Search Space Pruning and Global Optimization of Multiple Gravity Assist Trajectories with Deep Space Manoeuvres

V.M. Becerra, S.J. Nasuto, J. Anderson, M. Ceriotti and C. Bombardelli

Abstract— **This paper deals with the design of optimal multiple gravity assist trajectories with deep space manoeuvres. A pruning method which considers the sequential nature of the problem is presented. The method locates feasible vectors using local optimization and applies a clustering algorithm to find reduced bounding boxes which can be used in a subsequent optimization step. Since multiple local minima remain within the pruned search space, the use of a global optimization method, such as Differential Evolution, is suggested for finding solutions which are likely to be close to the global optimum. Two case studies are presented.**

I. Introduction

A gravity assist manoeuvre uses a celestial object's gravity in order to change a spacecraft's trajectory. When a spacecraft approaches a celestial object, a small amount of the object's orbital momentum is transferred to the spacecraft. This manoeuvre was used for the first time in the 1970's, when the spacecraft Mariner 10 used a gravity assist flybys of Venus to reach Mercury. Gravity assist manoeuvres (GAs) are frequently used to reduce fuel requirements. Most interplanetary trajectory design problems can be stated as optimization problems, where one of the fundamental goals is the minimization of fuel requirements, with consideration also given to intermediate planetary flybys, mission duration, type of arrival, launch and arrival windows, and velocity constraints. Traditionally, local optimization has been used to attempt to solve these design problems [1]. However, because of nonlinearities and the periodic motion of the planets, multiple local minima exist and, as a result, local optimization only helps to find local minima which are heavily dependent on the initial guesses employed and are not necessarily good solutions. The use of global optimization techniques has been proposed for tackling these problems, as these methods have a better chance of finding good solutions approaching the global optimum [2]. Genetic algorithms and similar techniques have been employed, but these techniques may face difficulties in tackling realistic missions due to the large size of the search space associated with these problems. A method known as GASP (Gravity Assist Space Pruning) has been proposed [3] in relation with problem of multiple gravity assist (MGA) trajectories with a known planetary sequence and no deep space manoeuvres. In such cases, it can be shown that the vast majority of the search space consists of infeasible, or very undesirable, solutions. This observation motivated the development of a method for producing reduced search spaces by pruning, thus allowing standard global optimization techniques to be applied more successfully to the reduced box bounds. The GASP method considers the sequential nature of the problem, as it prunes the search space on a phase by phase basis, and results in important computational savings, with search space reductions greater than six orders of magnitude, thus simplifying significantly the subsequent optimization. The method is based on grid sampling in two dimensions for each leg of the mission, with sequential pruning of the search space. The pruning method has been shown to have polynomial time and space complexity, so that it remains tractable as the number of decision variables increases. However, designing multiple gravity assist missions with no deep-space manoeuvres is limited in scope, since many possible trajectories cannot be considered, and, as practice shows, deep space manoeuvres are used in real missions. If the problem of multiple gravity assist with deep space manoeuvres could be pruned efficiently, then the computational cost of optimizing such trajectories may be significantly reduced. The introduction of deep space manoeuvres offers the further advantage of providing a reasonable approximation of multiple gravity assist trajectories with low-thrust arcs. If a transfer arc is no more simply ballistic but is shaped by one or more propelled manoeuvres (either impulsive or low-thrust) the number of degrees of freedom increases significantly. Hence, an efficient solution process would have to make use of additional information to reduce the number of possible alternatives (pruning the search space) so reducing the computational cost, and increasing the likelihood of finding good solutions.

This paper describes a method for pruning of the search space of multiple gravity assist optimization problems with deep space manoeuvres, for the particular case of powered swingbys. The method can be seen as an extension of the GASP method when deep space manoeuvres are considered. Since the pruned problem would still exhibit multiple local minima, the use of a global optimization method to find optimal solutions on the pruned space is proposed.

II. General problem formulation

The problem of interest may be formulated as a multi-stage optimization problem (MSOP).

MSOP: Find

$$\mathbf{x} = \left[\ \mathbf{x}_1^T, \quad \mathbf{x}_1^T, \quad \cdots, \quad \mathbf{x}_{s+1}^T \ \right]^T \in \Omega$$

V.M. Becerra, S.J. Nasuto and J. Anderson are with the School of Systems Engineering, University of Reading, UK, (phone: +44-118-3786703 fax: +44-118-3788220; email: v.m.becerra@reading.ac.uk). M. Ceriotti is with the Department of Aerospace Engineering, University of Glasgow, UK. C. Bombardelli is with the Advanced Concepts Team, European Space Agency, Noordwijk, the Netherlands

to minimise

$$f(\mathbf{x}, \mathbf{z}_1, \ldots, \mathbf{z}_{s+1})$$

subject to

$$\mathbf{z}_k = \mathbf{h}_k(\mathbf{x}_1, \ldots, \mathbf{x}_{k+1}),\ k = 1, \ldots, s$$
$$\mathbf{g}_k(\mathbf{z}_k) \leq 0,\ k = 1, \ldots, s+1$$

