A METHODOLOGY FOR SEQUENTIAL LOW THRUST TRAJECTORY OPTIMIZATION USING PREDICTION MODELS DERIVED FROM MACHINE LEARNING TECHNIQUES

A Thesis
Presented to
The Academic Faculty

By

John Casey

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in the
School of Aerospace Engineering

Georgia Institute of Technology

May 2019

A METHODOLOGY FOR SEQUENTIAL LOW THRUST TRAJECTORY OPTIMIZATION USING PREDICTION MODELS DERIVED FROM MACHINE LEARNING TECHNIQUES

Submitted for Approval by:

Dr. Dimitri Mavris, Advisor
Guggenheim School of Aerospace Engineering
Georgia Institute of Technology

Dr. Alicia Sudol
Guggenheim School of Aerospace Engineering
Georgia Institute of Technology

Dr. Bradford Robertson
Guggenheim School of Aerospace Engineering
Georgia Institute of Technology

Date Submitted: May, 2019

ACKNOWLEDGEMENTS

I would first thank my thesis advisor Dr. Dimitri Mavris. His guidance and advice were indispensable tools in navigating my way through this thesis and graduate school. I would also like to thank Dr. Alicia Sudol and Dr. Bradford Robertson for their invaluable advice as I worked on this thesis. I couldn't have done it without you!

Next, I would like to thank my labmates. In particular, Caleb Harris, Gene Chen, Jimmy Tang, and Justin Fan helped enormously. From taking coffee breaks to late night work sessions, you guys were always a joy to be around.

I would like to thank my parents for putting up with me and supporting me in whatever I did. Also, thanks for giving birth to me.

Finally, I would like to thank my girlfriend Ana Enriquez for always having my back and never wavering in her support for me, even at my lowest points.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Spacecraft trajectory sequence optimization has been a well-known problem for many years. Difficulty in finding adequate solutions arises from the combinatorial explosion of possible sequences to evaluate, as well as complexity of the underlying physics. Since there typically exists only minuscule amounts of acceptable solutions to the problem, a large search of the solution space must be conducted to find good sequences. Low thrust trajectories are of particular interest in this field due to the significant increase in efficiency that low thrust propulsion methods offer. Unfortunately, in the case of low thrust trajectory problems, calculations of the cost of these trajectories is computationally expensive, so estimates are used to restrict the search space before fully solving the trajectory during the mission planning process. However, these estimates, such as Lambert solvers, have been shown to be poor estimators of low thrust trajectories. Recent work has shown that machine learning regression techniques can be trained to accurately predict fuel consumption for low thrust trajectories between orbits. These prediction models provide an order of magnitude increase in accuracy over Lambert solvers while retaining a fast computational speed.

In this work, a methodology is developed for integration of these machine learning techniques into a trajectory sequence optimization technique. First a set of training data composed of low thrust trajectories is produced using a Sims-Flanagan solver. Next, this data is used to train regression and classification models that respectively predict the final mass of a spacecraft after a low thrust transfer and predict the feasibility of a transfer. Two machine learning techniques were used: Gradient boosting and artificial neural networks. These predictors are then integrated into a sequence evaluation evaluation scheme that scores a sequence of targets to visit according to the prediction models. This serves as the objective function of the global optimizer. Finally, this objective function is integrated into a Genetic Algorithm that optimizes sequences of targets to visit. Since the objective function of this algorithm uses predictions to score sequences, the final sequence

is evaluated by a Sims-Flanagan low thrust trajectory solver to evaluate the efficacy of the method. Additionally, a comparison is made between the global optimization results with two different objective functions: One based that score sequences using the machine learning predictors, and one that uses Lambert solvers to score sequences. This allows for a measurement of the this method's improvement in the global optimization results.

Results of this work demonstrate that the developed methodology provides a significant improvement in the quality of sequences produced by the Genetic Algorithm when paired with the machine learning predictor based objective function. Both gradient boosting and artificial neural networks are shown to be accurate predictors of both the fuel usage and feasibility of low thrust trajectories between orbits. However, gradient boosting is found to offer improved results when evaluating sequences of targets to visit. When paired with the Genetic Algorithm global optimizer, both the gradient boosting prediction model and the artificial neural network model produce similar results. Both are shown to offer a significant improvement over the Lambert solver based objective function while maintaining similar speeds.

The positive results this methodology yields lends support to the notion that the use of machine learning techniques has the potential to improve the optimization of sequences of low thrust trajectories. This work lays down a framework that can be applied to preliminary mission planning for space missions outfitted with low thrust propulsion methods. Such missions include, but are not limited to, multiple main-belt asteroid rendezvous, debris removal from Earth orbit, or an interplanetary tour of the solar system.

CHAPTER 1
INTRODUCTION AND BACKGROUND

Space missions often involve flybys and visitations to multiple bodies within the Solar System. Due to the tight constraints on spacecraft mass, mission planners seek to minimize the amount of spacecraft propellant necessary to achieve the scientific objectives of the mission. This minimization of fuel usage can be achieved by using a single spacecraft to visit multiple bodies that are of scientific interest. Some examples of missions that included multiple planetary flybys include the *Voyager* program,[1] the *Pioneer* program,[2] the *Dawn* mission,[3] and the *New Horizons* mission.[4] In planning these missions, planners must carefully optimize the trajectory of spacecraft. In particular, the *Dawn* mission involved a spacecraft outfitted with a low thrust ion engine that dramatically improved engine efficiency. This allowed the spacecraft to visit three different celestial bodies- Mars, Vesta, and Ceres- as well as become the first spacecraft to orbit two extraterrestrial bodies. As can be seen in Figure 1.1, the trajectory of the *Dawn* spacecraft took nearly a decade to complete, involved multiple revolutions around the sun, and included many phases of thrusting and coasting. Due to the non-linear and undulating nature of orbital transfer windows, and underlying complexity of spacecraft dynamics, optimizing spacecraft trajectories that visit multiple bodies is an extremely difficult problem to solve.

As complex as missions like *Dawn* may be, there are some common threads that allow for an insightful breakdown of the problem. In general, the problem can be presented as follows. There are some number of target bodies that are to be visited by a spacecraft. For the purposes of this thesis, a field of asteroids in the Asteroid Belt of the Solar System will be considered to be these target bodies. In the planning stages of this mission, a primary goal will be to find the sequence of asteroids to visit that will maximize the amount of visits. Since the number of potential targets in the Asteroid Belt are in excess of a million

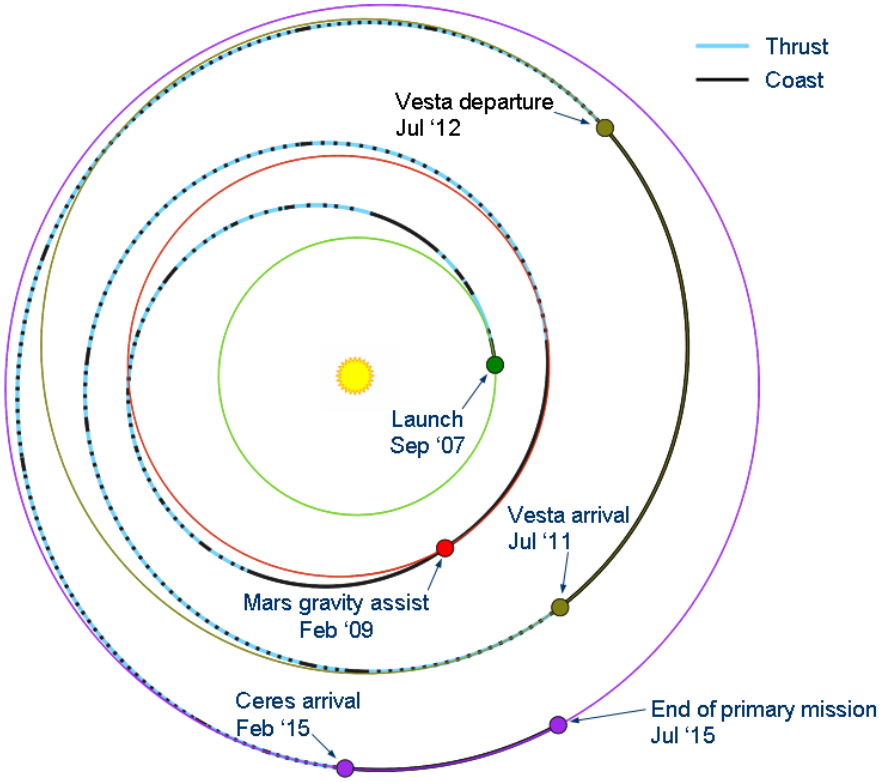

Figure 1.1: The orbital trajectory of the *Dawn* spacecraft[5]

asteroids, analyzing every potential sequence is impossible. This problem of finding the best sequence among near endless possible combinations is actually a fairly well known problem: The Travelling Salesman Problem.[6]

In the Travelling Salesman Problem (TSP), the following question is asked: "Given a list of cities and the distance between each city, what sequence of visitation minimizes the total path taken?" This question has pronounced parallels to the trajectory sequence optimization problem posed in the previous paragraph. This comparison allows for a re-formatting of the problem: "Given a list of target asteroids and their orbital characteristics, what sequence of visitations minimizes the total spacecraft fuel required?" Making this parallel helps to characterize the problem as a variant of the Travelling Salesman Problem. This parallel also highlights the additional layer of complexity to this problem. Looking at the differences between the two questions, the main difference between the two is that

in the prototypical TSP, the distances between cities is a known constant. However, in this problem, only the orbital characteristics of the asteroids are known. All things in orbit are constantly moving and accelerating, this means that the fuel required for a transfer between any two asteroids is also constantly changing with time. Thus, this spacecraft trajectory sequence optimization problem can be characterized as a unique time-variant version of the Travelling Salesman Problem.

This novel problem has been recognized through the European Space Agency's Global Trajectory Optimization Competition (GTOC). Though the scenarios and particular objectives change from competition to competition, the underlying problem of this trajectory sequence optimization problem remains constant. Some examples of the problems posed are: Maximizing the visitations of asteroids using a spacecraft with low thrust propulsion (GTOC 4,[7] GTOC5[8]); Remove a field of orbital debris with a spacecraft (GTOC 9);[9] Finding the best sequence of asteroids to collect samples from (GTOC 3).[10] While these are all different in objective, they all have a common thread of a combinatorial optimization combined with orbital trajectory optimization.

By examining the construction and solution of these GTOC-class problems, a general approach to solving these types of problems can be formulated. This approach deals with the two main elements of the optimization problem: an *outer loop* and an *inner loop*. In short, the outer loop deals with global optimization of sequences, while the inner loop deals with the optimization of evaluating transfers in a sequence, producing the cost function used by the outer loop.

1.1 The Outer Loop

As previously mentioned, the outer loop deals exclusively with finding the best sequence of targets to visit. In this sense, it is effectively blind to the physics of the problem, and operates solely based on the cost of a sequence returned by the inner loop. In practice, the blindness of the outer loop can mean that a large variety of optimization algorithms can be

used to find solutions. Indeed, a sampling of GTOC[11] solutions shows that there are many optimization techniques used as the outer loop. Some examples of the global optimization techniques include branch and prune methods, genetic algorithms, and global search trees.

As an example of how a global optimization technique would be applied, consider an evolutionary algorithm.[12] In this case, a set of initial sequences of targets would be generated and then evaluated by the inner loop. Using the cost of each sequence as evaluated by the inner loop, the top performing sequences would be iterated upon by the evolutionary algorithm in some way. These new sequences would then be evaluated by the inner loop, and sent back to the outer loop. This process would repeat until a stop condition is met.

1.2 The Inner Loop

The inner loop deals exclusively with solving the physics of the problem and generating the cost function to be used by the outer loop. If given a sequence of targets to visit, it goes through the sequence, finding the optimal trajectory transfer between each target. In this manner, it solves the timing and fuel consumption of each transfer in the sequence.

The inner loop has several layers of complexity. The first is solving the actual trajectory between the initial orbit and the target. At any particular time, the spacecraft can initiate a transfer to the next target. Given a time of flight, departure time, and desired orbit, the overall trajectory and fuel cost can be determined. For an impulsive maneuver, the solution to this particular part of the inner loop is found by a straightforward algorithm known as Lambert's Problem.[13] For low thrust maneuvers, however, the underlying mathematics is much more complex and difficult to solve.[14]

Now that the cost of a transfer at any particular time has been determined, the lowest cost transfer must be found. Since both the spacecraft and target are constantly moving and changing directions, the cost of transferring also changes accordingly. The lowest cost transfer is found by varying the departure date and time of flight, until an optimal transfer cost is found. Some algorithms can help find good transfer windows, such as a grid search

or a gradient search, but this presents its own unique optimization problem.

The process of finding and initiating the best transfer must be repeated for each target that is to be visited in the sequence. Furthermore, as time progresses through the sequence, the target positions and spacecraft positions changes drastically, which effects the cost of the next transfer. In the end, the inner loop steps through each transfer, performing its own optimization problem for every step, until it reaches the end of the sequence (or another constraint is met), and calculates the score of the sequence, which is typically the number of targets the spacecraft can feasibly visit. This score is then returned to the outer loop.

In particular, the inner loop of this problem serves as a computational bottleneck that can limit the amount of sequences evaluated by the outer loop. When considering impulsive maneuvers, the spacecraft transfers can be calculated by solving Lambert's Problem. This is a straightforward problem to solve and has many robust and fast methods of solving it.[15] On the other hand, calculating optimal trajectories for spacecraft with low thrust propulsion is much more computationally difficult than a Lambert solver.

1.3 Low Thrust Transfers

Spacecraft using low thrust propulsion systems (such as electric propulsion) are of particular interest due to the vastly increased efficiency over chemical propulsion systems. A spacecraft using low thrust propulsion methods can expect to see an order of magnitude increase in engine efficiency. Typical low thrust engines have specific impulses on the order of 3000 seconds, while chemical rockets have impulses around 300 seconds. For example, the *Dawn* spacecraft was outfitted with a Xenon based electric low thrust propulsion system.[3] In total, *Dawn* carried 425kg of propellant during it's mission. The efficiency of its engine allowed for a total ΔV in excess of 11km/s.[16] If *Dawn* had used a traditional chemical propulsion system with the same amount of propellant, the maximum ΔV available would have been less than 2km/s. This massive boost in available ΔV allowed for *Dawn* to be the first spacecraft to orbit more than one extraterrestrial body (asteroid Vesta

in 2011 and dwarf planet Ceres in 2015). Without such efficient engines, this feat would have required orders of magnitude greater fuel use.

However, this performance comes at a cost when considering trajectory optimization. The low thrust nature means that in order to gain any meaningful velocity, the low thrust engines must operate for long periods of time, typically on the order of months. For mission planners, this means that the relatively straightforward process of solving Lambert's Problem can no longer be applied. Instead, complex and computationally expensive calculations must be performed to determine valid trajectories. When applied to a GTOC-class problem, this can cause a computational bottleneck to form, limiting the search space of the global optimizer. In this way, a low thrust trajectory sequence optimization problem presents a unique challenge due to both the TSP type overarching problem, and the computationally intensive solutions of the Inner Loop.

Examining GTOC 4 helps provide insight into this issue. GTOC 4 presented competitors with the problem of maximizing the number of asteroids visited by a spacecraft outfitted with a low thrust propulsion system.[7] An overview of the submitted solutions shows that many contestants used a Lambert Solver to provide initial estimates of the low thrust trajectory maneuvers.[11] These estimates were then used to decide which sequences to examine in more detail. However, Lambert's Problem is a very poor estimator of low thrust trajectories,[17] meaning that the best sequences found by using this estimate may have not been selected if the physics had been fully solved.

This estimator issue presents a gap in the current approach: There is no adequate estimator or surrogate to provide fast, reliable solutions to low thrust trajectory problems.

1.4 Problem Identification and Overview

In GTOC type problems related to low thrust trajectories, there are multiple levels complexity. The first level of complexity is of the travelling salesman problem variety. The amount of possible sequences means that a search of the solution space is subject to com-

binatorial explosions. The second level of complexity is in evaluation of sequences. Given the nature of orbital mechanics, evaluating a sequence of targets to visit is a complicated and unpredictable process. Finally, at the base level, there is the optimization of transfer trajectories between targets. Solving this problem is a fundamental aspect in solving the global optimization problem. However, the computational complexity of solving low thrust trajectories makes this a particularly difficult problem to solve.

Thus, a method of accurately estimating the cost of low thrust transfers would allow for a larger search space to be examined. This would allow for the generation and evaluation of better target sequences. The method of using Lambert Solvers to estimate the cost is actually quite a poor predictor, so different prediction methods are considered in this thesis. As some machine learning techniques are particularly well suited to problems with nonlinear responses, they are of particular interest to this work. The development of such an estimator would likely allow for a measurable improvement in global optimization.

CHAPTER 2
PROBLEM FORMULATION

The previous chapter introduced the background of the problem that this thesis will address. An in depth examination of the topic and current methods and approaches will allow for a more comprehensive formulation of the problem and the development of research questions.

Given the overall goal of improving the performance of global optimization of sequences of low thrust trajectories, a literature review of current approaches was made. Since the focus of this thesis relates to the topic of low thrust trajectory estimators, the shortfalls of existing methods for low thrust trajectory estimation will be examined in more detail. Finally, the possible solutions to these shortfalls is addressed through a literature review of relevant sources.

With the insights gained by this review, a research objective will be formed that allows for the formulation of research questions and corresponding hypotheses. These research questions help to inform the construction of a methodology that will allow for the questions to be answered by experiments.

2.1 Physics

This section serves as background on the physical aspects of orbits and low thrust transfers. It introduces the terminology and some fundamental equations of orbital mechanics and low thrust transfers.

2.1.1 Assumptions

This project makes assumptions to simplify the underlying computations. The first is that there are no considerations given to perturbations by orbital bodies beyond the central grav-

itation body. In this manner, all targets will follow keplerian orbital paths. Additionally, orbit propagation in maneuvering will follow these laws as well.

2.1.2 Orbital Mechanics Nomenclature

This section introduces the relevant orbital mechanics concepts and nomenclature that will be used in this thesis. When considering the two body problem, orbits are generally described in two ways. The first is straightforward: A position and velocity vector that describes the position from the center of mass of the other body and the velocity with respect to that origin.

The second common form is known as the orbital element representation of an orbit.[18] This form represents an orbit as a collection of elements that characterize the shape and orientation of an ellipse that approximates the shape of an orbit. The six elements are:

- The semimajor axis (a), which is the average of the apoapsis and periapsis of the orbit. This describes the overall size of the orbit.
- The eccentricity (e) describes the elongation of the orbit when compared to a circle. For elliptical or circular orbits, it is a number between zero and one. An eccentricity of zero is a circular orbit while an eccentricity of one is a parabolic escape orbit.
- The inclination (i) describes the angle between an orbital plane and a reference plane. In essence, it describes the tilt of the orbit.
- The Longitude of the Ascending Node (Ω) gives the angular location of the ascending node, which is where the orbit passes upward through the reference plane.
- The Argument of Periapsis (ω) gives the angular location of the periapsis of the orbit, measured from the Longitude of the Ascending Node
- The True Anomoly (θ) describes the angular position of the body within the orbital ellipse, as measure from the Argument of Perapsis.

A table summarizing how orbital elements are represented in shown in Table 2.1.2:

r	Position Vector in inertial frame
v	Velocity Vector in inertial frame
a	Semi-major axis
e	Eccentricity
i	Inclination
Ω	Longitude of Ascending Node
ω	Argument of Periapsis
θ	True Anomaly

Figure 2.1 provides a depiction of how orbital elements describe a Keplerian element.

