In this work we address a multi-objective problem of finding optimal low thrust gravity assist trajectories for interplanetary and orbital transfers. For this, we use recently developed pruning techniques for incremental search space reduction – which we will extend for the current situation – in combination with subdivision techniques for the approximation of the Pareto sets. These techniques are particularly promising for the numerical treatment of these multi-objective design problems since they are characterised (amongst others) by highly disconnected feasible domains, which can easily be handled by these set oriented methods. We analyse the complexity of the novel pruning techniques, and finally demonstrate the usefulness of our approach by showing some numerical results on two realistic cases.
1 Introduction

NASAs Deep Space 1 and recently ESAs SMART-1 have shown the effectiveness of low-thrust systems as primary propulsion devices. Such new scenarios make the task of mission analysts more difficult than ever. In fact, the design of low-thrust transfers generally requires the solution of an optimal control problem, which has no general solution in closed form. Different methods have been developed to tackle these trajectory design problems. However, all of them need to be initialised with a first guess solution. The generation of a suitable first guess turns out to be a tricky and quite time consuming task. Studies on the generation of first guess solutions for low-thrust transfers, date back to the late nineties with the works of Coverstone et al. (5, 15), where multi-objective genetic algorithms were first used to compute first guess solutions for an indirect method. The derivation of approximated analytical solution was addressed in the works of Markopoulos (10), Bishop and Azimov (1, 2). Inspired by the work of Tanguay (19), Petropoulos and Longuski (14) proposed a shape-based approach, which represents the trajectory (connecting two points in space) with a particular parameterised analytical curve (or shape) and computes the control thrust necessary to satisfy the dynamics. Although the resulting trajectory is not the actual solution of an optimal control problem, by tuning the shaping parameters it is possible to generate solutions, sufficiently good to initialise a more fine optimisation process. More precisely in the work by Petropoulos, a thrust arc is represented by an analytical curve, known as exponential sinusoid, which consists of a five free parameters shape in polar coordinates. This shape is suitable for the approximation of planar motion, and the reduced number of shaping parameters does not allow to satisfy all the possible boundary conditions on position, velocity, time of flight and magnitude of the control acceleration; for 3D problems the propellant consumption for out of plane motion, is only estimated. By implementing the exponential-sinusoid trajectory model in the software code STOUR, Petropoulos and Longuski extended their systematic search for optimal ballistic MGA transfers to the global solution of Low-Thrust Gravity Assist (LTGA) transfers (11, 14, 13). Recently, it has been shown that whenever a Multiple Gravity Assist (MGA) optimisation problem is characterised by a simple $\Delta v$-matching for the swing-bys and no deep-space manoeuvres are present, there exists a polynomial-time algorithm that provides an efficient solution to the problem (13). Namely a branch and prune technique exists, the complexity of which is quartic with respect to dimensionality, i.e. in the number of swing-bys, and cubic in the resolution of the discretisation of time variables. If Multi LTGA trajectories (MLTGA) are considered it would be desirable to have an equivalent polynomial-time algorithm. In this paper we present an incremental pruning algorithm for the solution of the MLTGA problem and an analysis of the
computational complexity of such an algorithm. Further, we show one possible way to integrate the output set efficiently into the related (multi-objective) optimisation process.

The remainder of this paper is organised as follows: in Section 2 we state the background required for understanding the work. In Section 3 we propose an incremental pruning algorithm for MLTGA problems and analyse its complexity in Section 4. Section 5 deals with the numerical treatment of the multi-objective trajectory design problem, in Section 6 we present some numerical results, and we finally conclude in Section 7.

2 Background

In this section we state the required background for the understanding of the sequel: we introduce the concept of multi-objective optimisation, state the trajectory design problem, and finally describe subdivision techniques which will be used to attack the resulting problems.

2.1 Multi-Objective Optimisation

In a variety of applications in industry and finance a problem arises that several objective functions have to be optimised concurrently. One important feature of these problems is that the different objectives typically contradict each other and therefore certainly do not have identical optima. Thus, the question arises how to approximate one or several particular 'optimal compromises' or how to compute the entire set of optimal compromises – the Pareto set – of this multi-objective optimisation problem (MOP). For the solution of both problems there already exist a huge variety of efficient algorithms (see e.g. (12), (6) and references therein).

Mathematically speaking, an MOP can be stated in its general form as follows:

\[ \min_{x \in S} \{ F(x) \}, \quad S = \{ x \in \mathbb{R}^n : h(x) = 0, \ g(x) \leq 0 \}, \]

where \( F \) is defined as the vector of the objectives, i.e.

\[ F : \mathbb{R}^n \to \mathbb{R}^k, \quad F(x) = (f_1(x), \ldots f_k(x)), \]

with \( f_1, \ldots, f_k : \mathbb{R}^n \to \mathbb{R} \), \( h : \mathbb{R}^n \to \mathbb{R}^m \), \( m \leq n \), and \( g : \mathbb{R}^n \to \mathbb{R}^q \). A vector \( v \in \mathbb{R}^k \) is said to be dominated by a vector \( w \in \mathbb{R}^k \) if \( w_i \leq v_i \) for all \( i \in \{1, \ldots, k\} \) and \( v \neq w \). A vector \( v \) is called non-dominated with respect to a set \( P \), if none of the vectors \( p \in P \) dominate \( v \).

