# Machine Learning of Optimal Low-thrust Transfers between Near-Earth Objects

Alessio Mereta, Dario Izzo, and Alexander Wittig

European Space Agency
Advanced Concepts Team
Keplerlaan 1
2201AZ Noordwijk, The Netherlands
{Alessio.Mereta,Dario.Izzo,Alexander.Wittig}@esa.int

**Abstract.** During the initial phase of space trajectory planning and optimization, it is common to have to solve large dimensional global optimization problems. In particular continuous low-thrust propulsion is computationally very intensive to obtain optimal solutions. In this work, we investigate the application of machine learning regressors to estimate the final spacecraft mass $m_f$ after an optimal low-thrust transfer between two Near Earth Objects instead of solving the corresponding optimal control problem (OCP). Such low thrust transfers are of interest for several space missions currently being developed such as NASA's NEA Scout.
Previous work has shown machine learning to greatly improve the estimation accuracy in the case of short transfers within the main asteroid belt. We extend this work to cover also the more complicated case of multiple-revolution transfers in the near Earth regime. In the process, we reduce the general OCP of solving for $m_f$ to a much simpler OCP of determining the maximum initial spacecraft mass $m^*$ for which the transfer is feasible. This information, along with readily available information on the orbit geometries, is sufficient to learn the final mass $m_f$ for the same transfer starting with any initial mass $m_i$. This results in a significant reduction of the computational cost compared to solving the full OCP.

**Keywords:** machine learning, regression, astrodynamics, near Earth objects, low thrust transfers

## 1 Introduction

In the early phase of space missions design the careful selection of the trajectory of the spacecraft is paramount. The availability of an efficient trajectory determines many further mission parameters, and can enable missions otherwise not feasible due to restrictions on the available fuel mass or cost.

In order to obtain optimal trajectories compatible with the overall mission parameters, such as propulsion system and available fuel mass, a complex global optimization problem must be solved. It consists of a large discrete combinatorial

part, typically involving questions such as the selection of possible targets or flyby sequences, as well as a continuous part due to the optimal control of the spacecraft during the travel from one target to the next.

With impulsive chemical propulsion it is possible to approximate the effect of propulsion by discrete instantaneous changes in the spacecraft velocity. Combined with efficient methods for the solution of Lambert's problem (computing an inertial orbit connecting two points in space), this leads to an overall relatively compact search space that can be handled efficiently by current global optimizers to obtain good solutions also for quite challenging problems with reasonable computational effort [1].

Chemical propulsion, however, is not the most efficient propulsion method in terms of fuel consumption. Low-thrust methods such as electric propulsion have a much higher specific impulse, meaning they deliver a higher impulse per fuel mass and thus allow lighter spacecraft carrying less propellant. This advantage comes at the cost of very long time scales over which the acceleration is applied. Solar electric engines produce very little thrust (on the order of 1-100 mN) but are operated continuously over months.

This means that the optimal control problem (OCP) for moving from one target to the next also becomes continuous instead of discrete. The solution now consists of a function indicating the thrust direction as well as the thrust magnitude as a function of time. The optimal trajectory is the trajectory requiring the least amount of fuel mass to reach a target and match its velocity within a given transfer time.

There are several numerical and mathematical methods for solving this OCP. Direct methods transform the continuous problem into a non-linear programing problem by discretizing the trajectory into short arcs of constant thrust magnitude and direction and imposing appropriate boundary conditions [2] [3]. Indirect methods use a technique from control theory known as Pontryagin's principle [4], which transforms the problem into a two-point boundary value problem to be solved.

Unfortunately, both methods are computationally difficult to solve. In the context of the global trajectory optimization problem, an efficient way to estimate the fuel consumption of a given low-thrust transfer allows much faster evaluation of the feasibility of a given trajectory. Once a potentially feasible trajectory has been identified by the optimization process, it can then be refined using a full solver for the OCP. The goal is therefore to obtain the final mass $m_f$ of a spacecraft after a particular transfer as a function of the initial mass $m_i$, the initial and final positions $\boldsymbol{r}_i$ and $\boldsymbol{r}_f$, and velocities $\boldsymbol{v}_i$ and $\boldsymbol{v}_f$, and the transfer time $\Delta T$ without solving the associated OCP.

