Parallel Monotonic Basin Hopping for Low Thrust Trajectory Optimization

Steven L. McCarty,* Melissa L. McGuire+
NASA Glenn Research Center, Cleveland, Ohio, 44135, USA

Monotonic Basin Hopping has been shown to be an effective method of solving low thrust trajectory optimization problems. This paper outlines an extension to the typical serial implementation by parallelizing it over any number of available compute cores. The Parallel Monotonic Basin Hopping algorithm described herein is shown to be a faster way to locate feasible solutions and improve locally optimal solutions in an automated way without requiring a feasible initial guess. The increased speed achieved through parallelization enables the algorithm to be applied to more complex problems that would otherwise be impractical for a serial implementation. Three low thrust example cases are used to demonstrate the effectiveness of the algorithm. Finally, a direct comparison between serial and parallel implementations demonstrates the expected improvement in solve time.

I.Introduction and Motivation

The process of optimizing low-thrust trajectories is complex and solutions are often non-intuitive. Multiple gravitating bodies, long duration finite burns, many Optimization Variables (OV), and complicated constraints can make convergence difficult to achieve. Such problems often require an experienced mission designer to carefully supervise the optimizer, making small tweaks to constraints and solver settings, while working towards convergence. As the complexity of the problem grows, so too does the time it takes to locate a feasible solution. If an experienced mission designer is able to successfully converge a complex problem to a locally optimal solution, there may still exist a more optimal solution lurking nearby. It was these factors that motivated the development of a method to aid the mission designer in locating feasible solutions and improving optimal solutions to complex low-thrust trajectory optimization problems in an automated way.

II.Monotonic Basin Hopping

Monotonic Basin Hopping (MBH) is a stochastic global optimization algorithm that has been shown to be effective for low-thrust spacecraft trajectory optimization problems. Further, it is known to be robust and require minimal human supervision during the optimization. The NASA-developed Evolutionary Mission Trajectory Generator, EMTG, is a notable example.[1,2] MBH works by repeatedly perturbing the problem's OV with some random distribution, known as hopping, and attempting a local optimization for each perturbed state. If a more feasible, or eventually optimal, solution is found, that solution provides a new set of OV that are then similarly perturbed. This process repeats serially until a specified time limit or maximum number of hops is reached.

Figure 1 is a simple example pseudo-code using MBH to minimize an objective function, where `x` is an array of OV, `x0` is the initial guess, `x*` is the array producing the current best objective function value, `xf` is the array after local optimization, and `b` is an array of random numbers. The distribution of random numbers is chosen such that `x` remains in the general vicinity of `x*` in order to take advantage of the assumed clustering of optimal solutions.[3] Selection of the random numbers and type of distribution is an important aspect of tuning the MBH algorithm[4], which is discussed further in section IV.

* Mission Design Engineer, Mission Architecture and Analysis Branch, Steven.McCarty@nasa.gov
+ Lead Mission Design Engineer, Mission Architecture and Analysis Branch, Melissa.L.McGuire@nasa.gov

```
1: x* = x0
2: while time < time_limit:
3:   b = random_numbers
4:   x = x* + b                          # hop
5:   run local optimization on x
6:   if objective < best_objective:
7:     x* = xf
```

Figure 1. Simple MBH Pseudo Code

Serial (i.e. single core) implementations of MBH can be coupled with more efficient simplified dynamics models, including low-thrust approximation using the Sims-Flanagan transcription with 2-body Kepler propagation.[2] With simplified dynamics, the amount of time needed for the local optimizer to attempt to solve a single instance of the problem small enough that a serial approach is useful. This efficiency lends itself well to the large number of function evaluations generally required by MBH. For very complex low-thrust trajectory optimization problems, with multiple gravitating bodies, fully integrated dynamics, long duration finite burns, and many OV, local optimization of a single instance of the problem can require significantly more time than the simplified alternative.

The Parallel Monotonic Basin Hopping (PMBH) algorithm described in this paper is an attempt to compensate for the increased solve time necessitated by more complex trajectory optimization problems by spreading the MBH algorithm across any number of available cores.

III. Parallel MBH

The algorithm described in this paper extends the typical MBH algorithm by parallelizing it over any number of available cores. This is achieved by splitting duties amongst a single head node and *n* workers. The head node is responsible for continuously receiving results from each worker, comparing the result to the best-known solution, and alerting the workers when a better solution should be used as the starting point (`x*`). Each worker continuously runs a serial MBH algorithm, sending each solution to the head node (without waiting for a reply) and checking after each hop if an updated starting point has been delivered. Once the workers are initially spawned, they are free to continuously hop without waiting for other workers to finish or for the head node to evaluate their results. This allows a large number of workers to collaboratively search the solution space without wasting wall clock time. In doing so, the more time expensive function evaluations required for complex higher-fidelity trajectory optimization problems can be compensated for by running many instances in parallel.

