Improving the Evolutionary Optimization of Interplanetary Low-Thrust Trajectories Using a Neural Network Surrogate Model

Stubbig, L.J.; Cowan, K.J.

Publication date
2021

Document Version
Final published version

Published in
Advances in the Astronautical Sciences

Citation (APA)
Stubbig, L. J., & Cowan, K. J. (2021). Improving the Evolutionary Optimization of Interplanetary Low-Thrust Trajectories Using a Neural Network Surrogate Model. In R. Wilson, J. Shan, K. Howell, & F. Hoots (Eds.), *Advances in the Astronautical Sciences* (Vol. 175). Article AAS 20-658 (Advances in the Astronautical Sciences; Vol. 175).

AAS 20-658

IMPROVING THE EVOLUTIONARY OPTIMIZATION OF INTERPLANETARY LOW-THRUST TRAJECTORIES USING A NEURAL NETWORK SURROGATE MODEL

Leon Stubbig* **and Kevin Cowan†**

Building on recent advances in the fields of low-thrust trajectory optimization based on shaping methods, Artificial Neural Networks, and surrogate models in Evolutionary Algorithms, an investigation into a novel optimization routine is conducted. A flexible Python tool to evaluate linked trajectories in a two-body model based on hodographic shaping is implemented and used to develop a novel evolutionary optimization approach where a Genetic Algorithm is assisted in finding new candidate solutions by an online surrogate. The algorithm and different surrogate designs are experimentally investigated on two example problems based on the Dawn trajectory and the GTOC2 problem. Employing the surrogate yields new candidate solutions that improve the population's fitness especially when the surrogate is used to approximate the shaping computation. Additionally, the use of a surrogate pretrained on a general data set of low-thrust transfers is tested and found to considerably improve the initial quality of the model, meaning that more good candidate solutions are found early on, accelerating the algorithm's convergence.

INTRODUCTION

Low-thrust propulsion systems have gained popularity in recent years not only for Earth-orbiting satellites but also for interplanetary missions where their increased efficiency enables missions previously not feasible. Examples include NASA's Dawn mission which concluded in 2018 after a post-launch velocity increment, ΔV, of almost 11 km s^{-1}, the highest of any spacecraft flown to date.[1] The design of low-thrust trajectories is more complex than for chemically propelled spacecraft because low-thrust engines have to be operated almost continuously to accumulate the required ΔV. The change in velocity can therefore no longer be approximated as a discrete event and continuous methods mainly derived from optimal control theory have to be employed.[2] As these methods are computationally intensive and require a good initial guess for convergence, so-called shaping methods have been developed specifically for the preliminary design phase. The first shape that was recognized to be a suitable representation of a low-thrust transfer was the exponential sinusoid,[3] followed by a variety of other shapes and methods. Shaping methods efficiently generate transfers by assuming an analytical function that satisfies the applicable boundary conditions and evaluating the dynamics of the problem afterwards, avoiding excessive iterations and computation time. The generated transfers are inherently sub-optimal but are very useful as an initial guess for local optimization and, depending on the chosen shaping function, have been shown to yield realistic thrust profiles well suited for a preliminary exploration of the design space.

The ambitious nature of modern space missions, which often involve multiple flybys, target bodies, and science goals, further complicates the preliminary mission design phase. Multiple trajectory phases have to be linked, greatly increasing the number of mission concepts that have to be considered. The total mission ΔV then depends on each phase being flown as efficiently as possible and every improvement in one phase allows for an increase in the total science return, e.g. by increasing the length of scientific observations at another

* MSc Graduate, Faculty of Aerospace Engineering, Delft University of Technology, The Netherlands.
E-mail: leon.stubbig@gmail.com. The manuscript for this paper was part of the author's MSc thesis.
† Education Fellow + Lecturer, Faculty of Aerospace Engineering, Delft University of Technology, The Netherlands.
E-mail: k.j.cowan@tudelft.nl.

target. A variety of evolutionary optimization algorithms are successful in finding global optima.[4,5] This class of algorithms does not require gradient information and is built around populations of candidate solutions that are evaluated and 'evolved' to create new, more capable solutions. By balancing local and global exploration chances are high that the global optimum can be found. Here, a classical Genetic Algorithm (GA) is employed to develop the approach, which can be extended to other population-based algorithms in future work.

The goal of this work is to improve the preliminary optimization of linked low-thrust trajectories by including an approximate model in evolutionary optimization. This approach is known in literature as surrogate modeling.[6] The main idea is to make use of previous fitness evaluations by constructing an approximation of the computationally expensive fitness function. The model can potentially be used at a variety of places in the optimization algorithm, the most straightforward being the replacement of the original, costly fitness function for parts of the evolution. Another possibility is to evaluate the surrogate for new candidate solutions to be fed back into the original population. There are many options to construct the surrogate, ranging from simple linear fits to intricate statistical models. Here, we investigate the use of Artificial Neural Networks (ANNs). In the wake of the recent boom in Machine Learning (ML) ANNs have been heavily developed[7,8] and represent a promising approach to extract the maximum of information from the available data set.

The paper is structured as follows: First, the astrodynamical model used to represent the individual low-thrust transfers and the approach to building linked trajectories are introduced. Then, the data-driven surrogate approach based on ANNs is presented. The global optimization algorithm is shown in the following section, showcasing several possible applications for the surrogate. Finally, the constructed algorithm is applied to two example problems and conclusions are drawn.

ASTRODYNAMICS MODEL

In this work, the hodographic shaping method[9] is employed to represent interplanetary low-thrust transfers. This method was developed to facilitate the rapid, preliminary computation of low-thrust trajectories. In comparison to other shaping methods, it is particularly suited for the development of three-dimensional (3D), linked, trajectories because it allows to satisfy 3D boundary conditions at departure and arrival on position, velocity, and Time of Flight (ToF). It therefore enables the straightforward construction of multi-phase trajectories when combined with functions to evaluate ephemerides and flybys in a linked conics approach, see the example trajectory in Figure 1. We implemented the time-driven version[9] of the shaping method in a software tool for preliminary low-thrust mission design that was written in Python to provide an interface to the available stack of optimization and data science/ML packages. This section presents the implemented astrodynamics models that can be flexibly combined to build linked trajectories.

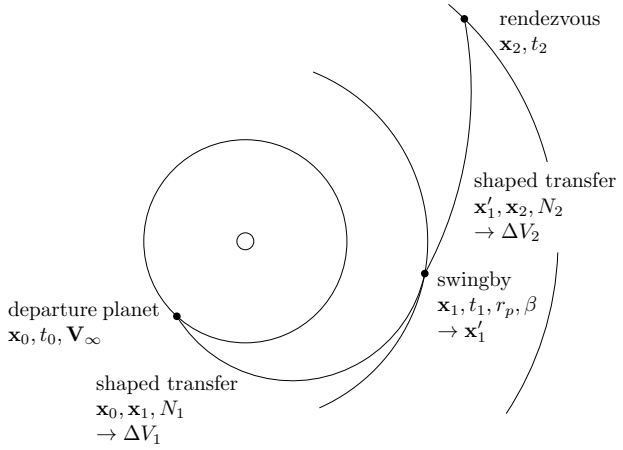

Figure 1. An example trajectory with one flyby and rendezvous

In hodographic shaping, the low-thrust trajectory is represented by three velocity functions in cylindrical coordinates. Each of these velocity functions $V_\square$ describes the velocity's time evolution in this direction ($\mathbf{e}_r$, $\mathbf{e}_\theta$, or $\mathbf{e}_z$) and is assembled by a variable number of analytically integrable and differentiable base functions v_i which are multiplied by shape coefficients c_i:

$$V_\square = \sum_{i=1}^{n} c_i v_i(t), \tag{1}$$

where t is time and n is the number of base functions. Radius r, vertical distance z, and polar angle θ can be computed from the respective base function by analytical integration. Similarly, the respective accelerations are calculated by analytically differentiating the base function. The values of 3 coefficients per velocity function are determined from the boundary conditions on position, velocity, and ToF, while any additional free parameters have to be chosen by an optimization algorithm. Here, we follow the original implementation[9] in using a total of 6 free parameters, i.e. $n = 5$, and the recommended base functions*. The hodographic shaping method then computes the ΔV necessary to fly a transfer assuming a two-body system and taking the initial and final state vectors, the number of heliocentric revolutions N, and the ToF as input.

The orbital models are kept simple and follow the capabilities (and limitations) of the chosen shaping method in a linked conics approach. The orbit's initial characteristic energy C_3 provided by the launch vehicle is applied to the departure planet's velocity $\mathbf{V}_\mathrm{p}$ to compute the initial velocity condition of the interplanetary transfer $\mathbf{V}_1$:

$$V_\infty = \sqrt{C_3}, \qquad \mathbf{V}_\infty = V_\infty \begin{pmatrix} \sin\varphi\cos\theta_1 \\ \sin\varphi\sin\theta_1 \\ \cos\varphi \end{pmatrix}, \qquad \mathbf{V}_1 = \mathbf{V}_\mathrm{p} + \mathbf{V}_\infty, \tag{2}$$

where V_∞ is the spacecraft's hyperbolic excess velocity and azimuthal angle φ and polar angle θ_1 denote the impulsive shot's direction in spherical coordinates attached to the spacecraft. As the Sphere of Influence (SOI) is small in comparison to the scale of interplanetary trajectories, its size is assumed to be negligible. Additionally, the launch vehicle's burn time is assumed to be short, leading to the used approximation where any V_∞ is applied instantly as an impulsive shot at the departure position.