Previous work [5] has shown that machine learning techniques are very successful in estimating the optimal $m_f$ for relatively short transfer arcs (typically a quarter or half a turn) between asteroids in the main belt an order of magnitude better than the Lambert estimate in Eq. 3.

In this paper, we apply the same ideas to the regime of Near-Earth objects (NEOs). There are several important differences in this regime compared to the

main asteroid belt. NEOs have a significantly shorter period, around 1 year as opposed to about 3-6 years, due to their proximity to the Sun. Transfers between NEOs are thus not only short arcs but can be more complicated including several revolutions around the Sun. This results in a more complicated structure of the solutions to the OCP, which now feature several thrust and coast arcs with a pronounced bang-bang control structure (see Figure 1). Due to these differences, we will see that the simple indicators used in [5] are not applicable any more.

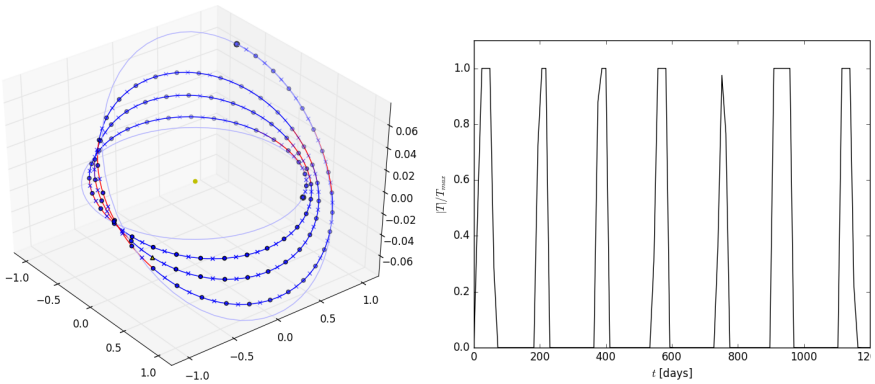


**Fig. 1.** Example of an optimal trajectory involving multiple revolutions (*left*) with thrust arcs highlighted in red, and the corresponding bang-bang control profile (*right*).

The rest of this paper is structured as follows: in Section 2 we describe the optimal control problem and the generation of a database of low-thrust trajectories between NEOs used in the training. Section 3 will present the algorithms and analyze the attributes used for the machine learning, the results of which are presented in Section 4. We conclude with some remarks and an outlook on possible future work in Section 5.

## 2 Generation of Optimal Low-thrust Trajectories

In this section we first give a short presentation of the spacecraft dynamics and the OCP we need to solve. We then proceed to describe our method for generating a database of optimal transfers used in the training.

For the remainder of this work, we assume a small spacecraft (CubeSat) of initial dry mass of $m_{sc} = 20$ kg and an electric low-thrust propulsion system with specific impulse $I_{sp} = 3000$ s and a maximum thrust of $T_{max} = 1.7$ mN. The mission of the spacecraft is to visit several NEOs starting from a parking orbit around Earth. The mission is similar to actual space missions currently being developed such as NASA's NEA Scout [7], although we use solar electric propulsion instead of solar sails.

### 2.1 Spacecraft Dynamics

The motion of the spacecraft is modeled as a restricted two-body problem of the spacecraft around the Sun. The spacecraft of initial mass $m_i$ is located at position $\boldsymbol{x}_i$ with velocity $\boldsymbol{v}_i$ at time $t_i$. We want to reach the final position $\boldsymbol{x}_f$ with velocity $\boldsymbol{v}_f$ at time $t_f$.

