i} + 0\hat{j} + 0\hat{k}\right\}^T [km]$$
$$v_{c,1,2} = \left\{0\hat{i} + 7.2599176\hat{j} + 2.64239\hat{k}\right\}^T [km/s]$$
$$r_{t,1} = \left\{7978\hat{i} + 0\hat{j} + 0\hat{k}\right\}^T$$
$$v_{t,1} = \left\{0\hat{i} + 6.11301\hat{j} + 3.529345\hat{k}\right\}^T$$
$$r_{t,2} = \left\{42240\hat{i} + 0\hat{j} + 0\hat{k}\right\}^T$$
$$v_{t,2} = \left\{0\hat{i} + 3.07186\hat{j} + 0\hat{k}\right\}^T$$

Fig. 1: Position/Velocity Vectors at the Epoch Time

The three variables used in the formulation of this problem are: time of launch (TOL), TOF, and Δv. In order to obtain the position vector of the chaser vehicle upon departure, equation 1 was numerically integrated using MATLAB's ODE45 function to the specified TOL. To obtain the position of the target vehicle at rendezvous, equation 1 is again numerically integrated to the specified rendezvous time (TOL+TOF).

The function to be minimized is the sum of the required Δv for the chaser vehicle to leave its initial orbit and the Δv for the chaser to rendezvous with the target:

$$f = \Delta v_1 + \Delta v_2 \quad (3)$$

2.2 CONSTRAINTS

The objective of each case is to obtain a trajectory that is relatively close to the optimality of the Hohmann transfer (in terms of Δv), yet is not nearly as expensive in rendezvous time. As a result, weights were introduced to penalize solutions that contain rendezvous times comparable to the Hohmann case. The reason why these weights were used is because why would a mission planner want to use extra Δv for a rendezvous when he or she could wait marginally longer to use a Hohmann transfer?

The constraints used for the LEO-to-LEO example are as follows:

$$if(TOF + TOL) > 300, \Delta v = \Delta v + 1$$
$$if(TOF + TOL) > 330, \Delta v = \Delta v + 3$$

Fig. 2: Penalty Weights

With the vehicles in the initial configuration as given in figure 1, if one were to attempt a Hohmann transfer on the LEO-to-LEO case, the mission planner would have to wait 360 minutes for the vehicles to obtain a geometry favorable for the Hohmann transfer. Then, if the Hohmann time of transfer were added to the phasing time, the total rendezvous time would be 410 minutes. As a result, the constraints in figure 2 were set so that solutions could be found with total rendezvous times much lower than that of the Hohmann example. The minimum rendezvous time in the LEO-to-GEO case is 390 minutes. This is due to the fact that the orbital periods in the LEO-to-LEO case are so close, that it takes a few orbits for the argument of latitude between the two vehicles to differ by 180^O. As a result, the same constraints were used for the LEO-to-GEO transfer.

3 EVOLUTIONARY ALGORITHMS

3.1 ALGORITHM OVERVIEW

EAs are stochastic, population-based heuristics. Because this problem is real-coded, as opposed to binary, real-coded EAs were evaluated. As a result, three real-coded algorithms were utilized in solving this problem. The first, Differential Evolution (DE), is a heuristic developed by Rainer Storn and Kenneth Price in 1996[3] and has shown promise in solving problems with large search spaces. DE is a classical EA in the sense that it functions on the basic premise of Darwinian natural selection: mutation, crossover, and selection. The second heuristic used in this problem is Particle Swarm Optimization

(PSO) developed by Russell Eberhart and James Kennedy in 1995; which, unlike DE, does not mimic Darwinian natural selection; but rather imitates the motion of a swarm or bees, or a school of fish[4]. The final EA used for analysis of this problem is Evolutionary Strategies using Covariance Matrix Adaptation (CMA-ES) developed by Nikolaus Hanson[5] in 2005, which also uses Darwinian natural selection as its underlying principle.