A point \( x \in S \) is called optimal or Pareto optimal, if \( F(x) \) is not dominated by
any vector $F(y), y \in S$. The solution set – the so-called Pareto set – consists typically not of finitely many points as for scalar optimisation problems, but forms a $(k-1)$-dimensional object.

### 2.2 Problem Formulation

A multiple low-thrust gravity assist transfer is here modeled as a sequence of multiple arcs connecting $N+1$ celestial bodies. The i-th low-thrust arc connects celestial body $i$ to celestial body $i+1$ and is computed independently of the others just using the position of celestial body $i$ at time $t_i$ and the position of celestial body $i+1$ at time $t_{i+1}$. The sequence of thrust legs is then assembled inserting a powered swing-by at each swing-by planet. The powered swing-by restores the discontinuity, in the velocity, that occurs at each celestial body. Each thrust leg is modeled through a shape-based method based on exponential sinusoids ((14, 8)).

The overall process for the composition of an MLTGA trajectory with the exponential sinusoid model can be summarised with the following steps (see Fig. 1):

- For each departure date $t_0$ and set of transfer times $T_i$
- Compute a low-thrust arc through the exponential sinusoid model from A to B
- Compute $v_1$
- Compute a low-thrust arc through the exponential sinusoid model from B to C
- Compute $v_3$
- Compute $v_2$ with pericenter radius $\geq r_p$
- If $v_2 \neq v_3$ compute matching $\Delta v_i$ at the pericentre of the hyperbola
- Compute the launch impulsive manoeuvre $\Delta v_0$
- Compute the arrival impulsive manoeuvre $\Delta v_N$
- Compute the the low-thrust $\Delta v$
- Compute the sum of all $\Delta v$

Therefore, an optimal MLTGA transfer would minimise the total propellant consumption due to the low-thrust engine and the total $\Delta v$ required at each planet to correct the gravity assist manoeuvre. Then, a first objective function $J$ can be defined as follows:

$$J = 1 - e^{-(\frac{\Delta V_{GA} + \Delta V_0}{I_{sp1}})}$$ \hspace{1cm} (2.1)$$

where $\Delta V_{GA}$ is the sum of all the $\Delta V$s (variation in velocity) required to correct every gravity assist manoeuvre, $\Delta V_0$ is the departure manoeuvre, while $\Delta V_{LT}$ is the sum of the total $\Delta V$ of each low-thrust leg. The two
specific impulses $I_{sp1}$ and $I_{sp2}$ are respectively for a chemical engine and for a low-thrust engine. In this way the use of low-thrust is favorite with respect to the chemical corrections.

A second objective for an ML TGA transfer can be simply given by the flight time required to transfer the spacecraft from the departure planet to the destination one. This objective is of great importance for the design process since the transfer can take several years (see e.g. the examples in Section 6).

\[
\begin{align*}
\text{minimise:} & \quad J(y) \\
\text{minimise:} & \quad T_{tot} = \sum_{i=1}^{N} T_i \\
\text{subject to:} & \quad r_p \geq r_{min}
\end{align*}
\] (2.2)

with the solution vector defined as follows:

\[
y = [t_0, T_1, k_{2,1}, n_1, \ldots, T_i, k_{2,i}, n_i, \ldots, T_N, k_{2,N}, n_N]^T
\] (2.3)

Where $k_{2,i}$ is the i-th shaping parameter for the exponential sinusoid and $n_i$ the number of revolutions around the Sun.

### 2.3 Subdivision Techniques

The subdivision techniques in (16), (7) have been primarily designed for MOPs without equality constraints. Algorithms of this type start with a compact subset $Q \subset \mathcal{S}$ of the parameter space, typically with one or more $n$-dimensional
boxes. Each box gets subdivided into a set of smaller boxes, and according to certain conditions it is decided which box could contain a part of the Pareto set and is thus interesting for further investigation. The other, unpromising boxes are discarded from the collection. This process, i.e., subdivision and selection, is performed on the current box collection until the desired granularity of the boxes is reached. The approach is of global nature, that is, in principle capable of detecting the entire set of Pareto points, see Fig. 2 for an example. Subdivision algorithms are very effective for the numerical treatment of moderate dimensional models, but, however, may become inefficient compared other approaches like e.g. evolutionary strategies for higher-dimensional models. To combine the strength of both approaches, a combination of subdivision techniques with evolutionary strategies has been proposed in (18). Though it was not required to use this hybrid method for the numerical results presented in the sequel, it is an interesting candidate for the treatment of further (higher-dimensional) models. Note that the initial box collection – as well as any further collections – (a) can in principle consist of any number of boxes and that (b) the collection does not have to form one connected component. Both observations are very important for our purpose.