The equations of motion are given by

$$\begin{aligned} \ddot{\boldsymbol{x}} &= -\mu \frac{\boldsymbol{x}}{|\boldsymbol{x}|^2} + \boldsymbol{u}(t)/m, \\ \dot{m} &= -\frac{|\boldsymbol{u}(t)|}{I_{sp} g_0}, \end{aligned} \tag{1}$$

where $\mu \approx 1.327 \cdot 10^{20}$ m$^3$/s$^2$ is the gravitational parameter of the Sun, $I_{sp}$ is the specific impulse of the low-thrust engine, and $g_0 \approx 9.8066$ m/s$^2$ is the standard gravity on Earth. The control $\boldsymbol{u}(t)$ is the thrust vector. The goal of the OCP is to find the control $\boldsymbol{u}(t)$ minimizing

$$S = \int_{t_i}^{t_f} |\boldsymbol{u}(t)| \ dt, \tag{2}$$

subject to the constraints

$$\begin{aligned} \boldsymbol{x}(t_i) &= \boldsymbol{x}_i, \\ \dot{\boldsymbol{x}}(t_i) &= \boldsymbol{v}_i, \\ \boldsymbol{x}(t_f) &= \boldsymbol{x}_f, \\ \dot{\boldsymbol{x}}(t_f) &= \boldsymbol{v}_f, \\ |\boldsymbol{u}(t)| &\leqslant T_{max} \quad \forall t \in [t_i, t_f], \\ m_f &\geqslant 0. \end{aligned}$$

Note that we impose that the final mass $m_f = m(t_f)$ of the spacecraft at time $t_f$ must be positive, but no further restriction is placed on the amount of propellant on the spacecraft. Furthermore, note that minimizing Eq. 2 is equivalent to maximizing $m_f$.

To solve this problem, we use a direct collocation method based on the Sims-Flanagan model [3]. In this model the trajectory is discretized into a number of segments and on each segment the low thrust is approximated by a small impulsive transfer in the middle of the segment. Solving the resulting non-linear programming problem (NLP) using e.g. SNOPT [6] yields discrete values for the control $\boldsymbol{u}(t)$ along the trajectory, as well as the optimal final mass $m_f$.

While approximate analytical and heuristic expressions have been proposed to estimate $m_f$, they only work for short transfer arcs and can be quite inaccurate. The simplest method for estimating $m_f$ is assuming it the same as the impulsive $\Delta v$ from a Lambert transfer:

$$m_{f_L} = m_i \exp \frac{-\Delta V_L}{I_{sp} g_0}. \tag{3}$$

However, while already quite inaccurate in general, in our particular application to multi-revolution transfers this estimate will be shown to be practically useless.

### 2.2 Training Database Generation

We begin by selecting possible targets from the catalog of known asteroids. To obtain asteroids in an Earth-like orbit, we require that $0.8 < a < 1.2$, $i < 5°$, and $e < 0.4$. This yields 24 asteroids, and we also include Earth itself as the 25th object in the list. Figure 2 shows the orbits of all selected asteroids with the orbit of Earth.

To construct our database of training data, we start by randomly selecting two objects from the list of targets. Next, we select a random initial epoch $t_i$ between the year 2020 and 2025, as well as a transfer time $\Delta T = t_f - t_i$ in the interval $[60, 1500]$ days. No transfer is expected to be feasible below 60 days, and due to mission constraints we are not interested in transfer times longer than about 4 years.

With this information, we solve a simplified OCP to the original one given above. Instead of fixing the initial mass and maximizing for the final mass, we simply maximize the initial mass. This yields the maximum initial mass $m^*$ of a spacecraft that can still make the given transfer within the limitations of the propulsion system. Note that this problem is significantly easier to solve than the original OCP, since there will be no coast arcs. This fixes the thrust magnitude to $T_{max}$ (always on), and leaves only the thrust direction to be determined while removing all but one inequality constraint from the problem.