Figure 2 and Figure 3 contain pseudo code for the head node and each worker, respectively, where `X*` is the array of OV corresponding to the best objective function value at the head node.

```
1: X* = x0
2: send X* to workers
3:
4: for solution received from workers:
5:   if objective < best_objective:
6:     X* = xf
7:     best_objective = objective
8:     send X* to workers
```

Figure 2. Pseudo Code for PMBH Head Node

```
1: while time < time_limit:
2:   if X* received from head_node:
3:     x* = X*
4:   b = random_numbers
5:   x = x* + b                          # hop
6:   run local optimization on x
7:   send solution to head_node
```

Figure 3. Pseudo Code for PMBH Workers

In the implementation of PMBH described in the pseudo code above, all information is relayed by the head node allowing the processes at the workers to be independent of one another. This independence enables the algorithm to, in theory, scale linearly with the number of cores available. That is, the time to reach a particular solution should decrease linearly with the number of cores. A closer look at scaling can be found in section VI.

IV.Implementation

The details of how best to implement the PMBH algorithm will depend on the application and familiarities of the user. For example, the implementation described in this paper has been written in Python[5] and uses Copernicus[6] as the trajectory optimizer. Some of the specific implementation details are discussed in this section.

A. Trajectory Optimizer

In the context of this paper, the PMBH algorithm is a wrapper around some sort of mission design tool with an accompanying optimizer. In this case, Copernicus[6] was chosen as the mission design tool because of its extensive use in the Mission Architecture and Analysis Branch at NASA GRC. Copernicus is a very general mission design tool that enables the formulation of arbitrarily complex spacecraft trajectories with complicated constraints and objective functions. SNOPT[7], a general-purpose system for constrained local optimization, is used as the gradient based optimizer within Copernicus.

Copernicus has a number of built in command-line operation options that made for a more straightforward integration with PMBH. On of specific use is a command-line option to randomize the OV by a given percentage before running the optimizer, which eliminated the need to read and write files in order to do so. The next section, Random Number Generation, provides more detail about how the randomization percentage is determined.

The authors have also successfully implemented variations of PMBH with other mission design tools, including EMTG and GMAT[8], the details of which are not included here.

B. Random Number Generation

The random perturbation of the OV utilizes an internal Copernicus feature that randomizes each optimization variable according to Equation 1, where Δ_i is change in the individual optimization variable, x_i is its initial value, r_i is a uniform random number between -1 and +1 (different for each variable in the vector), and f is a user specified maximum percent change. It is through the maximum percentage change, f, that numbers from a particular probability distribution are used to drive the randomization.

Equation 1. Copernicus Internal OV Randomization

$$\Delta_i \begin{cases} \frac{r_i f}{100} \cdot x_i, & if\ x_i \neq 0 \\ \frac{r_i f}{100}, & if\ x_i = 0 \end{cases}$$

The selection of random numbers for use in perturbing the OV vector can play an important role in how quickly MBH improves the solution. It has been found that random numbers pulled from long tailed probability distributions, Pareto distributions in particular, can improve the efficiency and robustness of MBH algorithms.[4] The Pareto distribution has the probability density function shown in Equation 2, where a defines the shape and m is the scale that defines the minimum value. In this PMBH implementation, m is set to zero and a is chosen based on the sensitivity of the problem type. Large values of a tend to decrease the magnitude of Δ_i and maintain a search radius closer to the current solution, while the opposite is true for small values of a.

Equation 2. Pareto Probability Density Function

$$p(x) = \frac{am^a}{x^{a+1}}, for\ x > m$$

It has been found that $a = 2$ provides good performance through experimentation. Decreasing a by a small amount after each unsuccessful hop – and resetting it to the initial value after each better solution is located – is found to improve performance by widening the search radius as improved solutions become less frequent.

C. Feasible Solution Solving

The pseudo code algorithms above only update the best solution vector, `X*`, when a more optimal solution is found. When starting with an infeasible initial guess, such an algorithm would need to locate an optimal solution through random search before a better `X*` could be determined. To more efficiently reach the first converged solution from an infeasible initial guess, a two-step solving method is included. With the feasibility value of a given solution assigned based on the magnitude of the worst constraint violation, the first step is to search for increasingly feasible solutions, updating `X*` along the way, until a solution is found which meets a user specified feasibility threshold. At that point, step two continues searching for more optimal solutions as usual.

Figure 4 shows the modified head node pseudo code including the two-step solving. The worker pseudo code remains unchanged.

```
1: X* = x0
2: send X* to workers
3:
4: for solution received from workers:
5:   if no feasible solution found yet:
6:     if feasibility < best_feasibility:
7:       X* = xf
8:       best_feasibility = feasibility
9:       send X* to workers
10:  else:
11:    if objective < best_objective:
12:      X* = xf
13:      best_objective = objective
14:      send X* to workers
```