Figure 2. Application of DS-Subdivision on an MOP $F: Q \subset \mathbb{R}^3 \rightarrow \mathbb{R}^2$ ((7)), where $Q$ is defined by box-constraints.

3 LTGASP – Low Thrust Gravity Assist Space Pruning

In this section we propose the LTGASP algorithm which is designed to efficiently detect and prune infeasible parts of the space of a given MLTGA problem.
3.1 The MLTGA Problem Formulation

In the following we consider the MOP which is a composition of (2.2) and the flight time. To be more precise, given an arbitrary but fixed sequence \((n_1, \ldots, n_N)\) of numbers of revolutions we consider:

\[
\begin{aligned}
\text{minimise:} & \quad \begin{cases} 
J(\tilde{y}) \\
t_N - t_0
\end{cases} \\
\text{subject to:} & \quad r_p \geq r_{\text{min}}
\end{aligned}
\tag{3.1}
\]

where the solution vector \(\tilde{y}\) is given by:

\[\tilde{y} = [t_0, T_1, T_2, \ldots, T_N, k_{2,1}, \ldots, k_{2,N}],\]

and where the parameters have the following ranges:

\[t_0 \in \mathcal{I}_0,\]
\[T_i \in \mathcal{I}_i, \quad i = 1, \ldots, N,\]
\[k_{2,i} \in \mathcal{I}_{k_{2,i}}, \quad i = 1, \ldots, N.\]

Define \(\mathcal{I}\) as the entire search space, i.e.

\[\mathcal{I} := \mathcal{I}_0 \times \ldots \mathcal{I}_N \times \mathcal{I}_{k_{2,1}} \times \ldots \mathcal{I}_{k_{2,N}}.\tag{3.2}\]

Introducing the map (analogue to (9))

\[f : x = [t_0, T_1, \ldots, T_N] \rightarrow X = [t_0, t_1, \ldots, t_N],\]

defined by the component wise relation \(t_i = t_0 + \sum_{j=0}^{i} T_j, \quad i = 0, \ldots, N,\) and setting

\[\mathcal{I}^* := f(\mathcal{I}) \times \mathcal{I}_{k_{2,1}} \times \ldots \times \mathcal{I}_{k_{2,N}}\]

we can reformulate (3.1) as:

\[
\begin{aligned}
\text{minimise:} & \quad \begin{cases} 
J(X) \\
t_N - t_0
\end{cases} \\
\text{subject to:} & \quad r_p(X) \geq r_{\text{min}}
\end{aligned}
\tag{3.3}
\]
Here we give some notations which are helpful for the statement of the different pruning techniques. Every (feasible) trajectory from planet $p_{i-1}$ to planet $p_i$ in the current setting is determined by the parameters $t_{i-1}, t_i$, and $k_{2,i}$. Given these three values, denote the resulting trajectory from $p_{i-1}$ to $p_i$ by

$$T(t_{i-1}, t_i, k_{2,i}).$$

Further, denote by $D(I^*_{i})$ and $D(I_{k_{2,..}})$ the discretisations of $I^*_i$ and $I_{k_{2,..}}$. Thus, the entire discretised search space is given by

$$D(I^*) = D(I^*_0) \times \ldots \times D(I^*_i) \times D(I_{k_{2,..}}) \times \ldots \times D(I_{k_{2,N}}).$$

Now we are in the position to state the pruning techniques which will be done in the following.

### 3.2 The Pruning Techniques

In the following we will propose the pruning techniques which are used for the LTGASP algorithm.

**Initialisation.** Mark all $t_i \in D(I^*_i)$, $i = 0, \ldots, n$, as valid as well as all trajectories

$$T(t_{i-1}, t_i, k_{2,i}), \quad \forall t_{i-1} \in D(I^*_{i-1}), \quad t_i \in D(I^*_i), \quad k_{2,i} \in D(I_{k_{2,..}}), \quad i = 1, \ldots, N.$$  

**$\Delta V$ constraining.** The maximal allowable $\Delta V_i$ is the main pruning criterion of the LTGASP algorithm in phase $i$. It works on the sampled space $D(I^*_{i-1}) \times D(I^*_i) \times D(I_{k_{2,..}})$ and prunes out all those points corresponding to trajectories having a velocity change larger than a given budget $\Delta V^\text{max}_i$. Algorithm 1 describes the $\Delta V$ pruning for the transfer from planet $p_{i-1}$ to planet $p_i$, i.e. for phase $i$. Denote by $\Delta V_i(T)$ the velocity change required by a given trajectory $T$.

**Departure velocity constraining.** This criterion prunes out all trajectories where the departure velocity (and thus the corresponding thrust required by the spacecraft) is larger than a given threshold.