If $m^*$ is less than the dry mass $m_{sc}$ of our spacecraft, we discard this data point, since in that case obviously there is no possible transfer compatible with our mission requirements. Otherwise, we continue to generate entries in our

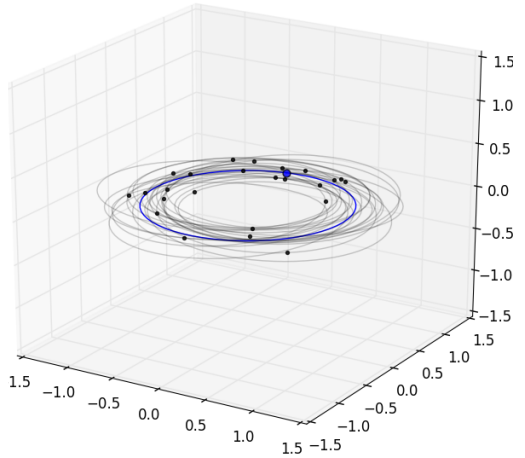


**Fig. 2.** Orbits of the selected asteroids for the mission (*black*) and Earth (*blue*).

database by dividing the interval $[0.3m^*, m^*]$ into four equal parts and selecting one random value from each. For each of these four initial masses $m_i$, we solve the initial OCP to obtain the optimal value of $m_f$ for that particular transfer. We then store the indices of the two asteroids $i_0$, $i_1$, the epoch and transfer time $t_i$ and $tof$, $m_i$ and $m_f$ as well as $m^*$ in the database.

In this fashion we generate a database of 60,000 examples of optimal transfers, from which we proceed to extract relevant features to train a set of standard regressors.

## 3 Machine Learning

In Figure 3 we visualize all the data points (trajectories) in the database. The left plot reports the ratio between the $\Delta V$ (which is directly related to the fuel mass) required for the optimal trajectory and the $\Delta V^*$ for the maximum initial mass trajectory. This quantity appears correlated with the ratio $m_i/m^*$, as is clear from the darker region of the plot. This finding is in agreement with the results in [5].

On the right side of Figure 3, the final mass $m_f$ is plotted against the maximum initial mass $m^*$ (*blue*), displaying again a clear correlation. Since this figure contains data points for any initial mass $m_i$, there is a wide band structure. Highlighting only data points corresponding to a narrow range of $m_i$ (*red*), here between 0.5 $m^*$ and 0.55 $m^*$, suggests a nearly linear correlation for fixed $m_i$.

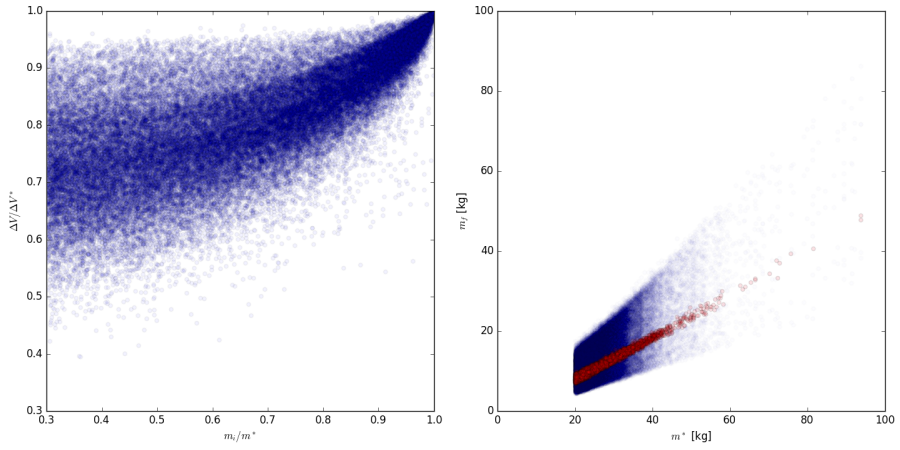


**Fig. 3.** Visualization of the training data.

### 3.1 Regression Algorithms

From the database of trajectories we randomly select 50,000 entries to use as the training set, and 10,000 as the test set. The split is performed not at the level of a single trajectory, but of a single choice of $\{i_1, i_2, t_i, \Delta T\}$. This ensures that trajectories differing only in $m_i$ are inserted together in either one of the two sets.

We train a set of regressors on the training set, using the Python Machine Learning library *scikit-learn* [8]. For each of them we evaluate the Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) on our test set. We use both base estimators (Decision Trees) and meta-estimators, namely Extra Trees, Random Forest, Gradient Boosting, Bagging and AdaBoost.