Figure 4. Pseudo Code for PMBH Head Node with Feasible Solution Solving

V. Demonstration Cases

The capability described in this paper, to automatically converge and/or improve solutions to low thrust trajectory optimization problems, has demonstrated extensive utility since it was developed. Outlined below are a few demonstration cases that show the types of results achieved. A general description of each case is provided, followed by a brief discussion of how PMBH was able to improve the solution. The cases include a low thrust cislunar orbit transfer from a Lunar Distant Retrograde Orbit (DRO) to a southern EML2 Near Rectilinear Halo Orbit (NRHO), a low thrust spiral from a High Earth Orbit (HEO) to an NRHO, and a roundtrip Mars conjunction class mission departing from an NRHO. These demonstration cases highlight the ability of PMBH to improve a locally optimal solution, converge an infeasible solution, and finally to converge and improve a solution to a very large optimization problem. All demonstration cases were run on a 28-core (2 Intel Xeon E7-4830v4 @ 2.00 GHz) Linux workstation with one head node and 27 workers.

A. Cislunar Orbit Transfer

The first demonstration case for this algorithm is a low thrust transfer from a L2 southern NRHO with a 9:2 lunar synodic resonance to a 70,000 km DRO. This transfer proves to be a fitting application for this algorithm due to the large number of local minima within the solution space and the non-intuitive nature of the optimal solutions. The initial guess for this transfer was a locally optimal solution generated by an experienced mission designer. This example demonstrates the ability of the algorithm to quickly improve solutions from a good initial guess.

Problem Definition

The NRHO and DRO are formulated in such a way that they can be constructed and phased as part of optimization process. The fully integrated trajectory uses the DE421 ephemeris with point mass gravity from the Earth, Moon, and Sun. The continuously time varying finite burns use representative polynomial thrust and mass-flow curves for three 13.3 kW Hall thrusters. The optimization objective function is minimum propellant mass. In total, this problem contains 55 optimization variables and 21 nonlinear constraints. Table 1 describes the details of the example problem. Figure 5 shows a representative initial NRHO and final DRO in the Earth-Moon rotating frame to better illustrate the transfer.

Table 1. Cislunar Orbit Transfer Example Parameters

Metric	Value
Initial Mass	24,500 kg
SEP Power @ 1AU	40 kW ($1/R^2$)
Nominal Engine Isp	2600 s
Objective Function	Minimum Propellant
Gravitating Bodies	Earth, Moon, Sun
Ephemeris	DE421
Copernicus Segments	17
Optimization Variables	55
Nonlinear Constraints	22

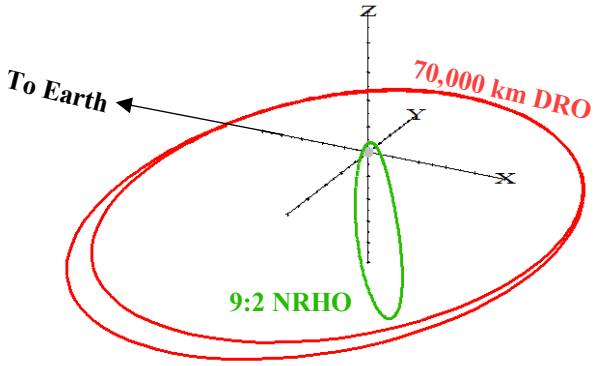

Figure 5. Representative NRHO and DRO in Moon-Centered, Earth-Moon Rotating Frame

Results

The PMBH run time in this case was limited to one hour, during which no human supervision was required. Figure 6 plots the objective function evolution over the course of the run time. The initial guess was a locally optimal solution that required 180 kg of propellant. After 750s (12.5 minutes) of run time and 224 hops, the propellant required decreased by 61% to 70 kg. No improvement in the solution was found after this time. The solution space was explored at an average rate of 18 hops per minute.

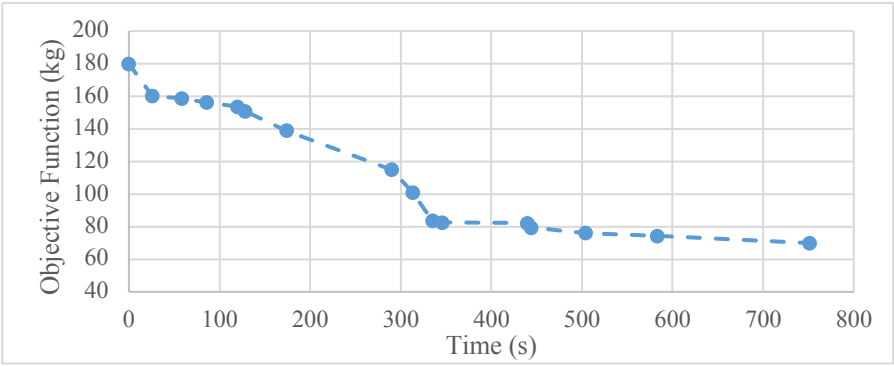

Figure 6. Plot of Objective Function (Propellant Mass) vs. PMBH Run Time

Figure 7 shows a side-by-side comparison of a portion of the initial and final solutions in the Earth-Moon rotating frame. The NRHO is in green, the DRO is in red, coast arcs are blue, and thrust arcs are orange. While the final solution looks less chaotic, it is not immediately obvious which would produce a more efficient transfer or why. This demonstrates the ability of PMBH to improve solutions when dealing with trajectories in non-intuitive solution regimes.