3.2.1 DE'S Mutation Scheme

In order to use DE, there are a series of parameters that must be set. One of those parameters is the mutation weight. This value, F, ranges between 0 and 2, and serves to amplify the difference between the two trial vectors. Figure 3 serves to illustrate how DE uses mutation:

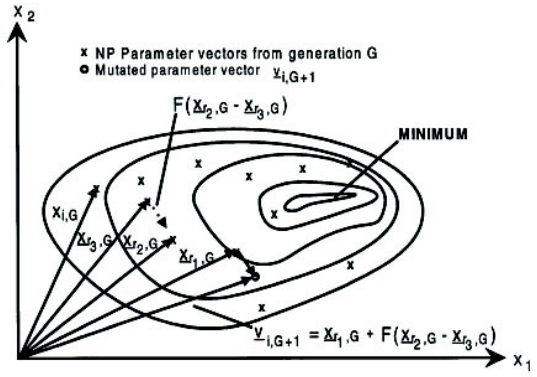

Fig. 3: Storn/Price's Mutation Schematic[3]

Equation 4 describes the mutation scheme:

$$v_{i,G+1} = x_{r1,G} + F\left(x_{r2,G} - x_{r3,G}\right) \quad (4)$$

where the r subscripts refer to random integers. For this to work, the number of population members must be greater than 4. This is because a total of four population members are used in the process of mutation (r_1,r_2,r_3), and they are chosen to be different than the i'th index on $v_{i,G+1}$. So essentially, DE takes the weighted difference (F) between two randomly selected trial vectors from the population, and then adds that difference to a third vector. That sum becomes the mutated vector.

3.2.2 Crossover

Another user-defined parameter is the Crossover Ratio (CR). CR is a ratio that is bounded by 0 and 1. Crossover works on each dimension of the vector. For dimension 1, if a random number is less than CR, then the value of dimension 1 for the mutant vector becomes the value of dimension 1 in the trial vector. If the random number is greater than CR, then the trial vector keeps its value for dimension 1. That process is repeated for all of the dimensions of the vector (in this case, 2 dimensions). Figure 4 illustrates how crossover works in DE:

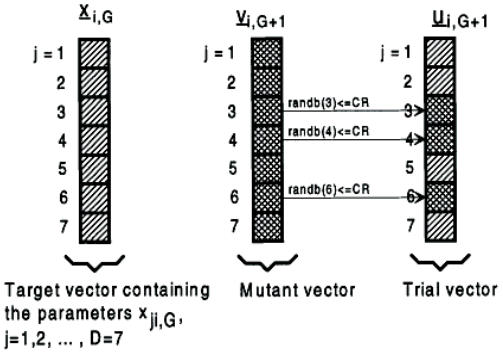

Fig. 4: Storn/Price's Crossover Schematic[3]

Equation 5 describes how DE handles crossover:

$$u_{ji,G+1} = \begin{cases} v_{ji,G+1} if(rand(j) \le CR) \\ x_{ji,G} if(rand(j) > CR) \end{cases} \quad (5)$$

3.2.3 Selection

To decide which vectors are allowed to enter the new generation, an elitist approach is utilized. Elitism ensures that the vector with the lowest fitness value will enter the new generation. In essence, the $u_{ui,G+1}$ trial vector and the $x_{i,G}$ target vectors from equation 5 are compared. If the trial vector yields a lower fitness value, then that vector enters the new population; otherwise, the target vector will enter the new population.

3.2.4 DE Variants

Storn and Price's DE code has 9 variants that can be used. Often the variant of DE is specified by: DE/x/y/z

where x specifies the vector to be mutated (either randomly chosen from the population, or the vector with the lowest fitness value); y specifies the number of difference vectors to be used (the number of difference vectors present in the parenthesis in equation 5); and z specifies the crossover type – either exponential or binary.