Planetary flyby maneuvers are implemented in a similar fashion.[10] The flyby planet's SOI is assumed to be small and the change in velocity is assumed to happen over a short time. To compute the spacecraft's velocity after the flyby its heliocentric velocity in Cartesian coordinates $\mathbf{V}_\mathrm{in}$ at the location of the flyby is projected into the planetocentric reference frame:[11]

$$\tilde{\mathbf{V}}_\mathrm{in} = \mathbf{V}_\mathrm{in} - \mathbf{V}_\mathrm{p}, \tag{3}$$

where the tilde denotes velocities in the Cartesian planetocentric frame. The angle between incoming and outgoing planetocentric velocity, the turn angle δ, is then computed from the eccentricity of the flyby hyperbola e:

$$e = 1 + \frac{r_\mathrm{p}}{\mu_\mathrm{p}} \tilde{\mathbf{V}}_\mathrm{in}^2, \qquad \delta = 2\arcsin\left(\frac{1}{e}\right), \tag{4}$$

where r_p is the periapsis of the flyby hyperbola and μ_p is the planet's gravitational parameter. The planetocentric outgoing velocity $\tilde{V}_\mathrm{out}$ is then computed from the turn angle and the orientation of the flyby plane β, the latter being an input parameter that is measured in the B-plane between the aiming point B, the planet, and a plane parallel to the ecliptic passing through the planet:[11]

$$\tilde{\mathbf{V}}_\mathrm{out} = \tilde{V}_\mathrm{in}\left(\mathbf{i}\cos\delta + \mathbf{j}\cos\beta\sin\delta + \mathbf{k}\sin\beta\sin\delta\right), \tag{5}$$

where $\mathbf{i}$, $\mathbf{j}$, and $\mathbf{k}$ are unit vectors defined as:

$$\mathbf{i} = \frac{\tilde{\mathbf{V}}}{|\tilde{\mathbf{V}}|}, \qquad \mathbf{j} = \frac{\mathbf{i}\times\mathbf{V}_\mathrm{p}}{|\mathbf{i}\times\mathbf{V}_\mathrm{p}|}, \quad \text{and} \quad \mathbf{k} = \mathbf{i}\times\mathbf{j}. \tag{6}$$

*$V_r = V_\theta =$ C Pow Pow2 PSin05 PCos05 and $V_z =$ CosR5 P3CosR5 P3SinR5 P4CosR5 P4SinR5

Finally, the outgoing velocity is projected back into the heliocentric reference frame:

$$\mathbf{V}_{\mathrm{out}} = \tilde{\mathbf{V}}_{\mathrm{out}} + \mathbf{V}_{\mathrm{p}}, \tag{7}$$

yielding the spacecraft's Cartesian velocity in the heliocentric frame $\mathbf{V}_{\mathrm{out}}$. Powered flybys are not considered as the low-thrust propulsion system is assumed to have a negligible effect on the spacecraft's state during the relatively short duration of the flyby.

These models are complemented by functions to convert positions, velocities, and state vectors between the cylindrical reference frame used during shaping and the Cartesian frame employed for flybys and impulsive shots. Ephemerides are pulled from NASA's planetary data system[12] using ESA's Pykep package*. Dwell times at planetary bodies are represented by an *orbit following* behavior. Here, the spacecraft is assigned the same state vector as the body it is visiting while not expending any energy.

MACHINE LEARNING MODEL

During the evolution of a population-based optimization algorithm the fitness function is evaluated many times. In the conventional use case these data points get discarded as soon as a population member is replaced. Some algorithms exist where the use of a history of previous evaluations has been proved to be beneficial. For example, in Particle Swarm Optimization (PSO) each particle keeps track of the best previously visited position in the search space which is then used to guide its path in the following generations. The classical GA, as well as many other optimization algorithms, does not employ such a feature and we manually construct a surrogate model to make use of the data points generated by the continuous evaluation of the fitness function.

As mentioned before, there is a variety of ways to construct a model from data. Previous research on surrogate assisted optimization has mainly been done on polynomial models, Gaussian Process Regression (Kriging), Artificial Neural Networks (ANNs), and Support Vector Machines.[13] After fitting the model to the available data set it provides a means to approximate the fitness function of new solution vectors. In this section, the approach to building this model is described, while the following section showcases the optimization approach and the model's use case therein.

A data set generated by the application of the astrodynamics model described in the previous section will generally consist of a set of parameters describing the trajectory (Departure and arrival dates, V_∞, ToF, flyby parameters) related to figures of merit (ΔV_i, $\Delta V_{\mathrm{total}}$, max. thrust). While it is necessary to reduce the size of the input vector as much as possible for the optimization run, i.e. to avoid redundant inputs, this is not automatically true for the ML model. For example, a linked trajectory's timing is efficiently encoded using a departure date and the ToFs of subsequent phases, meaning that the first transfer's arrival date coincides with the second transfer's departure date. The initial and final state vectors are then derived from ephemerides and possibly impulsive shots or the flyby model linking the transfers. The ML model may however benefit from the full state vectors as inputs as the conversion from date to state vector is non-trivial but computationally fast. We therefore experimented with building predictive models for different inputs sets, termed *feature engineering* in ML.

The most straightforward model is trained directly on the optimization's parameter vector. In case of the first example problem, which is derived from the Dawn mission, the parameter vector consists of the launch date, the initial V_∞, the arrival velocity at the flyby planet, the flyby distance and plane angle, the ToF of each transfer as well as the duration of the stay at Vesta (11 real inputs). Evaluating the trajectory, i.e. computing the optimization problem's fitness function, yields a total ΔV necessary to fly this mission. An initial data set to develop the model is created by running a random search using uniformly distributed input parameters from the set bounds. This data set is split into training, validation, and test sets, where training and validation sets are employed during model creation and the model's performance is evaluated on the test set. This approach aims to avoid overfitting to any of the individual data sets and to measure generalization ability on previously unseen data.[14]

*D. Izzo, esa/pykep, Version 2.3, 2019, `https://esa.github.io/pykep/`

To assess a model's predictive quality, two figures of merit are computed. The first one is Mean Absolute Percentage Error (MAPE):

$$\mathrm{MAPE} = \frac{100\%}{n} \sum_{i=1}^{n} \left| \frac{a_i - f_i}{a_i} \right|, \tag{8}$$

where f_i are predictions, a_i are the true values and n is the number of test samples.

The second figure of merit is derived from the models' application: In order to guide a population-based algorithm, the model needs to distinguish good from bad candidate solutions more than accurately predict the correct output value to guide the algorithm towards convergence. Thus, we define a *sorting accuracy* which represents the ratio of samples whose prediction correctly placed them at the top of the test set. In the following, we use a measure of "Top 25 out of 100" meaning that the reported sorting accuracies are the share of the best 25 samples that the predictive model correctly placed in the top 25 out of a total of 100 test samples.

The features in the data set have different ranges and units, necessitating scaling.[14] For example, the departure date is given as a Modified Julian Date 2000, while the flyby plane's angle is given in radians. Scaling is therefore necessary to generate model coefficients of a similar scale, which enables faster learning in ANNs.[14] The two standard options for scaling are Standardization and Normalization: Standardization removes the mean and scales the data to unit variance. This approach presumes that the data follows a normal distribution which cannot be guaranteed here. It was therefore chosen to use the other option of linear normalization to input and output features:

$$x_{\mathrm{scaled}} = \frac{x - x_{\min}}{x_{\max} - x_{\min}} (\max - \min) + \min, \tag{9}$$

where $x_{\min}$ and $x_{\max}$ are the per-feature minimum and maximum values, respectively, and $[\min, \max]$ is the range of the scaled data. The input features are scaled to a range of [0, 1], and the output values are scaled to [-1, 1]. The values for $x_{\min}$ and $x_{\max}$ can be automatically extracted from the data set or manually set. The latter was done if the range of input features was known a priori from the bounds of the optimization problem. For cases where the bounds of variables are not known the maximum and minimum values are determined from the data itself.

A linear regression model* is fit to subsets of the initial data set using a least-squares fit and serves as a baseline for more sophisticated models. The performance of the linear regression model is shown in Figure 2. It can be seen that linear regression provides a poor fit, even for a large number of samples. The maximum achieved sorting accuracy is 60 %, while the MAPE of the regression reaches 37 % for a sample size of 2000. Note that the reported samples size is the full data set which was split into training and validation sets at a ratio of 80/20. Regarding the model's performance, it is especially of interest that the model does not improve after increasing the number of training samples beyond 500, clearly indicating that a linear model is not sufficient for the data at hand. The linear relationship is simply not flexible enough to represent the non-linear nature of this problem.

In the following, we therefore investigate the use of Artificial Neural Networks (ANNs) on the given regression problem. ANNs have a long history dating back to the 1950s but received a renewed interest in the past decade when advances in training and initialization algorithms made it possible to train deep networks consisting of multiple hidden layers.[7] Neural Networks now represent the best solution to many problems in ML ranging from pattern recognition to natural language processing, with a variety of specialized flavors and architectures. They have also been used in the area of surrogate modeling,[16] where they are recommended for high-dimensional problems with limited amounts of data.[13] In comparison to other methods, they are very flexible in input and output shapes, size, and computational effort. Here, we rely on a classical fully-connected, feed-forward ANN to perform the non-linear regression task of estimating the necessary ΔV to fly a certain trajectory. A feed-forward ANN consists of an input layer, a variable number of hidden layers, and an output layer. The number of nodes, called neurons, in the input and output layers is determined by the regression problem, while the number of neurons in the hidden layers is flexible. In recent years a number of software

*Normalization and the linear model have been implemented using the scikit-learn package (MinMaxScaler and LinearRegression)[15]

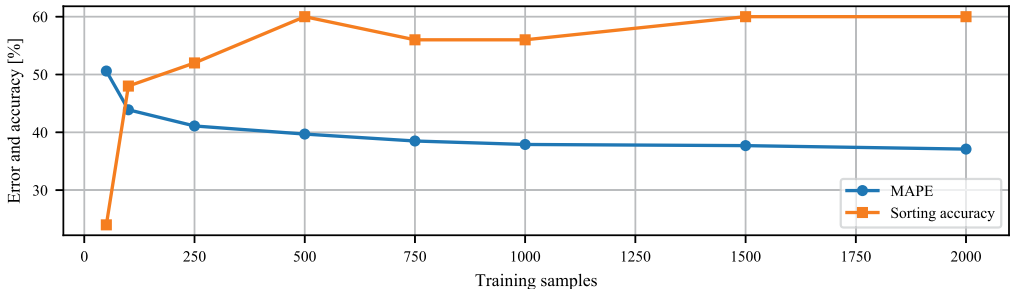

Figure 2. Regression accuracy of a linear regression model

frameworks have been published that allow for a fast implementation of ANNs. Here, we use the Keras API* in combination with the Theano backend†.