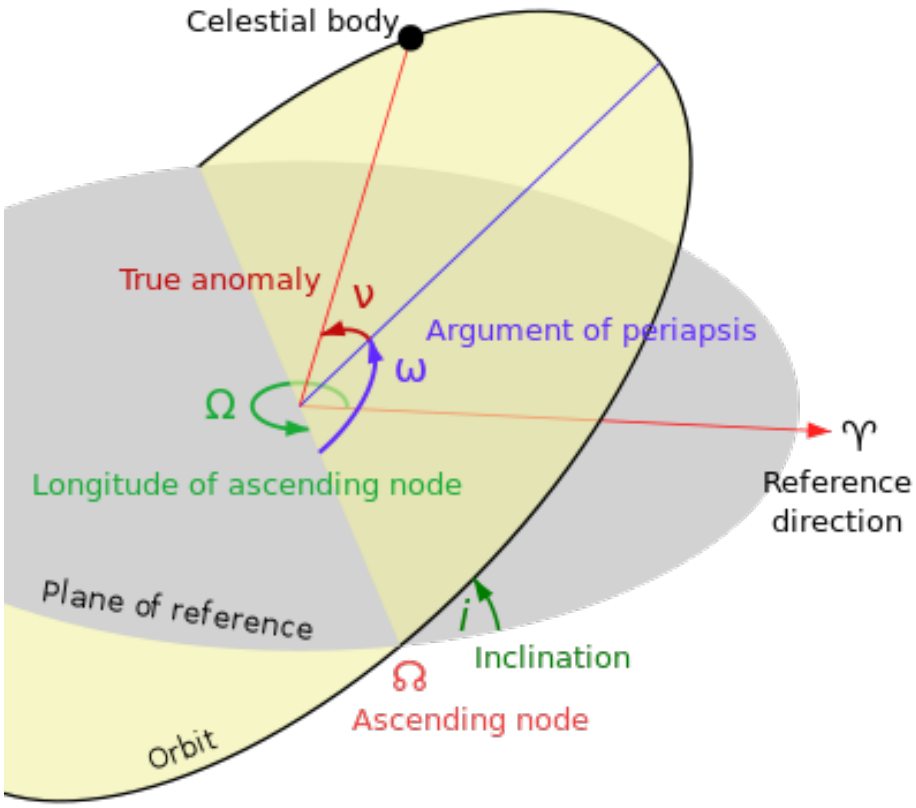

Figure 2.1: How orbital elements are used to describe a Keplerian orbit

2.1.3 Governing Equations

In general, the equation[18] that governs a body in a Keplerian orbit can be given by Equation 2.1:

$$\mathbf{a} = \frac{\mu}{r^2} \tag{2.1}$$

Where μ is the gravitational parameter of the main gravitational body and *r* is the radius. When the assumptions stated at the beginning of this section are applied to this equation, this governing equation traces out conic sections. The radius of the orbit can be described in terms of the Keplerian elements, as seen in Equation 2.2

$$\mathbf{r} = \frac{p}{1 + e \cdot cos(\theta)} \tag{2.2}$$

Where p is the semi latus rectum of the described ellipse[18] and is the True Anomaly. The position of the satellite in orbit can also be described as a function of time by Equation

$$M = \frac{2\pi t}{P} \tag{2.3}$$

Where M is the mean anomaly of the orbit, t is the time elapsed since periapsis, and P is the period of the orbit. The mean anomaly of the orbit describes the relation between the angle that would describe location of the satellite were the orbit circular instead of elliptical. The method of transforming between M and θ is known as "Kepler's Problem," and its solution method is well documented.[18] However, it is important to note that these equations show that describing Keplerian orbits is a relatively straightforward process. In this project, these are the equations of motion that describe the orbits of any bodies not undergoing thrust.

When an impulsive transfer is considered, Keplerian dynamics are applied. The change in velocity is considered instantaneous, so the matter becomes propagating the orbit forward in time after the impulse is applied. However, there is an optimization problem in

here to be considered. If two arbitrary points around a central body in space are considered, is it possible to find an orbit that contains both points? The solution to this problem is known as Lambert's Problem,[13] which gives the feasible Keplerian orbit connecting both points. Solution methods for Kepler's problem (referred to as Lambert solvers) are well known and robust. Fast, accurate solutions to this problem can be found in work by Izzo[15] and there are also many open source libraries that offer functionality for solving this problem.

When a low thrust trajectory is considered, this changes the dynamics of the orbit. Instead of Equation 2.1, the acceleration of a constantly thrusting spacecraft is described by Equation 2.4:

$$\mathbf{a} = \frac{\mu}{r^2} + \mathbf{T} \tag{2.4}$$

Where $\mathbf{T}$ is the thrust vector of the spacecraft. This simple addition of a thrust vector means the simple equations describing Keplerian orbits no longer apply. It also means that solving Lambert's Problem no longer can give accurate trajectories between points. Unfortunately, calculating these low thrust transfers turns into a problem that can only be solved numerically.[14] There are currently two approaches. The first is known as a direct method, which involves solving the physical propagation of the spacecraft under thrust. The second is known as an indirect method,[19] in which a set of complex boundary equations representing the spacecraft and its motion are solved. Using either direct methods[20] or indirect methods requires a significant increase in computation power when compared with solving Lambert's problem.

2.2 Low Thrust Trajectories

This section introduces computational methods for low thrust trajectories.

In general, due to the long stretches of low thrust (often months or years), the dynamics

of a low thrust spacecraft is much more complex than that of impulsive thrusts, which have long sections of coasting. Figure 2.2 shows an example of trajectories typical of low thrust transfers. Figure 2.2 shows a spacecraft equipped with low thrust engines transferring between three different asteroids. Blue sections of the trajectory represent coasting phases, during which the engine is off, and red sections of the trajectory represent periods of thrust. There are also significant sections where the spacecraft rendezvous with the asteroids for a period of time before initiating the next transfer. The end result is a complex, winding series of transfers between asteroids. Each section of the low thrust trajectory must be optimized, as well as other factors, such as the departure times and flight time of the spacecraft.

While impulsive thrust transfers typically are solved using Lambert solvers, there are two general approaches to solving low thrust trajectories. The first approach is known as a direct method. This approach solves the problem by directly simulating the physics. In the most commonly used direct method, known as the Sims-Flanagan method,[21] the trajectory between two targets is discretized into many nodes. Each node is described as an impulsive maneuver that approximates the change in velocity of the spacecraft over the node. Each node has a specific thrust direction. Starting from the beginning, the spacecraft is propagated forward according to the thrust vectors at each node until the spacecraft reaches the midpoint node. Similarly, the spacecraft is then propagated backwards from the end point according to the thrust vector at each node until the midpoint node. The states of the spacecraft at the midpoint node for the forward propagation and backwards propagation are then compared. If the states are identical, then a feasible low thrust trajectory has been found. If not, then the thrust vectors, mass change, or time of flight must be altered and the trajectory evaluated again. This process repeats until a feasible trajectory is found or a stop condition is met. The beginning and end of the trajectory also optimized for a set of constraints and variables, such as the propellant mass used and flight time, producing a final, optimized low thrust trajectory between targets. In this problem's case, the optimized variable would be the final spacecraft mass or a combination of the flight time and spacecraft mass. Since

this approach solves the trajectory by breaking up the trajectory into a sequence of impulsive maneuvers, it can be subject to inaccuracies when discretization is too coarse. Figure 2.3 shows a conceptual implementation of the Sims-Flanagan method. The red circles in the graphic represent impulsive burns where the spacecraft executes a single, instantaneous burn. The spacecraft's trajectory is then propagated into the next segment (represented by lines parallel to the trajectory), until the trajectory is changed by another burn. The spacecraft continues the propagation until reaching a match point, at which point the mismatch between the end and start states of the two adjacent segments is compared.

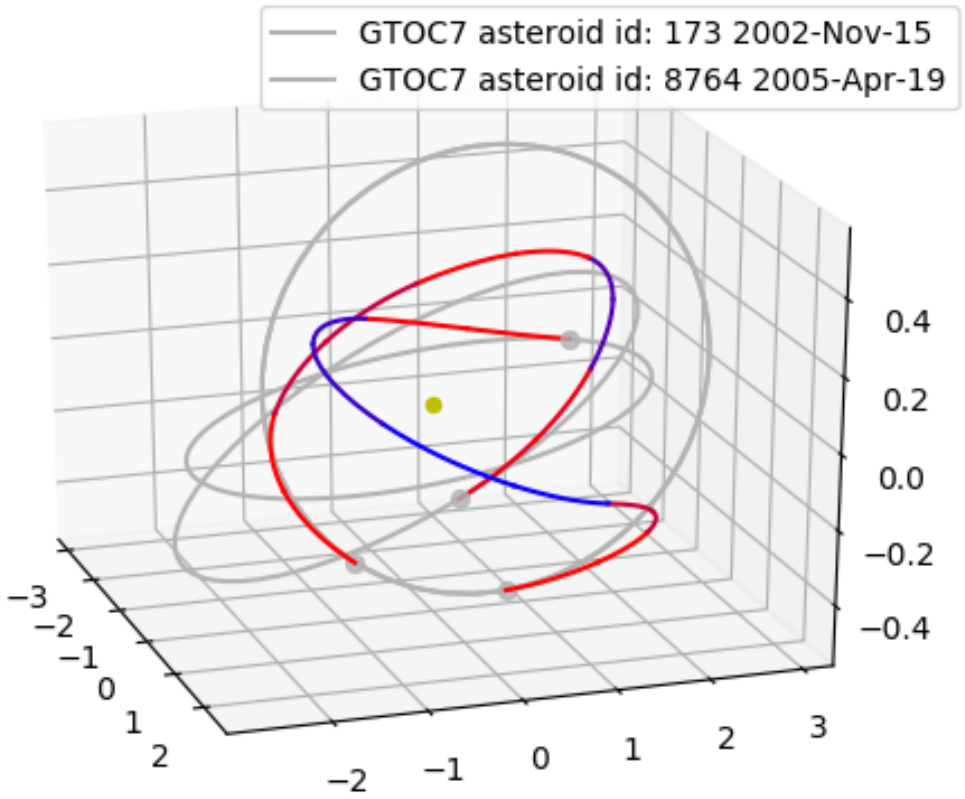

Figure 2.2: An example of low thrust trajectories between asteroids. Blue sections of the trajectory indicate a coasting phase, red sections indicate a thrust phase.

The second approach is known as an indirect method. Indirect methods do not solve the physics of the problem, but are instead calculated by the the development of a two-point

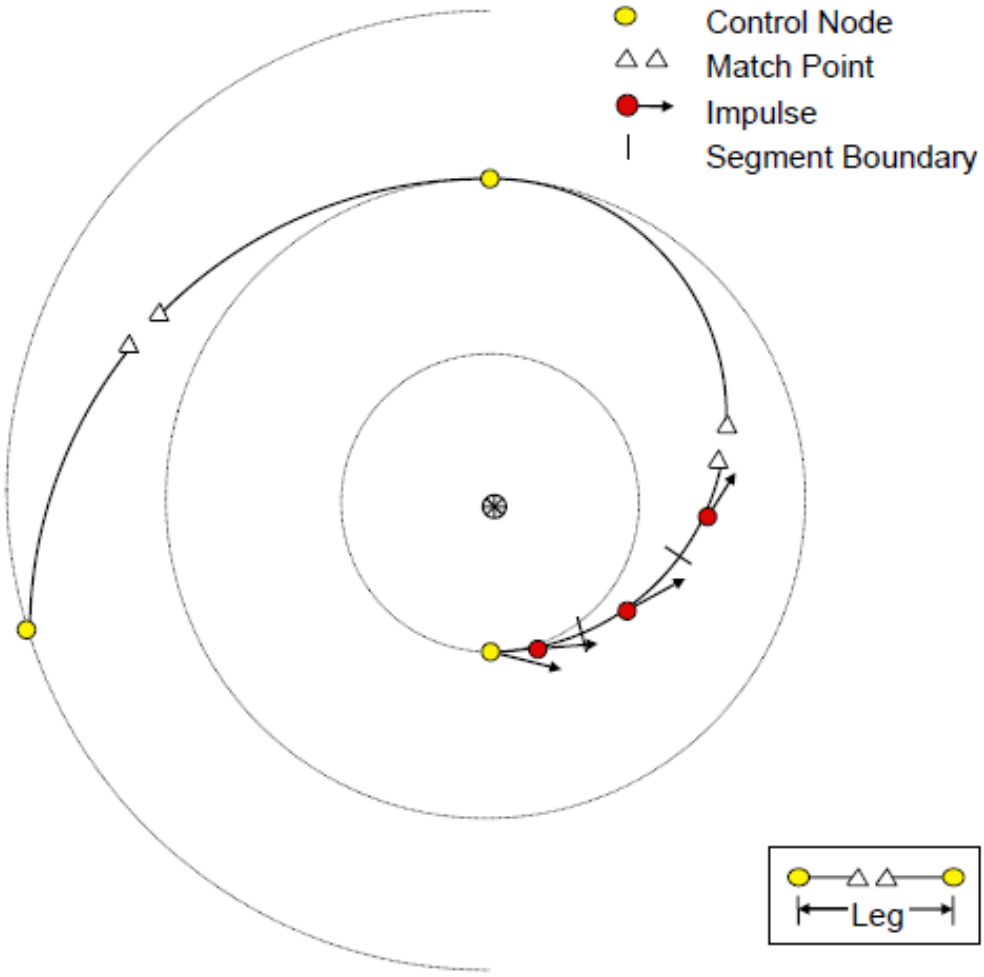

Figure 2.3: A conceptual graphic of how the Sims-Flanagan calculates a low thrust trajectory transfer

boundary value problem that is solved by optimizing for the target conditions. Though it may offer more numerically accurate results than direct methods, it can be extremely sensitive to the starting parameters and can often respond in non-intuitive ways.[20]

A common thread among both of these approaches is that they are both computationally demanding. At higher resolutions, both methods methods can have near unlimited computational load, and as such, often utilize parallel processing to solve in a reasonable time.[20] The fact that this problem is computationally expensive to solve exacerbates the issues present in solving the overall global optimization problem. Not only are there near

Table 2.1: Table showing Lambert Solver error

Spacecraft Fuel Mass	1000 kg
Spacecraft Dry Mass	500 kg
Lambert Solver Prediction MAE	462.89 kg

limitless combinations of sequences, but evaluating those sequences is very difficult to do as well. Thus, it is of interest to examine methods that quickly and accurately approximate solutions to the low thrust trajectory optimization problem.

2.2.1 Lambert Solvers in Current Global Optimization Techniques

Though Lambert solvers are considered to be poor estimators of low thrust trajectories,[22,17] they have often been used in to provide initial estimates in low thrust trajectory optimization problems. For example, in the GTOC4 competition, impulsive thrusts were used by many teams to approximate low thrust trajectories between asteroids. The top three scoring teams, Moscow State University, Aerospace Corporation, and ESA's Advanced Concepts Team, all used Lambert solvers as initial estimates for the feasibility and cost of low thrust trajectory legs.[11] However, using Lambert solvers to estimate low thrust legs has been shown to be a poor form of estimation. This presents the issue of forming poor starting estimates when selecting sequences to further analyze.

In work by Izzo, several thousand low thrust asteroid to asteroid trajectories were generated. The cost of these trajectories was then estimated using a Lambert solver, allowing for a comparison between the real solution and the estimate.[17] Table 2.1 shows the characteristics of the vehicle, along with the resulting prediction error generated by using a Lambert solver when compared to the real low thrust trajectories. Given the Lambert solver's mean absolute error (MAE) approached 50% of the total fuel of the vehicle, this demonstrates that Lambert solvers are an extremely poor indicator of low thrust trajectory fuel usage.

Even though many participants of GTOC4 used a Lambert solver to generate the initial search space of their global optimizer,[23] this poor accuracy implies, at the very least, that

using such an inaccurate estimator would result in a sub-optimal search of the solution space. In the worst case, the sub-optimal start to searching the solution space may miss far better sequence solutions. Therefore, using an estimator that offers better predictions of mass usage would allow for a better characterization of the search space in the initial stages of the global optimization problem.

Another example can be seen in the ESA's solution to the GTOC5 competition, which poses the problem of maximizing visits to asteroids with a spacecraft equipped with low thrust propulsion.[8] The ESA team's solution involved using Lambert solver solutions to evaluate sequences of asteroids to visit before evaluating them with a low thrust solver. The final report by the ESA's team specified that this approach methodology was inadequate, and, ultimately, a factor in their results:

> The assumption that a chemical trajectory can approximate a low-thrust one is very strong and, whenever possible, should be avoided. This requires the development of more and more efficient tools for low-thrust optimization, but also a correct choice for the solution strategy, i.e. one that does require only the strictly necessary amount of trajectory legs to be computed. In the case of the GTOC5 problem, this was probably a main factor to our algorithm failure in finding any n = 18 solution.[24]

The author's note that use of a Lambert solver (referred to as a chemical trajectory) should be avoided when estimating low thrust trajectories, and the development of better approximations is needed.

2.2.2 Low thrust trajectory estimation using machine learning techniques

This section provides background on the current research being conducted on developing low thrust trajectory estimators using machine learning techniques.

For trajectory optimization problems involving impulsive maneuvers, the solution is typically found by using the initial state of the spacecraft (initial position and velocity),

Table 2.2: The results of Machine Learning estimators in predicting the final mass of a spacecraft after a low thrust transfer

Predictor	MAE (Kg)	RSME (Kg)
Lambert's Predictor	31.44	42.77
Random Forest	8.64	13.88
Bagging	8.64	13.81
AdaBoost	24.6	30.4
Extra Forest	8.56	13.77
Gradient Boosting	7.86	12.48
Decision Tree	12.33	19.21
Extra Tree	13.47	20.99

the desired final orbit (final position and velocity), and a transfer time. Using a Lambert solver allows for the determination of the required departure and arrival for the transfer between two orbits - allowing for a calculation of the required reaction mass for the transfer. Unfortunately, this straightforward solution cannot be applied to low thrust trajectories. . However, the advantages of the Lambert solver, mainly it's ability to take in the initial orbit, time of flight, and desired orbit and output an optimal ΔV are characteristics that are beneficial when developing sequence generation optimization techniques. Thus, it is of interest to develop a predictor that can function similarly to a Lambert solver, but instead predict the ΔV for a low thrust transfer.

Work by Izzo[17] examined the development of predictors that are derived through machine learning techniques. This work used a data set of 100,000 low thrust trajectories between asteroids to train several different machine learning algorithms in predicting the final mass of the spacecraft after a transfer. The use of several different machine learning methods served as an overview of their accuracy. Table 2.2[17] shows the relative MAE and RSME of both the Lambert solver predictions (serving as the baseline) and various machine learning techniques. The Lambert prediction was by far the worst performing estimation, while Gradient Boosting offered the highest accuracy.

Work by Mendez[25] developed an approach for response surface regressions for low thrust trajectories. By generating a data set of low thrust transfers between Earth an Mars

with varying flight times, spacecraft properties, and orbital characteristics, a surrogate model was developed using an artificial neural network (ANN). This surrogate model was able to accurately predict the mass ratio needed for a low thrust transfer between orbits. Though the use case in this paper was constrained to an Earth Mars transfer window with a narrow departure date and flight time (varying by a few months), this work demonstrated the viability of using ANN models to predict the fuel usage required for low thrust trajectories.

In summary, work has shown that the machine learning techniques gradient boosting[17,22] and artificial neural networks offer high accuracy in predicting the fuel usage in low thrust trajectories. Additionally, the techniques for generating the training data are similar in that they involve generating a large amount of low thrust trajectories in different scenarios. The positive results in using the predictors for different flight regimes indicates that the surrogate models can be applied across a wide range of orbital transfers given enough training data. Thus, this thesis focuses on using both gradient boosting and artificial neural networks to predict the final mass of a spacecraft after a low thrust transfer. The techniques for generating low thrust trajectory data sets are also of use in this work.