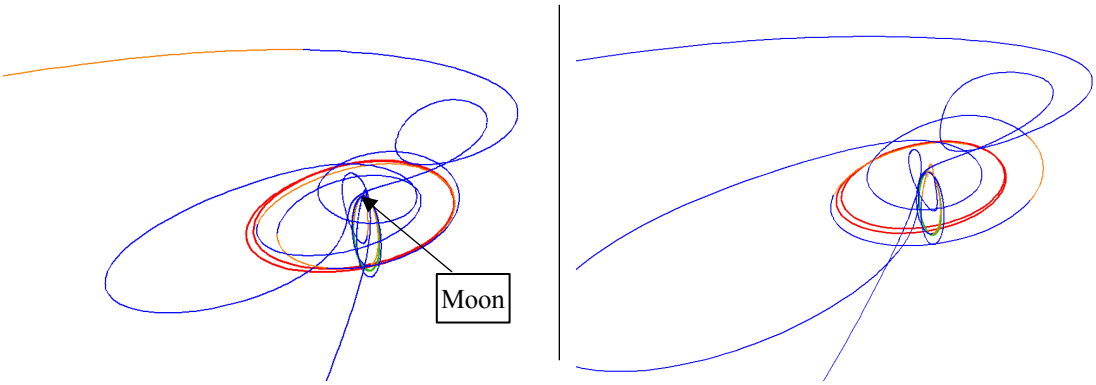

Figure 7. Comparison of Initial (left) and Final (right) Solutions in the Earth-Moon Rotating Frame.

B. Spiral to NRHO

The second demonstration is a low thrust transfer from a 24-hour period HEO to an NRHO with a 9:2 lunar synodic resonance, including a 100+ day low-thrust spiral. Due to the sensitivities of this problem type, such as the long low-thrust spiral, reaching convergence can be tedious for a mission designer. The initial guess for this demonstration was an infeasible solution generated while an experience mission designer worked to converge the problem. This example demonstrates the ability of the algorithm to converge infeasible solutions, and then improve upon the solution once initial convergence is reached.

Problem Definition

The NRHO is formulated in such a way that it can be constructed and phased as part of optimization process. The fully integrated trajectory uses the DE421 ephemeris with point mass gravity from the Earth, Moon, and Sun. The continuously time varying finite burns use representative polynomial thrust and mass-flow curves for three 13.95 kW Hall thrusters. The optimization objective is minimum propellant mass. In total, this problem contains 55 optimization variables and 21 nonlinear constraints. Table 2 describes the details of the example problem.

Table 2. GTO to NRHO Transfer Example Parameters

Metric	Value
Initial Mass	11,000 kg
SEP Power @ 1AU	42 kW ($1/R^2$)
Nominal Engine Isp	2600 s
Objective Function	Minimum Propellant
Gravitating Bodies	Earth, Moon, Sun
Ephemeris	DE421
Copernicus Segments	14
Optimization Variables	55
Nonlinear Constraints	21

Results

The PMBH run time in this case was limited to one hour. Figure 8 plots the objective function and feasibility evolution over the course of the run time. The first feasible solution was reached after 200s and 17 hops. After 3600s (60 minutes) of run time and 292 hops, the propellant required decreased by 7% from 984 kg to 918 kg. The solution space was explored at an average rate of 5 hops per minute. Most importantly, PMBH was able to quickly turn the infeasible initial guess into a feasible solution with no human supervision required.

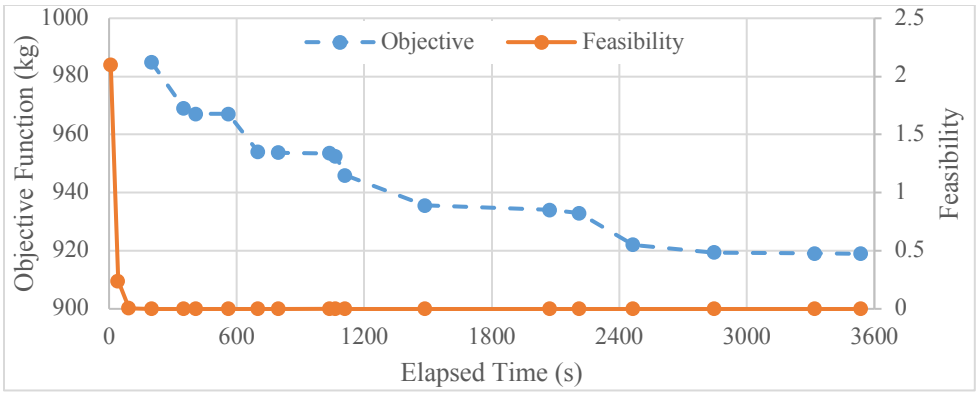

Figure 8. Plot of Objective Function (Propellant Mass) and Feasibility vs. PMBH Run Time

Figure 9 shows the initial guess and final optimal trajectory the Earth-centered J2000 frame. Thrust arcs are red, coast arcs are blue, the Moon's trajectory is grey, and Earth is the blue dot in the center. In addition to actually being feasible, the final trajectory has visibly shorter thrust arcs resulting in less propellant usage.