In order to define a suitable network architecture, several parameters were set a priori based on literature and preliminary experiments. Firstly, the search for a suitable ANN was restricted to feed-forward neural networks with a low to medium number of layers. As the task is a non-linear regression on a comparatively small, structured data set, there is no need for a specialized deep architecture. Keeping the number of free parameters in the network to a reasonable size also affects the time needed for training and is one of the measures taken against overfitting, which is a common problem especially when training a large network on a limited amount of data. The network's weights are randomly initialized using the uniform Glorot method, where each weight is drawn from a uniform distribution between limits depending on the number of neurons in the connected layers.[17] Each bias is initialized as zero.

The Rectified Linear Unit (ReLU) has been shown to learn faster than other activation functions[7] and is used in all hidden layers. The activation of the output layer is chosen to be linear, in order to not restrict the range of the regression output. The ANN is trained using the Adaptive Moment Estimation (Adam) optimizer.[18] It is used with a Mean Squared Error (MSE) loss, a batch size of 10, and an 80/20 split of training and validation data. To combat overfitting we employ early stopping with a patience of 130 epochs‡, meaning that training is stopped once the validation error does not improve for more than 130 epochs. The returned model is then the one that exhibited the minimal validation error. The learning rate is reduced periodically employing a similar heuristic: If the validation loss does not improve for more than 50 epochs the learning rate is reduced by 80 %, augmenting local convergence during training. Reducing the learning rate when the loss plateaus also allows to start training at a learning rate of 5×10^{-3}, which is higher than the default of 1×10^{-3} and accelerates the initial learning.

Finally, the relationship between the number of hidden layers, neurons per hidden layer and the size of the training data set is investigated in a grid search. We vary the ANN architecture from 1 to 5 hidden layers of size 16 to 128. The size of the data set is increased from 50 to 1500 samples, showing the performance gain that can be expected when more data becomes available. Each architecture was trained 5 times from random initial states with different seeds and averaged to show performance and training time. Looking at the MAPE and sorting accuracy shown in Figure 3 it is apparent that more data is the single most important factor in increasing the model's performance. In comparison to the linear model, the ANN is able to heavily profit from more data and reaches MAPEs below 15 % for most medium and large architectures. ANNs with 1 and 2 hidden layers perform worse in terms of MAPE while deeper architectures perform fairly similar. The number of hidden layers had a bigger effect on performance than the number of neurons per layer. It was decided to perform most experiments with an ANN of 3 hidden layers with 64 neurons each (3/64). Staying at the lower

*F. Chollet et al., Keras, Version 2.2.4, 2019, `https://keras.io`

†Theano Development Team, Theano, Version 1.0.3, 2019, `http://deeplearning.net/software/theano/`

‡An epoch has passed once the ANN was trained on each training example once.

end of possible network sizes significantly reduces the time needed for training to converge and is therefore beneficial for the setting at hand. For example, using 1000 samples the 3/64 network concluded training on average after 44 s while a bigger 3/128 network needed about 5 times as long (199 s*) to converge. During online use of the surrogate, the ANN was trained from scratch for each new data set. In practice, it is also possible to start with a trained model and only train a few more epochs when new data becomes available. As the training is already performed in batches and new data comes from the same distribution the network then increases its performance. This approach is especially valuable for larger data sets and was employed when a pretrained model was used during the optimization.

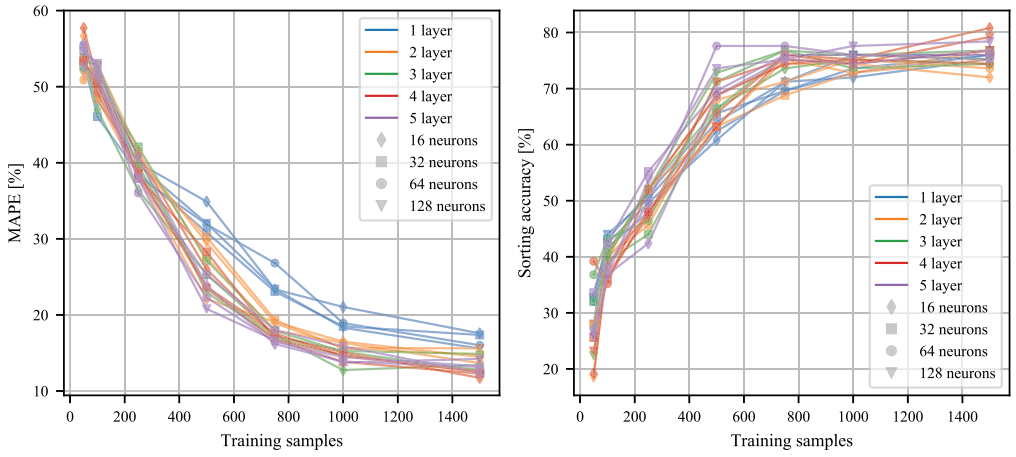

Figure 3. Resulting regression errors and sorting accuracy of the ANN architecture grid search

After choosing the 3/64 architecture, we test the model's robustness with regard to the fixed parameters. This is done by varying batch size and learning rate around the chosen values. The results are reported in terms of regression error and sorting accuracy in Figure 4. It can be seen that the chosen values for the initial learning rate (5×10^{-3}) and batch size (10) perform well especially for sample sizes of 500 and above. A bigger batch size or smaller learning rate would overall reduce the accuracy of the trained model. For smaller data sets a small gain in accuracy could be gained by smaller batch sizes. Here, we stick to one architecture in order to keep the approach consistent.

As shown in the following, it can also be beneficial to train on other features in the data set than the plain candidate solutions. For example, the relationship between the date and a planet's state is non-linear but can be implemented as a fast lookup of tabulated ephemeris data. Training directly on state vectors can therefore simplify the estimation problem, thus reducing the amount of necessary training data, while requiring very little computational effort. This approach is taken when the ANN is used to estimate the cost of a single transfer. Here, the initial and final state vectors in cylindrical coordinates as well as the ToF are chosen as inputs. This way we include the prior knowledge of the main features that determine the cost of a transfer. It is also beneficial that the cylindrical state is nearly periodic over the course of an orbit, meaning the model mainly has to interpolate between known values. The polar angle θ is encoded as $\cos\theta$ and $\sin\theta$, in order to provide two continuous input features. As the number of input/output parameters and the type of problem are very similar we are using the same ANN architecture found above for these related problems as well.

In conclusion, using Artificial Neural Networks to build non-linear regression models is a flexible and performant method that easily outperforms the baseline linear method, c.f. Figures 2 and 3, and is able to quickly improve when more data is available.

*All reported computation times are from a current laptop (CPU: Intel Core i7-7700HQ@2.8 GHz, GPU: Nvidia Quadro M1200)

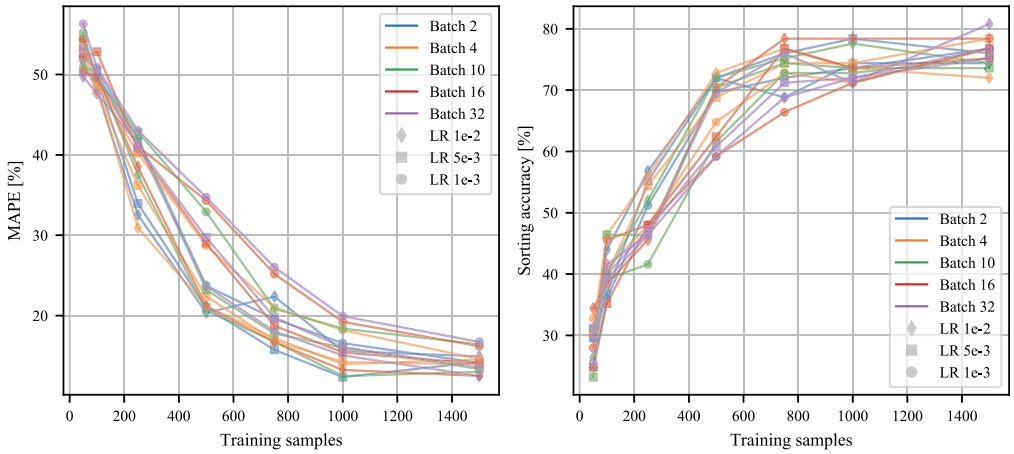

Figure 4. Investigation of the chosen ANN architecture's robustness for variations in learning rate and batch size

OPTIMIZATION APPROACH

The optimization process is structured in two nested loops. The inner loop optimizes the free parameters of the shaping functions defining each individual low-thrust transfer. Here, the Nelder-Mead simplex algorithm was chosen after several algorithms were tested. The outer loop performs the global optimization, i.e. setting the departure date, flight times, and parameters that define the linked trajectory. There, a classical GA was chosen and used to test different ways to include the ML model. Both optimization loops are presented here.

Shape Optimization

The shaping method takes the initial and final state vectors that correspond to a given departure date, ToF, and N as inputs. It then computes the trajectory's ΔV. As mentioned above, we chose to employ two Degrees of Freedom (DoF) in each velocity function, totaling six free parameters. These parameters are determined by a Nelder-Mead simplex algorithm (NM), following the original paper which investigated the use of the Differential Evolution (DE) and NM algorithms, and found that the latter performs favorably.[9] This result was confirmed in the preparations for this work where 10 global and derivative-free local optimization algorithms were run on a representative example transfer (Earth→Mars, $N = 2$, departure date = 9985..10075 mjd2000, ToF = 1010..1100 days). It was found that the problem is well suited for local optimizers,[19] especially as the lowest-order solution represents a suitable initial guess that is always available. NM consistently found the optimum in fewer fitness evaluations than similar algorithms. It was therefore decided to stick to NM to determine the free parameters in the hodographic shaping method.