Because of the rugged search space topology, DE/best/2/bin was used. With high enough population sizes, using 2 difference vectors increases the diversity. Due to the plethora of infeasible spaces, high diversity was essential in order to escape from local minima that may be surrounded by infeasibility.

3.3.1 PSO Basics

As mentioned previously, PSO works by mimicking the motion of a school of fish, or a flock of birds. A main

driving principle behind PSO is that social sharing of information among individuals in a population may provide an evolutionary advantage[4]. In contrast to many EAs, most heuristics encourage competition between members of a population, but PSO fosters cooperation between members.

3.3.2 Position and Velocity

Upon initialization, the particles are randomly distributed throughout the search space. Then each particle is given a velocity governed by equation 6:

$$
\begin{aligned}
& v_{id}^{n+1} = \chi\left(\omega v_{id}^{n} + c_1 r_1^{n}\left(p_{id}^{n} - x_{id}^{n}\right) + c_2 r_2^{n}\left(p_{gd}^{n} - x_{id}^{n}\right)\right) \\
& where \\
& v_{id}^{n+1} : Updated \quad Velocity \\
& v_{id}^{n} : Current \quad Velocity \\
& x_{id}^{n} : Current \quad Position \\
& \chi : Constriction \quad Factor \\
& p_{id}^{n} \; Individual \quad Best \quad Position \\
& p_{gd}^{n} : Global \quad Best \quad Position \\
& c_1 : Cognitive \quad Parameter \\
& c_2 : Social \quad Parameter \\
& \omega : Inertia \quad Weight
\end{aligned}
\tag{6}
$$

As is evident from equation 6, the velocity for each particle is comprised of the individual's current and best position, as well as the global best position. The cognitive parameter, social parameter, and inertia weight are constants which are defined by the user. One popular prescription for the social and cognitive parameters is that their sum must be bounded by zero and four. In addition, it is advised that the inertia weight be set to one initially, but then it decreases incrementally until it reaches zero at the maximum number of iterations. This encourages low selection pressure at the beginning, but the search pressure then increases as the particles begin to converge to an optimum. The constriction factor is often set to one, but it simply serves as a scaling factor; that is, it amplifies the velocity. If the search space is excessively large, then it is advantageous to use a larger constriction factor, such that the particles can analyze the search space in fewer iterations. Finally, the r_1 and r_2 values in equation 6 are simply random numbers (between zero and 1) which are generated upon each generation. The second governing equation for PSO describes the position of the particles in each successive generation:

$$
\begin{aligned}
& x_{id}^{n+1} = x_{id}^{n} + v_{id}^{n+1} \\
& where \\
& x_{id}^{n+1} : Updated \quad Position \\
& x_{id}^{n} : Current \quad Position
\end{aligned}
\tag{7}
$$

From equation 7, the updated position is simply the sum of the current position plus the velocity. From a mechanics standpoint, this equation is valid assuming that the time step is one time unit.

3.3.3 PSO Pseudo Code

Initially, the population members are randomly distributed throughout the search space. Then the fitness of each particle is measured. The particles begin to move towards the global best particle, this is done by calculating the velocity for each particle (6), and then using the position equation (7) to update the position of each particle as they move towards the global minimum. In the next generation, again the fitness of each particle is measured. If there is a new global best particle, then all of the particles will migrate towards that particle; otherwise, they will continue to move towards the initial global best particle. This procedure will continue until all of the particles have converged to an optimum.