Ω is the Cartesian product of s+1 hyper-rectangles $\Omega = \Omega_1 \times \Omega_1 \times \cdots \times \Omega_{s+1}$, where $\Omega_k = \{\mathbf{x}_k \in \mathrm{R}^{n_k} | \mathbf{x}_L^{(k)} \leq \mathbf{x}_k \leq \mathbf{x}_U^{(k)}\},\ k = 1 \ldots, s+1$, the objective function is assumed to be scalar $f : \Omega \times \mathrm{R}^{q_1} \times \cdots \times \mathrm{R}^{q_{s+1}} \rightarrow \mathrm{R}$, the vectors $\mathbf{z}_k \in \mathrm{R}^{q_k}$, k=1,...,s+1, are intermediate variables associated with each stage, and each of the functions $\mathbf{h}_k : \Omega_1 \times \Omega_2 \times \cdots \times \Omega_{k+1} \rightarrow \mathrm{R}^{m_k}$and $\mathbf{g}_k : \mathrm{R}^{q_k} \rightarrow \mathrm{R}^{d_k}$is associated with a particular stage k. Note that the calculation of the constraint function $\mathbf{g}_k$ depends on intermediate variables calculated at stage k, and the objective function depends on the values of the intermediate variables $\mathbf{z}_k$, k=1,..., $s+1$, hence a specific order must be followed to evaluate the objective function f and the constraint functions $\mathbf{g}_k$.

The presence of inequality constraints in the MSOP requires careful consideration. Although bounds on the decision variables are easy to manage, more general inequality constraints are more difficult to handle in global optimization. A plausible method is to prune the search space based on feasibility (i.e. constraint satisfaction). This has an important benefit: the size of the search space is reduced hence simplifying the optimization task. One simple method of pruning is to grid sample the search space with a suitable resolution so that unfeasible areas can be detected by evaluating the constraint functions, and subsequently eliminated, leaving a reduced search space where optimisation can be applied. However, the cost of grid sampling with reasonable resolutions may be prohibitive when the search space dimension is larger than two or three.

The mission considered by the GASP method includes powered gravity assist at intermediate planets, and if required a braking manoeuvre at the arrival planet for orbit insertion. The problem addressed by GASP can be cast as a MSOP. Here, the decision vector $\mathbf{x}$ consists of the launch date and transfer times between planets, the intermediate variables $\mathbf{z}_k$ are the $\Delta\mathbf{v}$'s applied at each planet. The functions $\mathbf{h}_k$ represent the calculations that are required to find the intermediate variables (solution of Lambert problems, swing-by models), the objective function is the sum of the magnitudes of the $\Delta\mathbf{v}$'s. The constraint functions $\mathbf{g}_k$ are related to upper bounds on the $\Delta\mathbf{v}$'s at each planet (as thrusters have limits), as well as lower bounds for the periapsis radius at each swing-by planet (to keep a safe distance from the planet).

III. Modelling a mission leg with deep space manoeuvres

We have used a model that is able to compute the trajectory from one planet to the next planet in a mission, with an arbitrary number of intermediate deep space manoeuvres. The model requires specifying the initial and final positions of the spacecraft, the time of flight T_{of} between the planets, and the positions and the timing of all deep space manoeuvres involved, and it returns the velocity vectors at the begining and at the end of the leg, together with the magnitudes of the deep space impulsive manoeuvres.

With reference to Figure 1, the initial position of the spacecraft, $\mathbf{r}_{p1}$ can be found by an ephemeris calculation related to the initial planet given the departure date t_0. Similarly, the final position of the spacecraft $\mathbf{r}_{p2}$ can be found by an ephemeris calculation related to the next planet, given the arrival time t_{arr}, which can be calculated as $t_{\mathrm{arr}} = t_0 + T_{\mathrm{of}}$.

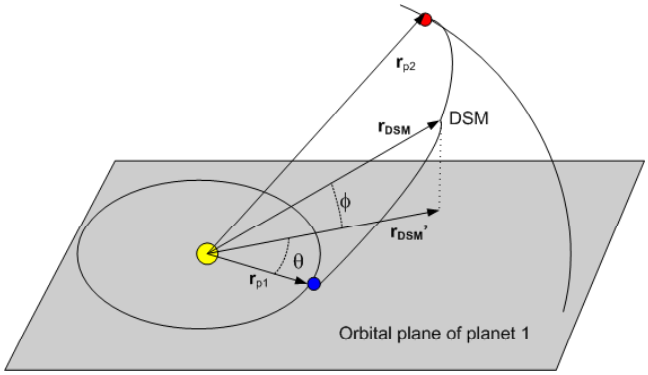

Fig. 1. Mission leg model with a deep space manoeuvre

Each deep space manoeuvre is characterized in polar coordinates with the following parameters:

- r: dimensionless distance from the sun. The vector from the sun to the deep space manoeuvre is denoted as $\mathbf{r}_{\mathrm{DSM}}$. The value r is equal to zero when $|\mathbf{r}_{\mathrm{DSM}}|$ is equal to $|\mathbf{r}_{p1}|$, and r is equal to 1 when $|\mathbf{r}_{\mathrm{DSM}}|$ is equal to $|\mathbf{r}_{p2}|$. Notice that r may be outside the interval [0,1] when the orbits of the planets are eccentric.
- θ: in-plane angle (angle between $\mathbf{r}_{p1}$ and the projection of $\mathbf{r}_{DSM}$ on the orbital plane of the first planet). This projection is denoted as $\mathbf{r}'_{\mathrm{DSM}}$ in Figure 1.
- ϕ: out of plane angle (angle between $\mathbf{r}_{\mathrm{DSM}}$ and the projection of $\mathbf{r}_{\mathrm{DSM}}$ on the orbital plane of the first planet)

In addition, the timing of the deep space manoeuvre is parameterized as a fraction $\alpha \in [0, 1]$ of the time of flight, such that the timing of the deep space manoeuvre is expressed as $\alpha \times T_{\mathrm{of}}$.

In the model, a patched conic, two-body problem is considered. The manoeuvres are assumed to be impulsive. In case of a single deep space manoeuvre, then the leg trajectory is found through the solution of two Lambert problems. We used an implementation of Battin's method for the Lambert solution. See [4], [5] for further details on the algorithms involved.

Notice that in case of a mission with multiple phases, any two consecutive deep space flight phases can be computed independently, without considering the swingby of the planet. This is due to the fact that the initial velocity is not needed to compute the Lambert arc. Rather, the initial and

final velocities are outputs from the Lambert solver. Once two consecutive legs are computed, both the incoming and outgoing velocity at the planet become available and the swingby (with the powered model) can be computed.

In this way, in order to create the whole trajectory, the only requirement is that the time at each planet is the same for all the phases arriving or departing from that planet. At this point, no constraints are considered on the incoming and outgoing velocities. Thus, it is possible to analyze (optimize, prune) the deep space flight phases first, then match them with the swingbys and prune again on the basis of the feasibility of the swingby.

This approach is not possible with a model of swingby which in some way computes the outgoing velocity, because in this case, it would be necessary to match both time and velocity in order to combine deep space flight phases with swingby phases.

IV. Modelling a powered swingby

The gravity assist calculations consist of matching the incoming and outgoing planetocentric velocities around the swingby planet, and computing a minimum passing distance, which is also known as the pericenter radius. If the pericenter radius is unacceptably low, it is possible to calculate a suitable velocity impulse to be applied to achieve a minimum desired altitude. When the manoeuvre requires such an impulse, it is known as powered swingby. A gravity assist manoeuvre is illustrated in Figure 2.

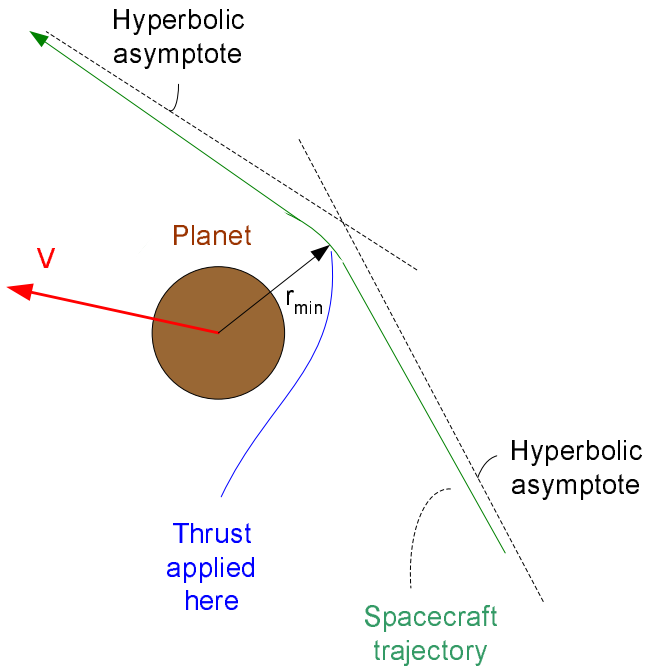

Fig. 2. Illustration of a gravity assist manoeuvre

The gravity assist model takes the planetocentric incoming and outgoing velocities, the minimum pericenter radius, and the gravitational constant of the planet, and returns the required impulsive Δv [4].

V. The GASP algorithm with powered swingbys and deep space manoeuvres

Inspired by the original GASP algorithm, its extension considering deep space manoeuvres and powered swingbys described in this section, also prunes the search space considering each phase of the mission in a sequential manner. The introduction of a deep space manoeuvres increases significantly the number of decision variables.

In order to keep the description of the method simple, we consider a two phase mission with one deep space manoeuvre at each phase, and one gravity assist manoeuvre. We also assume that a braking manoeuvre is performed for orbit insertion purposes at the arrival planet [6]. The method can be extended to more phases and more deep space manoeuvres per phase in a straightforward manner.