The NM stopping criterion was set to a relative fitness tolerance of 10^{-3} and a maximum number of fitness evaluations of 10^4. Using an initial guess of 0 for each of the six free parameters, corresponding to the lowest-order solution that already satisfies the boundary conditions, the optimizer converges within 400 to 1000 iterations, not triggering the limit of function evaluations. The optimization of an individual transfer takes about 0.15 s to 0.3 s running on a single thread.

Global Trajectory Optimization

A GA was chosen to develop the surrogate assisted optimization tool. As the surrogate is applied on the level of individual fitness function evaluations the approach is also applicable to more advanced algorithms while having a straightforward behavior simplifies the development and serves as a baseline for future extensions.

GAs have successfully been applied to the optimization of shaped, low-thrust trajectories for example by Wall and Conway.[20] Extended versions using a surrogate model have not yet been used in the context of shaped trajectory optimization. We use the GA provided by Pygmo* which is "rather classical", encoding the parameter vector in a so-called chromosome, and using selection, crossover, and mutation operations in combination with an elitist reinsertion strategy: After the genetic operators have been applied the combined pool of parent and offspring candidate solutions is sorted and the best are selected to form the new population.

The GA has several control parameters. Similar to the ML model search a number of these have been chosen a priori while other have been explored by running the algorithm on an example problem with different settings: The selection scheme is a 'tournament' with a size of 2. This means that the algorithm chooses the best out of two random parent individuals for reproduction. The crossover method is 'exponential' with a probability of 0.9. In this method, a random point is selected in the parent chromosome and the partner genes are inserted from there with the set probability until it stops. Both of these choices follow the default settings in Pygmo. The mutation scheme was set to 'uniform' to increase the population's diversity in comparison to for example drawing mutations from a Gaussian distribution.

Suitable settings for mutation probability and population size were found by averaging 5 runs of the first example problem, see the Dawn Problem below, each with different settings to show the expected convergence behavior. The results are shown in Figure 5, including a random search for comparison. The random search was set up drawing random values from uniform distributions within the problem bounds to explore the search space and to generate the data set used to develop the model. The number of generations computed for each setting was adapted to compute 3000 evaluations of the fitness function in total. Similar to the GA the shown plot is the average of 5 runs, taking the best individual solution as the optimum which gets updated as the random search progresses. It can be seen that small population sizes, as well as high mutation probabilities, are beneficial for the GA to converge. We therefore use a population size of 10 and a mutation probability of 10 % in the outer loop of the Dawn example problem. The same analysis was done for the second example, the Asteroid problem, leading to a population size of 50 and a mutation probability of 10 %.

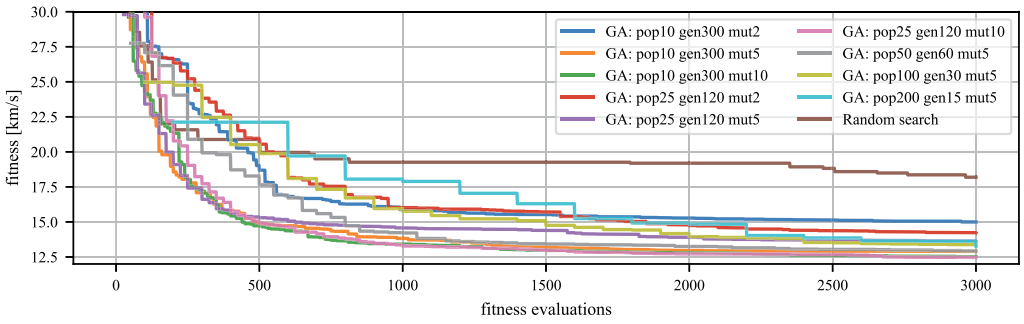

Figure 5. Comparison of different GA settings for population size and mutation probability, average of 5 runs each, optimizing the Dawn Problem

Including the Surrogate, Model Management, and Evolution Control

The GA converges to a good solution by combining individuals from the current generation to create new candidate solutions and keeping a random element in the crossover and mutation steps in order to globally explore the search space. As only better solutions are reinserted into the population the top solution improves over the generations. An interesting idea to improve the search for an optimum is to keep track of previous function evaluations and to build an approximate model using ML techniques, for example, based on the ANN presented earlier.[6,13,21] This approximate model is often called *surrogate* and makes it possible to utilize the

*F. Biscani and D. Izzo, esa/pagmo2, Version 2.10, 2019, https://doi.org/10.5281/zenodo.1045336

information generated from previous function evaluations during the optimization, for example, to speed up convergence or to reduce the computational burden of a large number of fitness function computations. In spacecraft trajectory optimization, the first steps to include a surrogate model have been undertaken in the context of high-thrust missions using DE[22] and orbit transfers with constant thrust.[23]

The successful application of such a surrogate model approximating the fitness function depends on its accuracy to represent the original fitness landscape. The location of local optima is thereby more important than the absolute value of the approximation. Especially the existence of false optima introduced by the surrogate may throw the optimization off track and hinder convergence. It is therefore important to include a strategy for *model management* and *controlled evolution* which mainly includes the regular inclusion of the original fitness function to avoid the convergence towards a false optimum. There are many approaches to manage the inclusion of the surrogate, e.g. replacing the fitness function, filtering candidate solutions, or generating new solution.[6,13,21] Here, we present the approach developed for the given problem of optimizing linked low-thrust trajectories.

One of the challenges of a model that is trained on previous fitness function evaluations is the small size of the data set and the resulting poor initial quality of the surrogate, likely including false optima. During preliminary tests that used both original and surrogate evaluations in parallel, it was observed that a single false optimum in the surrogate was enough to derail the optimization as individuals with unrealistic small fitness values are never reevaluated with the original fitness function. The latter is a wanted behavior of the GA as it saves computational effort over time. We therefore chose a conservative approach to evolution control: The surrogate is trained in regular intervals on the growing data set and searched using a variety of optimization algorithms. Then, the best solutions found during the surrogate search are reevaluated using the original fitness function. Only if a solution is better than the previous GA population it gets reinserted, following the elitist reinsertion strategy of Pygmo's GA implementation. Then the GA continues with the original evolution, increasing the size of the data set and converging towards the true optimum. After a set number of generations, a *sequence*, a new ANN is trained and searched, potentially resulting in new candidate solutions.

As the evaluation of the surrogate is computationally cheap (about 120 µs per fitness evaluation vs. about 2.3 s for a fitness evaluation with 3 transfers), a total of 4 local (Cobyla, Bobyqa, Nelder-Mead, and Subplex) and 6 global optimization algorithms (GA, DE, DE1220, Particle Swarm Optimization (PSO), and Artificial Bee Colony (ABC)) are employed during the surrogate search to find new optima in the approximate model. Each local algorithm is run 3 times with different, random starting points in a multistart approach, totaling 17 different optimization runs. Each run is performed with default parameters which can be found in the Pygmo and NLopt documentations. The local algorithms are run until convergence with an absolute tolerance of 10^{-2} or a maximum of 5000 fitness evaluations. The global algorithms are run with a population size of 20 for 200 generations except for ABC which uses more fitness evaluations in the scouting phase[24] and is run for 100 generations. In terms of computational effort, one original fitness evaluation takes as long as running 3 local or 2 global optimizers using the surrogate. The total effort of searching for new candidate solutions utilizing the surrogate is therefore about 7 original fitness evaluations plus the effort of reevaluating the resulting candidates using the original fitness function.

The initial lack of data was identified as a major problem, see the Results section below, which is why we also investigated the use of a pretrained surrogate. Here, the surrogate is trained on a general data set containing precomputed low-thrust transfers. The more general, i.e. unrelated to the optimized trajectory, this data set is, the more easily it can be used for a variety of problems and does not have to be recomputed for each specific mission design problem. Pretraining improves the model's initial quality immensely but includes the additional effort of computing a general data set and pretraining the ANN. During the optimization the model is then only incrementally trained, meaning that it is not replaced but allowed to adapt to the region of interest by executing some optimization epochs on the new data. We observe incremental training times of 5 s to 60 s on the CPU, while pretraining a new model is a larger computational effort that is better suited for the GPU. Pretraining took in the order of 10 min for the two presented case studies. To avoid forgetting of previously seen data this *incremental* training is started at a lower initial learning rate of 10^{-4}. The learning rate is reduced on the detection of a validation loss plateau and training is stopped early as before. Both approaches, the fully online generation of the surrogate and the optional use of a pretrained model, are shown in Figure 6.