3.4.1 The CMA Evolutionary Strategy

The driving principle behind CMA-ES is that an adaptive covariance matrix is utilized in order to sample the new population of search points[5]. First, the new population is sampled using a multivariate normal distribution, described by equation 8. This is in contrast to DE and PSO where they invoke use of a uniform distribution for all random number generations.

$$
\begin{aligned}
& x_k^{g+1} \sim \mathrm{N}\left(m^{g}, \left(\sigma_g\right)^2, C^{g}\right) \\
& k = 1, \ldots, \lambda
\end{aligned}
\tag{8}
$$

For the initial generation, the covariance matrix is set to the identity matrix, I. Then selection and recombination are used to determine which population members evolve to the subsequent generation. Selection and recombination are controlled by equation 9:

$$
\begin{aligned}
& m^{g+1} = \sum_{i=1}^{\mu} \omega_i x_{i:\lambda}^{g+1} \\
& \sum_{i=1}^{\mu} \omega_i = 1 \\
& where \\
& \omega_i > 0
\end{aligned}
\tag{9}
$$

In addition, CMA-ES makes use of an adaptive step size, governed by equation 10:

$$
\begin{aligned}
& p_{\sigma}^{g+1} = \left(1 - c_{\sigma}\right) p_{\sigma}^{g} + \sqrt{c_{\sigma}\left(2 - c_{\sigma}\right)\mu_{eff}}\, C^{g-1/2} - \frac{m^{g+1} - m^{g}}{\sigma^{g}} \\
& \sigma^{g+1} = \sigma^{g} \exp\left(\frac{c_{\sigma}}{d_{\sigma}}\left(\frac{\left\| p_{\sigma}^{g+1} \right\|}{\mathrm{E}\left\|\mathrm{N}(0, I)\right\|} - 1\right)\right)
\end{aligned}
\tag{10}
$$

And finally, CMA-ES adapts the covariance matrix for the subsequent generation:

$$p_c^{g+1} = (1-c_c)p_c^g + h_\sigma^{g+1}\sqrt{c_c(2-c_c)\mu_{eff}}\,\frac{m^{g+1}-m^g}{\sigma^g}$$
$$C^{g+1} = (1-c_{\text{cov}})C^g + \frac{c_{\text{cov}}}{\mu_{\text{cov}}}\left(p_c^{g+1}p_c^{(g+1)^T} - \delta\left(h_\sigma^{g+1}\right)C^g\right) + \quad (11)$$
$$+c_{\text{cov}}\left(1-\frac{1}{\mu_{\text{cov}}}\right)\sum_{i=1}^{\mu}\omega_i OP\frac{x_{i:\lambda}^{g+1}-m^g}{\sigma^g}$$

CMA-ES will continue to loop until a stopping criterion has been satisfied.

3.4.2 The CMA Evolution Strategy Parameters

Equation 12 describes some of the default parameter settings for CMA-ES:

$$\lambda = 4 + \lfloor 3\ln(n) \rfloor$$
$$\mu = \left\lfloor \frac{\lambda}{2} \right\rfloor \quad (12)$$
$$\omega_i = \frac{\ln(\mu+1)-\ln(i)}{\mu\ln(\mu+1)-\sum_{j=1}^{\mu}\ln(j)}, i = 1,\ldots,\mu$$

Equation 12: CMA-ES Parameters

CMA-ES is unique in the sense that it is meant to utilize small populations (λ). Using the λ-equation, the population size for the two problems at hand was 6, and that proved marginally effective. The number of crossover points is defined by μ. In this case, that was set to 3. The above formulae are simply a guideline – as they are defaults. The user can change those values if desired, but CMA-ES has been shown to converge in the fewest function evaluations with those prescribed formulae. One important parameter that the user must set is sigma. Sigma is the initial coordinate-wise standard deviations for the search. It is recommended that sigma is set to 20-50% of the search space (for each direction).

4 RESULTS AND DISCUSSION

4.1 SEARCH SPACE ANALYSIS

One of the complicating factors in this problem is the rugged search space topology. Infeasible spaces are scattered throughout the decision space. These infeasible spaces stem from the fact that transfer angles in the Lambert's problem were restricted to be less than 180^O. As a result, for a given set of chaser and target position vectors, only a certain number of TOFs exist for an elliptic transfer trajectory between the two points. There are two cases which result in infeasible spaces, the first being those TOFs that result in a parabolic or hyperbolic trajectory, because for those cases, the required Δv approaches infinity. The other case which results in an infeasible space is if the TOF is too great for a transfer angle below 180^O. An example of this would be if a spacecraft were simply doing a simple 100-km orbit-raising maneuver, and the input TOF to Lambert's algorithm was 300 minutes, then a trajectory (with a transfer angle less than 180^O) could not be fit to the above criteria – thus introducing an infeasible space.