For ensemble methods the base estimators are Decision Trees and the number of estimators is set to 1000. The other parameters are optimized by grid search using 5-fold cross-validation. As a result, for Gradient Boosting we set a maximum depth of 6 and a learning rate of 0.1. For AdaBoost a square loss is used, with a learning rate of 1.2. All the other parameters are set to their default, since any change results in a degradation of performance.

We also experiment with a Neural Network regressor, implemented using the library Lasagne [9]. The architecture consists of 2 hidden layers of 80 units each with *ReLu* activations, and an output layer of a single linear unit. Training is performed with 500 epochs of Stochastic Gradient Descent using a batch size of 16 and a learning rate of $10^{-3}$.

### 3.2 Attributes

In order to perform regression to learn the optimal final mass $m_f$, domain knowledge suggests a number of possible attributes, listed in Table 1.

**Table 1.** Possible attributes considered to learn $m_f$

| Attribute | Description |
|---|---|
| $m_i$ | Initial mass |
| $\Delta T$ | Transfer time |
| $\boldsymbol{r}_i, \boldsymbol{v}_i$ | Starting position and velocity |
| $\boldsymbol{r}_f, \boldsymbol{v}_f$ | Arrival position and velocity |
| $\cos\theta$ | Cosine of the inclination between starting and arrival orbit |
| $\lvert\Delta a\rvert$ | Difference between semi-major axes of the orbits |
| $\lvert\Delta e\rvert$ | Difference between eccentricities of the orbits |
| $m^*$ | Maximum initial mass |
| $m_D^*$ | MIMA |
| $m_L^*$ | Maximum initial mass using the Lambert transfer |

This list includes some obvious choices: in order to solve the original OCP, knowledge of the initial mass $m_i$, the transfer time $\Delta T$, and the initial and final states $\boldsymbol{r}_i$, $\boldsymbol{v}_i$ and $\boldsymbol{r}_f$, $\boldsymbol{v}_f$ is strictly required.

We also include some other readily available orbital parameters. These relate directly to the geometrical shape of the departure and target orbits, and thus are expected to be a good measure of how different the two orbits are and hence how complicated the transfer is.

None of these attributes, however, encode any information on the actual OCP that needs to be solved. We therefore expect that it is necessary to also include some information related to the solution of the OCP, without, of course, actually specifying the solution. This is why we include the value $m^*$, the maximum initial mass making the specified transfer feasible. While this is not the solution $m_f$ to the full OCP we want to solve, as mentioned before we still need to solve a simpler reduced OCP to obtain $m^*$.

To avoid solving the reduced OCP, [5] introduces two approximations for $m^*$: the commonly used cost of the impulsive Lambert transfer $m_L^*$ given by Eq. 3, and the new Maximum Initial Mass Approximation (MIMA) $m_D^*$, a simple analytical expression that in their case results in a relevant improvement over $m_L^*$. We also include these in the possible set of attributes.

### 3.3 Feature selection

We perform a preliminary tree-based feature selection based on the Extra Trees Regressor from *scikit-learn*. Such analysis shows that, unsurprisingly, the most informative attribute is $m_i$, followed by $m^*$ and $\Delta T$. After that, also the three orbital parameters $\cos\theta$, $\Delta a$, and $\Delta e$ carry some relevant information.

The starting and arrival states, or any combination of them, does not contribute to the learning. While mathematically sufficient for solving the OCP (together with the constant maximum thrust $T_{max}$, $\Delta T$ and $m_i$), the relationship between those inputs and the final mass $m_f$ is highly non-linear and hence non-trivial.

More surprisingly, given their previous success, both approximators $m_L^*$ and $m_D^*$ turn out to be utterly uninformative as well. This result can be explained by the relatively short transfers considered in [5], usually requiring less than one revolution. For multi-revolution trajectories such as those considered in our case, neither one of the two measures has any correlation to $m^*$. This is particularly obvious when plotting both values against $m^*$ in Figure 4. It is therefore currently unavoidable to solve the simplified OCP for $m^*$ numerically each time when querying the model.