**Forward pruning.** An application of the $\Delta V$ pruning in each phase typically reduces the search space volume of an MLTGA problem significantly. As a consequence many values of the arrival time $t_i$ in phase $i$ become nonfeasible departure times in phase $i + 1$. To be more precise: if there is no feasible
Algorithm 1 ΔV pruning

1: for all valid \( t_{i-1} \in D(I^*_i) \) do
2:   for all valid \( t_i \in D(I^*_i) \) do
3:     if \( t_i - t_{i-1} \in \mathcal{I}_i \) then
4:       for all \( k_{2,i-1} \in D(I^*_k_{i-1}) \) do
5:         if \( \Delta V(T(t_{i-1}, t_i, k_{2,i})) > \Delta V_{max} \) then
6:           mark \( T(t_{i-1}, t_i, k_{2,i}) \) as not valid.
7:       end if
8:     end for
9:   end if
10: end for
11: end for

trajectory that arrives at a planet on a given date because they have all been pruned out according to the various criteria introduced, then there will be no departures from that planet on that date. Thus, all the corresponding points will also be pruned.

Algorithm 2 describes the forward pruning from phase \( i \) to phase \( i + 1 \) with respect to \( t_i \in D(I^*_i) \).

Algorithm 2 Forward pruning

1: for all valid \( t_i \in D(I^*_i) \) do
2:   If \( T(t_{i-1}, t_i, k_{2,i}) \) is not valid for all
3:     \( (t_{i-1}, k_{2,i}) \in D(I^*_i) \times D(I^*_k) \), mark \( t_i \) as not valid as well as all trajectories \( T(t_i, t_{i+1}, k_{2,i+1}) \)
4: end for

Backward constraining. This technique is analogue to the previous one: clearly, if a departure time in phase \( i + 1 \) becomes infeasible because of pruning, also the relative arrival date in phase \( i \) has to be pruned out.

Algorithm 3 describes the backward pruning from phase \( i + 1 \) back to phase \( i \) with respect to \( t_i \in D(I^*_i) \).

Algorithm 3 Backward pruning

1: for all valid \( t_i \in D(I^*_i) \) do
2:   If \( T(t_i, t_{i+1}, k_{2,i+1}) \) is not valid for all \( (t_{i+1}, k_{2,i+1}) \in D(I^*_{i+1}) \times D(I^*_k_{i+1}) \), mark \( t_i \) as not valid as well as all trajectories \( T(t_{i-1}, t_i, k_{2,i}) \)
3: end for
Gravity assist maximum thrust constraint. The gravity assist maximum thrust constraint prunes the trajectories having a difference between incoming velocities of trajectories in phase \(i\) (denote this velocity by \(V_i^{e\text{nd}}(T)\) for a given trajectory \(T\)) and outgoing velocities of trajectories in phase \(i+1\) (denote by \(V_{i+1}^{s\text{tart}}(T)\)) during a gravity assist larger than some threshold, \(A_v\). This threshold has to be set separately for each gravity assist. Further, an appropriate tolerance, \(L_v\), based on the Lipschitzian constant of the current phase plot has to be taken into account. Algorithm 4 describes the gravity assist maximum thrust constraint pruning between phase \(i\) and phase \(i+1\).

\begin{algorithm}
\begin{algorithmic}[1]
\State \textbf{for all} valid \(t_i \in D(I^*_i)\) \textbf{do}
\State \hspace{1em} \(v_{\text{min}}^f := \min_{t_{i-1},k_2,i} V_{\text{end}}(T(t_{i-1},\bar{t}_i,k_{2,i}))\) \Comment{forward}
\State \hspace{1em} \(v_{\text{max}}^f := \max_{t_{i-1},k_2,i} V_{\text{end}}(T(t_{i-1},\bar{t}_i,k_{2,i}))\)
\State \hspace{1em} \textbf{for all} valid \(t_{i+1} \in D(I^*_i+1)\) \textbf{do}
\State \hspace{2em} \textbf{for all} valid \(k_{2,i+1} \in D(I^*_{k_{2,i+1}})\) \textbf{do}
\State \hspace{3em} \textbf{if} \(V_{i+1}^{s\text{tart}}(T(\bar{t}_i,t_{i+1},k_{2,i+1})) \notin [v_{\text{min}}^f - A_v - L_v, v_{\text{max}}^f + A_v + L_v]\) \textbf{then}
\State \hspace{4em} \text{mark} \(T(\bar{t}_i,t_{i+1},k_{2,i+1})\) as not valid.
\State \hspace{3em} \textbf{end if}
\State \hspace{2em} \textbf{end for}
\State \hspace{1em} \textbf{end for}
\State \hspace{1em} \(v_{\text{min}}^b := \min_{t_{i+1},k_{2,i+1}} V_{i+1}^{s\text{tart}}(T(\bar{t}_i,t_{i+1},k_{2,i+1}))\) \Comment{backward}
\State \hspace{1em} \(v_{\text{max}}^b := \max_{t_{i+1},k_{2,i+1}} V_{i+1}^{s\text{tart}}(T(\bar{t}_i,t_{i+1},k_{2,i+1}))\)
\State \hspace{1em} \textbf{for all} valid \(t_{i-1} \in D(I^*_{i-1})\) \textbf{do}
\State \hspace{2em} \textbf{for all} valid \(k_{2,i} \in D(I^*_{k_{2,i}})\) \textbf{do}
\State \hspace{3em} \textbf{if} \(V_{\text{end}}(T(t_{i-1},\bar{t}_i,k_{2,i})) \notin [v_{\text{min}}^b - A_v - L_v, v_{\text{max}}^b + A_v + L_v]\) \textbf{then}
\State \hspace{4em} \text{mark} \(T(t_{i-1},\bar{t}_i,k_{2,i})\) as not valid.
\State \hspace{3em} \textbf{end if}
\State \hspace{2em} \textbf{end for}
\State \hspace{1em} \textbf{end for}
\end{algorithmic}
\end{algorithm}