It is important to note the method used to sift out infeasible spaces. When a function evaluation was performed, and the TOL and TOF combination were deemed infeasible, the Δv returned would be 20 km/s. Once the function evaluator returns such a high Δv, that value will not be the best in the population, so it will simply be ignored, and not allowed to enter the next generation.

4.2 LEO-to-LEO

For a LEO-to-LEO transfer, the Δv required for a Hohmann transfer is 0.6561 km/s, and the minimum time to rendezvous with the target vehicle is 410 minutes. Using Storn and Price's DE, a rendezvous trajectory was found, with the rendezvous time being a mere 130 minutes, and the Δv expenditure being only 1.4242 km/s. In summary, the trajectory DE found requires 50% more fuel, but the rendezvous time is 68% that of the Hohmann case.

4.2.1 LEO-to-LEO With DE

Using DE, the above results were obtained using a CR and F value of 0.5 and 0.95 respectively. The reasoning behind having a high F was to help combat the problem introduced by numerous infeasibility islands surrounding local minima. Higher mutation weights can decrease search pressure – thus allowing the algorithm to perform a thorough search of the decision space.

First, 10 population members were used. Intermittant convergence was seen across random seeds – i.e. about 40-50% convergence. The seeds which misconverged were getting trapped inside a valley with a large infeasible space surrounding it. As a result, the population size was increased to 30. Once this was done, another random seed analysis was performed, and 90% of the seeds converged to the optimal region.

Furthermore, many values for CR and F were attempted, but F=0.95 proved to be most effective. F seemed to be the dominant of the two user-defined parameters. This can be attributed to the many infeasible spaces. F values greater than 1 proved to be too greedy, as sub-optimal convergence was seen. However, F values that were too low resulted in non-convergence.

4.2.2 LEO-to-LEO With PSO

The LEO-to-LEO case was also solved using PSO. In order to do this, c_1 and c_2 were set to 2.8 and 0.8 respectively. These values were chosen using a trial and

error analysis. Previous analysis of PSO with various test functions did not indicate any corrolation of parameters between different problems. As a result, proper parameter tuning becomes a trial and error process. The constriction factor was set to 1. Due to the large search space, higher constriction factor values were tried, but that simply resulted in a search that was increasingly greedy; and thus unsuccessful as sub-optimal convergence was witnessed. And the inertia weight was set such that it equaled 1 initially, and then approached 0 as the population converged.

Initially a population size of 20 was used, however, intermittant success was seen across random seeds. Therefore, the population size was doubled, and success was seen across all random seeds.

4.2.3 LEO-to-LEO With CMA-ES

The most difficult step in implemating this problem into CMA-ES was determining the proper sigma values. Knowing that sigma should be 20-50% of the search space, sigma was set to [100;100]. When this was done, quick convergence was seen; however, across random seeds, this algorithm converged to sub-optimal regions, yielding a success rate of 30%. In addition, numerous values for sigma were attempted in hopes of finding a more reliable parameter setting, but nothing seemed to eclipse the 30% success rate.

This analysis was done using Hanson's prescribed formulae for population sizes. Larger population sizes were tried, but better success rates were not seen. In addition, the number of crossover points was also altered, and that did not prove effective either. As well, the recombination weights were also changed, again, to no avail.

4.2.4 LEO-to-LEO Termination

The concept of termination is quite amorphous because, a priori, the global minimum is not known. There is no certainty that the minimum outlined above is the global minimum. As a result, each program was allowed to run for 1000 generations. This was done a few times, and each time the same minimum was obtained. Therefore, for the random seed analysis performed above, the seed was counted as successful if it converged to a value below 1.43.