2.3 Machine Learning Background

Gradient Boosting

Gradient boosting is a supervised machine learning technique that trains many weak predictors and combines them into a weighted sum that represents the overall predictor's result.[26,27] In the context of this project, these weak predictors are set of decision trees with a variable number of leaves. This model is trained in a step-wise fashion, gradually building and training more weak predictors to form a much stronger prediction model. As its name implies, the gradient of the objective function (an error function in this case) is used by the algorithm to inform the next iteration of training.

In the literature search, gradient boosting was found to be an effective technique at

building prediction models for low thrust trajectories.

Artificial Neural Networks

Artificial Neural Networks (ANN) are a family of nonlinear regressions and classifiers that are loosely inspired by biological neural networks.[28] In this machine learning technique, the ANN is modelled as having a network of artificial neurons. Each neuron can have many connections to other neurons in the network. As information passes from one connection to the next, each neuron decides whether or not to activate and continue passing the information to connected neurons. Figure 2.4 shows a conceptual overview of the general structure of Neural Networks.[29] As more data is trained on this, the pathways of the neurons are changed and weighted differently until the ANN produces the desired results. Given the highly nonlinear nature of solving low thrust transfers, ANN have been shown to offer high accuracy results in estimating the cost of transfers.[22,25]

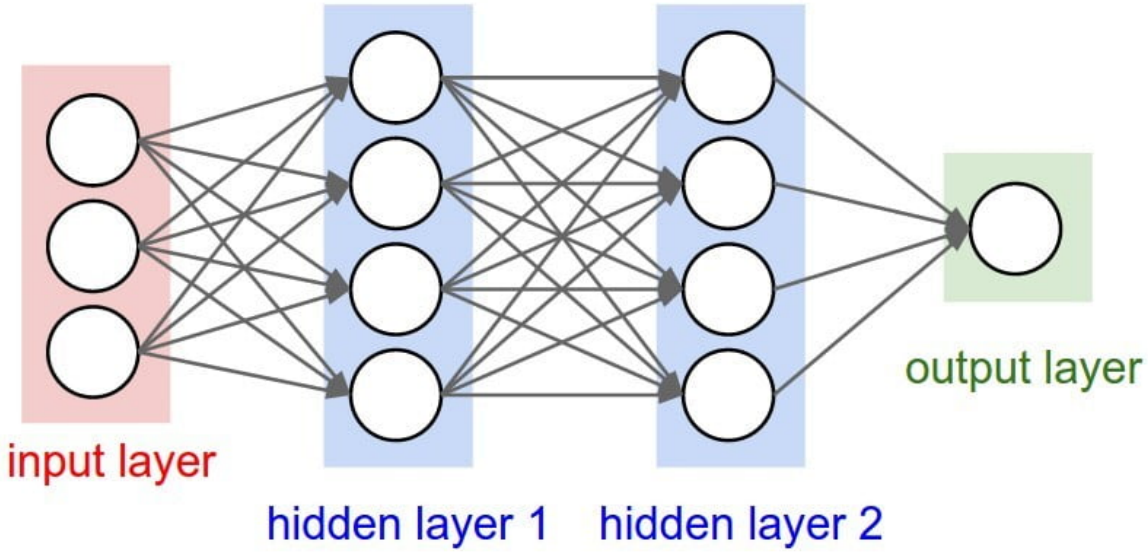

Figure 2.4: A conceptual overview of the structure of an Artificial Neural Network

Classification

As only a portion of the trajectories that will be evaluated by the physics solver will result in feasible trajectories, it is of interest to use some sort of prediction model that predicts whether or not a given set of inputs will result in a feasible solution being found by the trajectory solver. Machine learning classification techniques allow this type of prediction.[30] Therefore, machine learning classifiers will be implemented to predict the feasibility of a potential trajectory. Since the training data that results in infeasible trajectories is not used to train the regression models, this data can be used to train a classification technique. As both gradient boosting and artificial neural networks can offer classification schemes, these will be used to form the classification model. This will allow for a prediction of whether or not a proposed trajectory is likely to be feasible.

2.4 Global Optimization Techniques for Sequential Low Thrust Trajectory Optimization

This section provides an introduction to the outer loops that are currently used to solve low thrust sequential trajectory optimization. Since GTOC competitions include similar problems and are participated in by leaders of the field, the results of these competitions offer an excellent insight into the leading edge solutions of this class of problem.

As an example, the objective of the 4th Global Trajectory Optimization Competition was to maximize the number of asteroid flybys (among a catalog of thousands) and rendezvous with a previously visited asteroid at the conclusion of the sequence. Given the amount of participating teams (25), a wide variety of optimization algorithms were used.[11] These can generally be classified into three broad categories: Evolutionary, greedy,[31] and branch and prune algorithms.[32]

In evolutionary algorithms,[12,33] sequences are generated, evaluated, and then iterated upon in a process that mimics biological evolution. The benefits of these types of algo-

rithms is that they generally allow the optimizer to find sequences where there may be a high up front cost that leads to better overall results. However, these types of algorithms tend to explore a large search space when a very small area is feasible, meaning they can be inconsistent in finding near optimum sequences.

Greedy algorithms perform the optimization by forming a sequence one target at a time, finding the path of least resistance. After each step, it finds the lowest cost next step, and repeats this until exit conditions are met (typically time or fuel constraints). These algorithms have an advantage in the simplicity of operation and in that they generally find an acceptable, if not near optimum, sequence. However, due to the one step ahead nature of the algorithm, they have a tendency to get stuck in local valleys and may miss sequences that have high up front costs that open up into better overall sequences.

Branch and prune algorithms are another type that perform well. They are similar to greedy algorithms in that they generally construct sequences as they go along, but they operate by splitting the search space up into smaller chunks known as ”branches.” These branches are essentially partially constructed sequences that are evaluated in parallel, and then ”pruned” by selecting the best performer. This process is repeated by branching from the best performer. This has the benefit of immediately finding well performing areas in the search space, while also having some amount of planning and foresight into the optimization process. In this manner, it has the advantages of both evolutionary and greedy algorithms, while minimizing the negative aspects of it. Indeed, the top scoring team in GTOC4, Moscow State University, used a branching and pruning method to find the winning sequence.

Though branching and pruning algorithms have typically offered the best performance, the objective of this thesis is to neither find nor develop the best performing algorithm. Instead, it is to evaluate the effects of using accurate low thrust trajectory predictions on the overall sequence optimization. Therefore, instead of selecting the best performing algorithm, this thesis aims to use the algorithm that can best inform on this subject. To this

end, this thesis proposes the use of genetic algorithm in this endeavor.

The use of a genetic algorithm allows this study to draw conclusions on multiple fronts. Since the primary objective is to evaluate the effect of a machine learning derived prediction model integrated into the inner loop on the outer loop optimization, the genetic algorithm will fulfill this goal. However, a secondary objective is to evaluate the effects of errors that the use of predictions will have on solutions in this field. Since, at its core, the optimization process in this area is sequence generation and evaluation, the propagation of errors in the evaluation of these sequences may have a large effect on the final score of a sequence. Since a genetic algorithm evaluates full sequences, as opposed to single steps in greedy algorithms or partial sequences in branching and pruning methods, the effects of errors can be more thoroughly examined in genetic algorithms. As these error propagation results are analyzed, conclusions about other algorithms may be inferred as well.

2.5 Research Objectives

Thus far, a gap in the current approach to solving this low thrust trajectory sequence optimization problem has been identified. This gap is that current methods use a Lambert solver to provide initial estimates of low thrust trajectories. However, the poor performance of Lambert solvers in approximating low thrust trajectories means that the outer loop may incorrectly select sequences that are not really good low thrust trajectories.

This leads into the overarching research objective of this thesis, which is defined as follows:

Research Objective: Develop an approach for improved sequential low thrust trajectory optimization through improving fuel cost estimations of low thrust trajectories.

Given the identified gap, an important question to ask is if a viable estimator for low thrust trajectories can be developed. Recent work has shown that machine learning is a viable technique to creating estimates that predict the fuel cost of low thrust trajectory solutions.[17,22,12,25] The resulting predictors offer an order of magnitude increase in accuracy over Lambert solvers while maintaining similar computation speeds. While such machine

learning techniques derived estimators have been developed, there has not been extensive work done in pairing such a prediction model with a traditional global optimizer.

This leads into the main research question of this thesis:

Research Question 1: Can machine learning techniques be utilized to predict the cost of low thrust trajectories such that more optimal trajectory sequences are found?

The following hypothesis is therefore defined as:

Hypothesis 1: An accurate fuel cost predictor for low thrust trajectories developed through machine learning will allow for better sequences to be found.

As the use of Lambert solvers in low thrust trajectory optimization problems has been shown to be unreliable in identifying favorable trajectories,[24] using better estimators should allow for an improvement in Outer Loop global optimization results. The use of higher accuracy estimators is advantageous for two reasons: The first is that they are significantly faster at estimating the costs of low thrust trajectories than using a physics solver. The second is that they are much more accurate than Lambert solvers. This combination of speed and accuracy allows for a more expansive and more accurate exploration of the search space than would be achievable with an optimization scheme using only a Lambert solver or a physics solver integrated into the inner loop.

Though machine learning has been shown to be a viable method of creating surrogates for low thrust trajectories, there is still the question of the best machine learning technique to be used in this global optimization scheme. Therefore, it is of interest to use the best prediction model possible for the problem. This leads to the additional research question:

Research Question 2: What is the best machine learning technique for building a low thrust trajectory prediction model?

In the literature examined in Section 2.2.2, gradient boosting offers the highest accuracy predictions.[17,22] Therefore, it is expected that this trend will hold. In order to test this hypothesis, gradient boosting predictors will be compared to artificial neural network predictors, which have also been shown to be fast and accurate predictors.[22,25] This forms the second hypothesis:

Hypothesis 2: Gradient boosting will offer a more accurate prediction model than artificial neural networks.

When considering the fact that all estimates will have some level of error, it is also of interest to investigate the effect that this error will have on the overall results of the sequence evaluation. Since the inner loop portion of the optimization problem deals with evaluating entire sequences of targets to visit, any error in calculating the costs associated with the first transfer will necessarily be introduced to the next item in the sequence. Since the inner loop must also optimizes for transfer timing, it must make predictions on two levels. The first level is the minimum fuel cost for a transfer. The second prediction is of the optimal timing (departure date and time of flight) of a transfer. Inaccuracies in either will have an impact on the scoring of sequences by the inner loop. Therefore, another question is considered:

Research Question 3: How sensitive are the global optimization results to the accuracy of the prediction models?

Given the ESA's results in GTOC5,[24] it is expected that increasing the accuracy of cost estimates for low thrust trajectories will also allow the global optimizer to find more optimal sequences of targets to visit. Therefore, the global optimization results should be highly sensitive to the accuracy of the prediction models. This leads into Hypothesis 3:

Hypothesis 3: The global optimization results are sensitive to the accuracy of prediction models.

Comparing the global optimization results with predictors of differing accuracy would allow for a confirmation or refutation of this hypothesis. Therefore, the global optimizer will be integrated with three different inner loops. The first inner loop version will have a Lambert solver based estimator. Since Lambert solver estimates is a common method of preliminary low thrust trajectory design, this will serve as the baseline that all other results will be compared against. The other two inner loop versions will have the trained prediction models serving as low thrust trajectory estimators. Additionally, the prediction models will

be trained with varying amounts of data to examine the effects that less accurate prediction models have on the global optimization results.

Thus far this work has introduced background, the problem, and the current methods. After research into current literature, a gap was identified in current methodologies. Since low thrust trajectory optimizers tend to be very computationally expensive to execute, it is of interest to use fast, accurate estimators to speed up the global optimization. Current methods have typically solved Lambert's problem to approximate low thrust trajectories, but this has been shown to be a poor estimator. Fortunately, recent work has shown that machine learning techniques can be used to develop fast and accurate predictions. However, integration of these prediction models into traditional global optimization algorithms has not yet been investigated. Thus, the goal of this thesis is to develop and examine the efficacy of such an approach. A genetic algorithm has been chosen to perform the global optimization, as it may allow for some inference on the effects of low thrust trajectory prediction models on other global optimization algorithms.

Now that the background and goal of this thesis has been firmly established, the technical approach to solving this problem must be considered. Specifically, the integration between the outer loop of the global optimizer, the inner loop of the physics solver, and the training and implementation of the machine learning techniques.

CHAPTER 3
TECHNICAL APPROACH

The goal of this work is to develop and examine the feasibility an approach to combine two established techniques: Evolutionary algorithms in low thrust orbital sequence generation and low thrust trajectory estimators trained utilizing machine learning. This examination will be performed by developing software that integrates machine learning into an established optimization scheme. Results will then be compared between the scheme with and without the prediction model.

The previous chapters introduced the background of the problem and the research questions that this thesis addresses. A more detailed examination of the questions and current methods will allow for the formulation of the problem and the technical approach. The research questions are as follows:

1. Can machine learning techniques be utilized to estimate low thrust trajectories such that more optimal trajectory sequence can be found?

2. What is the best machine learning technique for building a low thrust trajectory prediction models?

3. How sensitive are the global optimization results to the accuracy of the prediction models?

The first question relates to the overall thesis and is characterized by the shortcomings of traditional methods. Examining this question allows for the identification of the gaps in current methods to be exposed. As has been established in previous sections, the main gap in current methods is the lack of an adequate predictor to perform fast, accurate cost estimates for low thrust trajectories. Thus, a methodology for performing fast, accurate low thrust trajectory cost estimates is needed. This project solves this gap through the

application of machine learning techniques to quickly and accurately estimate the costs of low thrust trajectories.

The second question deals with the selection of an appropriate machine learning technique. Given the non-linearity of the problem, simple estimators (such as Lambert Solver's) do not offer adequate accuracy. As they have been shown to be more accurate predictors than Lambert solvers machine learning algorithms will be used. Gradient boosting and artificial neural networks are two methods that have performed well for this particular purpose, and so both of these methods will be used. A comparison will be made between the levels of accuracy of the two prediction models, as well as their effects on the global optimization. Since all estimators have some level of error, the effect of these errors on sequence evaluation in the inner loop, as well as the overall optimization scheme should be examined. This examination will help answer the third research question.

A successful implementation of a low thrust trajectory cost estimator into an outer loop optimizer has broader implications on the field of trajectory optimization. In addition to quantitatively examining the effect on the chosen global optimizer, answering these outlined questions also helps to qualitatively examine the potential effects that machine learning prediction model implementation may have on other schemes. The issue pertaining to poor predictions in the inner loop of this problem is not exclusive to any single global optimization algorithm, but is instead a shared trait. Thus, the optimization algorithm selected for this thesis is intended not to give insight into the larger problem of sequential low thrust trajectory optimization.

Through a modelling and simulation environment, this work addresses the posed research questions by performing various experiments. The environment accurately reproduces both impulsive and low thrust maneuvers, as well as the movement of celestial objects according to Keplerian dynamics. This modelling environment can be used to evaluate sequences of objects to visit, according to the low thrust solvers.

3.1 Software

All software development was in the Python programming language and used additional package libraries. Such libraries include, but are not limited to, SciPy,[34] NumPy,[35] PyKep,[36] PyGMO,[37] and scikit-learn.[38]

PyKEP is an open source Python library developed by the ESA Advanced Concepts division that offers a wide range of astrodynamics and spacecraft trajectory functions.[37] For many computational aspects of astrodynamics (including low thrust trajectory solutions), PyKEP functions boosted in C++, allowing for a significant speedup of calculations over native Python code. This is the main astrodynamics library used during this project.

PyGMO is an open source Python library also developed by the Advanced Concepts Division at the European Space Agency. It serves as a multipurpose large scale optimization library.[37] As it was developed by the same group, it is well suited to solve PyKEP trajectory optimization. PyGMO has several core classes that are used in optimization. These include, but are not limited to, the Algorithm class, the Problem class, and the Population class.

SciPy is an open source Python library used for scientific and technical computing. It includes many optimization modules, such as the Differential_evolution module, which was use as the genetic algorithm for this project.

Scikit-learn is an open source Python library that is used for machine learning applications. The gradient boosting regression, gradient boosting classifier, artificial neural network regression, and artificial neural network classifier were all used to create the machine learning prediction models for this project.

3.2 Evolutionary Algorithms in Sequence Optimization

Evolutionary techniques have been among the most successful techniques for finding near-optimal low thrust trajectory sequences. Though there are many evolutionary algorithms that may perform better, a genetic algorithm is used due to the fact that this work is intended

to serve as a proof of concept for this methodology. Many algorithms in this field (such as ant colony optimization,[39] simulated annealing, or branch and prune methods[32]) require analysis of partial or entire sequences of targets. Using an algorithm that evaluates full sequences of low thrust trajectories serves as a worst case scenario in terms of error propagation in sequence evaluation. Therefore, it is of interest to use an algorithm that evaluates full sequences, rather than only partial. For example, if a genetic algorithm paired with the machine learning predictor based inner loop shows significant improvement, then it is very likely that a branch and prune method will also show significant improvement, as branch and prune methods only evaluate partial sequences before checking with physics solvers. Since a genetic algorithm optimizer runs full sequences, using a genetic algorithm with the trained predictors allows for inference of the feasibility of other similar methods.

A genetic algorithm is intended to mimic the process of evolution by natural selection.[40] Figure 3.1 shows a flow chart of how a genetic algorithm functions. First, a parent population of initial solutions is generated. In the case of this problem, each individual will be a full sequence of target bodies. Next, every individual in the parent population has its fitness evaluated by some metric. For this problem, fitness will be the amount of targets that the space vehicle can visit given fuel and time constraints. For trajectory sequences, this fitness evaluation is calculated by the inner loop of physics solvers that computes the optimal trajectory between each successive target. Next, the top performing solutions in that set are used to create the next generation of solutions by breeding and mutation. In breeding, partial sequences will be swapped between individuals to generate children. This children will also have some probability of having their sequences randomly mutated. With the new child population, each solution has its fitness evaluated and the process of breeding repeats. This cycle continues until some criteria is met, either through performance evaluation or a set number of iterations.

A main concern and area of examination in this project is how the inaccuracy of a prediction model will influence the end result. As a sequence is analyzed, the Inner Loop

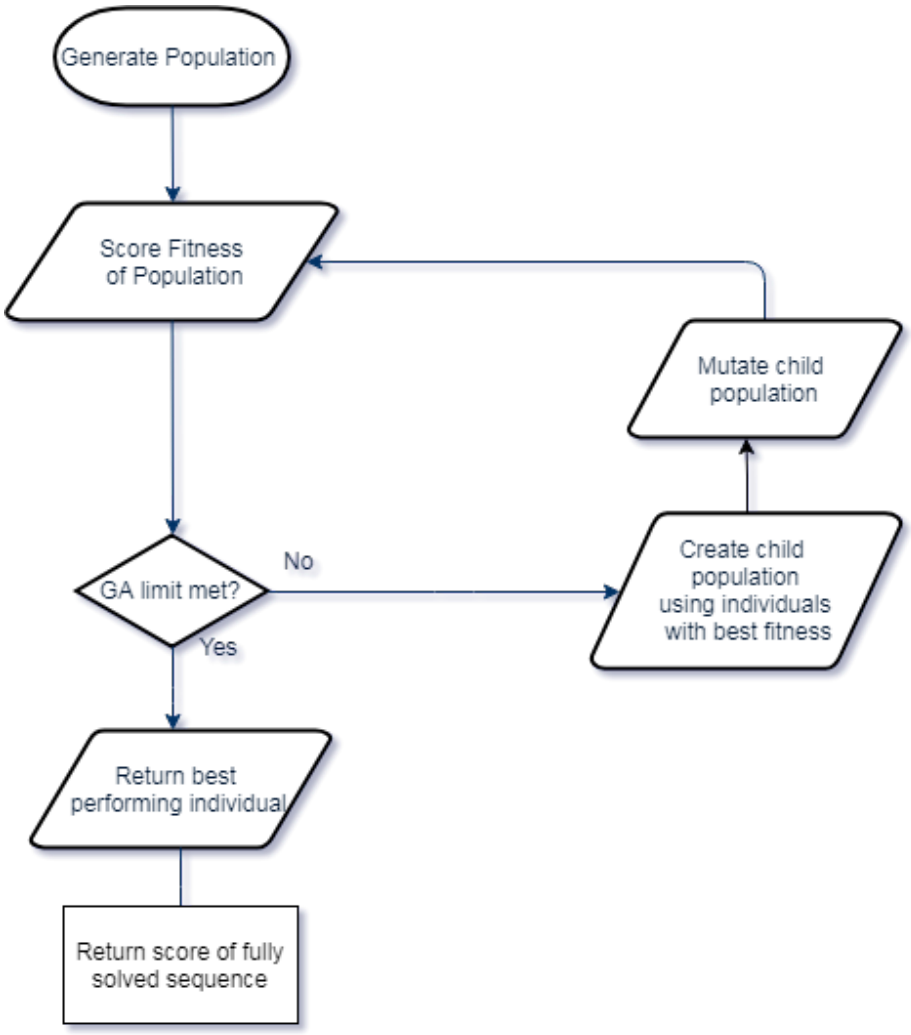

Figure 3.1: Flow chart of a genetic algorithm

finds the optimal trajectory between each target, in order. If a prediction model is used, there is some inherent loss of accuracy. As an entire sequence is evaluated by the prediction model, this runs the risk of errors compounding over the sequence to significantly reduce the efficacy of the optimizer. The sensitivity of the Outer Loop Optimization results to this error propagation will be examined.