The notation used below is as follows. The super-index in parenthesis (e.g. $^{(1)}$, $^{(2)}$). indicates the phase of the mission. Δv_{dep} is the impulsive manoeuvre at departure, Δv_{DSM} is an impulsive manoeuvre at deep space, Δv_{ga} is an impulsive manoeuvre at a gravity assist planet, Δv_{b} is a braking manoeuvre that is performed for orbit insertion purposes, $\mathbf{v}_{\mathrm{in}}$ is the arrival velocity at a swingby planet, $\mathbf{v}_{\mathrm{out}}$ is the departure velocity from a swingby planet, t_0 is the mission launch date, t_{arr} is an arrival time at the end of one phase, t_{dep} is the departure time at the beginning of a phase, T_{of} is the time of flight of a given phase.

A. First phase

The decision vector associated with this leg is as follows:

$$\mathbf{x}_1 = [t_0,\, T_{of}^{(1)}, r^{(1)}, \alpha^{(1)}, \theta^{(1)}, \phi^{(1)}]^T \tag{1}$$

In contrast to the original GASP algorithm, where there are only two decision variables $(t_0,\, T_{of}^{(1)})$ associated with the first leg, there are now six decision variables associated with the first leg. The threefold increase in the number of decision variables makes it too expensive to carry out a grid sampling, as is shown in the first case study below.

Assume that $\mathbf{x}_1$ is initially limited to the hyper-rectangle $\Omega_1 \subset \Re^6$, such that each element of $\mathbf{x}_1$ is initially bounded between lower and upper limits: $x_{1,j}^l \leq x_{1,j} \leq x_{1,j}^u$, $j = 1, \ldots, 6$.

We are interested in pruning the search space by finding an estimate of the feasible regions of the search space associated with the first leg, with respect to constraints on the departure impulse $\Delta v_{\mathrm{dep}}^{(1)}$, and the deep space manoeuvre impulse $\Delta v_{\mathrm{DSM}}^{(1)}$. To find such an estimate, we propose the use of a local optimization algorithm, which can be started from multiple random vectors within the admissible region Ω_1, and stopped when feasible vectors are found. This procedure is then followed by the application of a clustering algorithm to find an estimate of the region. We have used in this work the mean shift clustering algorithm [7], which does not require an a priori specification of the number of clusters. Thus the algorithm to prune the search space associated with the first phase is as follows:

Algorithm 1: Pruning the first phase:

1) Generate randomly N_1 starting vectors within the admissible region for the first phase: $\bar{\mathbf{x}}_1^i \in \Omega_1 \subset \Re^6$, $i = 1, \ldots, N_1$.
2) Start a constrained local optimization algorithm, such as sequential quadratic programming, from each initial vector $\bar{\mathbf{x}}_1^i$, $i = 1, \ldots, N_1$ to solve the following problem:

$$\min_{\mathbf{x}_1} f_1(\mathbf{x}_1) = \Delta v_{\text{dep}}^{(1)} + \Delta v_{\text{DSM}}^{(1)} \tag{2}$$

subject to

$$\begin{aligned} \begin{bmatrix} \Delta v_{\text{dep}}^{(1)} \\ \Delta v_{\text{DSM}}^{(1)} \end{bmatrix} &= \mathbf{h}_1(\mathbf{x}_1) \\ \Delta v_{\text{dep}}^{(1)} &\leq \Delta v_{\text{dep}}^{\max} \\ \Delta v_{\text{DSM}}^{(1)} &\leq \Delta v_{\text{DSM}}^{\max} \end{aligned} \tag{3}$$

where $\Delta v_{\text{dep}}^{(1)}$ is the impulsive manoeuvre at launch, $\Delta v_{\text{DSM}}^{(1)}$ is the impulsive manoeuvre performed at deep space during phase 1. Note that the constrained optimization algorithm may be stopped as soon as a feasible vector satisfying the inequality constraints is found. If a feasible vector is found, it is recorded as $\hat{\mathbf{x}}_1^i$. If no feasible vector is found starting from $\bar{\mathbf{x}}_1^i$, then the optimization starts again with the next value of i. This step ends with a collection of feasible vectors $\hat{\mathbf{x}}_1^i$ $i = 1, \ldots, M_1$, where $M_1 \leq N_1$. If no feasible vectors are found, the algorithm stops.
3) Given the set of feasible vector found in step 2, run the clustering algorithm to find a set of P_1 clusters, so that each vector $\hat{\mathbf{x}}_1^i$ is assigned to a cluster C_j, where $i = 1, \ldots, M_1$ and $j = 1, \ldots, P_1$.
4) This step uses the information in the clusters to form one bounding box for each cluster C_j, $j = 1, \ldots, P_1$. Let $\mathbf{x}^{\min,j} \in \Re^6$ be a vector so that each of its elements $x_i^{\min,j}$ is the minimum i-th coordinate value for all the vectors in cluster C_j. Similarly, let $\mathbf{x}^{\max,j} \in \Re^6$ be a vector so that each of its elements $x_i^{\max,j}$ is the maximum i-th coordinate value for all the vectors in cluster C_j. Then the bounding box for cluster j is defined as

$$B_j^{(1)} = \{\mathbf{x} \in \Omega_1 \subset \Re^6 | \, \mathbf{x}^{\min,j} \leq \mathbf{x} \leq \mathbf{x}^{\max,j}\}. \tag{4}$$