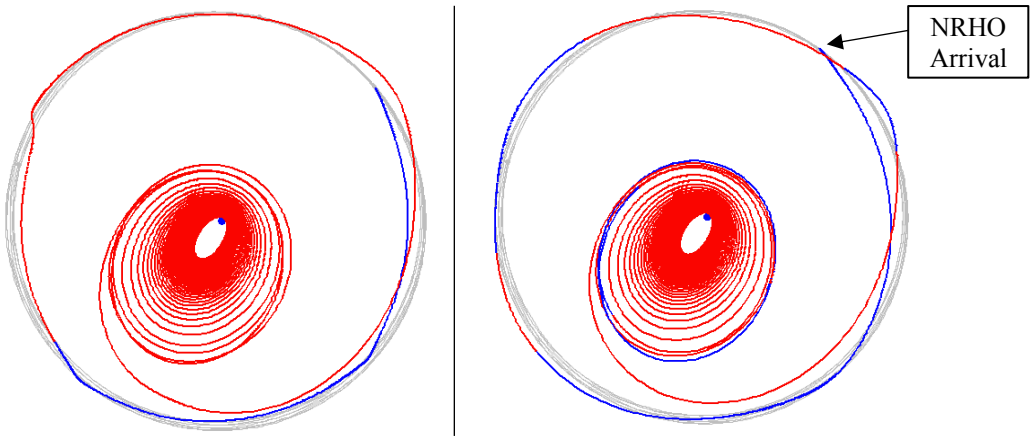

Figure 9. Comparison of Initial (left) and Final (right) Solutions in the Earth-Centered J2000 Frame.

C. Roundtrip Mars Mission

The third and final demonstration is a round-trip Mars conjunction class mission with a hybrid vehicle utilizing low and high thrust propulsion. Due to the sensitivities of this problem, including LGA departure sequence, higher order Mars gravity, and multiple propulsion systems, reaching convergence can be tedious. Complicating that effort further, a single call to SNOPT can take upwards of 45-minutes to execute. The initial guess for this transfer was an infeasible solution generated while an experience mission designer worked to converge the problem. This example demonstrates the ability of the algorithm to locate feasible solutions from an infeasible starting point, and then improve upon the solution once initial convergence is reached. Also, the parallelization of PMBH enables this class of problem to be run in a reasonable amount of time compared to a serial MBH implementation. See section VI for a comparison of serial MBH vs. PMBH.

Problem Definition

The end-to-end optimized mission begins in a Lunar NRHO, completes a Lunar Gravity Assist (LGA) departure sequence, and then executes a chemical burn to depart for Mars. The electric propulsion system propels the vehicle to Mars where it captures into a 5-sol orbit with a chemical burn targeting a landing latitude of 18.8 degrees. During the 300-day stay at Mars, the low thrust system maneuvers the vehicle to target the appropriate orbital elements to rendezvous with a Mars Ascent Vehicle and depart using another chemical burn. The low thrust system then pushes the vehicle back to Earth intercept.

The fully integrated trajectory uses the DE421 ephemeris with point mass gravity from the Earth, Moon, and Sun and 4x4 spherical harmonics for Mars. The continuously time varying finite burns use representative polynomial thrust and mass-flow curves for 26 13.3 kW Hall thrusters. The optimization objective is minimum initial mass for a fixed final mass. In total, this problem contains 58 optimization variables and 27 nonlinear constraints. Table 3 describes the details of the example problem.

Table 3. Mars Round Trip Example Parameters

Metric	Value
Final Mass	59,000 kg
SEP Power @ 1AU	318 kW ($1/R^2$)
Nominal Engine Isp	2600 s
Objective Function	Minimum Initial Mass
Gravitating Bodies	Earth, Moon, Sun, Mars
Ephemeris	DE421
Copernicus Segments	35
Optimization Variables	58
Nonlinear Constraints	27

Results

The PMBH run time in this case was limited to 6 hours. Figure 10 plots the objective function and feasibility evolution over the course of the run time. The first feasible solution was reached after 24 minutes and 39 hops. After 5.5 hours of run time and 822 hops, the initial mass required decreased from 101.2 mt to 98.3 mt. The solution space was explored at an average rate of 2.5 hops per minute. Most importantly, PMBH was able to quickly turn the infeasible initial guess for a very complex and sensitive problem into a feasible solution with no human supervision required.