3.3 Fitness Evaluation and Inner Loop

The inner loop serves as the optimizer within the fitness evaluation. It deals with handling the physics of the the problem and finding optimal trajectories and transfer timings between each target orbit. Generally speaking, the fitness of a sequence is evaluated by how many targets it can visit before running out of fuel. However, calculating this can be a complex task on its own. As an example, an inner loop sequence could be the visitation of a set of asteroids called A,B, and C. A potential sequence could be as C-A-B, that is, the spacecraft visits asteroid C in a flyby, then asteroid A, and asteroid B. In order to evaluate this sequence, an orbital transfer from C to A must be calculated. This includes finding the best transfer timing (departure and arrival) and the trajectory that offers the least fuel consumption. After a solution is found for that, the end state of the flyby is used as the initial conditions for the next transfer. Then a transfer from A to B is calculated by repeating the same process. This process is repeated for all objects in the sequence until the sequence is finished or the some exit condition is met. The final output of the inner loop is the number of objects visited, the total time, and the total fuel used.

Figure 3.2 shows a flow chart breakdown of the inner loop. As detailed above, a sequence of targets is fed into the inner loop. The cost and feasibility of the first transfer is calculated. If no constraint is violated, the next target is considered. This process continues until a constraint is violated or the sequence is completed. The score of the sequence is then returned to the outer loop algorithm.

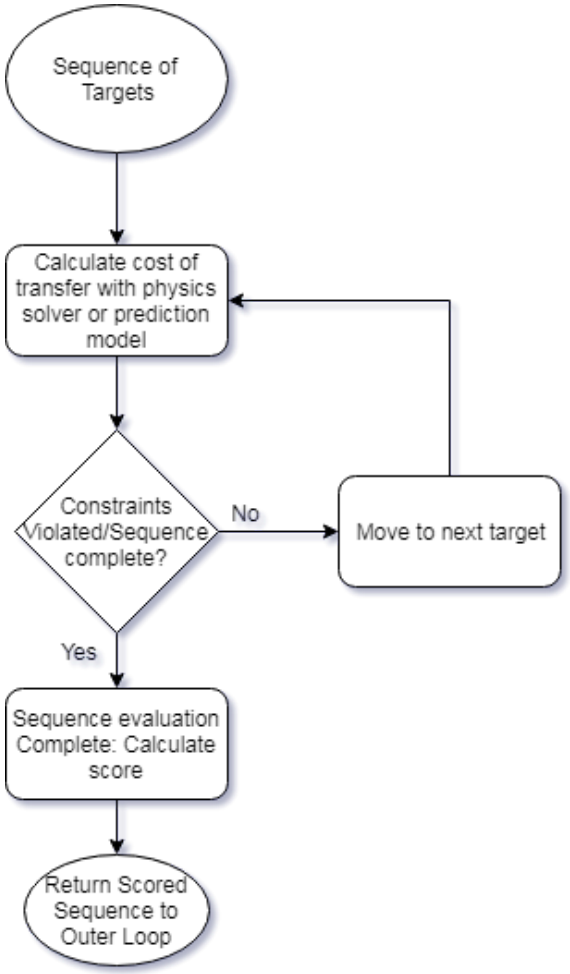

Figure 3.2: Flow chart of the inner loop

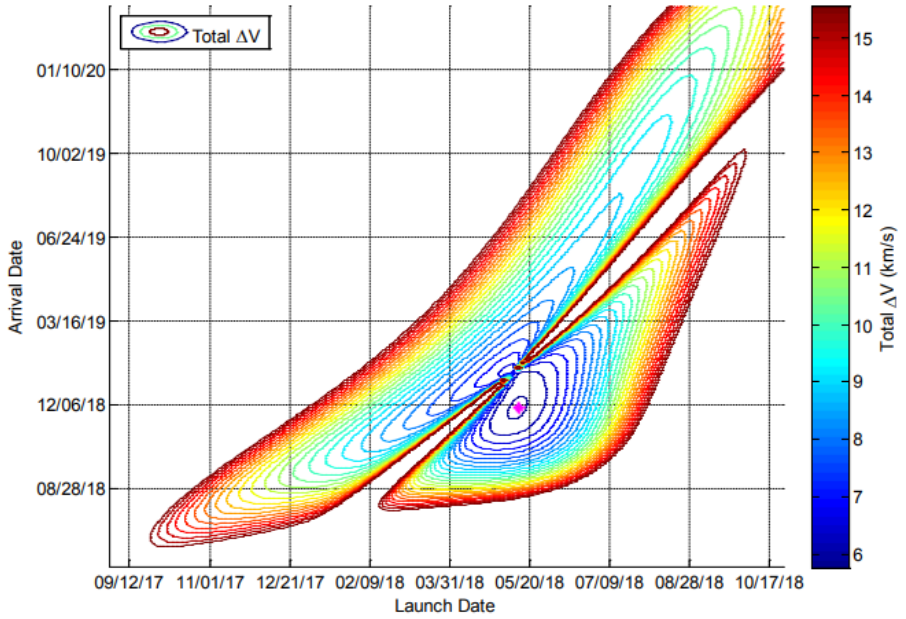

Figure 3.3: An example of an impulsive transfer pork chop plot between Earth and Mars[43]

3.3.1 Finding Optimal Transfer Windows During Inner Loop Evaluation

When planning an impulsive transfer to a target in a different orbit, the fuel cost of the transfer changes with time. During the inner loop process, finding the best time to initiate the transfer and arrive at the target is an extremely important aspect to evaluating the sequence correctly. When dealing with impulsive transfers, these opportunities can be represented by something called a pork-chop plot, which is a graph with the departure and arrival dates on the x and y axis, and the cost of each transfer as a surface in the graph.[41] An example of an Earth-Mars transfer pork chop plot can be seen in Figure 3.3. This is a common method of finding transfer windows for planetary missions.[42] Small areas of low cost transfers can be visualized on this pork chop-plot. Since the physics that creates this plot is nearly impossible to easily predict, optimization methods are employed to find the best transfer windows. One such method is a simple grid search, where the cost is evaluated at dozens or hundreds of different points arranged in a grid of departure dates and arrival dates, and the best of those is selected as the transfer window.

Unfortunately, generating low thrust trajectory pork-chop plots has been shown to be a

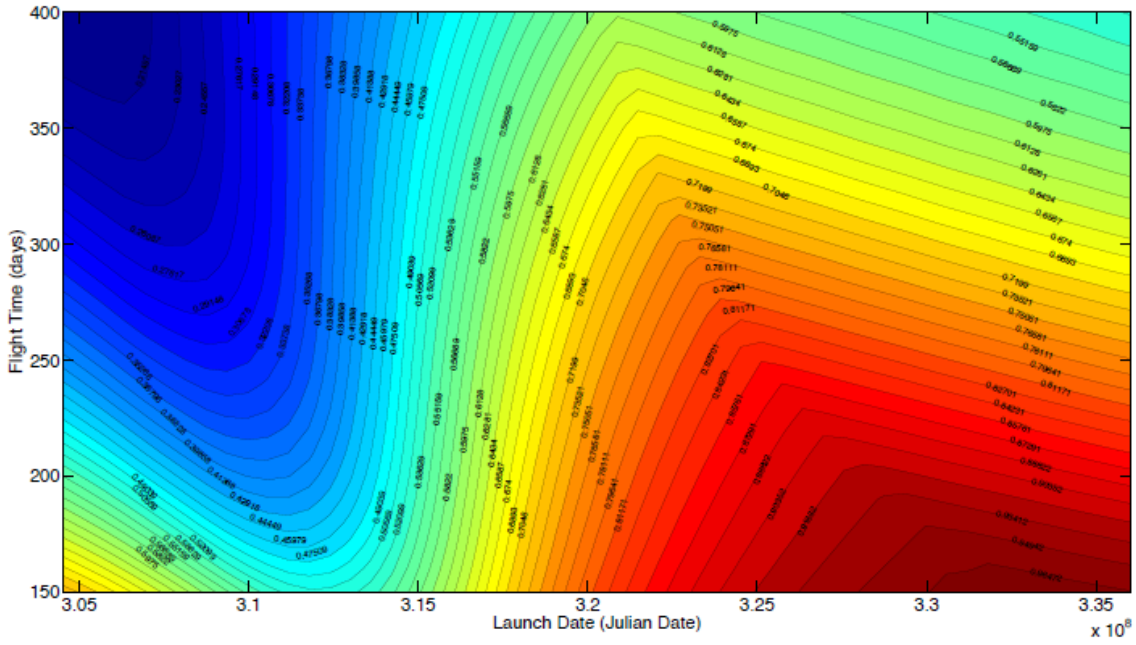

Figure 3.4: Low thrust plot that shows the optimal transfers window

difficult problem due to the highly nonlinear and varied shape of the trajectory, as well as the computational cost of computing dozens or hundreds of different trajectories. However, when an accurate low thrust trajectory cost prediction method is used, it is possible to quickly construct a low thrust analogue to these pork-chop plots. This approach allows for an estimation of not only the lowest cost transfer, but also the departure date and time of flight of the optimal transfer. Figure 3.4[44] shows an example of a low thrust trajectory pork chop plot developed by Patel used for finding optimal transfer timing. This plot shows a contour plot of the cost of an Earth to Mars transfer over a variety of departure and arrival dates. Using a grid search like this or can be applied to this data to find the low thrust transfer departure date and arrival time that minimizes fuel cost.

3.4 Application of Machine Learning Techniques in Inner Loop Evaluation

This section describes the problems associated with evaluating the inner loop and why machine learning can be beneficial to this particular problem. As previously mentioned, the inner loop of the problem deals with the physics inherent in the problem and evaluates the

cost of any particular sequence. Extended discussion on methods of solving the underlying physics (low thrust trajectory optimization methods) can be found in works by Sims and Finlayson,[21] Sims and Flanagan,[20] and Patel.[44]

The method of low thrust trajectory optimization to be used in this project is a direct method described by Sims and Flanagan.[21] As a direct method, the Sims-Flanagan approach offers better robustness in solving low thrust trajectories than indirect methods, as well as less sensitivity to starting conditions.[45] This method solves the low thrust trajectory problem by dividing a trajectory into many discrete segments in which an impulsive maneuver is performed and propagated in each.[20] This leverages the ability to average the rate of changing orbital elements, but at high fidelity it can be computationally expensive. The Python library PyKep[36] offers a robust module for performing these calculations, and is used in the simulation environment.[37]

In this work, to offset the high cost of performing these calculations, an estimator is developed using machine learning to provide accurate estimates of low thrust trajectory fuel usage. In works by Izzo, Gradient Boosting has been shown to be a high performing algorithm for estimating the final mass of the spacecraft after a transfer.[22,17] Additionally, artificial neural networks have been show to be effective estimators of low thrust trajectories.[25,22] As such, both will be used to provide fast and accurate estimates in place of the Sims-Flanagan physics solver. Additionally, both of these machine learning techniques can be trained as classifiers.[30] A classifier will be trained to predict the feasibility of a trajectory to serve as an additional criteria when evaluating the inner loop.

The attributes used for training of the machine learning algorithm to calculate the final mass are shown in Table 3.1:

Table 3.1: Machine Learning Attributes

Attribute	Explanation
ΔT	Transfer Time
m_i	Initial Spacecraft mass
$cos(\Delta i)$	Cosine of difference in inclination of two orbits
$\Delta\Omega$	Difference between the RAAN of the orbits
$\Delta\omega$	Difference in arguments of periapse
$\Delta\theta$	Difference in true anomaly
$\mid \Delta a \mid$	Difference Between the semimajor axis of the orbits
$\mid \Delta e \mid$	Difference Between the eccentricities axis of the orbits
$m^\star$	Maximum initial mass

The maximum initial mass is defined by Izzo as a maximum initial mass approximation.[17] This serves as an estimate for the required starting value of the initial mass in the trajectory leg optimization process. It is given by Equation 3.1:

$$m^\star = 2\frac{T_{max}}{a_D}\left(1 + exp(\frac{-a_D\Delta T}{I_s p g_0})\right)^{-1} \tag{3.1}$$

Where a_D is the average acceleration, $I_s p$ is the specific impulse, ΔT is the transfer time, and T_{max} is the maximum thrust of the engine.

3.5 Machine Learning Integration into Global Optimizer

In order to maximize the effectiveness of machine learning in this framework, there are several things to be considered. The first is how the training and validation data for the estimator is generated. The second is how the estimator is integrated into the overall optimization scheme.

3.5.1 Training Data

This project uses the estimation technique developed by Izzo and Mereta.[17,22] In this technique, a large number of low thrust transfers (50,000) between asteroids are simulated and the results saved to a database. Using features such as the transfer time, starting mass, maximum initial mass, and orbital characteristics of the initial and target orbits, the estimators are trained. The end result is an estimator that can output the estimated final mass of the vehicle at arrival.

In the same vein, the training data for this project is generated before the global optimization. The starting point for the training data is the generation of sets of orbital transfers. Each set of orbital transfers is defined by two sets of Keplerian Orbital elements (one for the initial orbit and one for the target orbit), a time of flight, and a spacecraft starting mass. These transfers are then evaluated by the low thrust Sims-Flanagan trajectory solver. The solution to each trajectory has two aspects: The first is the final mass of the spacecraft, which is used to train the regression estimators. The second is the feasibility of the trajectory. Since not all inputs into the solver will results in a feasible low thrust trajectory, the solver returns whether or not it found a feasible low thrust trajectory. Both of these results are saved for training. The machine learning techniques will then be trained on this data in order to generate fast, accurate estimators for use in the inner loop. The main reason for this approach (as opposed to training the estimators during the global optimization process) is due to the limits of available computational resources. Additionally, generating the set of training data before global optimization allows for an iterative optimization process in regards to developing the estimators.

3.5.2 Prediction Model Integration into Global Optimizer

This section describes the methodology of integrating the prediction models into the optimization scheme. The general idea of the this integration is relatively simple: when properly trained, the predictions should take the place of the low thrust physics solver.

This allows for a significant speedup of the evaluation process, which is a computational bottleneck in the global optimization process.

Since implementing machine learning training into the actual genetic algorithm would be unwieldy at best, data for training the prediction model will be generated before performing the global optimization, and the trained model will be tuned and saved for use in global optimizer. Figure 3.5 shows a flow chart of the implementation of the genetic algorithm to globally optimize low thrust trajectory sequences with a modified innner loop.

The genetic algorithm starts with a population of chromosomes that each represent a sequence of targets to visit. This sequence is then evaluated by an objective function that returns the score of the sequence. This objective function uses the prediction model to evaluate each transfer and calculate the estimated final mass of the spacecraft at the end of the low thrust transfer. Additionally, it uses a grid search to find the optimal departure date and time of flight for the transfer. The final mass and position of the spacecraft, as well as the ending epoch, are used for the next transfer. Furthermore, when the classification scheme is implemented, the feasibility of the transfer will be predicted, and if the trajectory is predicted to be infeasible, a constraint will be violated. This process repeats until the spacecraft runs out of fuel or a total time limit is exceeded. At this point, the objective function returns a score that is a combination of the total targets visited and the mass used. Since the overall objective is to maximize target visitation, the number of feasible visitations is heavily weighted. The score of a sequence is calculated by adding the number of targets reached to the fraction of fuel remaining on the spacecraft at the end of the sequence. Thus, if two sequences were able to visit the same number of targets, the sequence with the higher final mass (lower total fuel spent) scores higher.

In conjunction with the inner loop objective function, the genetic algorithm produces a population of sequences and evaluates them. After the scores are returned for each sequence, the best performing sequences are combined, mutated, and reevaluated using the

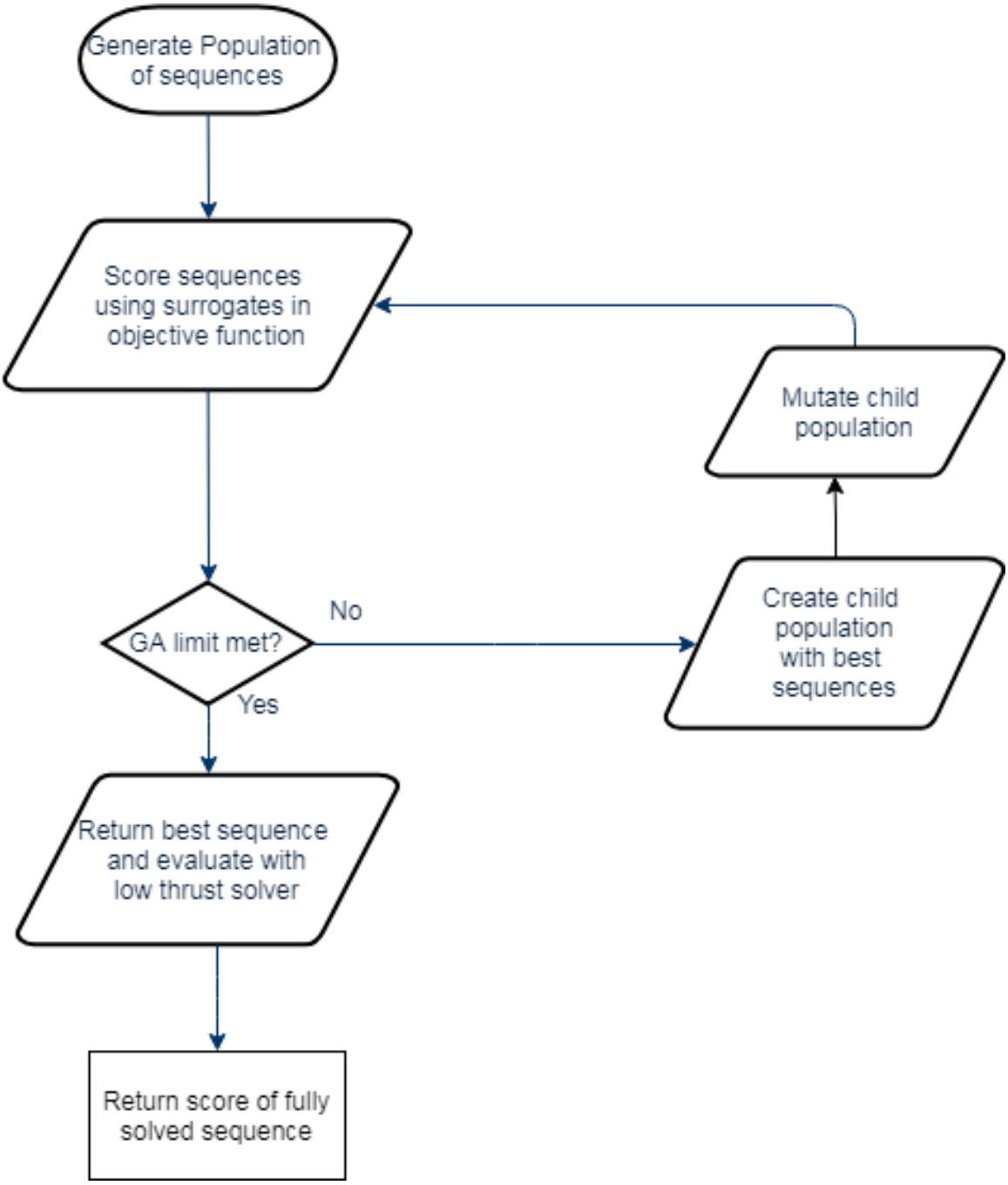

Figure 3.5: Flow chart of genetic algorithm with an integrated prediction model

objective function. This process is repeated until a stop condition (either a set number of iterations or when the score stops improving) is met.