Note that each $B_j^{(1)}$ is a subset of Ω_1, the original search space for the decision variables associated with phase 1. The set of bounding boxes $B_j^{(1)}$, $j = 1, \ldots, P_1$ represents the initially pruned search space for phase 1 (this set is updated later, in a backward constraining step). Denote $\mathcal{B}^{(1)}$ as the set of bounding boxes B_j, $j = 1, \ldots, P_1$.

B. Second phase

We are using the patched conic approach to model the trajectory, so that the time of arrival at the end of the first phase is identical to the time of departure for the second phase. In other words, it is assumed that the gravity assist manoeuvre occurs instantaneously. We have found reduced bounding boxes $B_j^{(1)}$, $j = 1, ...P_1$ for the decision variables associated with phase 1.

Given values for the launch time t_0 and the time of flight for the first leg $T_{\text{of}}^{(1)}$, the time of arrival at the end of phase 1, which is equal to the time of departure for phase 2, is given by:

$$t_{\text{arr}}^{(1)} = t_{\text{dep}}^{(2)} = t_0 + T_{\text{of}}^{(1)} \tag{5}$$

We have found in phase 1 a set of intervals of feasible values for the arrival time $t_{\text{arr}}^{(1)}$. Such intervals are derived from the first two co-ordinates of the bounding boxes for phase 1, $B_j^{(1)}$, $j = 1, \ldots, P_1$. Since $t_{\text{arr}}^{(1)} = t_{\text{dep}}^{(2)}$ then it only makes sense to consider values of $t_{\text{dep}}^{(2)}$ within the same intervals. This was the main idea that was exploited in the design of the original GASP method [3]. Let us denote the feasible intervals for $t_{\text{dep}}^{(2)}$ as $\mathcal{I}_j, j = 1, \ldots, P_1$. Note that such intervals may, in general, overlap. Denote $\mathcal{I}$ as the union of all intervals $\mathcal{I}_j$, $j = 1, \ldots, P_1$.

Given that we are assuming a powered swingby, the arrival and departure velocities are decoupled (the departure velocity does not depend on the arrival velocity, provided any bound constraints on the Δv magnitude are not hit), so we can compute the second phase without having computed first the powered swingby. Assume that the second phase also involves a single deep space manoeuvre, so that the decision vector for the second phase is:

$$\mathbf{x}_2 = [T_{\text{of}}^{(2)}, r^{(2)}, \alpha^{(2)}, \theta^{(2)}, \phi^{(2)}]^T \tag{6}$$

where $T_{\text{of}}^{(2)}$ represents the time of flight for the second leg, and $\{r^{(2)}, \alpha^{(2)}, \theta^{(2)}, \phi^{(2)}\}$ are parameters associated with the deep space manoeuvre that takes place in the second phase. Let us denote the initial admissible region for $\mathbf{x}_2$ as Ω_2.

In order to compute the second phase, it is necessary to specify values for $t_{\text{dep}}^{(2)}$ and $\mathbf{x}_2$. Define an augmented vector associated with the second phase as follows:

$$\mathbf{X}^{(2)} = [t_{\text{dep}}^{(2)}, \mathbf{x}_2^T]^T \in \Re^6 \tag{7}$$

Let us define the admissible region for this vector as $\bar{\Omega}_2 = \mathcal{I} \times \Omega_2$.

We can formulate the pruning algorithm for the second phase as follows.

Algorithm 2: Pruning the second phase:

1) Generate randomly N_2 starting vectors within the admissible region for the second phase: $\bar{\mathbf{X}}_2^i \in \bar{\Omega}_2 \subset \Re^6$, $i = 1, \ldots, N_2$.
2) Start a constrained local optimization algorithm, such as sequential quadratic programming, from each initial vector $\bar{\mathbf{X}}_2^i$, $i = 1, \ldots, N_2$ to solve the following problem:

$$\min_{\mathbf{X}_2} f_2(\mathbf{X}_2) = \Delta v_{\text{DSM}}^{(2)} + \Delta v_{\text{b}}^{(2)} \tag{8}$$

subject to

$$\begin{aligned} \begin{bmatrix} \Delta v_{\mathrm{DSM}}^{(2)} \\ \Delta v_{b}^{(2)} \end{bmatrix} &= \mathbf{h}_2(\mathbf{X}_2) \\ \Delta v_{\mathrm{DSM}}^{(2)} &\leq \Delta v_{\mathrm{DSM}}^{\max} \\ \Delta v_{b}^{(2)} &\leq \Delta v_{b}^{\max} \end{aligned} \tag{9}$$

where $\Delta v_{\mathrm{DSM}}^{(2)}$ is the deep space manoeuvre, and $\Delta v_{\mathrm{b}}^{(2)}$ is the braking manoeuvre at the final planet. Note that the constrained optimization algorithm may be stopped as soon as a feasible vector satisfying the inequality constraint is found. If a feasible vector is found, it is recorded as $\hat{\mathbf{X}}_2^i$. If no feasible vector is found starting from $\bar{\mathbf{X}}_2^i$, then the optimization starts again with the next value of i. This step ends with a collection of feasible vectors $\hat{\mathbf{X}}_2^i$ $i = 1, \ldots, M_2$, where $M_2 \leq N_2$. If no feasible vectors are found, the algorithm stops.