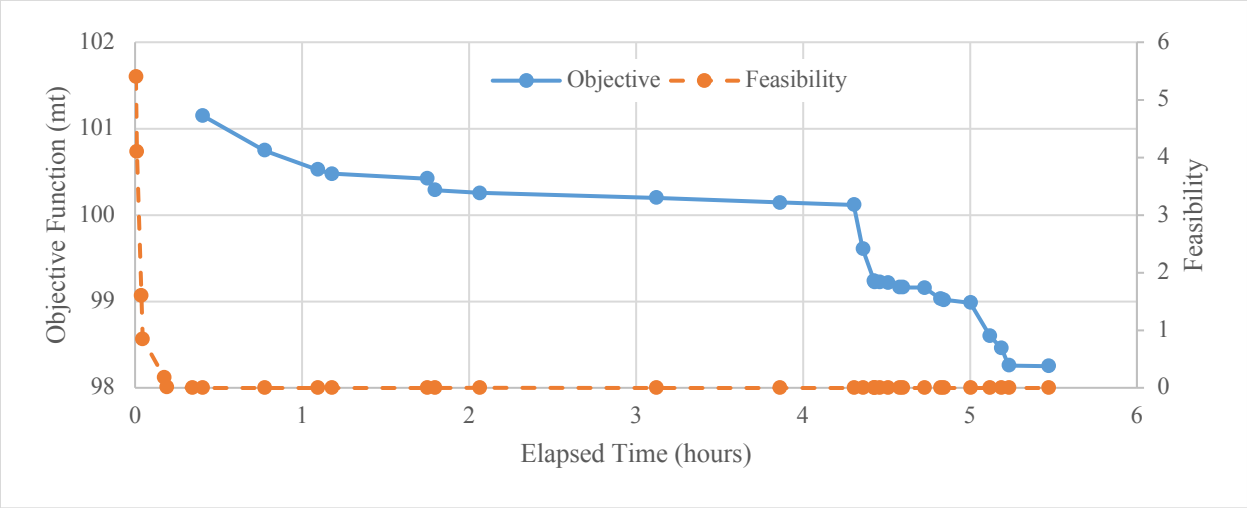

Figure 10. Plot of Objective Function (Initial Mass) and Feasibility vs. PMBH Run Time

Figure 11 shows the initial guess and final optimal trajectory the Sun-centered Ecliptic J2000 frame. Thrust arcs are red, coast arcs are blue, Earth's trajectory is thin blue, and Mars' trajectory is thin red. There are only minimal visual differences between the two trajectories at this scale, as much of the sensitive portions are near Earth and Mars.

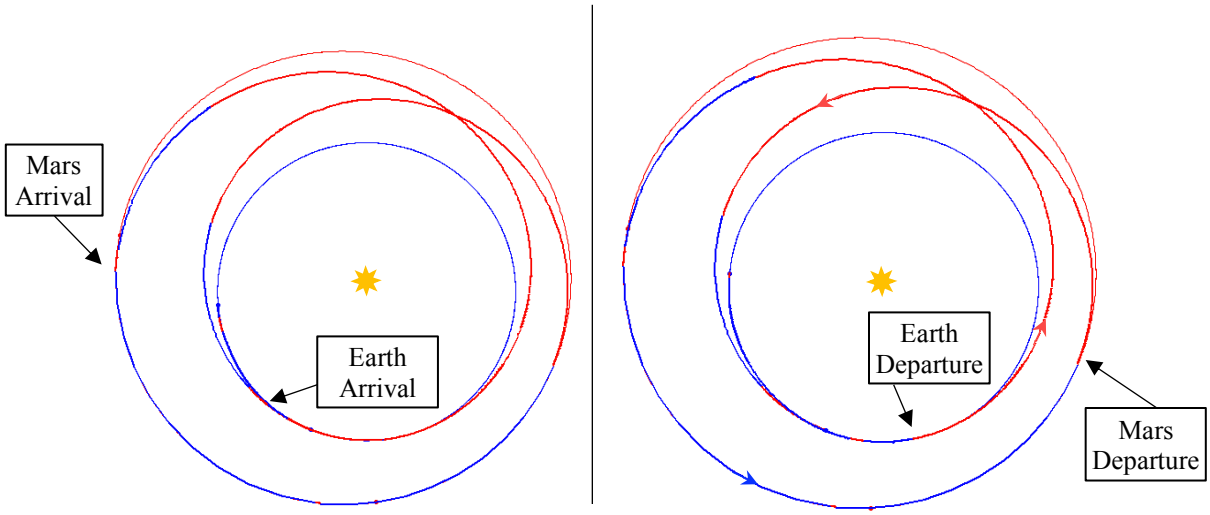

Figure 11. Comparison of Initial (left) and Final (right) Solutions in the Sun-Centered Ecliptic J2000 Frame.

VI.MBH vs. PMBH Scaling

The purpose of parallelizing MBH is to increase the speed at which the algorithm can search the solution space. One way to measure this speed is the number of hops performed by the workers per minute (HPM). In order to characterize this performance, 20 trials of each demonstration case (10 for Mars) were completed with 1 worker (i.e. serial MBH) and with 27 workers. Multiple trials for each case are necessary because of the stochastic nature of the algorithm. The results from all trials were then averaged to obtain a single HPM value for each case/workers combination. All PMBH run times were limited to one hour, except the Mars roundtrip cases, which were limited to 3 hours. In addition to measuring the speed of the algorithm, HPM is also a relative measure of how complex a problem is. For example, a lower average HPM means that more time is required by SNOPT to evaluate each hop.