The result of the genetic algorithm is a sequence of targets that maximizes the number of targets visited by a spacecraft according to the machine learning prediction. Once a sequence has been optimized by the genetic algorithm, it will then be evaluated by a Sims-Flanagan solver to calculate the true score of the optimized sequence.

In the case of a GTOC-class problem, this prediction-based sequence would likely be used as a seed for a genetic algorithm that uses the full low thrust solver in the objective function. However, given the somewhat limited computational resources available for this thesis, the fully evaluated sequence will be considered the end of the global optimization. Instead of further optimizing the sequence with a low thrust solver, multiple genetic algorithm optimizations using the prediction models will be performed. The results of all of these will be then evaluated by the Sims-Flanagan solver and then their scores averaged. This allows for a general score to be evaluated for each machine learning prediction model.

The process will be repeated with an inner loop that uses a Lambert Solver, rather than the trained estimators. This allows for a comparison of results between the traditional method of using Lambert Solvers and the new method of integrated prediction models.

The genetic algorithm will be performed by using the scipy.optimize differential_evolution module.[34] Using this allows for modularity when implementing the different objective functions. The default settings in differential_evolution optimizer were used.

3.5.3 Solving Low Thrust Trajectories

As mentioned, solving for any particular low thrust trajectories is a computationally involved process. This section details the technical set up and libraries used for this thesis.

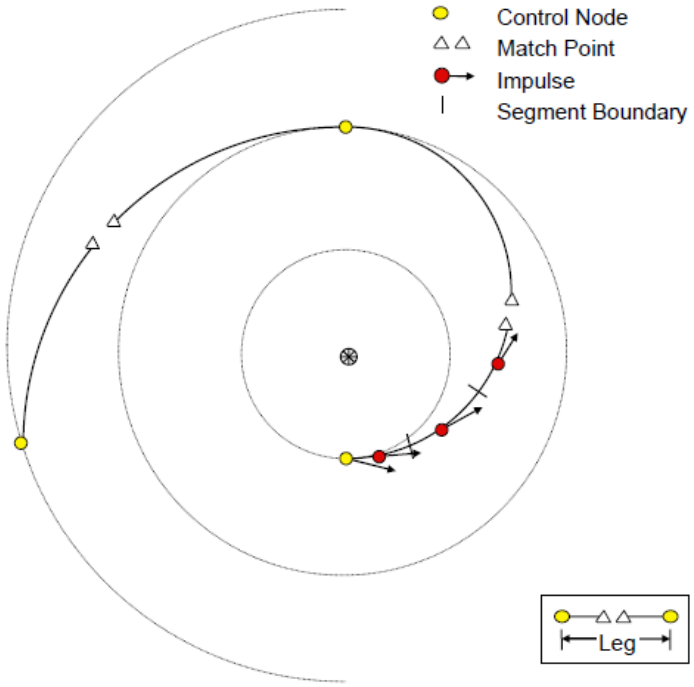

Figure 3.6: A conceptual visualization of the Sims-Flanagan Method[20]

Creating Low thrust Trajectories Using PyKEP

PyKEP's Sims-Flanagan module was used for the generation of low thrust trajectories in this work. The set up for this functionality will be briefly described. Figure 3.6 gives a conceptual visualization of the Sims-Flanagan method.

First, a PyKEP Spacecraft object is instantiated. This object is instantiated with characteristics that define the mass, isp, and thrust of the vehicle. Next, the end and start states of the spacecraft are decided. This includes the initial and ending position, velocity, and mass of the spacecraft. These state are then used to instantiate a PyKEP Sims-Flanagan leg, a class that is used to describe Sims Flanagan legs. In order to properly instantiate a Sims-Flanagan leg, the user must input the throttles of the spacecraft over the leg at each decision node. Each node along the trajectory has 3 throttles (one in each Cartesian direction), so, for example, if leg was set at 10 segments, there would be a throttle vector length of 30. In Figure 3.6, each thrust vector is represented as a red dot with an arrow coming

out of it.

After the Sims-Flanagan Leg is properly set up, it takes the initial spacecraft state and propagates it forward according to the inputted throttles and gravitational accelerations. At the same time, the end state of the spacecraft is propagated backwards according to the inputted throttles and gravitational acceleration. After the first half of the leg is propagated forward, and the last half is propagated backwards, the feasibility of the entire leg is checked. If the endpoints of the two propagation (backwards and forwards) do not match position and velocity, this particular leg is not feasible (this is seen at the white triangles in Figure 3.6). The Sims-Flanagan object in PyKEP then returns the mismatched states of the leg.

It should be noted that PyKEP's Sims-Flanagan class does not actually find a feasible leg; instead it determines whether or not a particular set of spacecraft start, end, and throttle states produces a viable trajectory. In order to generate feasible trajectories, the class must be used in conjunction with a separate optimization method that iterates upon different inputs to find feasible trajectories. In this sense, the PyKEP component of the trajectory solver serves as the objective function.

Figure 3.7 shows a flowchart breakdown of the low thrust trajectory solver. It receives inputs for a starting orbit, target orbit, transfer time, and returns the mismatch between two Sims-Flanagan Legs. The mismatch state is then used to determine if a given trajectory is feasible or not. If not, the trajectory chromosome is iterated upon. This entire process is repeated several times using a monotonic basin hopping algorithm to help avoid local minimums.[46] This procedure produces the end chromosome, which ideally will be a feasible, optimized trajectory.

Solving Low Thrust Trajectories Using PyGMO

The first step in solving a low thrust trajectory with a PyKEP and PyGMO pairing is instantiating a PyGMO Problem class. This Problem object functions as the objective function

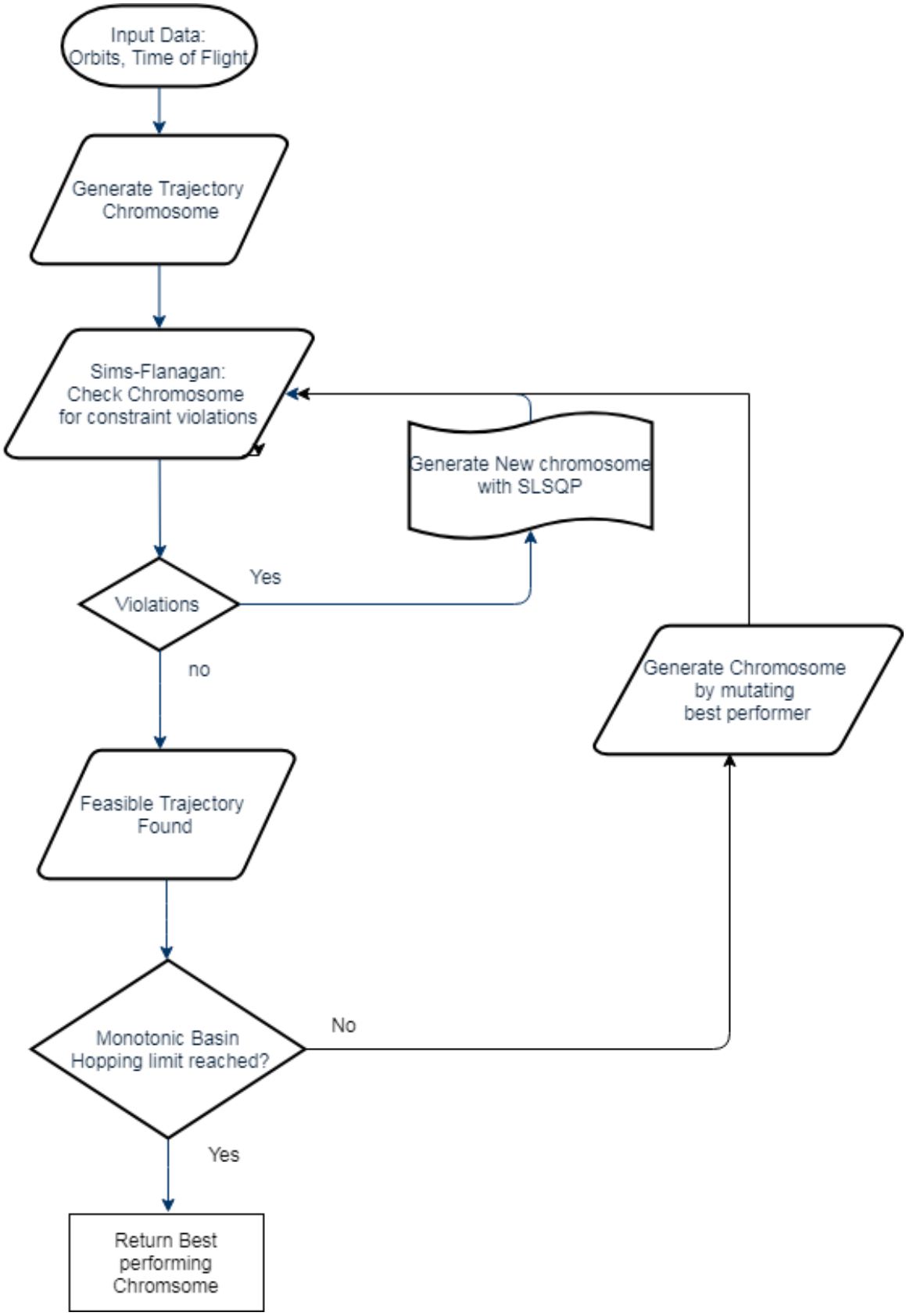

Figure 3.7: Diagram of a low thrust trajectory solver

of the optimizer. The Problem object has a few characteristics of note. The first is the fitness function. This is a function within the Problem class that defines how the problem is evaluated. Given the input of a decision chromosome, it returns the objective evaluation (in the case of a low thrust trajectory, this would be the final mass of the spacecraft) along with the evaluation of any constraints. The second class characteristic of note is the getbounds() function. This defines the bounds on the inputs of the decision chromosome. Finally, the Problem class allows for the number of equality and inequality constraints on the problem to be set. When a fitness evaluation is returned, if these constraints are not met, the solution is not considered viable.

Next, this Problem is inputted into an Algorithm object. This Algorithm class defines the optimization technique that will be performed. In the case of this project, PyGMO's native SLSQP method was used. Next, using the Population class, a population of decision chromosomes for the problem was instantiated. Then this population was evolved using the Population class's native Evolve function. The Evolve function optimizes the Problem according to the Algorithm class's selected method. The resulting best performing chromosome is returned.

Furthermore, PyGMO allows for the addition of meta-algorithms that wrap around the Algorithm class. For this particular problem, the Monotonic Basin Hopping (MBH) meta-algorithm was used. This algorithm is intended to avoid the solution settling into local optima though iteration. It does this by taking the best solution from the last iteration, slightly changing it, and the re-running the optimization with the previous best solution as a starting seed. This process is repeated until a the best solution no longer improves for a set number of MBH iterations.

In a brief summary, the process to find a single feasible trajectory using PyKEP and PyGMO is as follows: A PyGMO Problem is set up that uses PyKEP's Sims-Flanagan module to evaluate the feasiblity of a low thrust trajectory. The input to this Problem is a decision chromosome that includes spacecraft end and start properties, as well as a se-

quence of throttles. The constraints of the problem are also defined here (e.g. limits on the starting mass or ending mass of the spacecraft). An Algorithm is wrapped around this Problem and then optimized using the Population class. The solution of this Algorithm is then iterated upon by the Monotonic Basin Hopping meta-algorithm.[47] Ideally, this optimization returns a decision chromosome that describes a feasible low thrust trajectory between two points in space.

3.6 Experiments

This section details experiments that are designed to answer the research questions posed in Chapter 2.5.

3.6.1 Simulation Scenario

The main requirement for testing and validating the approach described in Chapter 3.5 is to evaluate sequences of transfers between asteroid like bodies. Fortunately, the GTOC competitions provide a wide database of real celestial bodies. In particular, the GTOC7 competition has a database of nearly 15,000 asteroids in the asteroid belt. Additionally, PyKEP offers a function to quickly retrieve the orbital characteristics of any of asteroid from this database. Both this database and function were selected to serve as the basis for the modelling and simulation environment of the asteroid field.

The spacecraft must rendezvous with each asteroid in the sequence, in order. In terms of orbital mechanics, a rendezvous indicates that the spacecraft's transfer orbit will take it to an orbit identical to the target orbit, with the position of the spacecraft matching the position of the asteroid in the orbit. After the spacecraft has rendezvoused with an asteroid, it may wait at the asteroid for up to 90 days before initiating the next transfer. The maximum time allowed for the entire simulation to run is 10 years. Table 3.3 shows these important constraints of the simulation. The spacecraft begins each sequence at the asteroid at the beginning of the sequence with the characteristics shown in Table 3.2. These characteristics

are fairly similar to both real life spacecraft with low thrust propulsion systems, as well as the spacecraft used in GTOC4[7] and GTOC9.[9]

Table 3.2: Spacecraft characteristics

Spacecraft Dry Mass	500kg
Propellant Mass	900kg
Specific Impulse	4000 seconds
Maximum Thrust	.1N

Table 3.3: Simulation characteristics

Characteristic	Setting
Maximum simulated time	10 years
Maximum Time allowed between transfers	90 Days
Maximum Transfer Time	1800 Days
Total Asteroid count	15,000

3.6.2 Building the Regression Models

Given the scenario mentioned in the previous section, the prediction model will be trained on data from this simulation. Two machine learning regression techniques will evaluated. Literature[17,22] has indicated that gradient boosting offers accurate estimates. Artificial Neural Networks have also been shown to offer accurate estimates.[25] A comparison of the effectiveness of the two methods in the overall sequence optimization results will be evaluated.

After generating the transfer training data, the two machine learning regressors will be trained and compared with validation results for each model. Next, both regression models will be integrated into the inner loop seperately. Running the global optimizer with the different inner loops will allow for an examination of the effects of the two different machine learning prediction models.

Additionally, another factor is that many trajectories evaluated by the physics solver are not feasible. Thus, the prediction model should take this into consideration as well. This

is accomplished through the use of a classification model, which predicts the feasibility of any proposed transfer.

Data Generation

In order to train the prediction models, low thrust trajectory data must first be generated. The data generation method was done by performing a Design of Experiments (DoE), which allows for the most efficient data gathering over a given set of experiments.[48] A Latin-Hypercube sampling[25] was used to generate the inputs for the training model. Each input set consisted of two pairs of orbital elements (one for the starting body and the other for the target body), as well as a time of flight for the spacecraft. A Latin-Hypercube sampling model was chosen because it allows for a more interior focused sampling (where feasible trajectories are more likely) rather than a more edge case focused sampling, which would waste computational power evaluating sequences unlikely to generate feasible trajectories.

Given the stated mission of visiting asteroids, generated data is based on real asteroid data taken from the GTOC7 competition asteroid database. This imposed upper and lower limits on some orbital constraints, mainly the semimajor axis, inclination, and eccentricity of the orbits. Taking these limits into consideration, a DoE was generated. Next, each transfer was solved for using a low thrust solver. The final decision vector, which includes the final mass, departure time, arrival time, and throttles, for each result was saved. Additionally, since many transfers are not feasible, the feasibility of the transfer was saved as a True or False Boolean type. Overall, 100,000 sets of trajectory inputs were generated and evaluated.

3.6.3 Training the Prediction Models

Both machine learning tools were part of the Python library scikit-learn.[38] The main reason that this library was chosen was due to the accessibility and modularity of different training

algorithms.

The large set of inputs and outputs from the data generation step was modified to format the inputs as show in Table 3.4. Some input features remain unmodified from the data generation input, such as the transfer time, while others are slightly modified, such as the difference between the orbital parameters.

Table 3.4: Machine Learning Inputs

Attribute	Explanation
ΔT	Transfer Time
m_i	Initial Spacecraft mass
$cos(\Delta i)$	Cosine of difference in inclination of two orbits
$\Delta\Omega$	Difference between the RAAN of the orbits
$\Delta\omega$	Difference in arguments of periapse
$\Delta\theta$	Difference in true anomaly
$\mid \Delta a \mid$	Difference Between the semimajor axis of the orbits
$\mid \Delta e \mid$	Difference Between the eccentricities axis of the orbits
$m^\star$	Maximum initial mass

After the training data is transformed from the data generation inputs, the training data inputs are scaled using scikit-learn's StandardScalar. The data is then further split for the training of the predictors by discarding any low thrust trajectories that resulted were determined to be infeasible. For the training of the classifiers, both feasible and infeasible trajectory results are used. The data is then split into a training and validation using a 75/25 split.

The regressions and classifiers were then trained using this data. The inputs to the parameters machine learning techniques were tuned according to experimentation. The outputs for the two different types of prediction models are shown in Table 3.5.

For the neural network regrssion, scikit-learn's MLPRegressor method was used. For the Gradient Boosting regresion, the GradientBoostingRegressor method was used. Since

Table 3.5: The Outputs for each machine learning technique

Predictor Method	Output
Regression	Final Spacecraft Mass (kg)
Classification	Low thrust trajectory feasibility (Boolean)

scikit-learn offered a classifier for both machine learning techniques, the MLPClassifier method was used for the artificial neural network's classifier, while the GradientBoosting-Classifier method was used for the gradient boosting classifier.

3.6.4 Sequence Evaluation

Since the prediction models have some level of error in prediction, evaluating a sequence of transfers means that these errors may compound and cause increasingly erroneous final mass calculations. So while the first transfer in a sequence may offer good final mass predictions, after several transfers, the prediction performance is expected to degrade.

This will be examined by evaluating a set of random sequences with the inner loop with the different prediction models integrated. The sequences were generated by simply making a random list of targets to visit by randomly selecting asteroids from the GTOC7 catalogue. At the end of each transfer, the mass of the spacecraft will be saved. The sequence will then be evaluated by the low thrust transfer solver, with the masses saved at the end of each transfer. The average error for each consecutive step is then analyzed.

Since the inner loop must optimize for transfer windows, this experiment will also allow for an evaluation of how well the prediction models characterize transfer windows. As none of the surveyed literature examined the evaluation of sequences by machine learning trained estimators, the differences that the two machine learning algorithms may produce is unknown.

3.6.5 Evaluating the Global Optimization Scheme After Integration of Machine Learning Models

Since the overall goal of this thesis is to evaluate the effect of integrating a fast, accurate prediction model into a global optimizer, a comparison will be made for runs with and without the optimizer. As the goal of the global optimizer is to find the sequence of maximum targets to visit, the results will be evaluated by the number of targets visited. This comparison will answer the primary research question of the paper. If the proposed hypothesis is correct, there should be a notable increase in targets visited by the global optimizer integrated with the low thrust trajectory estimator.