3) This step checks the feasibility of each of the vectors found in Step 2 with respect to the gravity assist manoeuvre. From each of the vectors found in Step 2, $\hat{\mathbf{X}}_2^i$ $i = 1, \ldots, M_2$, extract the departure time $t_{dep}^{(2,i)}$, and take the corresponding departure velocity vector $\mathbf{v}_{\mathrm{out}}^{(2,i)}$ (which is computed as part of the evaluation of the second leg). Then, given values for $t_{dep}^{(2,i)}$, and $\mathbf{v}_{\mathrm{out}}^{(2,i)}$, start a constrained local optimizer from N_3 randomly generated vectors $\mathbf{x}_1^j \in \mathcal{B}^{(1)}$, $j = 1, \ldots, N_3$, to solve the following problem:

$$\min_{\mathbf{x}_1} f_1(\mathbf{x}_1) = \Delta v_{\mathrm{dep}}^{(1)} + \Delta v_{\mathrm{DSM}}^{(1)} \tag{10}$$

subject to

$$\begin{aligned} \mathbf{x}_1 &\in \mathcal{B}^{(1)} \\ \begin{bmatrix} \Delta v_{\mathrm{dep}}^{(1)} \\ \Delta v_{\mathrm{DSM}}^{(1)} \\ \mathbf{v}_{\mathrm{in}}^{(1)} \end{bmatrix} &= \bar{\mathbf{h}}_1(\mathbf{x}_1) \\ \Delta v_{ga}^{(1)} &= q_1(\mathbf{v}_{\mathrm{in}}^{(1)}, \mathbf{v}_{\mathrm{out}}^{(2,i)}, r_{\mathrm{min}}^{(1)}) \\ t_0 + t_{\mathrm{of}}^{(1)} - t_{\mathrm{dep}}^{(2,i)} &= 0 \\ \Delta v_{\mathrm{dep}}^{(1)} &\leq \Delta v_{\mathrm{dep}}^{\max} \\ \Delta v_{\mathrm{DSM}}^{(1)} &\leq \Delta v_{\mathrm{DSM}}^{\max} \\ \Delta v_{\mathrm{ga}}^{(1)} &\leq \Delta v_{\mathrm{ga}}^{\max} \end{aligned} \tag{11}$$

where $\mathbf{v}_{\mathrm{in}}^{(1)}$ is the arrival velocity at the gravity assist planet, $\Delta v_{\mathrm{ga}}^{(1)}$ is the impulsive manoeuvre performed at the gravity assist planet, $r_{\mathrm{min}}^{(1)}$ is the minimum allowed pericenter altitude during the gravity assist manoeuvre. If a feasible solution $\tilde{\mathbf{x}}_1$ is found out of the N_3 local optimizer runs, then $\tilde{\mathbf{x}}_1$ is stored, and the vector $\hat{\mathbf{X}}_2^i$ is confirmed as feasible, otherwise $\hat{\mathbf{X}}_2^i$ is discarded. This results in $Q_2 \leq M_2$ feasible vectors associated with the second phase, and $Q_1 \leq M_1$ feasible vectors associated with the first phase.

4) Given the set of feasible vectors found in step 3, run the clustering algorithm to find a set of P_2 clusters, so that each vector $\hat{\mathbf{X}}_2^i$, is assigned to a cluster $C_j^{(2)}$, where $i = 1, \ldots, Q_2$ and $j = 1, \ldots, P_2$.

5) This step uses the information in the clusters found in step 4 to form one bounding box for each cluster $C_j^{(2)}$, $j = 1, \ldots, P^{(2)}$. Let $\mathbf{X}^{\min,j} \in \Re^6$ be a vector so that each of its elements $X_i^{\min,j}$ is the minimum i-th coordinate value for all the vectors in cluster $C_j^{(2)}$. Similarly, let $\mathbf{X}^{\max,j} \in \Re^6$ be a vector so that each of its elements $X_i^{\max,j}$ is the maximum i-th coordinate value for all the vectors in cluster $C_j^{(2)}$. Then the bounding box for cluster j is defined as

$$\bar{B}_j^{(2)} = \{\mathbf{X} \in \bar{\Omega}_2 \subset \Re^6 | \, \mathbf{X}^{\min,j} \leq \mathbf{x} \leq \mathbf{X}^{\max,j}\}. \tag{12}$$

Note that the bounding boxes found in step 5 are associated with the augmented variable $\mathbf{X}_2$ defined in equation (7). It is straightforward to find the bounding boxes $B_j^{(2)}$ that correspond to the original variable vector $\mathbf{x}_2$. Denote the set of such boxes as $\mathcal{B}^{(2)}$.