Gravity assist angular constraint. The gravity assist angular constraint prunes infeasible swingbys from the search space on the basis of them being associated with a hyperbolic periapse under the minimum safe distance for the given gravity assist body.
This is determined over every arrival date \( t_i \in D(I_i^*) \) as follows: for all incoming trajectories \( T(t_{i-1}, \bar{t}_i, k_{2,i}) \) and all outgoing trajectories \( T(\bar{t}_i, t_{i+1}, k_{2,i+1}) \) check if the corresponding swingby is valid. In this case mark both incoming and outgoing trajectory as valid. Finally (i.e. after going through all arrival dates), all trajectories not marked as valid by this procedure will be pruned out.

Algorithm 5 describes the gravity assist angular constraint pruning between phase \( i \) and phase \( i + 1 \).

**Algorithm 5 Gravity assist angular constraint pruning**

1: for all \( \bar{t}_i \in D(I_i^*) \) do
2: for all valid incoming trajectories \( T(t_{i-1}, \bar{t}_i, k_{2,i}) \) do
3: for all valid outgoing trajectories \( T(\bar{t}_i, t_{i+1}, k_{2,i+1}) \) do
4: if the swingby for \( T(t_{i-1}, \bar{t}_i, k_{2,i}) \) and \( T(\bar{t}_i, t_{i+1}, k_{2,i+1}) \) is valid then
5: mark \( T(t_{i-1}, \bar{t}_i, k_{2,i}) \) as valid
6: mark \( T(\bar{t}_i, t_{i+1}, k_{2,i+1}) \) as valid
7: end if
8: end for
9: end for
10: end for
11: Invalidate all trajectories not marked as valid

**Breaking manoeuvre constraint.** As well as the departure velocity constraint, it is logical to add a constraint on the maximum breaking manoeuvre that a spacecraft can perform and prune out trajectories with an exceedingly high fuel demand.

### 3.3 The LTGASP Algorithm

Having stated the different pruning techniques we are now able to state the complete pruning algorithm.

Given an MLTGA problem (3.3), the LTGASP algorithm for the search space reduction reads as follows:

(0) perform the initialisation process.

(1) perform the \( \Delta V \) pruning, departure velocity pruning as well as the forward pruning (one ‘phase shift’) for phase 1.

(2) for \( l = 2, \ldots, n - 1 \)
   (a) perform the \( \Delta V \) pruning for phase \( l \).
(b1) perform the backward pruning from phase \( l \) down to phase 1.
(b2) perform the forward pruning from phase 1 up to phase \( l + 1 \).
(c) perform the gravity assist pruning for phases \( l - 1 \) and \( l \).
(d) perform the angular constraint pruning for phases \( l - 1 \) and \( l \).

(3) perform all the pruning steps described in step (2) plus the breaking manoeuvre constraint for phase \( n \).

Remark 3.1 This is just one possible way to combine the different pruning techniques. Note that the steps 2(a), 2(b) and 2(c) can be interchanged, and that the outcome of the resulting pruning algorithm depends on this choice. However, since the angular constraint pruning is the most time consuming technique (see below), it is logical to apply this technique at last in each phase. Further, it is also possible e.g. to apply the backward/forward pruning after each crucial pruning criterion (such as the angular constraint pruning). This technique is typically quite effective and has – in general – a running time which is almost negligible compared to the other pruning criteria.

4 Time and Space Complexity for the LTGASP Algorithm

This section determines the time and space complexity of the LTGASP algorithm. It will be shown that LTGASP scales quadratically in space and quintic in time with respect the number of gravity assist manoeuvres considered. For simplicity, the following analysis assumes that the initial launch window and all phase times are the same.
4.1 Space Complexity

Consider a launch window, a mission phase time, and the range of the $k'_s$ discretised into $l$ bins. Thus, for the first phase $l^3$ Lambert problems have to be sampled. Since the number of possible times the planet may be arrived at in phase $i$, $i = 1, \ldots, n$ can be assumed to be $(i + 1)l$ (see (9)) the $i$-th phase will require an amount of $(i + 1)l \cdot l \cdot l = (i + 1)l^3$ Lambert function evaluations (given by the discretisations of the departure times, the time of flight and $k_2$). This gives the series

$$O(n) = l^3 + 2l^3 + \ldots + nl^3 = l^3 \frac{n(1 + n)}{2}.$$ 

Therefore, the amount of space required for $n$ phases is only of order $O(n^2)$, rather than $O(k^{2n+1})$ for full grid sampling.