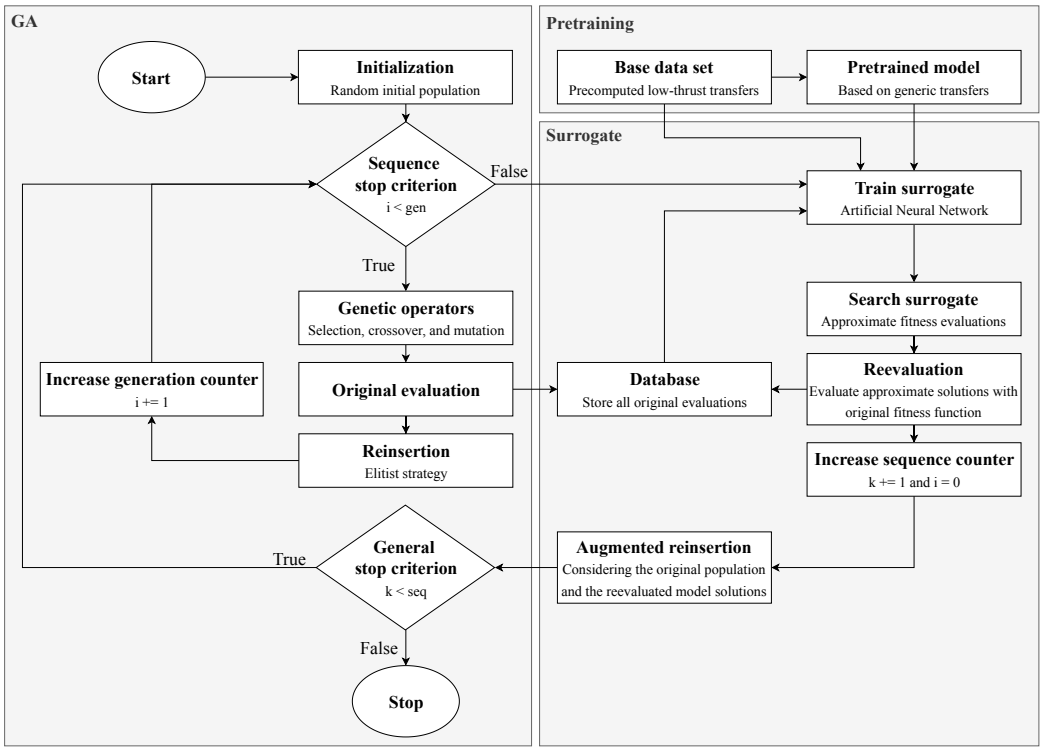

Figure 6. The augmented GA using an ANN surrogate to find new candidate solutions

We are exploring three different points of approximation for the surrogate: The first approach is to replace the full fitness function evaluation by the surrogate, with the ML model being trained directly on the candidate solution vectors of the previous generations. This approach, termed *full surrogate*, is the most general and readily translates to other problems. We also test two other strategies that are adapted to the problem at hand in order to incorporate prior knowledge about problems concerning interplanetary transfers. The focus is therefore put on the computationally expensive part of the fitness function, i.e. the inner ΔV computation of individual low-thrust legs of the linked trajectory. The first approach is to replace the general shape optimization by an ML model, i.e. to train a *general transfer surrogate*, the other is to replace each transfer by a separate model, termed *individual transfer surrogate*. The advantage of the first one being the availability of more online training data, while a model estimating individual transfers has to learn a problem that is valid for a smaller input parameter space, therefore reducing the approximation problem's difficulty. An extension on the latter approach is the use of a model that is pretrained on a general set of similar transfers, which is also investigated.

RESULTS

The presented approach is applied to two example problems that utilize low-thrust propulsion to explore the solar system. The first one is derived from the Dawn mission and involves a launch at Earth, a flyby at Mars, a stay at Vesta, and a transfer to Ceres. The second example is a simplified version of the problem posed for the second Global Trajectory Optimization Competition (GTOC2). It consists of four consecutive visits at different asteroids with a minimum stay time. Both problems and the results are presented here.

The Dawn Problem

The first example problem is inspired by NASA's Dawn mission which explored (4) Vesta and (1) Ceres in the asteroid belt. The spacecraft was launched in September 2007 and left Earth's SOI with a C_3 of $11.3\,\mathrm{km}^2\,\mathrm{s}^{-2}$.[25] Propulsion was achieved with three xenon ion engines with a specific impulse of 3100 s producing a maximum thrust force of 91 mN. With the spacecraft's launch mass of 1300 kg this translates to an initial maximum thrust acceleration at of $7 \times 10^{-5}\,\mathrm{m\,s}^{-1}$ per thruster. Dawn's trajectory was designed with a duty cycle of 95 %, meaning that the continuous thrust imposed in the shaping model is a reasonable approximation. The mission's trajectory, including thrust (blue) and coast periods (black), is presented in Figure 7. After launch, the spacecraft used its low-thrust propulsion system to reach Mars, where an unpowered flyby was performed in February 2009 yielding a velocity change of $2.6\,\mathrm{km\,s}^{-1}$.[26] The low-thrust transfer continues and Vesta is reached for a rendezvous in July 2011. The science phase in orbit around Vesta lasted one year until Dawn performed a transfer to Ceres which lasted from July 2012 to February 2015. Table 1 lists the flight and stay times as well as the boundary conditions defining the trajectory phases. In total the spacecraft achieved a ΔV of $11\,\mathrm{km\,s}^{-1}$, including the interplanetary transfers and proximity operations.

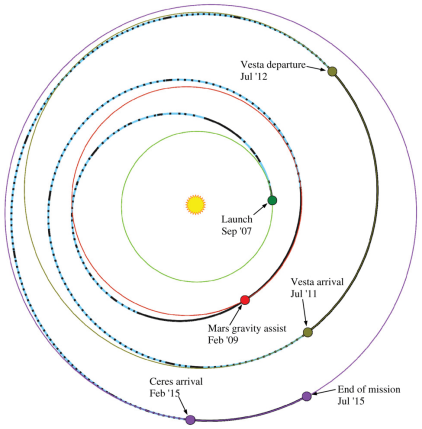

Figure 7. The trajectory of the Dawn mission[26]

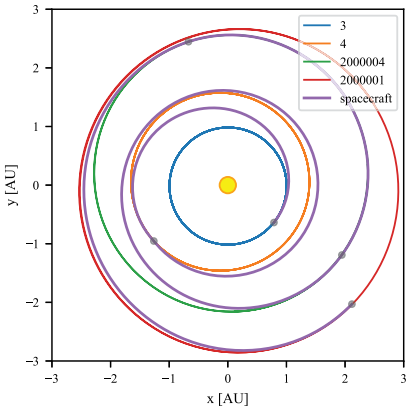

Figure 8. Optimized Dawn problem

Table 1. The Dawn mission: Phases and dates[26]

	Transfer 1	Transfer 2	Stay 1	Transfer 3
Departure body	Earth	Mars	Vesta	Vesta
Arrival body	Mars	Vesta	Vesta	Ceres
Departure condition	Orbit injection	Flyby	Orbit following	Rendezvous
Arrival condition	Flyby	Rendezvous	Orbit following	Rendezvous
Revolutions [ ]	0	1	–	0
Departure date [calendar date]	01 Sep 2007	12 Feb 2009	05 Sep 2011	01 Jul 2012
Departure date [mjd2000]	2800	3330	4265	4565
ToF [days]	530	935	300	945

An optimization problem was derived from Dawn's trajectory. Its free parameters and the corresponding bounds are presented in Table 2. Corresponding to the Dawn mission N was fixed to 0, 1, and 0 for transfers 1, 2, and 3, respectively.

As shown before, see Figure 5, the GA converges well on this problem. The overall best solution of $11.8\,\mathrm{km\,s}^{-1}$ was found during one of the runs with a population size of 10 and a mutation probability of 5 % after 2860 fitness function evaluations. All 5 runs reached fitness values below $15\,\mathrm{km\,s}^{-1}$ within 1400 fitness

Table 2. Optimization parameters of the Dawn Problem

Parameter	Unit	Lower bound	Upper bound	Best GA result
Departure date	mjd2000	2500	3500	2782.2
ToF 1	days	300	700	407.6
ToF 2	days	700	1100	1099.1
Stay 1	days	200	600	559.2
ToF 3	days	700	1100	912.7
Orbit insertion V_∞	$\mathrm{m\,s^{-1}}$	1.5×10^3	5×10^3	2.42×10^3
Mars arrival velocity (radial)	$\mathrm{m\,s^{-1}}$	-5×10^3	5×10^3	-2.21×10^3
Mars arrival velocity (tang.)	$\mathrm{m\,s^{-1}}$	1×10^4	5×10^4	2.17×10^4
Mars arrival velocity (vert.)	$\mathrm{m\,s^{-1}}$	-5×10^3	5×10^3	-6.37×10^1
Flyby radius r_p	m	$r_{\mathord{\male}} + 1.5 \times 10^5$	$r_{\mathord{\male}} + 1 \times 10^7$	$r_{\mathord{\male}} + 4.71 \times 10^5$
Flyby plane rotation angle β	rad	0	2π	2.71

evaluations. The resulting trajectory is shown in Figure 8 and the corresponding parameter values are included in Table 2. It can be seen that the optimizer converges to a slightly earlier launch date. ToF 1 and 3 are chosen shorter while ToF 2 and the stay time at Vesta are longer in the optimized solution than the original trajectory. ToF 2 is close to the imposed upper boundary which was kept as is to not overly increase the total flight time. The maximum thrust acceleration is $2 \times 10^{-4}\,\mathrm{m\,s^{-2}}$, which is 70 % of the initial thrust acceleration of two Dawn ion engines. The thrust is therefore achievable by current technology without having imposed any upper limit during the optimization. The total ΔV is in the same order of magnitude as the original mission with Transfers 1, 2, and 3 contributing $2.46\,\mathrm{km\,s^{-1}}$, $5.76\,\mathrm{km\,s^{-1}}$, and $3.57\,\mathrm{km\,s^{-1}}$, respectively. Of course, this is only a rough comparison as the real mission had to deal with many more constraints, a full dynamical environment, and also included the science operations at both bodies. Overall we conclude that the optimization approach based on a linked conics approximation and hodographically shaped low-thrust transfers is able to capture the character of the original trajectory while being rapidly computed, and therefore represents a suitable approach to preliminary low-thrust trajectory optimization.

Full surrogate As a first augmentation approach, the surrogate is trained solely on data from the GA. One sequence was chosen to be 200 fitness evaluations, split into the evolution of a population of 10 individuals for 20 generations. The mutation probability was set to 10 %. Figure 9 shows the evolution of 5 runs. Here, the dots indicate the minimum value found during the surrogate utilization (training, finding new candidate solutions, reevaluating using the original fitness function). It can be seen that the reevaluated candidate solutions are never better than the current champion in the GA population. In fact, during these 5 runs the best model solution was never good enough to be reinserted into the original population. This means that the GA evolution is unaffected by the use of the surrogate and would converge similarly without any augmentation. We attribute this mainly to the lack of available data. The 200 data points available after the first sequence are not enough to construct a useful approximation of the fitness function and, while the model improves as more data becomes available, it never catches up with the improvements found by the GA. The 11-dimensional fitness function is highly non-linear, especially due to the Mars flyby. There are also interdependencies between the variables as a longer ToF in one phase shifts the departure and arrival dates of later phases, making it a difficult problem to learn on limited data.