4.3 LEO-to-GEO TRANSFER

For a Hohmann transfer from LEO-to-GEO, the required Δv is 3.8938 km/s. As mentioned in section 2.2, the minimum rendezvous time is 390 minutes As a result, the EA strives to find a solution which is marginally higher in Δv expenditure, yet much lower in rendezvous time. For this case, the best solution found yielded a Δv of 7.177908 km/s. That solution was found at a TOL of 77.98 minutes, and a TOF of 228 minutes – giving a total rendezvous time of 306 minutes. Using this transfer, the chaser vehicle would arrive 86 minutes (22%) quicker, yet would require 45% more fuel as compared to the Hohmann transfer case.

4.3.1 LEO-to-GEO With DE

Using a population size of 20, and the same CR and F constants as in the LEO-to-LEO case, DE converged to the above-mentioned optimum in relatively few function evaluations.

With that initial population size of 20, a random seed analysis was performed. Using the above parameters, DE had a success rate of 100% across random seeds. This is in direct contrast to the population size used in the LEO-to-LEO example because this problem is simply not as difficult to solve, so smaller population sizes population sizes can still converge to the optimal solution.

4.3.2 LEO-to-GEO With PSO

Using PSO, the cognitive and social parameters were set to 3.4 and 0.2 respectively – which varies from the LEO-to-LEO case. Other than that, the same parameters were used in this case, as compared to the first. Using the above parameters, PSO was also able to converge to the same optimum as DE. In addition, PSO succeeded 100% of the time when a random seed analysis was performed.

Initially, the population size was set to 10 (half of that for the LEO-to-LEO case) to speed up convergence, due to the fact that this problem is conceptually easier to solve. However, intermittent convergence was witnessed across random seeds, so the population size was changed to 20, where constant success was seen.

4.3.3 LEO-to-GEO With CMA-ES

Finally, CMA-ES was used for this problem. The same values as the LEO-to-LEO case were used, with the exception of sigma. A larger sigma value was used because the search space for this problem is larger due to the fact that larger TOFs had to be accounted for. Similar to DE and PSO, CMA-ES solved this problem for the same optimum. However, CMA-ES only solved this problem 50% of the time when a random seed analysis was performed.

As described in section 4.2.4, the recombination weights, parent sizes, and crossover points were all changed in hopes of CMA-ES becoming more reliable; but again, success rates eclipsing the 50% plateau were not found.

4.3.4 LEO-to-GEO Termination

As mentioned in section 4.2.5, determining when to termination is certainly an inexact science. While running numerous simulations (with varying search pressure), the best solution found was 7.177908 km/s. As a result, runs

were terminated when they reached that value – so for the random seed analysis, termination was determined by whether or not the fitness value was below 7.18.

4.4 HEURISTIC COMPARISON

This section serves to compare the above algorithms to determine which one provided the best performance. For the first case, CMA-ES converged in the fewest number of function evaluations. Figure 5 shows a plot of the number of function evaluations for each heuristic to converge to the optimum.

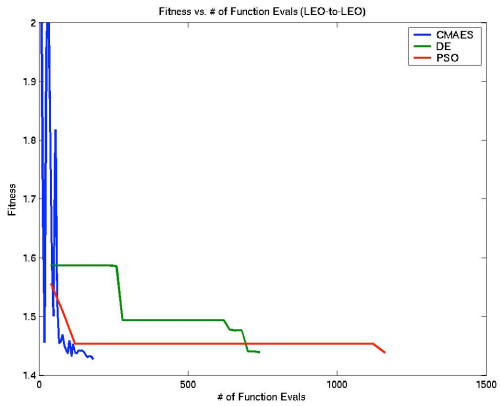

Figure 5: Number of Function Evaluations

Notice in figure 5 that CMA-ES converges in approximately 1/3 fewer function evaluations than DE, and 1/9 fewer function evaluations than PSO.