Similarly, the space complexity with respect to the resolution $l$ is of the order $O(l^3)$.

4.2 Time Complexity

The memory space requirement is directly proportional to the maximum number of Lambert problems that must be solved, and hence the time complexity of the sampling portion of the LTGASP algorithm must also be of the order $O(n^2)$.

For the further time complexity analysis we make the following assumptions (see above):

$$|D(I^*_i)| = (i + 1)l, \quad i = 0, \ldots, n,$$

$$|D(I_{k_2,i})| = l, \quad i = 1, \ldots, n. \quad (4.1)$$

**$\Delta V$ constraint complexity.** The $i$-th step requires of the order of $i^2l^3$ operations since by (4.1) it follows that $|D(I^*_i)| \cdot |D(I^*_i)| \cdot |D(I_{k_2,i})| \approx i^2l^3$. Thus, the time complexity applying the $\Delta V$ pruning is $O(n^3)$ with respect to the dimensionality and $O(l^3)$ with respect to the resolution.

**Forward and backward pruning complexity.** The forward pruning requires of the order of $i^2l^3$ flops for one ‘phase shift’ (i.e. the pruning of the departure times for phase $i + 1$ by analysing the data of phase $i$). Since for every phase $i$ there are $i$ such phase shifts (i.e. after the $\Delta V$ pruning of phase $i$ and after
the backward pruning), for this pruning criterion an amount of
\[
\sum_{j=1}^{n} \sum_{i=1}^{j} i^2 l^3 = l^3 \sum_{j=1}^{n} \frac{j(j+1)(2j+1)}{6}
\]
flops is required. Thus, the complexity of the forward pruning is \(O(n^4)\) with respect to the dimensionality and \(O(l^3)\) with respect to the resolution. Analogously, the complexity of the backward pruning is of the same order.

**Gravity assist thrust constraint complexity.** The \(i\)-th step requires of the order of \(2i^2 l^3\) operations. Thus, the time complexity applying the gravity assist thrust constraint is \(O(n^3)\) with respect to the dimensionality and \(O(l^3)\) with respect to the resolution.

**Gravity assist angular constraint complexity.** The \(i\)-th step requires of the order of \(2i^3 l^5\) operations. Thus, the time complexity applying the gravity assist angular constraint is \(O(n^4)\) with respect to the dimensionality and \(O(l^5)\) with respect to the resolution.

**Overall time complexity.** The overall complexity, taken from the most complex part of the algorithm (the gravity assist angular constraint), is quintic with respect to the resolution and quartic with respect to the dimensionality.

5 Attacking the MOPs

Here we propose one possible way to attack an MOP of the form (3.3) which involves the pruning techniques presented above. Obviously, these techniques have to be performed before the optimisation can start\(^1\). More interesting is how this candidate set, which can consist of up to hundreds of different connected components of different shape and size, can be utilised for the optimisation algorithm.

We propose do proceed in the following way (which is similar in spirit to (9) for the MGA case):

1. Perform the LTGASP algorithm on the given setting. Denote the resulting candidate set by \(C\).
2. Construct a box collection \(R\) starting from \(C\) such that all boxes are mutually non-intersecting and that \(R\) covers \(C\) "tightly" (see discussion below).

\(^1\)Here we consider sequential algorithms.
(3) Perform the subdivision techniques described in Section 2 using $R$ as domain.

Before we can present some numerical results in the next section we have to make some remarks on the approach:

A $n$-dimensional box $B$ can be represented by bounds $l, u \in \mathbb{R}^n$:

$$B = B_{l,u} = \{ x \in \mathbb{R}^n : l_i \leq x_i \leq u_i \forall i = 1, \ldots, n \}.$$ 

One way to construct a box collection $R$ which covers a candidate set $C$ is by looking at the connected components of the logical three-dimensional matrices $A_1$ which correspond to the $i$-th phase: set $a_{jkl} = 0$ if $T(t_j, t_k, k_2)$ is detected as not valid for the underlying sequence, where $t_j$ is the $j$-th element of $|D(I^*_{i-1})|$ ($t_k$ and $k_2$ analogous); else set $a_{jkl} = 1$. The boxes can e.g. be selected by taking the minimal and maximal coordinate values of each connected component (see Fig. 4 for an example). The corresponding $n$-dimensional boxes can be constructed on the basis of this sequence of three-dimensional boxes, and overlapping boxes have to be merged together. For a given set $C$ there does typically not exist one ‘ideal’ box collection as the following discussion shows: in order to speed up the computation, it is desired to keep the number of boxes small (since all boxes have to be evaluated). This can be done e.g. by merging ‘neighbored’ boxes together. However, since this increases the volume of $R$ the probability of picking infeasible solutions in the run of the search procedure will increase which will in turn decrease its performance. To avoid this, smaller boxes can be considered which will in turn increase the number of boxes required to cover $C$. As an example consider the set in Fig. 5 (b), which consists of 57 connected components. If components which are merely separated by one discretisation step are merged together\(^1\) the resulting box collection consists of 5 boxes, which is probably the best trade off solution for this case.