General transfer surrogate The case is similar for the *general transfer surrogate*, which is trained on the inputs to the shaping method and estimates the cost of the low-thrust transfers with one model. The convergence of 5 runs is shown in Figure 10. The number of reinsertions into the original populations are indicated with a number next to the found minimum. Here, this occurred early during run 5 where the surrogate search resulted in an improvement of the global optimum. The candidate solutions found using the surrogate are better, consistently finding solutions below $25\,\mathrm{km\,s^{-1}}$ starting with the second sequence, but are generally not good enough to increase the fitness of the original population. The average MAPE and sorting accuracy of the final model are 23.1 % and 67.2 %, respectively.

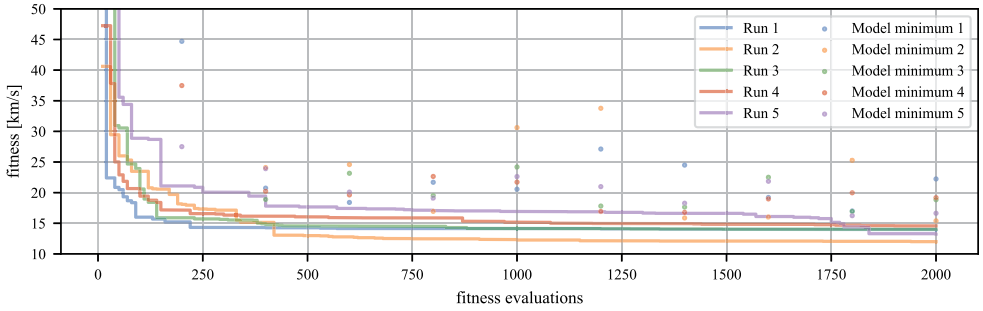

Figure 9. Full surrogate applied to the Dawn problem (Pop. 10, Gen. 20, Seq. 10)

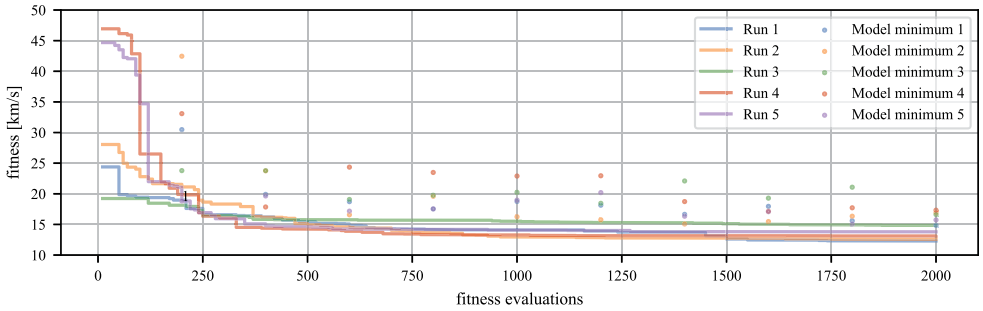

Figure 10. General transfer surrogate applied to the Dawn problem (Pop. 10, Gen. 20. Seq. 10)

Individual transfer surrogate A separate model is trained for each transfer, reducing the problem complexity. Especially the estimation for Transfer 3 improves, as the rendezvous conditions at departure and arrival simplify the approximated function. The effect is smaller for Transfers 1 and 2, which have more diverse boundary conditions due to orbit injection and the Mars flyby. The models' average final MAPE are 32.3 %, 11.2 %, and 2.4 % for Transfers 1, 2, and 3, respectively. The corresponding sorting accuracies reach 75.2 %, 89.6 %, and 96.0 %. The algorithm's convergence is shown in Figure 11. Occasional reinsertions happen, but the general effect on the GA is small.

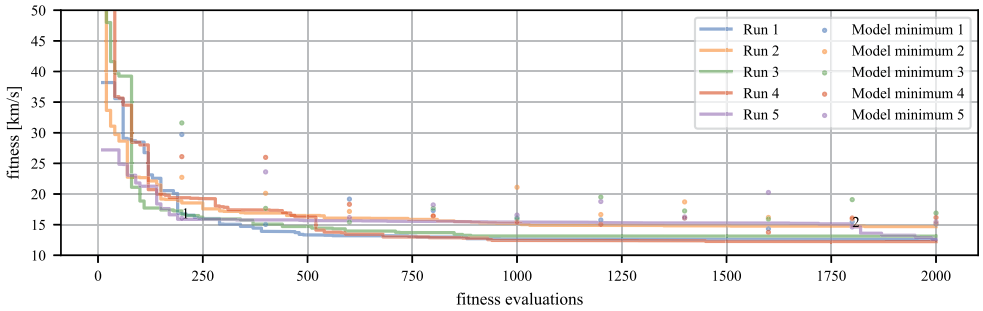

Figure 11. Individual transfer surrogate applied to the Dawn problem (Pop. 10, Gen. 20, Seq. 10)

Pretrained individual transfer surrogate Here, the initial quality of the surrogates is improved by pretraining the ANN on a data set comprised of transfers from Earth to Mars, Mars to Vesta, and Vesta to Ceres. We used the data from the random searches that were used to develop the ANN model. This data set was deemed to

be general enough for a proof of concept as the range for the departure date comprises more than two synodic periods between Earth and Mars with the later epochs being additionally shifted by the ranges in the ToF and stay time at Vesta. As the data set is now much larger than during the online cases and the computational effort of pretraining the model is detached from the actual optimization run, the ANN was slightly increased in size to an architecture of 3/124. The used data set has a size of 15 000 trajectories of which 100 are used for the computation of MAPE and sorting accuracy. The remaining trajectories are split into training and validation sets at a ratio of 90/10. Based on this much larger data set the regression MAPE of the pretrained surrogate is greatly improved in comparison to the online generated variants: It reaches 3.5 %, 2.2 %, and 0.1 % for models 1, 2, and 3, respectively. The sorting accuracy reaches 100 % for all three models.

The algorithm was then run with a population size of 10 and 5 generations per sequence, see Figure 12. It can be seen that the surrogate improves the champions especially after the first sequence where all runs receive a new champion. Note that the minimum found during the random search was 16.6 km s^{-1}, meaning that the surrogate evaluation yields better results than a linear interpolation would. The surrogate also keeps improving due to the incremental training between sequences and more reinsertions keep improving the fitness of the original population after the following sequences. Furthermore, all runs converge to good optima below 12.5 km s^{-1} which was not the case during the runs of the original GA or the full surrogate approach.

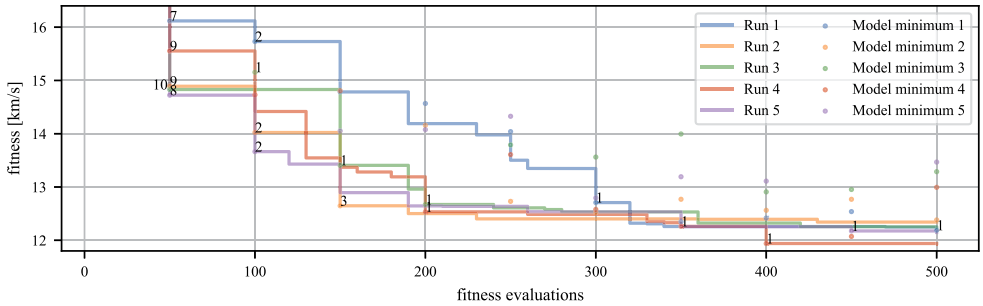

Figure 12. Pretrained individual transfer surrogate applied to the Dawn problem (Pop. 10, Gen. 5, Seq. 10)

The Asteroid Problem

As a second example, we apply the presented optimization to a multiple asteroid rendezvous mission, derived from the problem created by JPL for GTOC2*. The problem was to visit one asteroid out of 4 groups with a low-thrust mission in a given time-frame. The asteroid orbits are shown in Figure 13. All competition solutions visited an asteroid from Group 4 first and then went outwards to Groups 3, 2, and 1. We disregard the outer problem of finding a suitable sequence of asteroids to visit and concentrate on finding a low ΔV trajectory for given sequences. We also optimize directly for ΔV and not for the original fitness function which was the final mass over total ToF.

Previous work applied the spherical shaping method[27] to the winning trajectories of GTOC2,[28] keeping launch dates as well as ToFs and stay times constant. V_∞ was set to zero for the shaped solutions. Recreating these trajectories using the hodographic shaping method yields the results listed in Table 3 and shown in Figure 14 for the winning solution. It can be seen that the hodographic shaping method performs significantly better than the spherical method while not reaching the quality of the fully optimized winning solutions.