Using the scenario described in Chapter 3.5, the global optimization scheme attempts to maximize the number of asteroids visited by a spacecraft outfitted with low thrust propulsion. In order to evaluate the validity of the methodology, the same scenario was evaluated by the global optimizer paired with several different forms of the inner loop. The first global optimization was run with the an inner loop based on the Lambert solver. This will serve as the baseline result. Next, the global optimization was run with an inner loop integrated with the gradient boosting regression, and then with an inner loop integrated with the the artificial neural network regression. Finally, for each machine learning infused inner loop, the classification prediction model was implemented as an additional criteria for evaluating sequences. Additionally, the sensitivities of the global optimizer to the accuracy of the prediction model was examined. This was done through applying lower accuracy prediction models to the inner loop. These low accuracy models were obtained by restricting the amount of training data when training the machine learning techniques.

CHAPTER 4
RESULTS AND DISCUSSION

With the experiment setup and technical approach described in the previous chapter, experiments were performed. The results of these experiments are intended to answer the research questions posed in Chapter 2.

First, the low thrust solver was tested to ensure that it was capable of producing appropriate low thrust trajectories. Next, the training data set was generated using this low thrust trajectory solver. This training data was examined to ensure that it accurately represented the search space.

Next, the machine learning algorithms were trained to predict the final mass of the spacecraft after a low thrust trajectory, as well as the feasibility of low thrust trajectories. The accuracy of these prediction models was then analyzed. With the trained prediction models developed, their effectiveness at evaluating sequences was then evaluated by checking the results of the inner loop with the trained prediction models with a Sims-Flanagan solver.

Finally, the prediction models were integrated into the inner loop optimizer. This inner loop optimizer was then paired with a genetic algorithm to produce optimized low thrust trajectory sequences. These sequences were then evaluated by the Sims-Flanagan solver to evaluate the effects of the different inner loops.

The results of these experiments and analysis are discussed in this chapter.

4.1 Low Thrust Transfer Solver Baseline Comparison

In order to judge the reliability of the low thrust transfer solver, a comparison was made to a known optimal low thrust trajectory between Earth and Mars.[25] Table 4.1 shows the comparison between the two. Figure 4.1 shows a visualization of the calculated optimal

Table 4.1: Baseline comparison for low thrust solvers

Parameter	Calculated	Baseline
Departure Date	May 4, 2003	May 3, 2003
Arrival Date	November 22, 2003	November 19, 2003
Final Mass (kg)	546.95	554.4
Initial Mass (kg)	585	554.4
Departure Velocity (km/s)	1.6	1.7
Arrival Velocity (km/s)	3.1	1.6

Table 4.2: Table showing traits of training data

Parameter	Mean	Standard Deviation
Semimajor Axis (AU)	2.5	.4
Eccentricity	.2	.05
Inclination (Rad)	.3	.1
RAAN (Rad)	3.14	1.6
Argument of Periapse (Rad)	3.14	1.6
True Anomaly (Rad)	3.14	1.6
Initial Mass (Kg)	1000	100
Time of Flight (Days)	1500	300

trajectory. The trajectory solver used in this thesis gave very similar results established to the optimal solution.[49] The minor discrepancies between the two solutions are likely due to the optimization process getting stuck in sub-optimal valleys. Though there is a small difference, these results indicate that the Sims-Flanagan method used here can offer reasonable, fairly accurate solutions to low thrust trajectory problems, and is therefore appropriate for use in generation and validation of data.

4.2 Training Data

As described in Chapter 3.6.2, a DoE was used to generate a set of 100,000 transfers. Each transfer was two sets of Keplerian orbital elements, along with a starting mass and time of flight. The Latin-Hypercube DoE then filled out the input set, with the maximum and minimum values for each set through a standard deviation, shown in Table 4.2. All transfers were between objects that approximate the orbits of asteroids in the asteroid belt. Table 4.3 shows traits of this data set.

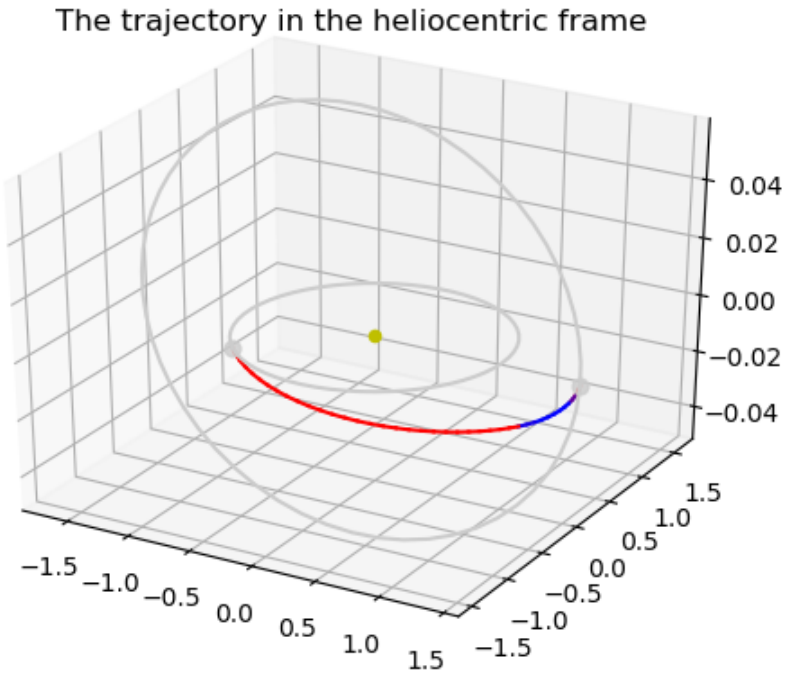

Figure 4.1: Calculated optimal low thrust trajectory between Earth and Mars

Table 4.3: Training Data

Parameter	Quantity
Total Transfers	100,000
Feasible Transfers	50,163
Infeasible Transfers	49,837

As the set of 100,000 possible trajectories produced only about 50,000 feasible trajectories, there was some concern that the set of feasible trajectories used as training data did not appropriately represent the entire search space. The set of inputs for generating the training data was compared to the set of inputs that returned feasible trajectories. For each set of inputs (such as the change in semi-major axis, eccentricity, etc.), the frequency of values between feasible and the entire data set was compared. Examining the data in this manner allows for an identification of any unexpected impacts of input variables. If the set of feasible trajectories represents the input space well, the frequency of variable values should be close to identical. Figure 4.2 shows a breakdown of the semi-major axis, eccentricity, inclination, and right ascension of the ascending node (RAAN), while Figure 4.3 shows the argument of periapse, true anomaly, time of flight, and initial spacecraft mass values. The blue bar gives the frequency of values for all inputs, while the blue bar shows the frequency of variable values that only represent feasible trajectories. A large difference in size between the two bars indicates that more or less feasible trajectories were generated than expected for that particular variable value. In Figure 4.2, the first three subplots of semi-major axis, eccentricity, and inclination changes show no unexpected behavior. This indicates that the set of feasible trajectories represents the inputted variables well. However, the subplot showing the right ascension of the ascending node shows lower values were far more likely to result in feasible trajectories than higher values. This indicates that low thrust transfers that have small RAAN changes were much more likely to produce feasible trajectories.

The subplot showing the time of flight in Figure 4.3 shows that the solver had a tendency to find feasible trajectories that had higher time of flight values. This is indicative that the average value for the time of flight in the entire data set was slightly too low. As the average semi-major axis in the data set used to generate the training data was 2.5AU, this corresponds to an average orbital period of about 1100 days. Due to the velocities involved in a transfer between two asteroids, the average transfer between them should also be about

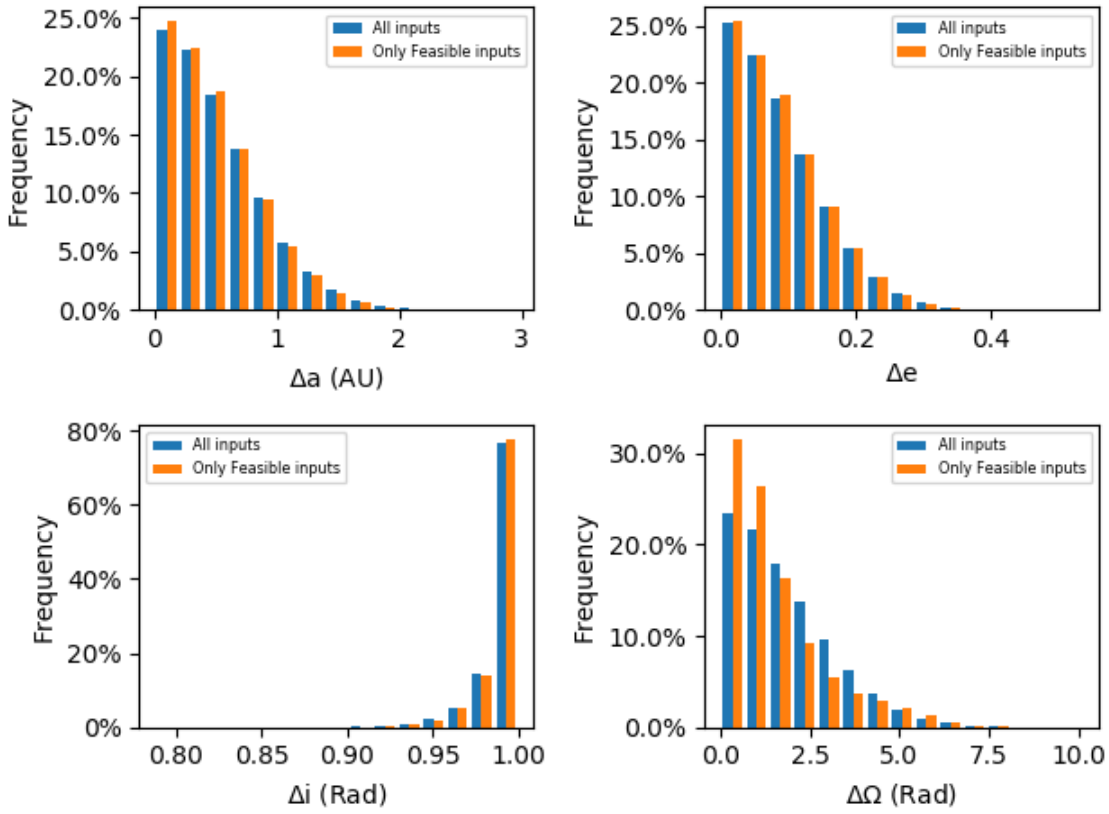

Figure 4.2: A proportional breakdown of all inputs compared to the inputs that resulted in a feasible trajectory. This shows the semi-major axis, the eccentricities, the inclinations, and the right ascension of the ascending node.

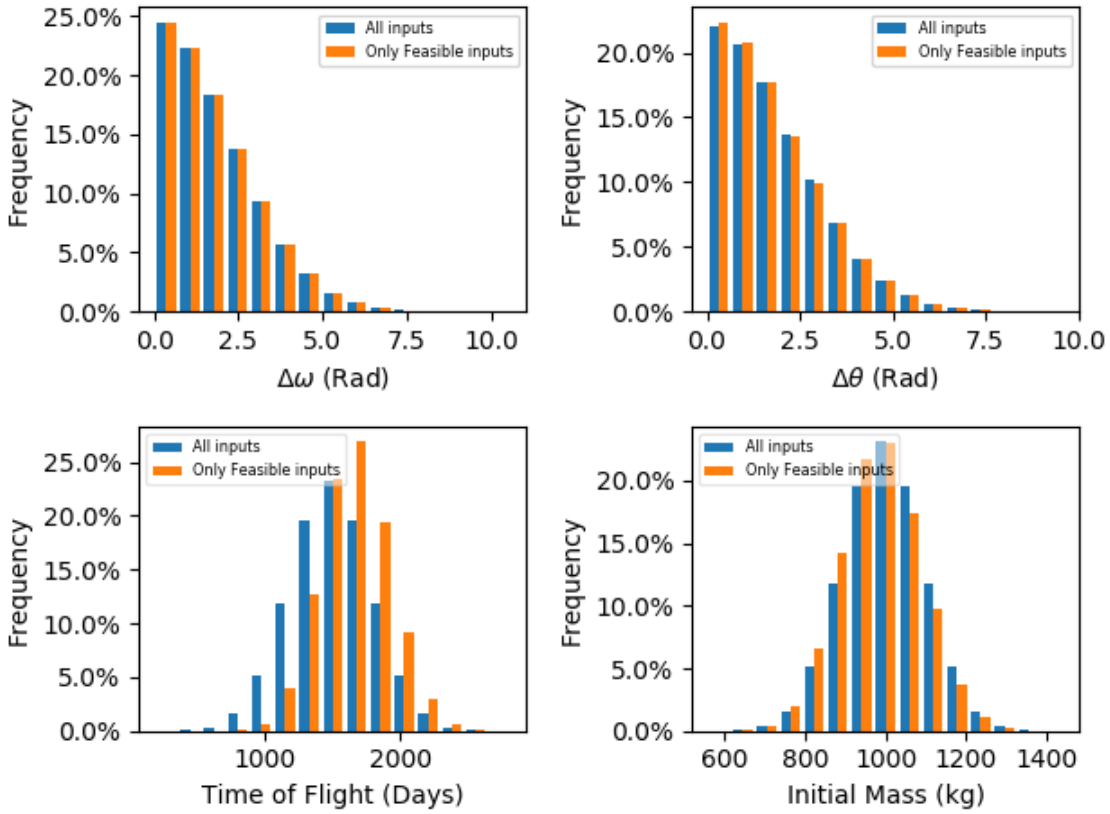

Figure 4.3: A proportional breakdown of all inputs compared to the inputs that resulted in a feasible trajectory. This shows the argument of periapse, the true anomaly, spacecraft time of flight, and the spacecraft initial mass.

1700 days. This value corresponds to the highest value seen on this subplot. However, since the average time of flight in the training data(Table 4.2) was 1500, it is likely that this is the cause for the discrepancy in frequencies.

A predictor screening test is shown in Figure 4.4. This shows that all inputs are roughly equally weighted in predicting the final mass of the spacecraft after a low thrust trajectory is completed. The RAAN and time of flight had the most influence in predicting the outcome, which is in agreement with what is seen in Figures 4.2 and 4.3. Though there are some minor discrepancies in how the training data represents the desired search space, these do not prevent the creation of accurate prediction models. For the most part, the total search space characterized the resulting training data set well, even if only half of the total points in the search space were found to have feasible low thrust trajectories. Since the search

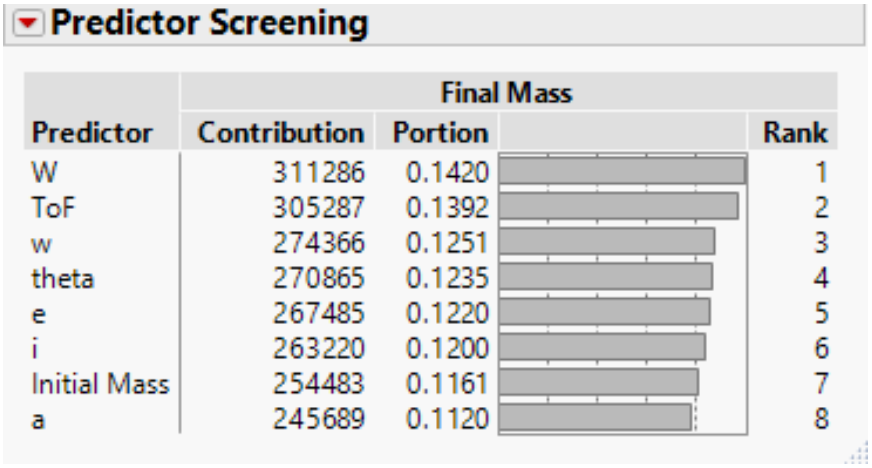

Figure 4.4: Predictor Screening of inputs

Table 4.4: Artificial Neural Network Settings

Parameter	Setting
Random State	0
Hidden Layer Sizes	(80,80)
Solver	'adam'
Max Iterations	5000

space was well characterized by the feasible low thrust trajectories that were found, it is likely that the low amount of found trajectories is due to the stochastic nature of the low thrust trajectory solver. Nonetheless, future work should consider and further investigate the minor discrepancies found here when generating training data sets.

4.2.1 Regression Model Settings

In order to generate the best predictions, the machine learning techniques were tuned through experimentation. Table 4.4 shows the settings used for the Artificial Neural Network regression. These settings were also used for the ANN classifier. Table 4.5 shows the settings used for the Gradient Boosting regression.

Table 4.5: Gradient Boosting regression Settings

Parameter	Setting
Random State	3
Max Depth	7
n_estimators	200
Learning Rate	.1

4.2.2 Prediction Comparisons Between Lambert Solver and Machine Learning Derived Estimators

Using the input data set generated by the DoE, fuel required for impulsive trajectories was generated using a Lambert solver. For reference, the spacecraft characteristics are shown in Table 4.6. The results of these were then compared to the trajectory results from the low thrust solver. This allowed for a comparison of accuracy between the Lambert solver and the machine learning predictions.

Table 4.6: Spacecraft characteristics

Spacecraft Dry Mass	500kg
Propellant Mass	900kg
Specific Impulse	4000 seconds
Maximum Thrust	.1N

As shown in the Table 4.7, the mean absolute error and mean squared error for the machine learning predictions is significantly better than a Lambert solver's estimate for mass consumption. As the estimators offer an accuracy that is better than the Lambert solver's prediction by an order of magnitude, the machine learning techniques should offer a much more robust method of evaluating trajectory sequences. The Lambert solver's coefficient of determination (r^2) was found to be -5.121. Though this number seems nonsensical, the scikit-learn documentation notes that a negative r^2 value indicates a model is arbitrarily worse than predictions that has no correlation to the true value.[50] This not only confirms the findings in literature that Lambert solvers are poor predictors of low thrust trajectories,

Table 4.7: Table showing error rates for various predictors

Predictor	MAE (kg)	RMSE (kg)	r^2
Gradient Boosting	40.21	50.41	.7880
ANN	39.89	48.90	.7905
Lambert Predictor	256.63	270.39	-5.121

but also implies that selecting trajectories based on their estimates may be worse than a random selection of trajectories.

Figure 4.5 shows a breakdown of the errors in final mass predictions by machine learning estimators. A positive error indicates that the true final mass of the vehicle was higher than the estimate; in other words, the prediction model estimated a higher fuel usage than needed in reality. Likewise, a negative error indicates that the prediction estimated a higher final mass than the true value, and low fuel requirements.

The first histogram shows the prediction error of the Lambert solver when compared to the Sims-Flanagan solved low thrust trajectories. Compared to the other, more accurate predictors, the Lambert solver has a wide variance, and in almost all cases significantly overestimates the fuel required for a transfer. These results are in general agreement with existing literature.[22,17,25]

The other two histograms in Figure 4.5 show the prediction errors of the artificial neural network and gradient boosting predictors. As expected, these are significantly more accurate than the Lambert solver. Both have means near zero, and a much tighter variance than the Lambert solver's predictions. This indicates they would be much more well suited to predicted low thrust trajectory costs than a Lambert solver.