For the stability of the transfer and due to the (long) resulting transfer times, up to date only few celestial bodies are involved for ‘real’ missions. Thus, the resulting domain of a given trajectory design problem can be considered to be low or moderate dimensional (say, $\leq 15$). We have chosen to use subdivision techniques for the approximation of the Pareto sets since the algorithms can easily cope with the disconnected domain and since they have proven their efficiency on many applications so far (e.g., (7), (17)).

---

\(^1\)That is, if in the logical matrix described above elements $a_{j,k,l}$ are set to 1 if $a_{j-1,k,l} = 1 = a_{j+1,k,l}$, analogous with $k$ and $l$. 
On the other hand, regarding the dimensionality of the models, certainly also other approaches can in principle serve to produce satisfying results. However, note that due to the particular structure of the domain an application is in most cases not straightforward. For instance, for multi-objective evolutionary algorithms (MOEAs, see e.g., (3)), the probably most widely used class of algorithms in this field, there does ad hoc exist no suitable crossover operator since points from different connected components of the feasible set (or from different boxes) do typically not have similar characteristics. Further, a suitable penalisation strategy is also not straightforward.

Figure 4. Relaxation of the candidate set (right) by using boxes (right), which are much easier to handle for most optimisation algorithms.

6 Numerical Results

Here we present some numerical results coming from two different settings. All computations have been done on an Intel Xeon 3.2 Ghz processor using the programming language MATLAB.

6.1 Sequence EVM

First we consider the two-phase sequence Earth – Venus – Mercury. Using the parameters shown in Table 1 the LTGASP algorithm computes the candidate set displayed in Fig. 5. Starting with a box collection containing 24 boxes and using the subdivision techniques described above the Pareto front shown in Fig. 6 was obtained (using a budget of $N = 500,000$ function calls). The huge diversity in the values of the front indicates that the multi-objective approach
Table 1. Parameter settings for the pruning of the EVM sequence.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>sequence</td>
<td>Earth – Venus – Mercury</td>
</tr>
<tr>
<td>launch window</td>
<td>[700, 1300] (days after 01.01.2000)</td>
</tr>
<tr>
<td>time of flight phase 1</td>
<td>[100, 800] (days)</td>
</tr>
<tr>
<td>time of flight phase 2</td>
<td>[500, 1500] (days)</td>
</tr>
<tr>
<td>$</td>
<td>D(I_0^n)</td>
</tr>
<tr>
<td>$</td>
<td>D(I_1^n)</td>
</tr>
<tr>
<td>$</td>
<td>D(I_2^n)</td>
</tr>
<tr>
<td>range of $k_{2,i}$, $i = 1, 2$</td>
<td>[0.01, 2]</td>
</tr>
<tr>
<td>$</td>
<td>D(I_{k_{2,1}})</td>
</tr>
<tr>
<td>$</td>
<td>D(I_{k_{2,2}})</td>
</tr>
<tr>
<td>$\Delta V_{1,\text{max}}$</td>
<td>5 (m/s)</td>
</tr>
<tr>
<td>$\Delta V_{2,\text{max}}$</td>
<td>5 (m/s)</td>
</tr>
<tr>
<td>max. departure velocity</td>
<td>50 (m/s)</td>
</tr>
<tr>
<td>max. terminal velocity</td>
<td>100 (m/s)</td>
</tr>
<tr>
<td>$r_{p_{\text{min}}} \text{ Venus}$</td>
<td>6750 (m)</td>
</tr>
</tbody>
</table>

Table 2. Function calls and running time for sequence EVM.

<table>
<thead>
<tr>
<th>Function calls</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>exposin-calls</td>
<td>156,864</td>
</tr>
<tr>
<td>gravity assists</td>
<td>36,534</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Running times</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Delta V$ pruning 1. phase</td>
<td>3940.79 sec.</td>
</tr>
<tr>
<td>$\Delta V$ pruning 2. phase</td>
<td>3884.91 sec.</td>
</tr>
<tr>
<td>backward/forward pruning</td>
<td>0.03 sec.</td>
</tr>
<tr>
<td>max. thrust pruning</td>
<td>15.15 sec.</td>
</tr>
<tr>
<td>ang. constr. pruning</td>
<td>9613.19 sec.</td>
</tr>
<tr>
<td>2nd backward/forward pruning</td>
<td>0.03 sec.</td>
</tr>
<tr>
<td>final backw./forw. pruning</td>
<td>5.93 sec.</td>
</tr>
<tr>
<td>total running time</td>
<td>17460.03 sec.</td>
</tr>
</tbody>
</table>

is indeed interesting for this mission: the total time of flight varies between 1200 and 1800 days which makes a difference of more than 1.5 years, and the mass fraction varies by 16 %, which is a significant value for such missions.