The scale factor is defined as the ratio of HPM for 27 workers divided by the HPM for 1 worker. In this case, the scale factor would be equal to 27 if the algorithm scales exactly linearly with the number of workers. In practice, sub-linear scaling is observed. This could be partially due to the way the specific hardware used is able to utilize turbo boost to increase the clock speed of a single core from 2.0 to 2.8 GHz, whereas it is unable to when fully utilizing all cores at maximum thermal design power. Table 4 shows the scale factors achieved for each of the demonstration cases.

Table 4. PMBH Scaling Results for 10 Trials

Demonstration Case	Average HPM (1 worker)	Average HPM (27 workers)	Scale Factor
1. NRHO to DRO	1.29	17.80	13.8
2. HEO to NRHO	0.21	4.77	22.7
3. Mars Roundtrip	0.12	2.46	20.5

The scale factors observed ranging from approximately 14-23 indicates that it would, on average, take the serial MBH implementation 14-23 times longer to reach the same solution as the PMBH algorithm with 27 workers. More specifically, for the Mars roundtrip example, it is expected that the serial MBH implementation would require 61.5 hours to arrive at the solution reached by the 27-worker PMBH algorithm in 3 hours.

While HPM is a good measure of the speed of the algorithm, it does not account for other factors which impact the time to reach a specific solution. One advantageous factor is that each worker in a parallelized scheme is more efficient than a serial worker because of the cooperative nature of the algorithm. For example, since the time to evaluate each hop can vary widely, a single parallel worker may evaluate 1 hop while another evaluates 5, yet both workers benefit from whatever information was gained from the full 6 hops. This should result in fewer total hops being required to solve a problem.

One disadvantage of the PMBH described here manifests because a new `X*` is only read by a worker after the current hop is complete. Specifically, some workers will be optimizing a hop from the previous `X*` while a new `X*` has been delivered to them, whereas a serial implementation would always be operating with the latest `X*`. As problems become more complex, more time is necessary to evaluate each hop, which means more time will be spent evaluating a hop from a previous solution while a better solution is waiting. This should result in more total hops being required to solve a problem.

Figure 12 contains plots of objective function vs. run time for all trials in the scaling study. Red dashed lines correspond to serial MBH trials. If the serial MBH trial only found a single feasible solution, that solution is plotted as a red x. Blue lines correspond to PMBH trials with 27 workers. Each line begins at the first feasible solution found and ends at the last. Since the NRHO to DRO transfer started with a feasible solution, the lines begin at run time equal to zero. The other two examples started with an infeasible initial guess, so some time is spent finding the first feasible solution.

From the NRHO to DRO Transfer plot, it is immediately clear that the blue PMBH trials find better solutions much faster than the red serial trials when starting from a feasible initial guess. In addition to improving the solution faster, the HEO to NRHO to Mars Roundtrip plots show that the PMBH trials find the first feasible solution faster when starting from an infeasible initial guess. Further, many of the serial MBH trials did not find a single feasible solution in the allotted time. In the HEO to NRHO Transfer trials, only 14 out 20 found at least one feasible solution. In the Mars Roundtrip trials, only 2 out of 10 found at least one feasible solution. Feasible solutions were found for all PMBH trials. This result demonstrates the ability of PMBH to solve a complex class of problems that would otherwise be impractical with a serial MBH implementation.