For the second case, DE converged in the fewest number of function evaluations – figure 6 shows a plot of the number of function evaluations as a function of fitness.

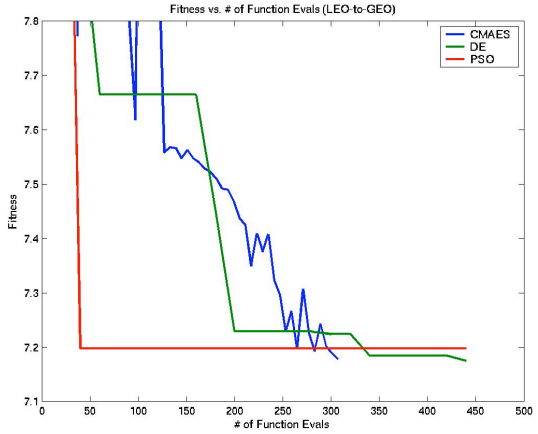

Figure 6: Number of Function Evaluations

Notice the effect of a different search space topology on the heuristics – CMA-ES converged the quickest for the first case, but that certainly did not occur in the second case.

However, the algorithm that converged the quickest is far from an adequate metric of algorithm efficacy. Another performance metric that must be considered is reliability. In order to measure reliability, figure 7 depicts the aforementioned random seed analysis performed on each algorithm. Figure 7 shows the number of failed runs (out of 10), and the average number of function evaluations for convergence.

	# Failed Per 10 Runs	Average # of NFE for Convergence
LEO-to-LEO		
DE	0	860
PSO	0	558
CMA-ES	7	249
LEO-to-GEO		
DE	0	474
PSO	0	190
CMA-ES	5	176

Figure 7: Random Seed Analysis

For the first problem, CMA-ES converged the quickest, but it also had a 30% success rate as compared to PSO and DE. For a problem where run time is not a significant factor, an algorithm with a high success rate outweighs a speedy algorithm with a poor success rate. As a result, PSO seemed to perform the best for case 1 because it had the 2nd fewest number of function evaluations, and it also had a perfect success rate.

For the second problem, again CMA-ES had the fewest number of function evaluations. However, it also had the poorest success rate at 50%. Again, because the run time is not extreme, the algorithm that has a higher success rate is better than one with quick convergence. Judging from the statistics from case 2, again PSO performed the best with its fairly low number of function evaluations coupled with its 100% success rate.

5 CONCLUSION

It seems as though the topology is more rugged in the LEO-to-LEO example as opposed to the LEO-to-GEO example. This variance in topology can be attributed to the fact that fewer infeasible spaces exist in the LEO-to-GEO case. This occurs because for such a long transfer, a larger number of TOFs create elliptical trajectories.

One can be confident the optima located by the EAs are the global optimums – although that is not a guarantee. Each heuristic was allowed to run for over 1000 generations with varying input parameters, and the optima outlined in this paper were the lowest that were seen. A 2-dimensional search space is not huge by any standards, therefore, one can be confident that the EAs found the global minimums

Both DE and PSO proved to be viable tools for solving each case. Where CMA-ES was quicker in terms of number of function evaluations, DE and PSO were much more reliable. In a case where run time is not significant, a reliable algorithm is much more desirable. A classical search method is not available for comparison due to the non-continuous nature of the search space; so as a result, heuristics are the sole means of a solution.

References

[1]Prussing, J.E. and Conway (1993), B.A. Orbital Mechanics, Oxford University Press Inc.

[2]Vallado, David A. (1999), Fundamentals of Astrodynamics and Applications.

[3]Storn, R., and Price, K. *Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces.* Journal of Global Optimization.

[4]Parsopoulos, K.E., *Recent Approaches to Global Optimization Problems through Particle Swarm Optimization*. 2002.

[5]Hanson, Nikolaus. *The CMA Evolution Strategy: A Tutorial*. 2005.