Figure 4.6 shows a breakdown of the final mass prediction errors on a single histogram. A clear distinction between the Lambert solver predictions and the machine learning estimator predictions can be seen. Both machine learning techniques offer much more accurate predictions, as well as significantly less bias and variance than the Lambert solver's predictions. The differences between the artificial neural network and gradient boosting predictors is not significant by inspection, but there are small differences. The artificial neural

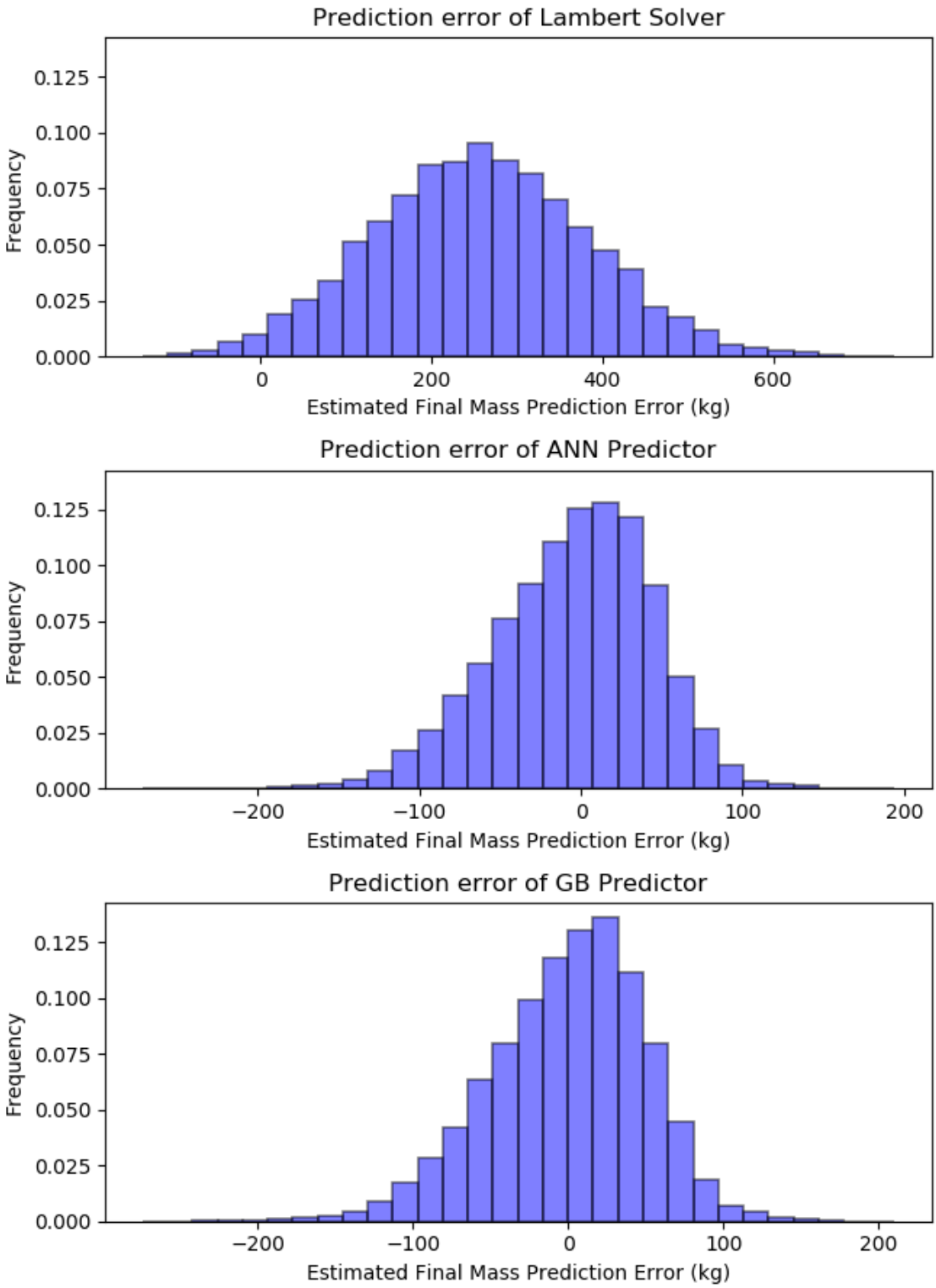

Figure 4.5: Histogram of errors in predicted fuel usage.

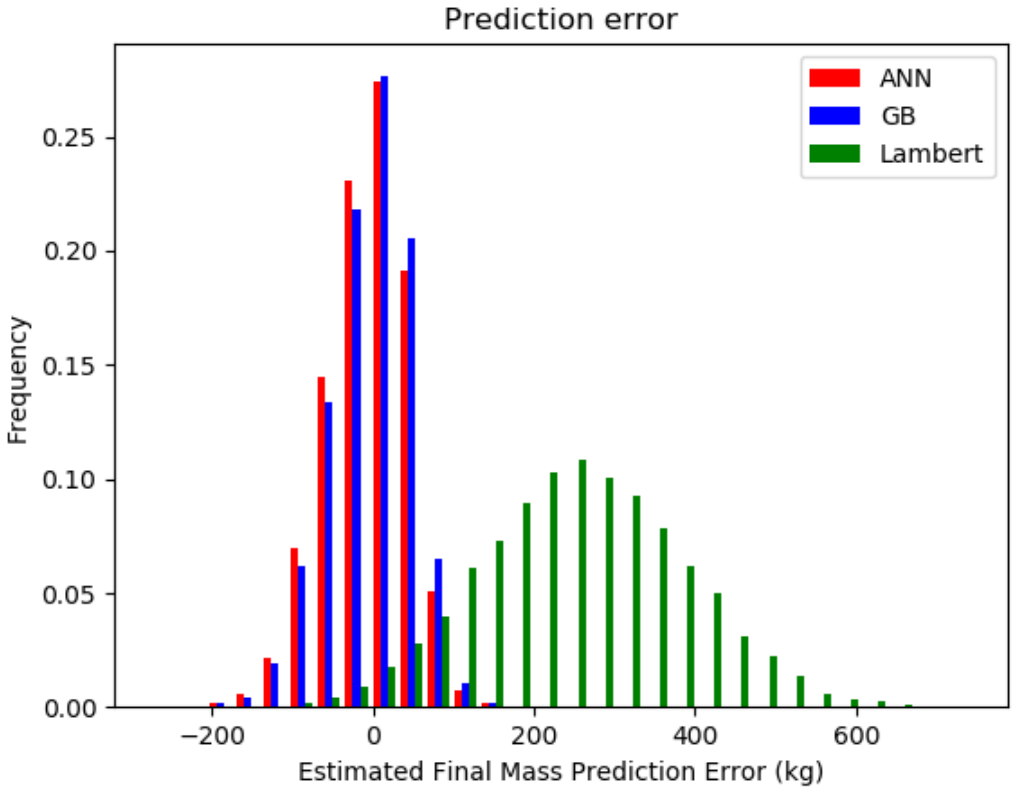

Figure 4.6: Compiled view of prediction errors.

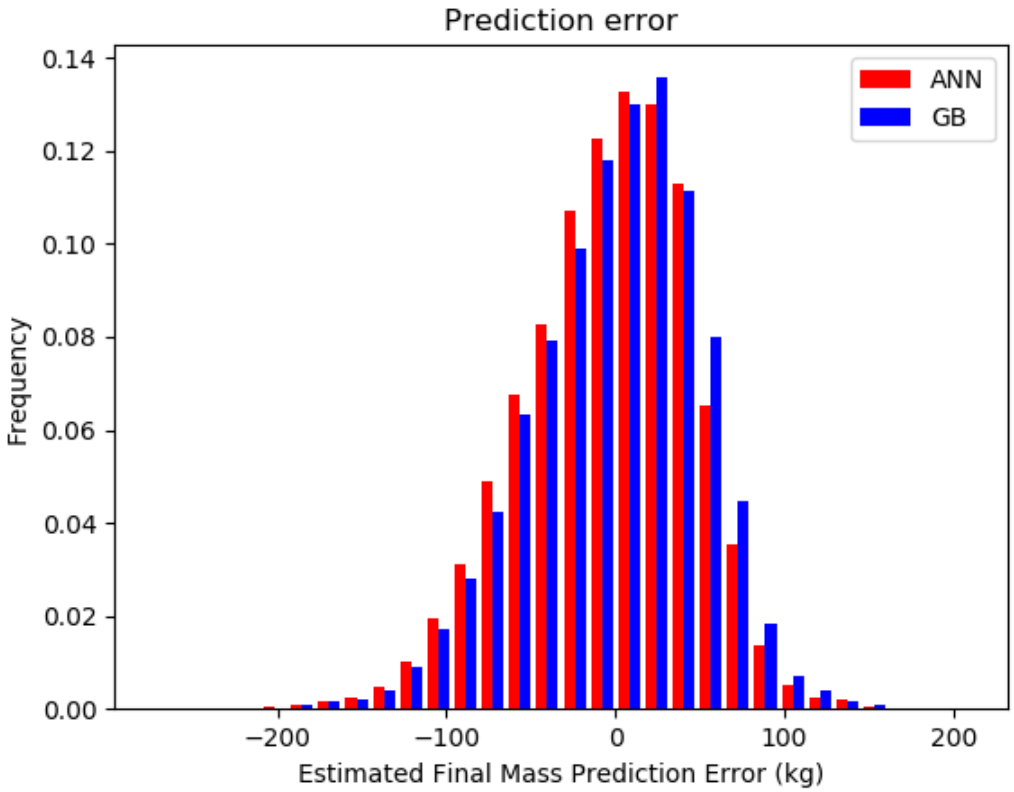

Figure 4.7: A compiled view of the prediction error histogram comparing both gradient boosting and artificial neural network.

Table 4.8: The Gradient Boosting regression accuracy as training data sample size changes

Number of Training Samples	MAE (kg)	RMSE (kg)	r^2
37,622	40.21	50.41	.7880
28,216	40.06	50.05	.7838
21,162	40.36	50.36	.7885
15,871	40.47	50.83	.7872
11,903	41.16	51.47	.7779

network prediction offers a slightly lower variance and mean than the gradient boosting prediction. That being said, the performance was nearly identical. Figure 4.7 shows a histogram comparison of the errors between the gradient boosting and neural network predictions. It can be seen that there is a slight difference between the two. The artificial neural network had predictions that were more weighted towards under-predicting the final mass of the vehicle (underestimating fuel usage), while the gradient boosting prediction displayed the opposite phenomenon.

The importance of training data quantity was also examined. Analyzing 100,000 trajectories yielded approximately 50,000 feasible solutions. Breaking this down further allowed for an analyses of the impact the quantity of data has on the estimator's accuracy. Table 4.8 shows the gradient boosting predictor's accuracy with varying amounts of training data. The lowest number of training data points tested was 11,903, compared to a maximum of 37,622. Reducing the number of training points did not significantly reduce the accuracy of the prediction. Table 4.9 displays the results of the same analysis on the ANN predictor. This model displays the same characteristics as the gradient boosting predictor: changing the number of training points does not significantly impact the accuracy of the prediction. The diminishing returns given by increasing the number of training points indicates that improving accuracy would likely come from additional tuning and modification of input parameters, rather than increasing amount of training data. This is an area that should be examined by future work.

Table 4.9: The Artificial Neural Network regression accuracy as training data sample size changes

Number of Training Samples	MAE (kg)	RMSE (kg)	r^2
37,622	39.89	48.90	.7901
28,216	39.55	49.66	.7872
21,162	39.48	49.62	.7947
15,871	40.01	50.54	.7896
11,903	40.61	51.14	.7834

Table 4.10: Gradient Boosting Classifier Settings

Parameter	Setting
Random State	1
Max Depth	7
n_estimators	100

Classification

During the generation of the training data, it was noted that many examined potential trajectories did not generate feasible solution. For example, in the evaluation of 100,000 low thrust trajectories, only about 37,000 produced feasible trajectory solutions. Due to this fact, it is of interest to not only develop a regression method for predicting propellant usage, but also develop a method for classifying whether or not any particular potential trajectory is likely to be feasible. This was examined through the implementation of a classifier. The training input data was the same as the training for the regression method, but a True or False Boolean was used as the output training data. The classifiers used were the GradientBoostingClassifier module and the MLPClassifier module provided by the scikit-learn library. The training parameters are shown in in Table 4.10 and Table 4.11. The training parameters for machine learning techniques were tuned through experimentation.

The accuracy of the classifiers is shown in Table 4.12. The overall accuracy of both classifiers was scored at .82, indicating that it can correctly identify whether or not a proposed trajectory is feasible 82% of the time. Though this is not perfect, it offers an additional level of filtering that can be implemented into the sequence evaluation. While not a perfect

Table 4.11: Artificial Neural Network Classifier Settings

Parameter	Setting
Random State	0
Hidden Layer Sizes	(80,80)
Solver	'adam'
Max Iterations	5000

Table 4.12: Table showing classifier performance

Classification Algorithm	Accuracy
Gradient Boosting	.821
Artificial Neural Networks	.823

prediction method, this implies that the classifier is effective at predicting the feasibility of low thrust trajectories.

Since the results of the classification tests were positive, the classifier was integrated into the inner loop to test the impact on global optimization results. Since the gradient boosting classifier had slightly better results than the artificial neural network classifier, the gradient boosting classifier was implemented in the inner loop for testing.

4.2.3 Sequence Evaluation Accuracy

A set of random sequences were evaluated by the prediction models and the Sims-Flanagan solver. This experiment is intended to evaluate the effect of error propagation when evaluating a sequence. Additionally, since sequence evaluation requires optimizing for transfer windows, running this experiment allowed for some characterization of how well each prediction model performed transfer timing optimization.

Table 4.13 displays the error of the gradient boosting prediction as the inner loop iterates through a sequence of targets. The mean absolute error is shown with along with the mean error. Since the machine learning estimator injected inner loop does a grid search to find the best possible departure date and time of flight, it is expected that MAE would be slightly worse than the results shown in Table 4.7. The trend is generally as expected, with errors having a tendency to increase with number of transfers. An interesting item to

Table 4.13: Error in final mass calculations in evaluating a sequence for Gradient Boosting Predictor

Order in Sequence	MAE (kg)	ME (kg)
First Transfer	93.51	31.86
Second Transfer	95.81	25.76
Third Transfer	116.45	62.63
Fourth Transfer	108.74	86.45

Table 4.14: Error in final mass calculations in evaluating a sequence for ANN Predictor

Order in Sequence	MAE (kg)	ME (kg)
First Transfer	174.50	-174.50
Second Transfer	320.45	-320.45
Third Transfer	408.48	-408.48
Fourth Transfer	454.721	-454.721

not is the difference between the MAE and ME. The MAE only increase by about 15kg, whereas the ME increases by over 50kg from first transfer to fourth. This indicates that the machine learning predictions retains precision while losing accuracy, with a tendency to overestimate fuel consumption.

Table 4.14 shows the sequence error breakdown applied to the trained Artificial Neural Network's mass prediction for a sequence. The first thing of note is that the gradient performs much better than the artificial neural network in all cases. Given the discrepancy between the MAE for the ANN seen here and in Table 4.7, this indicates that the ANN model is does a poor job of predicting the optimal transfer windows for the spacecraft. In other words, the ANN offers good results when only considering the trained points, but when varying the departure and arrival dates, it does a poor job of characterizing the appropriate time of flight. This may be indicative of an over-trained model, but further analysis would need to be performed to reach a conclusion. Additionally, in all cases the ANN predicted a fuel consumption significantly lower than the real trajectory.

This indicates that the prediction model developed using Gradient Boosting is more well suited to the task of evaluating transfer sequences than the Artificial Neural Network, and therefore should offer better global optimization results as well.

Table 4.15: Average time to evaluate a sequence for different predictors.

Predictor	Average time per sequence (s)
Gradient Boosting	.496
ANN	.446
Lambert Predictor	.0891
Sims-Flanagan Solver	443.5

4.2.4 Computation speed

The prediction models offered a significant improvement in the speed of evaluating low thrust trajectory sequences. Table 4.15 shows the differences in computation time when evaluating sequences with the Gradient Boosting regression, ANN regression, Lambert predictor, and Sims-Flanagan solver. The average sequence evaluation when using the physics solver was 443.5 seconds, while both prediction models averaged less than .5 seconds per sequence evaluation. This represents a computational speedup of nearly three orders of magnitude. Compared to the Lambert prediction model, the machine learning predictions were slightly slower in execution, but when considering their improved accuracy Table 4.7, there is a clear advantage over the Lambert solver's predictions. Specifically, the machine learning prediction models offer an accuracy improvement over the Lambert solver predictions by nearly an order of magnitude, while only slowing computations by approximately a factor of 5. It should be noted, however, that these time calculations do not consider the time needed for training data generation and training.

The training time for the different machine learning techniques was also examined, shown in Table 4.16. Interestingly, the artificial neural network model took the most amount of time to train. This is likely due to the size of the network's layers (80x80). Even though the ANN model took twice the amount of time as the Gradient Boosting regression model, the ANN model was still less accurate than. This indicates that Gradient Boosting is more well suited to estimating the cost of low thrust trajectories, both in terms of accuracy and training times.

Table 4.16: Training times for different predictors.

Predictor	Training time (s)
Gradient Boosting Regression	34.8
Gradient Boosting Classifier	25.8
ANN Regression	52.8
ANN Classifier	100.2

Table 4.17: Genetic Algorithm Settings

Parameter	Setting
Population Size	10
Maximum Iterations	20
Reruns	20

4.3 Global Optimization Results

4.3.1 Global optimization with Machine Learning Estimators

In order to test the global optimization, the GA was used with an inner loop that used the Gradient Boosting trained estimator, the neural network trained estimator, and the Lambert solver as a final mass prediction method. After a sequence was produced by the GA, the sequence was evaluated by the real low thrust trajectory solver. This allowed for a scoring of predictor performance in finding optimal sequences. This performance was measured by calculating how many targets in the sequence that the spacecraft could feasibly rendezvous with when calculated with the low thrust solver. This optimization process was repeated several times, allowing for an average score of the global optimizer with each prediction method.

Table 4.17 shows the settings used for the genetic algorithm in the scipy.optimize module. The low population and iteration settings for the genetic algorithm are due to the limitations of the computational hardware available for this project. However, the genetic algorithm was rerun twenty times with each different inner loop in order to find the average of the best sequences found by the GA.

The results of the global optimization function are shown in Table 4.18. The Lambert

solver, as expected, generated sequences that performed quite poorly when evaluated by the low thrust trajectory solver, averaging only 2.5 feasible transfers for each optimized sequence. Both trained estimators performed much better, with the gradient boosting estimator based inner loop producing an average of 5.2 feasible transfers per optimized sequence, and the artificial neural network estimator based inner loop producing an average of 4.8 feasible transfers per sequence. Additionally, the tests were rerun with prediction models that were trained with less data. The tests with the most amount of training data (37,622 data points) offered the best performance, while the tests with the least (11,903 data points) were slightly worse. However, the worst prediction model accuracy still offered a significant improvement over the baseline case. As the Lambert solver based inner loop is considered the baseline test case, all results display a significant improvement when compared to the baseline. The higher accuracy Gradient Boosting based inner loop offered an improvement of over 100% when used with the global optimizer, while the higher accuracy ANN based inner loop offered a slightly less significant improvement.

When examining Tables 4.7, 4.14, 4.13, and 4.18, this result is expected. Both machine learning types offered a significant improvement When compared to the Lambert estimates, but the ANN offered slightly worse performance in both single transfer prediction and sequence evaluation. However, when paired with the global optimizer, both machine learning techniques offer similar results.

Next, the objective function of the optimization function was integrated with the the trained classifier. As the objective function evaluates the score of a sequence, the feasibility of each transfer was predicted by the trained classifier. This classifier returned either True or False, with the True indicated that the transfer was feasible. If the classifier determined that the transfer was not feasible, then the sequence was considered finished and the objective function returned the sequence's score at that transfer.

The results of the combined regression and classifier can be seen Table 4.19. The global optimization saw a small improvement when paired with the ANN regression, but saw no

Table 4.18: Average score of sequences produced with tested regression models

Inner Loop Prediction Model	Average Sequence Score
Lambert (baseline)	2.5
Artificial Neural Network (37,622 training points)	4.8
Artificial Neural Network (11,903 training points)	4.6
Gradient Boosting (37,622 training points)	5.2
Gradient Boosting (11,903 training points)	4.7

Table 4.19: Average score of sequences produced with regression models and classifier

Inner Loop Prediction Model	Average Sequence Score
Artificial Neural Network (37,622 training points)	5.1
Artificial Neural Network (11,903 training points)	4.7
Gradient Boosting (37,622 training points)	5.2
Gradient Boosting (11,903 training points)	5.0

improvement when paired with the gradient boosting regression. Though there was small to no improvement in the global optimization results, improving this classifier may offer more significant improvements to the results.