Based on this analysis, we select the following features to perform our experiments, while training each regressor to predict the target variable $m_f$:

$$\{\Delta T,\ m_i,\ m^*,\ cos\theta,\ |\Delta a|,\ |\Delta e|\}.$$

In order to investigate the impact of $m^*$ on the performance we also run the same experiments excluding it from the input features.

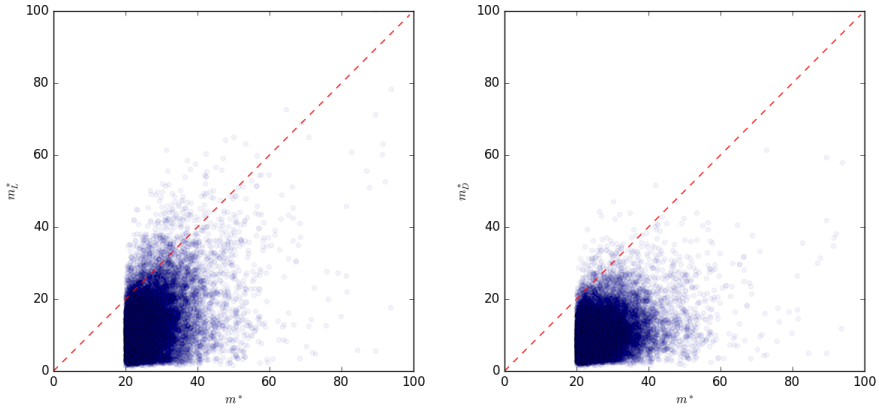


**Fig. 4.** Correlation of correct $m^*$ from OCP with estimated $m_L^*$ via Lambert (*left*) and $m_D^*$ via MIMA (*right*). The truncation of $m^*$ at 20 kg is due to the database generation described in Subsection 2.2.

## 4 Results

Our baseline for comparisons is the simple impulsive Lambert transfer cost estimate given in Eq. 3. Since our transfers include multi-revolution transfers, we compute the $\Delta V_L^i$ and corresponding $m_{f_L}^i$ for Lambert transfers with $i$ revolutions (in our case $i = 0, 1, \ldots, 5$) and take the minimum

$$m_{f_L} = \min_i m_{f_L}^i.$$

As expected, the regressors easily outperform this basic Lambert estimate, which is vastly wrong. Figure 5 shows a histogram of the error in the final mass $m_f$ (computed against the numerical solution of the OCP) over the test set for the best regressor (Gradient Boosting), the worst regressor (Decision Tree), and the Lambert baseline. It is clear that even the worst regressor outperforms the Lambert baseline estimate by an order of magnitude.

Table 2 shows the performance of each algorithm trained with its default parameters. Also quantitatively, it is clear that all learning algorithms improve on Lambert, whether $m^*$ is included in the attributes or not. It must be pointed out, however, that this is basically due to the horrible performance of the Lambert estimator for this problem.

Typical spacecraft such as the one considered in this work, carry on the order of $3 - 5$ kg of propellant. The average fuel consumption for the transfers in our database is 3.41 kg with a standard deviation of 1.45 kg. This means that even an error in the final mass of 0.3 kg after the transfer means a 10% error in the required fuel mass. This is why the improvement between the results including $m^*$ in the learning and those excluding it is very important. While it seems small compared to the Lambert baseline, the improvement relative to the typical transfer cost is still substantial.