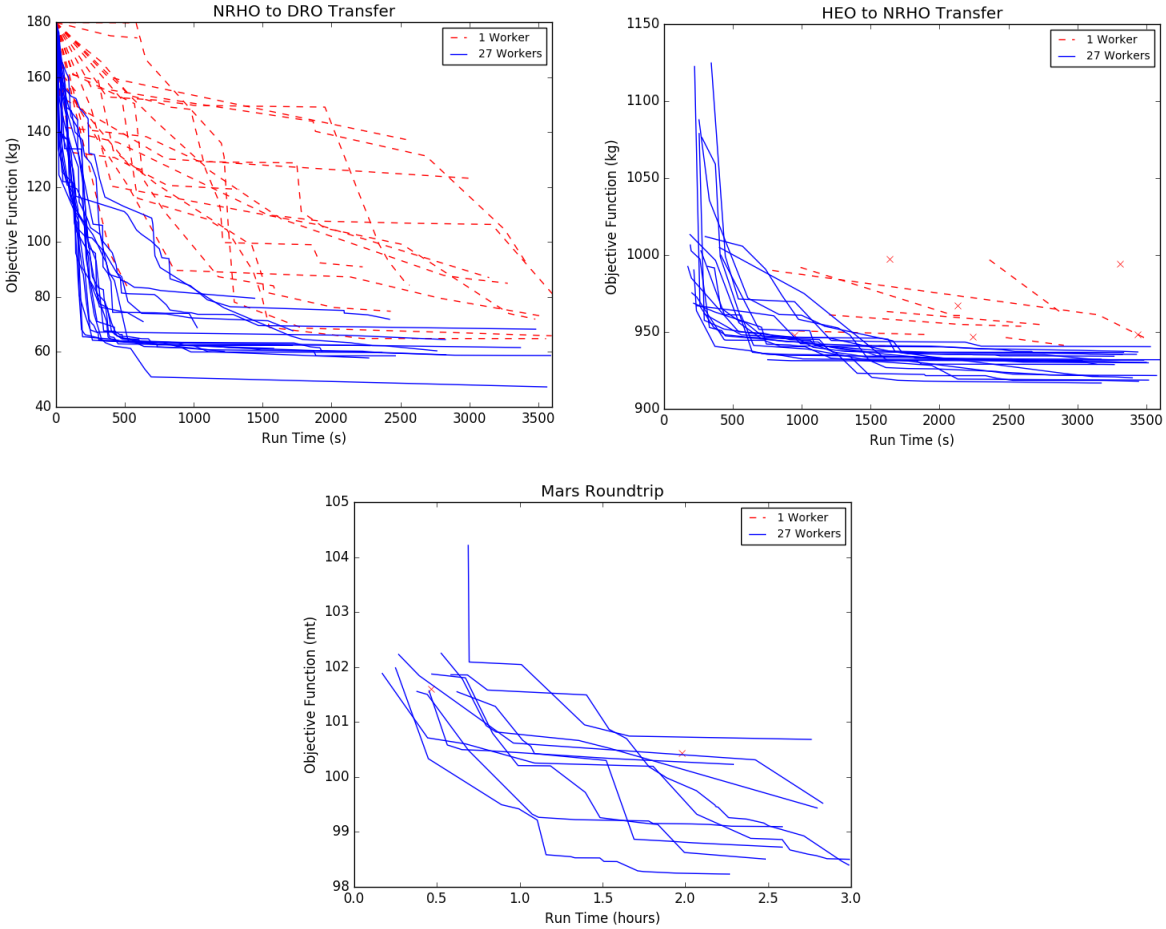

Figure 12. Plots of Objective Function vs. Run Time for all Trials of the NRHO to DRO Transfer (top left), HEO to NRHO Transfer (top right), and Mars Roundtrip (bottom center)

VII. Conclusion

In conclusion, PMBH has been shown to be a very useful tool for solving low thrust trajectory optimization problems. It is able to find feasible solutions from infeasible initial guesses and improve optimal solutions once feasibility is reached. Further, this method is able to achieve these results in an automated way without human supervision. The parallelization not only improves the speed at which solutions can be found and improved, but enables the solving of very complex low thrust trajectory optimization problems that would otherwise be impractical with a serial MBH implementation. Lastly, while PMBH was presented here in the context of complex low thrust trajectory optimization, it would also have application to a wide range of numerically simpler trajectory optimization problems.

Acknowledgments

The authors would to extend a warm thank you to our colleagues at NASA Glenn Research Center for their contributions. Specifically, Laura Burke for providing initial Copernicus input decks for the demonstration cases, Steve Oleson for securing the funding for the computing resources used in this effort, and Les Balkanyi for carefully reviewing the final manuscript.

References

[1] Vavrina, M., Englander, J., Ellison, D., "Global Optimization of N-Maneuver, High-Thrust Trajectories Using Direct Multiple Shooting," AAS/AIAA Spaceflight Mechanics Meeting, Napa, CA, February 2016.

[2] Englander, J., Vavrina, M., Ghosh, A., "Multi-Objective Hybrid Optimal Control for Multiple-Flyby Low-Thrust Mission Design," AAS/AIAA Space Flight Mechanics Meeting, Williamsburg, VA, January 2015.

[3] Conway, B., "Spacecraft Trajectory Optimization", Cambridge University Press, New York, 2010.

[4] Englander, J., Englander, A., "Tuning Monotonic Basin Hopping: Improving the Efficiency of Stochastic Search as Applied to Low-Thrust Trajectory Optimization", International Symposium on Space Flight Dynamics, Laurel, MD, May 2014.

[5] Python Software Foundation, Python Language Reference, version 2.7, http://www.python.org

[6] Williams, J., Senent, J., Ocampo, C., Mathur, R., Davis, E., "Overview and Software Architecture of the Copernicus Trajectory Design and Optimization System", 4th International Conference on Astrodynamics Tools and Techniques, Madrid, May 2010.

[7] Gill, P.E., Murray, W., and Saunders, M.A., "SNOPT: An SQP algorithm for large-scale constrained optimization." SIAM Journal on Optimization, Vol. 12 No. 4, 2002, pp. 979-1006.

[8] Hughes, Steven P., "General Mission Analysis Tool (GMAT)", International Conference on Astrodynamics Tools and Techniques (ICATT) Darmstadt, 2016.