6.2 Sequence EVEJ

Next we consider the three-phase sequence Earth – Venus – Earth – Jupiter. An application of the LTGASP algorithm using the parameter settings shown in Table 4 leads to the candidate set which is shown in Fig. 7. Starting with a total of 8 boxes, the Pareto front displayed in Fig. 8 was obtained using $N = 700,000$.
function evaluations. For a comparative study with another approach we have taken the multi-objective particle swarm optimisation algorithm proposed in (4), which was given the same budget of function evaluations albeit using the entire search space (3.2) as domain (i.e., without pruning). A comparative result can be seen in Fig. 8. For sequences which involve more celestial bodies – which will be interesting for future mission designs – more drastic differences in the performance are expected.
Table 3. Parameter settings for the pruning of the EVEJ sequence.

<table>
<thead>
<tr>
<th>sequence</th>
<th>Earth – Venus – Earth – Jupiter</th>
</tr>
</thead>
<tbody>
<tr>
<td>launch window</td>
<td>[4745, 5840] (days after 01.01.2000)</td>
</tr>
<tr>
<td>time of flight phase 1</td>
<td>[100, 200] (days)</td>
</tr>
<tr>
<td>time of flight phase 2</td>
<td>[300, 400] (days)</td>
</tr>
<tr>
<td>time of flight phase 3</td>
<td>[1000, 2000] (days)</td>
</tr>
<tr>
<td>$D(I^*_0)$</td>
<td>80</td>
</tr>
<tr>
<td>$D(I^*_1)$</td>
<td>100</td>
</tr>
<tr>
<td>$D(I^*_2)$</td>
<td>120</td>
</tr>
<tr>
<td>$D(I^*_3)$</td>
<td>100</td>
</tr>
<tr>
<td>range of $k_2,i, i=1,2,3$</td>
<td>[0.01, 2]</td>
</tr>
<tr>
<td>$D(I_{k_2,1})$</td>
<td>20</td>
</tr>
<tr>
<td>$D(I_{k_2,2})$</td>
<td>20</td>
</tr>
<tr>
<td>$D(I_{k_2,3})$</td>
<td>20</td>
</tr>
<tr>
<td>$\Delta V_{1,\text{max}}$</td>
<td>5 (m/s)</td>
</tr>
<tr>
<td>$\Delta V_{2,\text{max}}$</td>
<td>5 (m/s)</td>
</tr>
<tr>
<td>$\Delta V_{3,\text{max}}$</td>
<td>5 (m/s)</td>
</tr>
<tr>
<td>max. departure velocity</td>
<td>30 (m/s)</td>
</tr>
<tr>
<td>max. terminal velocity</td>
<td>30 (m/s)</td>
</tr>
<tr>
<td>$r_{\text{p, min}}$ Venus</td>
<td>6750 (m)</td>
</tr>
<tr>
<td>$r_{\text{p, min}}$ Earth</td>
<td>6750 (m)</td>
</tr>
</tbody>
</table>

Table 4. Function calls and running time for sequence EVEJ.

**Function calls**:  
exposin-calls : 24317  
gravity assists : 4543  

**Running times**:  
$\Delta V$ pruning 1. phase : 431.56 sec.  
backward/forward pruning : 0.03 sec.  
max. thrust pruning : 5.06 sec.  
ang. constr. pruning : 3475.93 sec.  
backward/forward pruning : 3.87 sec.  
$\Delta V$ pruning 3. phase : 1710.13 sec.  
backward/forward pruning : 0.05 sec.  
max. thrust pruning : 9.14 sec.  
ang. constr. pruning : 2373.77 sec.  
backward/forward pruning : 0.06 sec.  
final backw./forw. pruning : 5.60 sec.  
total running time : 8355.70 sec.
7 Conclusions

In this work we have presented a novel approach for the design of optimal MLTGA trajectories by using space pruning and a multi-objective approach. Based on existing pruning methods for MGA problems we have adapted and extended to the MLTGA case resulting in the LTGASP algorithm which scales quadratically in space and quintic in time with respect to the number of gravity assist manoeuvres considered. Further, we have proposed a method to attack the related multi-objective design problem (minimisation of flight time and fuel consumption) efficiently by integrating the pruning techniques in the optimisation process in a suitable way. Finally, we have presented some numerical results on two interplanetary transfers indicating the strength of our approach.
8 Acknowledgment

The authors acknowledge support from the European Space Agency through the Ariadna initiative, Ariadna study number 05/4106.

REFERENCES


REFERENCES