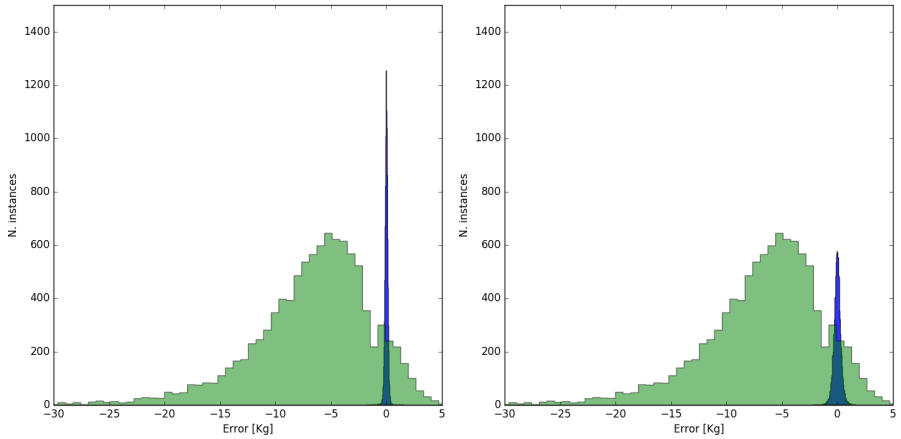


**Fig. 5.** The distribution of the prediction error for the best (*left, blue*) regressor Gradient Boosting and the worst regressor Decision Tree (*right, blue*). The Lambert baseline is shown in both cases for comparison (*green*).

**Table 2.** Performance comparison of different regressors and Lambert baseline

| | including $m^*$ | | excluding $m^*$ | |
|---|---|---|---|---|
| Algorithm | MAE [kg] | RMSE [kg] | MAE [kg] | RMSE [kg] |
| Decision Tree | 0.236 | 0.315 | 0.477 | 0.742 |
| Extra Trees | 0.147 | 0.195 | 0.425 | 0.596 |
| Random Forest | 0.154 | 0.205 | 0.376 | 0.533 |
| Gradient Boosting | 0.097 | 0.134 | 0.231 | 0.347 |
| Bagging | 0.154 | 0.205 | 0.376 | 0.533 |
| AdaBoost | 0.149 | 0.201 | 0.315 | 0.454 |
| Neural Network | 0.120 | 0.161 | 0.328 | 0.469 |
| *Lambert's predictor* | *6.98* | *8.76* | *6.98* | *8.76* |

## 5 Conclusions

We have shown that also in the case of multi-revolution transfers between NEOs the machine learning approach is vastly superior to the commonly used impulsive Lambert estimate. We did not perform any particular tuning of the parameters of the learning algorithms to achieve this result.

To obtain realistically usable results, however, we were not able to eliminate entirely the need for solving an OCP. But the problem is reduced to solving a significantly easier OCP for $m^*$ instead of the direct solution of $m_f$.

In future work we propose to remove this requirement by training a more sophisticated regressor to also learn the full solution of the OCP without the need of $m^*$.

## References

1. Izzo, D., Becerra, V. M., Myatt, D. R., Nasuto, S. J., Bishop, J. M.: Search space pruning and global optimisation of multiple gravity assist spacecraft trajectories. J. Glob. Opt. 38(2), 283-296 (2007)
2. Izzo, D.: PyGMO and PyKEP: open source tools for massively parallel optimization in astrodynamics (the case of interplanetary trajectory optimization). In: Proceed. Fifth International Conf. Astrodynam. Tools and Techniques, ICATT (2012)
3. Sims, J. A., Flanagan, S. N.: Preliminary design of low-thrust interplanetary missions. In: AAS/AIAA Astrodynamic Specialist Conference (1999)
4. Pontryagin, L. S., Boltyanskii, V. G., Gamkrelidze, R. V., Mishchenko, E. F.: The Mathematical Theory of Optimal Processes. English translation. Interscience (1962)
5. Hennes, D., Izzo, D., Landau, D.: Fast approximators for optimal low-thrust hops between main belt asteroids. In: IEEE Symposium Series on Computational Intelligence (2016)
6. Gill, P. E., Murray, W., Saunders, M. A.: SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization. In: SIAM Review 47:1, 99–131 (2005)
7. NASA NEA Scout `https://www.nasa.gov/content/nea-scout/`
8. Pedregosa, F. et al.: Scikit-learn: Machine Learning in Python. In: Journal of Machine Learning Research (12), 2825–2830 (2011)
9. Dieleman, S. et al.: Lasagne: First release. `http://dx.doi.org/10.5281/zenodo.27878` (2015)