The accuracy of the prediction models was also varied and then inserted into the global optimizer. The results of this can be seen in Tables 4.18 and 4.19. Each prediction model was tested twice, once with the highest accuracy (37,622 training points) and once with the a lower accuracy (11,903 training points). Some conclusions can be drawn from the effect that changing the accuracy of the predictions had on the global optimization. The first is that the addition of a classifier had measurable impact when paired with a lower accuracy model. Both the lower accuracy gradient boosting and all ANN prediction models based inner loop saw significantly improved global optimization scores when the classifier was integrated into the inner loop. However, the same improvement was not seen for the high accuracy gradient boosting prediction model. In fact, the high accuracy gradient boosting based inner loop did not see any improvement when the classifier was implemented, whereas the ANN based inner loop saw improvement for every case when the classifier was implemented. This may be due to the fact that the ANN prediction model had worse performance when evaluating sequences, meaning that it benefited more from the classifier.

Table 4.20: Estimated maximum time to run GA with different prediction models.

Predictor	GA optimization time
Sims-Flanagan Physics Solver	10 Days
Gradient Boosting Based Inner Loop	16 minutes
ANN Based Inner Loop	15 minutes
Lambert Prediction Based Inner Loop	3 minutes

Additionally, the total time to generate an optimized low thrust trajectory sequence can be examined. Given the settings used for the genetic algorithm (Table 4.17), the maximum number of objective evaluations possible for this setup is calculated to be 1990. Applying this number of evaluations to the average time to evaluate a sequence produces the maximum amount of time to generate an optimized low thrust trajectory sequence with the genetic algorithm settings used for this project. A breakdown of these times can be seen in Table 4.20. Generating an optimized sequence of low thrust trajectories using the full physics solver would take 10 days using the current GA settings. Considering the relatively small number of sequences that the GA evaluates here, this results shows using the Sims-Flanagan solver to explore the solution space is a computationally impractical approach. Both the gradient boosting and artificial neural network prediction models offer a huge improvement in terms of calculation speed, while retaining high accuracy. There was not a significant time difference noted between using the genetic algorithm with and without the classifier integrated into the inner loop. Though the Lambert prediction offers a significant speed boost, its poor performance negates the advantages.

4.4 Conclusions

These results validate the hypothesis of this thesis: Using trained machine learning prediction models for estimating low thrust transfer costs has been shown to offer a significant improvement over the existing method of cost estimation with Lambert solvers. Now, with the results of the experiments, the research objective and research questions posed in Chapter 2.5 can now be fully addressed. For the convenience of the reader, the research objective

and questions are restated:

Research Objective: Develop an approach for improved sequential low thrust trajectory optimization through improving fuel cost estimations of low thrust trajectories.

This research objective is based on research gap that was identified after a literature search. An investigation of current approaches to the global optimization problem revealed a common method of estimating the cost of low thrust trajectories to be through the use of a Lambert solver. Though the Lambert solver is an extremely poor predictor of low thrust trajectories, the computational cost of using a physics solver in the global optimization scheme prevented its use. The poor performance of Lambert solvers has led to the development of fast and accurate estimators for low thrust trajectories through the use of machine learning techniques. This led to the identification of the research gap: While there are global optimization techniques for finding sequential low thrust trajectories and there are methods for generating fast, accurate estimates of low thrust trajectories through machine learning, there has been little research on developing an effective approach to integrating accurate estimators into these global optimization schemes. This leads to the first research question:

Research Question 1: Can machine learning techniques be utilized to predict the cost of low thrust trajectories such that more optimal trajectory sequences are found?

Given the results of the literature search, machine learning techniques were identified as an effective approach for predicting the costs of low thrust trajectories. Furthermore, the Lambert solvers that were typically used were explicitly identified in existing literature to be poor predictors of low thrust trajectories and identified as a main cause of failure in finding adequate low thrust trajectory sequences during global optimization.[17,24] This led to the formation of the hypothesis that using machine learning techniques to estimate in the global optimization scheme would allow for better sequences to be found:

Hypothesis 1: An accurate fuel cost predictor for low thrust trajectories developed through machine learning will allow for better sequences to be found.

The results of the experiments performed in this work affirm Hypothesis 1. Using the

methodology outlined by this work, the global optimization results saw significant improvement when compared to the typical approach of using Lambert solvers to predict the cost of low thrust trajectories. In all tested cases of the machine learning prediction models paired with the genetic algorithm global optimizer, there was a near doubling of the amount of targets visited in the optimized sequences. Thus, the methodology developed in this work is shown to offer significant improvement over existing methods in the preliminary design of sequential low thrust trajectories.

The results of using the prediction models with a genetic algorithm in particular are also of some interest. Since integration with a GA offered a significant improvement over baseline, it stands to reason that other global optimization techniques would see improvements in results as well. As a Genetic Algorithm evaluates entire transfer sequences before scoring the sequence, the estimate errors tend to add up (see Tables 4.13 and 4.14). For sequences that evaluate a significant number of transfers, as seen in GTOC competitions, this compiling error would cause a loss of accuracy in scoring. Thus, a more ideal global optimization technique would allow for the machine learning estimates to be checked by a low thrust solver before the estimation error become significant. Branch and Prune methods may be well suited to this technique, as they build sequences typically evaluate many sequences with a small amount of transfers in each. Additionally, since Branch and Prune methods have been shown to be among the most successful methods for solving GTOC-class problems, it is expected that the benefit from using accurate prediction models be even more significant than what was seen in genetic algorithms.

The next research question dealt with the particulars of which machine learning technique was well suited to this problem:

Research Question 2: What is the best machine learning technique for building a low thrust trajectory prediction model?

It should be noted that this thesis does not necessarily identify the *best* machine learning technique for this problem, as an exhaustive examination of applying machine learning techniques to this problem would not be practical. Instead, two methods that existing

literature had identified as high performing regression techniques (gradient boosting and artificial neural networks) were selected. Given the highest performing machine learning technique identified in literature was gradient boosting, this was hypothesized to be be a better candidate than artificial neural networks:

Hypothesis 2: Gradient boosting will offer a more accurate prediction model than artificial neural networks.

Interestingly, in terms of predicting the final mass of a spacecraft after a low thrust trajectory, both gradient boosting and artificial neural networks offered near identical results. However, the gradient boosting offered much better sequence evaluation when paired with the inner loop. This indicates that the gradient boosting regression technique was better able to approximate the physics, resulting in better estimates for transfer timing between targets. Therefore, Hypothesis 2 is supported by these findings.

Finally, there was a question of how the global optimization responded to the accuracy of the prediction models:

Research Question 3: How sensitive are the global optimization results to the accuracy of the prediction models?

Since reviewed literature had indicated that the use of Lambert solvers as a low thrust trajectory estimator was a cause of poor global optimization performance, it was hypothesized that the global optimization would be sensitive to changes in prediction accuracy:

Hypothesis 3: The global optimization results are sensitive to the accuracy of prediction models.

The global optimization results that are seen in Table 4.18 and Table 4.19 indicate that there is significant positive response to the accuracy of the prediction models. The baseline case of a Lambert solver based inner loop paired with a global optimizer averaged a score of only 2.5 targets reached. The lowest accuracy prediction models (trained with only 11,903 data points) offered a significant boost to these results (4.7 for the gradient boosting

prediction and 4.6 for the ANN prediction). Furthermore, the overall scores increased with the accuracy of the prediction models. The high accuracy prediction models (with 37,622 training points) saw improved results when compared with their lower accuracy counterparts. While the results artificial neural network based inner loop increased a small amount (.1 increase in score on average), the gradient boosting estimator increased global optimization results quite significantly as its accuracy increased (.5 increase in score on average).

Finally, the addition of a classifier into the inner loop also improved the overall scores of the optimizer. The lowest accuracy ANN prediction model paired with the classifier actually achieved a score nearly as good as the the highest accuracy ANN prediction model alone. This indicates that the global optimization results are very sensitive to the accuracy of the prediction model, and small increases in accuracy of sequence evaluation can noticeably improve results.

With the research questions fully addressed, the main conclusion of this thesis may now be drawn. When considering the optimization of sequences of low thrust trajectories, the use of Lambert solvers to estimate costs of low thrust trajectories has been shown to inadequate approach. Machine learning methods offer significant advantages over Lambert solvers when predicting the costs of low thrust trajectories. Thus, a new approach was proposed in this thesis. This new approach formulates integration of fast, accurate low thrust trajectory prediction models derived from machine learning techniques. The testing of this approach against the approach of using Lambert solvers for cost estimates shows the effectiveness of using machine learning prediction models in this problem.

4.5 Future work

The positive results of this work indicate that machine learning derived prediction models for predicting the costs of low thrust transfers can be successfully integrated into global optimization schemes aimed at finding optimal sequences of asteroids to visit with low thrust

spacecraft. Though this thesis only examined the use case of a genetic algorithm, the next steps in this field would be to examine the results from machine learning algorithm integration into other global optimization techniques. Considering that the genetic algorithm used in this work was neither tailored for this specific problem nor expected to offer near optimal sequences, it is likely that a better global optimizer would offer a significant improvement in results.

While the machine learning regression methods investigated in this work were significantly more accurate at predicting the fuel cost of a low thrust trajectory than Lambert solvers, the improved predictions still have a quite large error margin. Given the fact that increasing the amount of training data did not significantly impact the prediction accuracy, it is possible that improvement of the predictors may be accomplished through the use of different input training features or different machine learning techniques. This is an area that should also receive consideration in future work.

The accurate prediction of transfer timing was also considered. An examination of the prediction errors for sequences of transfers (Table 4.13 and Table 4.14) indicates that there is an increasing level of inaccuracy as the sequence is evaluated. Furthermore, due to the inner loop attempting to optimize the transfer timing, the first transfer begins with a much higher level of inaccuracy. Therefore, it would be beneficial to develop an approach of fast, accurate approximations of the optimal low thrust transfer timing between two orbits.

The feasibility classifier, while successfully implemented, did not have a significant impact on the results. It is possible that improving the reliability of the classifier would also offer a more significant improvement to the global optimization results. Since minimal work was done on examining other possible classification techniques, further research could yield more fruitful results.

Though the scenario investigated in this thesis was an exploration of the asteroid belt, there is no reason that this technique could not be applied to many different scenarios that have underlying similarities. Some examples include, but are not limited to, debris removal

in Earth orbit, Jovian or Saturnian Moon missions, and outer solar system exploration. In fact, another area of interest would be to generate a trained surrogate that could be applied to more than one of these scenarios. It is possible that this may be accomplished through using canonical units for distance and time, as well as some normalization process with respect to the canonical units for the spacecraft characteristics. Though the surrogate would likely loose some accuracy in order for a feasible application in many different scenarios, it would still potentially offer better estimates than the Lambert solver.

Finally, this methodology could prove useful for participants of future Global Trajectory Optimization Competitions. Since these competitions are typically won by only a few points, using this method could potentially offer enough of an edge to win the competition.

REFERENCES

[1] C. E. Kohlhase and P. A. Penzo, "Voyager mission description," *Space Science Reviews*, vol. 21, pp. 77–101, 1977.

[2] M. Vasile and P. D. Pascale, "Preliminary design of multiple gravity-assist trajectories," *Journal of Spacecraft and Rockets*, vol. 43, no. 4, pp. 794–805, 2006.

[3] e. a. Thomas Valerie C., *The Dawn Mission to Minor Planets 4 Vesta and 1 Ceres*. New York: Springer, 2011.

[4] Y. Guo and R. Farquhar, "New horizons mission design," *Space Science Reviews*, vol. 140, pp. 49–74, 2007.

[5] M. Rayman, *Nasa jet propulsion laboratory blog — for dawn, a time to thrust and a time to coast*, 2013.

[6] W. Cook, *In pursuit of the traveling salesman: mathematics at the limits of computation*. Princeton University Press, 2011.

[7] R. Bertrnd, R. Epenoy, and B. Meyssignac, "Problem description for the 4th global trajectory optimization competition," Centre National d'Etudes Spatiales, France, Tech. Rep., 2009.

[8] I. S. Grigoriev and M. P. Zapletin, "GTOC5: Problem statement and notes on solution verification," *Acta Futura*, vol. 8, pp. 9–19, 2014.

[9] D. Izzo, "Problem description for the 9th global trajectory optimization competition," Advanced Concepts Team, European Space Agency, Netherlands, Tech. Rep., 2017.

[10] L. Calasurdo and M. Sentinella, "Problem description for the 3rd global trajectory optimization competition," Politecnico di Torini, Departimento di Energetica, Italy, Tech. Rep., 2007.

[11] R. Bertrnd, R. Epenoy, and B. Meyssignac, "Final results of the 4th global trajectory optimization competition," Centre National d'Etudes Spatiales, France, Tech. Rep., 2009.

[12] D. Izzo, C. Sprague, and D. Tailor, "Machine learning and evolutionary techniques in interplanetary trajectory design.," (arXiv preprint arXiv:1802.00180), 2018.

[13] J. E. Prussing and B. A. Conway, "Orbital mechanics," in, 2nd ed. 198, Madison Ave, New York, Ny: Oxford University Press, 2013, ch. 5.

[14] B. A. Conway, "Spacecraft trajectory optimization," in. Cambridge University Press, 2010, vol. 29, ch. 5.

[15] D. Izzo, "Revisiting lambert's problem," *Celestial Mechanics and Dynamical Astronomy*, vol. 121, pp. 1–15, 2014.

[16] J. Brophy, C. Garner, B. Nakazono, M. Marcucci, M. Henry, and D. Noon, "The ion propulsion system for dawn," Jet Propulsion Laboratory, 39th AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit, 2003.

[17] D. Hennes, D. Izzo, and D. Landau, "Fast approximators for optimal low-thrust hops between main belt asteroids," (In International Conference on Hybrid Artificial Intelligence Systems (pp. 543-553)), Springer, Cham, 2016.

[18] J. E. Prussing and B. A. Conway, "Orbital mechanics," in, 2nd ed. 198, Madison Ave, New York, Ny: Oxford University Press, 2013, ch. 2.

[19] J. Englander, M. A. Vavrina, and D. Hinckley, "Global optimization of low-thrust interplanetary trajectories subject to operational constraints," *AASAIAA Space Flight Mechanics Meeting*, 2016.

[20] J. Sims, P. Finlayson, and E. Rinderle, "Implementation of a low-thrust trajectory optimization algorithm for preliminary design," (AIAA/AAS Astrodynamics specialist conference and exhibit), 2006.

[21] J Sims and S Flanagan, "Preliminary design of low thrust interplanetary trajectories," (AIAA/AAS Astrodynamics Specialist Conference), Girdwood, Alaska: AIAA, 1999.

[22] A. Mereta, D. Izzo, and A. Wittig, "Machine learning of optimal low-thrust transfers between near-earth objects," (In International Conference on Hybrid Artificial Intelligence Systems (pp. 543-553)), Springer, Cham, 2017.

[23] I. S. Grigoriev and M. P. Zapletin, "Choosing promising sequences of asteroids," *Automation and Remote Control*, vol. 74, 12841296, 2013.

[24] D. Izzo, L. F. Simes, C. H. Yam, F. Biscani, D. D. Lorenzo, B. Addis, and A. Cassioli, "GTOC5: Results from the european space agency and university of florence," *Acta Futura*, vol. 8, pp. 45–55, 2014.

[25] E. Mendez, P. Mishra, S. Edwards, and D. Mavris, "Response surface regressions for low-thrust interplanetary mission design," in *AIAA SPACE 2016*, Georgia Institute of Technology, 2016.

[26] J. Friedman, "Stochastic gradient boosting," *Computational Statistics Data Analysis*, vol. 38, pp. 367–378, 2002.

[27] J. H. Friedman, "Greedy function approximation: A gradient boosting machine.," *Annals of statistics*, pp. 1189–1232, 2001.

[28] D. E. Rumelhart, "Artificial neural networks: A tutorial," *Computer*, vol. 29, pp. 31–44, 1996.

[29] L. Dormehl, "What is an artificial neural network? heres everything you need to know," *Digital Trends*, 2019.

[30] S. B. Kotsiantis, I Zaharakis, and P Pintelas, "Supervised machine learning: A review of classification techniques," *Emerging artificial intelligence applications in computer engineering*, vol. 160, pp. 3–24, 2007.

[31] G. Tsirogiannis and V. Markellos, "A greedy global search algorithm for connecting unstable periodic orbits with low energy cost.," *Celestial Mechanics and Dynamical Astronomy*, vol. 117, no. 2, pp. 201–213, 2013.

[32] O. Schütze, M. Vasile, O. Junge, M. Dellnitz, and D. Izzo, "Designing optimal low-thrust gravity-assist trajectories using space pruning and a multi-objective approach," *Engineering Optimization*, vol. 41, no. 2, pp. 155–181, 2009.

[33] A. H. Gad, "Space trajectories optimization using variable-chromosome-length genetic algorithms," PhD Thesis, Michigan Tech, 2011.

[34] E. Jones, T. Oliphant, P. Peterson, *et al.*, *SciPy: Open source scientific tools for Python*, [Online; accessed April 2019], 2001–.

[35] T. Oliphant, *Guide to NumPy*. Jan. 2006.

[36] D. Izzo, "The simsflanagan module," *https://esa.github.io/pykep/documentation/simsflanagan.html*, Accessed 2018.

[37] D. Izzo, "Pygmo and pykep: Open source tools for massively parallel optimization in astrodynamics (the case of interplanetary trajectory optimization)," 5th International Conference on Astrodynamics Tools and Techniques (ICATT), 2012.

[38] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[39] G. Radice and G. Olmo, "Ant colony algorithms for two impluse interplanetary trajectory optimization," *Journal of guidance, control, and dynamics*, vol. 29, no. 6, pp. 1440–1444, 2006.

[40] e. a. Wolfgang B., *Genetic Programming An Introduction*. San Fransisco, CA: Morgan Kaufmann, 1998.

[41] R. C. Woolley and C. W. Whetsel, "On the nature of earth-mars porkchop plots," 2013.

[42] J. A. Sims, A. J. Staugler, and J. M. Longuski, "Trajectory options to pluto via gravity assists from venus, mars, and jupiter.," *Journal of Spacecraft and Rockets*, vol. 34, pp. 347–353, 1997.

[43] R. Woolley and C. Whetsel, "On the nature of earth-mars porkchop plots," 23rd AAS/AIAA Spaceflight Mechanics Meeting, Jet Propulsion Laboratory, 2013.

[44] Patel, G. P. R., A. D., Zurbuchen, S. T. H., and D. J., "An algorithm for generating feasible low thrust interplanetary trajectories," (Proceedings of theInternational Electric Propulsion Conference), Michigan, USA, 2009.

[45] S. Crepaldi, "Low-thrust trajectory design and attitude strategies in multi-body dynamical regimes," Master's Thesis, Polytechnic University of Milan, 2018.

[46] J. A. Englander and A. C. Englander, "Tuning monotonic basin hopping: Improving the efficiency of stochastic search as applied to low-thrust trajectory optimization," 2014.

[47] C. H. Yam, D. D. L., and D. Izzo, "Low-thrust trajectory design as a constrained global optimization problem," *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, vol. 22, pp. 1243–1251, 2011.

[48] D. C. Montgomery, *Design and Analysis of Experiments*. John Wiley Sons, Inc.

[49] T Polsgroe, L Kos, R Hopkins, and T Crane, "Comparison of performance predictions for new low-thrust trajectory tools," *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, vol. AIAA 2006-6742, 2006.

[50] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux, "API design for machine learning software: Experiences from the scikit-learn project," in *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, 2013, pp. 108–122.