C. Backward constraining

Given the set of feasible vectors $\tilde{\mathbf{x}}_1^k$, $k = 1, \ldots, Q_1$, which are found in Step 3 of Algorithm 2, it is possible to run again the clustering algorithm and find, in a similar way as done before, a new set of $\bar{P}_1$ bounding boxes $\bar{\mathcal{B}}^{(1)}$, $j = 1, \ldots, \bar{P}_1$, for phase 1. This usually results in the shrinking of the previously found set of boxes for phase 1, and possibly in the elimination of some of them. Denote the new set of bounding boxes for phase 1 as $\bar{\mathcal{B}}^{(1)}$, which represents the final pruned search space for phase 1.

VI. Global optimization on the pruned search space

Note that, given the connection in time between the phases (the time of arrival of the first phase is the same as the time of departure of the second phase), it is normally possible to associate each of the bounding boxes found in the initial phase with one (or possibly more) bounding boxes associated with the second stage. Each of these associations defines a solution family. Suppose that box $\bar{B}_k^{(1)}$ from the first phase is associated with box $B_l^{(2)}$ from the second phase to form a solution family with index s. Denote the search space associated with solution family s as $B_s = \bar{B}_k^{(1)} \times B_l^{(2)}$. Assume that S solution families are identified.

Note that in the case of a two phase mission with two deep space manoeuvres, the decision vector is 11-dimensional:

$$\begin{aligned} \mathbf{x} = [t_0, \, T_{of}^{(1)}, r^{(1)}, \alpha^{(1)}, \theta^{(1)}, \phi^{(1)} \cdots \\ T_{\mathrm{of}}^{(2)}, r^{(2)}, \alpha^{(2)}, \theta^{(2)}, \phi^{(2)}]^T \end{aligned} \tag{13}$$

It is normally the case the multiple local minima will be present within each solution family. Hence, it is of relevance to use a global optimization method to solve the problem. The procedure is then as follows.

Algorithm 3: Global optimization on the pruned search space

1) Define two positive integers N_4 and N_5, where $N_5 >> N_4$, and $N_4 > \dim(\mathbf{x})$. For each solution family $s = 1, ...S$, use a global optimiser to solve the following problem to N_4 iterations:

$$\min_{\mathbf{x}} f(\mathbf{x}) = \Delta v^{(1)}_{\text{dep}} + \Delta v^{(1)}_{\text{DSM}} + \Delta v^{(1)}_{\text{ga}} + \Delta v^{(2)}_{\text{DSM}} + \Delta v^{(2)}_{\text{b}} \tag{14}$$

subject to

$$\begin{aligned} \mathbf{x} &\in B_s \\ \begin{bmatrix} \Delta v^{(1)}_{\text{dep}} \\ \Delta v^{(1)}_{\text{DSM}} \\ \mathbf{v}^{(1)}_{\text{in}} \\ \mathbf{v}^{(2)}_{\text{out}} \\ \Delta v^{(2)}_{\text{DSM}} \end{bmatrix} &= \mathbf{h}(\mathbf{x}) \\ \Delta v^{(1)}_{\text{ga}} &= q_1(\mathbf{v}^{(1)}_{\text{in}}, \mathbf{v}^{(2,i)}_{\text{out}}, r^{(1)}_{\text{min}}) \end{aligned} \tag{15}$$

Notice that the bounding boxes found by the pruning method approximate the feasible regions with respect to the inequality constraints associated with the original optimization problem (mainly constraints on the various Δv magnitudes). Because of this, it is possible to ignore such constraints at the final optimization step, and simply check the solutions found for feasibility with respect to such constraints. This is what we have done in the case studies presented below. Alternatively, the inequality constraints could be considered by the global optimization algorithm using methods such as those described in [8].

This results in S (sub-optimal) decision vectors $\mathbf{x}_k, k = 1, \ldots, S$, one for each solution family. Each of these solution vectors contains useful information to the mission analyst, since each of them corresponds to the best feasible solution (with N_4 iterations) found for the corresponding solution family (notice that each solution family is associated with a feasible launch window.), and hence each solution gives an upper bound for the objective function within each solution family. Out of these solution vectors, the one that gives the lowest value of the objective function is denoted as the best solution found with N_4 iterations. Denote as s^* the index of the solution family corresponding to the best solution found to N_4 iterations.

2) For solution family s^* selected in step 1, use a global optimiser to solve the the optimization problem defined by Equations (14) and (15) to N_5 iterations. Store the best value of the decision vector $\mathbf{x}^*$ found in this step, and the corresponding objective function value.

Notice that the purpose of step 1 is to do an brief evaluation of the solution families to select the one which is most likely to contain the best solution, based on the progress of the global optimizer after N_4 iterations. Step 2 then optimizes the solution family selected in step 1 to a greater number of iterations. It is assumed that all other tuning parameters of the optimization algorithm are the same in steps 1 and 2.