Table 4 lists the parameters of the example problem derived from the GTOC2 setting. The asteroid sequence† as well as the N of each transfer (0, 0, 0, 0) are fixed and the optimizer is tasked to find optimal values for

*A. Petropoulos, Problem Description for the 2nd Global Trajectory Optimization Competition, 2006
†SPK IDs: 3258076, 2000060, 2000058, 2002959; Pykep indices: 815, 300, 110, 47

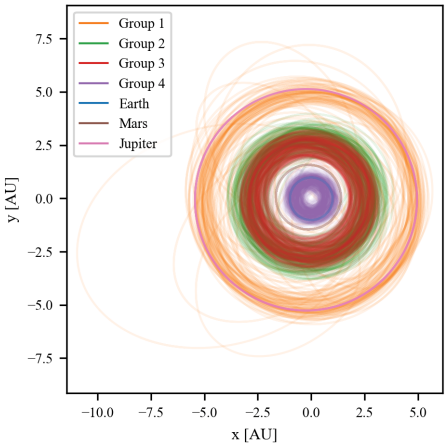

Figure 13. The GTOC2 asteroids

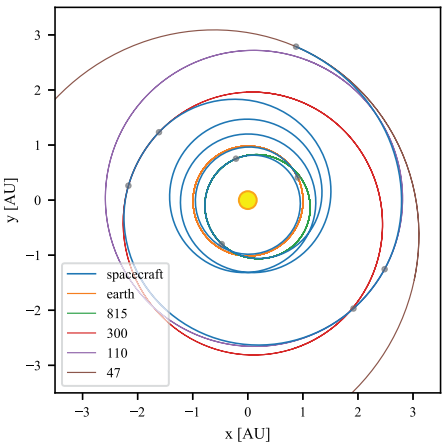

Figure 14. The recreated 1st trajectory

Table 3. Fixed solutions of the Asteroid problem

Winners[28]	N	ΔV_{total}	Spherical[28]	ΔV_{total}	Hodographic	ΔV_{total}
GTOC 1st	1, 2, 0, 0	20.1 km s^{-1}	Shaping 1	41.78 km s^{-1}	Shaping 1	39.56 km s^{-1}
GTOC 2nd	0, 3, 0, 0	19.4 km s^{-1}	Shaping 2	49.49 km s^{-1}	Shaping 2	27.78 km s^{-1}
GTOC 3rd	1, 2, 0, 0	23.3 km s^{-1}	Shaping 3	46.08 km s^{-1}	Shaping 3	33.81 km s^{-1}

the departure date, flight and, stay times as well as magnitude and direction of the injection impulsive shot. The bounds for the departure window are set to 10 years. The bounds for ToFs are kept wide, shifting towards longer durations for later transfers, while the stay times are kept close to the minimum time of 90 days set in the competition. The parameters of the best solution found, see Table 4, yielded a total ΔV of 26.9 km s^{-1}.

Table 4. Optimization parameters of the Asteroids problem

Parameter	Unit	Lower Bound	Upper bound	Best result
Departure Date	mjd2000	5479	9129	8311.5
ToF 1	days	100	1700	106.8
Stay 1	days	90	110	90.0
ToF 2	days	200	1900	517.1
Stay 2	days	90	110	102.1
ToF 3	days	400	1900	1299.5
Stay 3	days	90	110	90.0
ToF 4	days	600	2000	1096.6
Injection magnitude V_∞	m s^{-1}	0	3.5×10^3	3.22×10^3
Injection angle φ	rad	0	π	2.19
Injection angle θ_1	rad	0	2π	2.91

Full surrogate The surrogate assisted optimizations are again conducted with sequences of 200 fitness function evaluations. The case is similar to the application of the full surrogate optimization to the Dawn problem: Even though the surrogate's quality improves as more data becomes available, it not high enough to assist the optimization, see Figure 15.

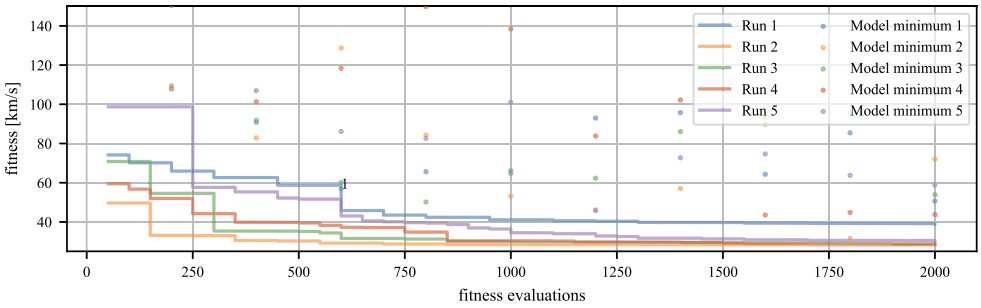

Figure 15. Full surrogate applied to the Asteroid problem (Pop. 50, Gen. 4, Seq. 10)

General transfer surrogate More reinsertions are observed when the surrogate is trained to approximate the cost of the low-thrust transfer, see Figure 16. This is partially due to the larger population size resulting in a higher chance for medium quality solutions to be better than some member of the GA population. But we also observe a number of very useful reinsertions where the reinsertion leads to a new champion, e.g. after each of the first 3 sequences of Run 1. Another aspect that leads to better results here in comparison to the Dawn case is that the transfers are more homogeneous: Only one departure includes an impulsive shot and there are no flybys. Additionally, there are four transfers per trajectory, resulting in more and better training data.

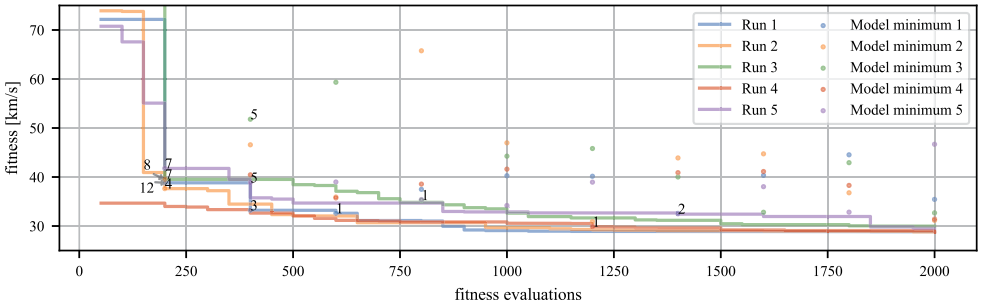

Figure 16. General transfer surrogate applied to the Asteroid problem (Pop. 50, Gen. 4, Seq. 10)

Individual transfer surrogate The results for the third variation of surrogate inclusion are shown in Figure 17. Similar to the general transfer surrogate useful candidate solutions are found in the surrogate especially in the beginning. But, reinsertions happen more slowly than for the general transfer surrogate and the overall algorithm does not reliably converge below 34 km s^{-1}. The suspected reason is again the reduced initial quality of the model due to less training data as there are now four models, each trained on samples from one transfer.

Pretrained general transfer surrogate Following the previous results, the pretrained approach is applied to the general transfer surrogate. Here, a new data set is created based on shaped transfers between separate from the predefined sequence in the optimization problem. Transfers are computed between random bodies from Earth to Group 4, Group 4 to 3, 3 to 2, and 2 to 1, with 110 000 samples per group resulting in a data set of 440 000 samples. The departure date is set to a range of [5479, 12053], representing the first 18 years of the competition window, and the ToF is varied in large windows of [50, 600] days for the first two transfers and [300, 1500] and [400, 1500] days for the two transfers further away from the Sun. The Earth to Group 4 transfer included a maximum V_∞ at launch of 3.5 km s^{-1} with φ in $[1/6\pi, 5/6\pi]$ and θ_1 in $[0, 2\pi]$, while the other three transfers have rendezvous boundary conditions. Due to the random inputs, some transfers have physically very unfavorable conditions leading to high ΔV (and a few NaN) values. We therefore filter the database for transfers below 1000 km s^{-1}, leaving 438 526 sample transfers. As previously the network was

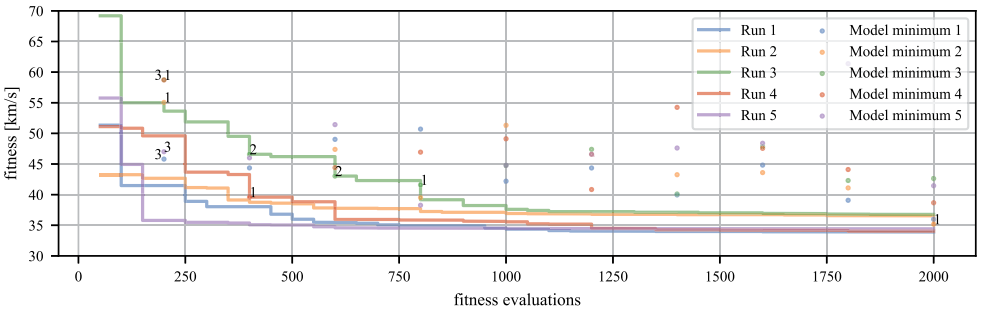

Figure 17. Individual transfer surrogate applied to the Asteroid problem (Pop. 50, Gen. 4, Seq. 10)

expanded to an architecture of 3/128 and the data set was split into training and validation sets at a ratio of 90/10. Training lasted for 275 epochs in 20 min on the GPU and the ANN reached a MAPE of 14.6 % and sorting accuracy of 88 %.

The algorithm is then run with sequences of 2 generations and its convergence is shown in Figure 18. The surrogate search after the first sequence results in a new champion for all runs and continues to produce useful solutions up until the 5th sequence. The result is a reliable convergence of all runs below 30 km s^{-1} within 850 fitness function evaluations.

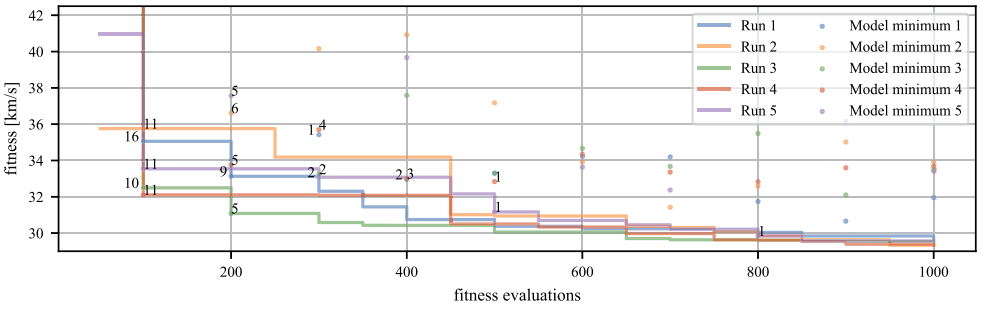

Figure 18. Pretrained general transfer surrogate applied to the Asteroid problem (Pop. 50, Gen. 2, Seq. 10)

Average convergence

The average convergence for the different implementations of the algorithm are shown in one plot together with the classical GA for both test cases. For the Dawn case, see Figure 19, there is on average little influence for all three versions of the online surrogate approach. The pretrained case, however, converges much faster, as the first surrogate search consistently leads to new champions and considerably improves the overall fitness of the population. Beyond 500 fitness function evaluations all approaches behave similarly, slowly converging towards 12 km s^{-1}.

Figure 20 shows the average convergence of the different approaches on the Asteroid problem. Here, both the general transfer surrogate and the individual transfer surrogate converge faster than the unassisted GA due to new solutions found utilizing the surrogate. These are visible as steps in the plots at 200, 400 and 600 fitness evaluations. The individual transfer surrogate does not converge to the same optimum within the observed 2000 fitness evaluations, the reason for which is unclear. An interference from the surrogate can be ruled out due to the taken approach to evolution control: If no better solutions are found during surrogate utilization, the original GA evolution is not affected. The pretrained general transfer surrogate assisted GA

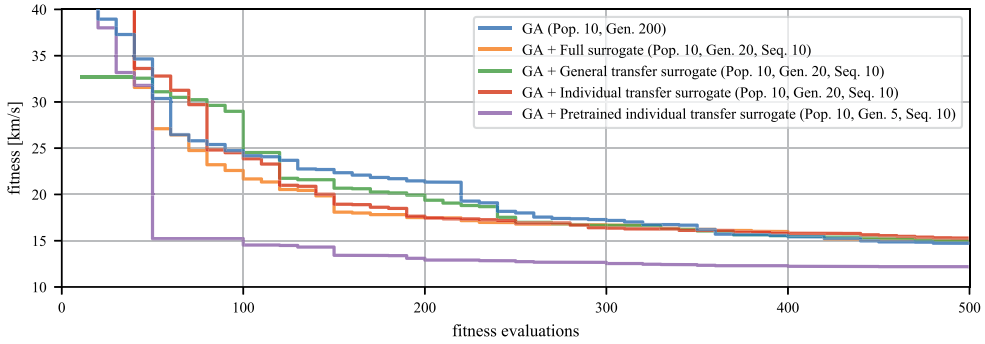

Figure 19. Averaged convergence of the investigated algorithms on the Dawn problem

converges the fastest, with a large step towards the minimum after the first sequence at 50 fitness evaluations. The surrogate's influence is less visible afterwards, but the plot of individual runs, see Figure 18, showed that the surrogate keeps supplying new candidate solutions until the third sequence, making this approach the most promising for future research.

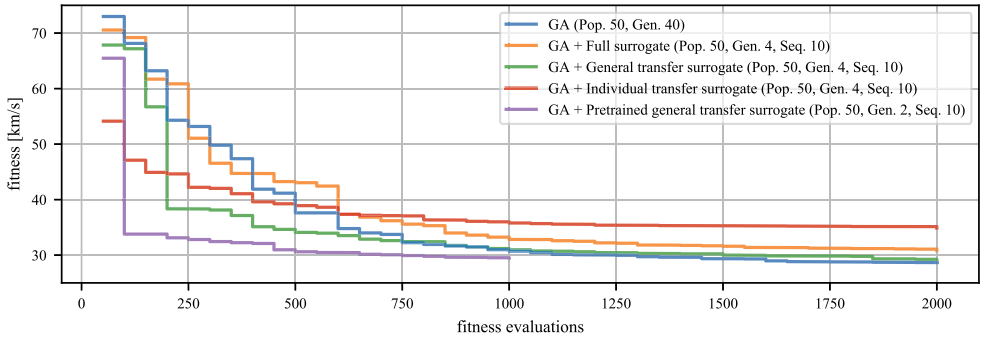

Figure 20. Averaged convergence of the investigated algorithms on the Asteroid problem

CONCLUSION

A flexible Python tool for preliminary optimization of interplanetary low-thrust trajectories was developed. It was employed in the development of a surrogate-assisted evolutionary optimization approach using an Artificial Neural Network to approximate the problem's fitness function and find new candidate solutions that are reinserted into the original population following an elitist strategy. Different ways to design the surrogate were investigated. The full surrogate approach of directly modeling the relationship between candidate solution and fitness suffers from the initially small size of the data set. Concentrating on the computationally expensive parts of the fitness function, the ΔV computation, and thereby simplifying the approximated problem and using more specialized transfer surrogates, leads to better results as more useful new solutions are found especially early on, increasing the overall fitness of the population. Extending the online approach with a pretrained model that was trained on a general data set consisting of shaped transfers improves the initial quality of the model greatly, consistently yielding new best solutions. The pretrained model continues to be useful by being incrementally trained on the new data points becoming available during the optimization run, allowing the surrogate to adapt to the region of interest.

We conclude that the inclusion of a surrogate can improve the evolutionary optimization of low-thrust multi-phase trajectories and especially the availability of a pretrained model can greatly speed up the preliminary search for trajectories. For future work we recommend the further investigation of Machine Learning surrogates in this context.

REFERENCES

[1] C. E. Garner *et al.*, "Ion propulsion: An enabling technology for the dawn mission," *23rd AAS/AIAA Spaceflight Mechanics Meeting, Kauai, Hawaii*, 2013.

[2] J. T. Betts, "Survey of numerical methods for trajectory optimization," *Journal of Guidance, Control, and Dynamics*, Vol. 21, No. 2, 1998, pp. 193–207.

[3] A. E. Petropoulos and J. M. Longuski, "Shape-based algorithm for the automated design of low-thrust, gravity assist trajectories," *Journal of Spacecraft and Rockets*, Vol. 41, No. 5, 2004, pp. 787–796.

[4] A. Shirazi, J. Ceberio, and J. A. Lozano, "Spacecraft trajectory optimization: A review of models, objectives, approaches and solutions," *Progress in Aerospace Sciences*, Vol. 102, 2018, pp. 76–98.

[5] S. Li, X. Huang, and B. Yang, "Review of optimization methodologies in global and China trajectory optimization competitions," *Progress in Aerospace Sciences*, 2018, pp. 60–75.

[6] Y. Jin, "Surrogate-assisted evolutionary computation: Recent advances and future challenges," *Swarm and Evolutionary Computation*, Vol. 1, No. 2, 2011, pp. 61–70.

[7] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, Vol. 521, No. 7553, 2015, p. 436.

[8] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks*, Vol. 61, 2015, pp. 85–117.

[9] D. J. Gondelach and R. Noomen, "Hodographic-shaping method for low-thrust interplanetary trajectory design," *Journal of Spacecraft and Rockets*, Vol. 52, No. 3, 2015, pp. 728–738.

[10] D. Izzo, "Global optimization and space pruning for spacecraft trajectory design," *Spacecraft Trajectory Optimization* (B. A. Conway, ed.), ch. 7, pp. 178–201, Cambridge University Press, 2010.

[11] D. Morante, M. Sanjurjo Rivo, and M. Soler, "Multi-Objective Low-Thrust Interplanetary Trajectory Optimization Based on Generalized Logarithmic Spirals," *Journal of Guidance, Control, and Dynamics*, Vol. 42, No. 3, 2018, pp. 476–490.

[12] W. M. Folkner, J. G. Williams, D. H. Boggs, R. S. Park, and P. Kuchynka, "The planetary and lunar ephemerides DE430 and DE431," *Interplanetary Network Progress Report*, Vol. 196, 2014, pp. 1–81.

[13] Y. Jin, "A comprehensive survey of fitness approximation in evolutionary computation," *Soft Computing*, Vol. 9, No. 1, 2005, pp. 3–12.

[14] Y. Bengio, "Practical recommendations for gradient-based training of deep architectures," *Neural networks: Tricks of the trade*, pp. 437–478, Springer, 2012.

[15] F. Pedregosa *et al.*, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, Vol. 12, 2011, pp. 2825–2830. Version 0.21.2.

[16] Y. Jin, M. Olhofer, and B. Sendhoff, "On evolutionary optimization with approximate fitness functions," *2nd Annual Conference on Genetic and Evolutionary Computation*, 2000, pp. 786–793.

[17] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," *13th International Conference on Artificial Intelligence and Statistics*, 2010, pp. 249–256.

[18] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *3rd International Conference on Learning Representations*, 2014.

[19] L. Stubbig, "Investigating the use of neural network surrogate models in the evolutionary optimization of interplanetary low-thrust trajectories," *MSc Thesis, TU Delft*, 2019.

[20] B. J. Wall and B. A. Conway, "Genetic algorithms applied to the solution of hybrid optimal control problems in astrodynamics," *Journal of Global Optimization*, Vol. 44, No. 4, 2009, p. 493.

[21] J. Zhang *et al.*, "Evolutionary computation meets machine learning: A survey," *IEEE Computational Intelligence Magazine*, Vol. 6, No. 4, 2011, pp. 68–75.

[22] C. Ampatzis and D. Izzo, "Machine learning techniques for approximation of objective functions in trajectory optimisation," *IJCAI-09 Workshop on Artificial Intelligence in Space*, 2009, pp. 1–6.

[23] Y.-S. Hong, H. Lee, and M.-J. Tahk, "Acceleration of the convergence speed of evolutionary algorithms using multi-layer neural networks," *Engineering Optimization*, Vol. 35, No. 1, 2003, pp. 91–102.

[24] M. Mernik *et al.*, "On clarifying misconceptions when comparing variants of the Artificial Bee Colony Algorithm by offering a new implementation," *Information Sciences*, Vol. 291, 2015, pp. 115–127.

[25] M. D. Rayman and K. C. Patel, "The Dawn project's transition to mission operations: On its way to rendezvous with (4) Vesta and (1) Ceres," *Acta Astronautica*, Vol. 66, No. 1-2, 2010, pp. 230–238.

[26] M. D. Rayman and R. A. Mase, "The second year of Dawn mission operations: Mars gravity assist and onward to Vesta," *Acta Astronautica*, Vol. 67, No. 3-4, 2010, pp. 483–488.

[27] D. M. Novak and M. Vasile, "Improved shaping approach to the preliminary design of low-thrust trajectories," *Journal of Guidance, Control, and Dynamics*, Vol. 34, No. 1, 2011, pp. 128–147.

[28] T. Roegiers, "Application of the Spherical Shaping Method to a Low-Thrust Multiple Asteroid Rendezvous Mission: Implementation, limitations and solutions," *M.Sc. Thesis, TU Delft*, 2014.