If a stochastic global optimization algorithm is employed, then steps 1 and 2 can be repeated a number of times to account for the variability of the results due to the randomness in the global optimization algorithm.

In this work, we have used Differential Evolution as a global optimizer, see [9].

VII. Case studies

A. Earth-Mars mission

A simple single phase mission has been designed to compare the results that can be obtained with the proposed pruning method with results obtained by grid sampling in several dimensions. The mission consists of a transfer from Earth to Mars with a deep space manoeuvre and an insertion manoeuvre at Mars. There are six decision variables. The initial ranges for the variables are:

$$\begin{aligned} t_0 &\in [2000,\, 3000] \\ T_{\text{of}} &\in [150,\, 450] \\ r &\in [-1,\, 2] \\ \theta &\in [-\pi/6,\, \pi/6] \\ \phi &\in [-\pi/8,\, \pi/8] \\ \alpha &\in [0.1,\, 0.9] \end{aligned} \tag{16}$$

The impulsive manoeuvres were constrained as follows:

$$\begin{aligned} \Delta v_{\text{dep}} &\leq 5\,\text{km/s} \\ \Delta v_{\text{DSM}} &\leq 2\,\text{km/s} \\ \Delta v_{\text{b}} &\leq 3\,\text{km/s} \end{aligned} \tag{17}$$

An insertion manoeuvre at Mars is specified with radius of pericenter $r_p = 3950$ km, and eccentricity $e = 0.98$.

For the grid sampling, 45 points were taken along the t_0 interval, 20 along the T_{of} interval, 20 along the r interval, 10 along the θ interval, 10 along the ϕ interval, and 10 along the α interval. This gives a total of 18 million points to be sampled, of which only 19 points were found to be feasible. The grid sampling required 36 million calls to the Lambert solver, and took almost 3.36 hours on an Intel Core 2 Duo 2.0 GHz PC running Matlab 2007a.

The sequential quadratic programming algorithm, as implemented in function `fmincon` of Matlab's Optimization Toolbox [10] was employed for the local optimization steps associated with the proposed pruning method. To perform the pruning, 150 random vectors were generated in phase 1 as described in section V ($N_1 = 150$), and the local optimizer was started from each vector, resulting in 89 feasible vectors ($M_1 = 89$). The mean shift clustering algorithm was run with a bandwidth value of 230 to find the approximate feasible regions. A total of 116,326 calls to the Lambert solver where done by the proposed pruning algorithm in this case, and the pruning took 128 seconds on the same PC. The clustering algorithm takes a negligible amount of time to execute compared with the overall time it takes to perform the pruning with the proposed method.

Figure 3 shows the projected bounding boxes found using the proposed pruning method, while figure 4 shows the projected points found using the grid sampling method. Notice that the feasible points obtained by means of grid sampling are located inside the bounding boxes found by the proposed pruning method.

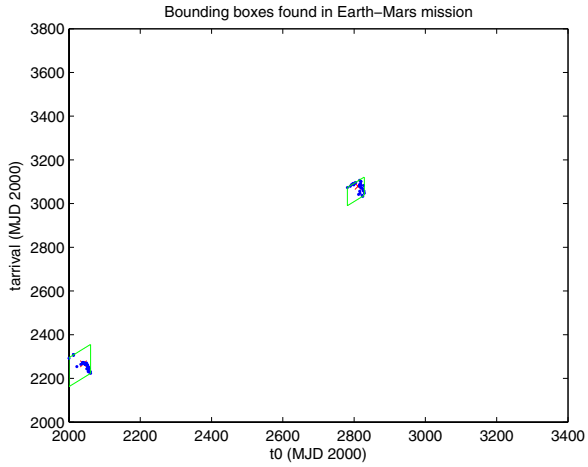

Fig. 3. Illustration on the located bounding boxes on the $t_0 - t_{\mathrm{arr}}$ plane for the estimates of the feasible regions for the Earth-Mars mission The actual bounding boxes are in six dimensions

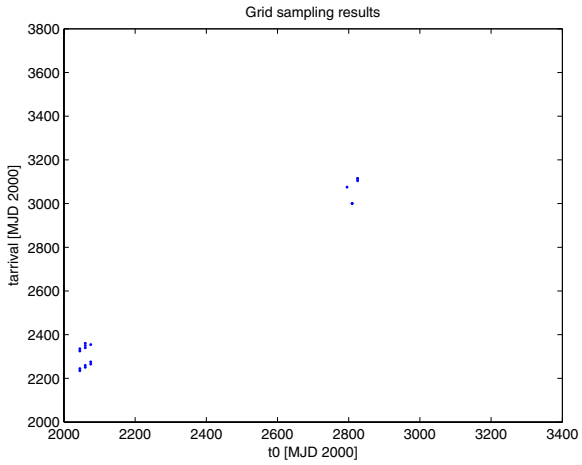

Fig. 4. Projection on the $t_0 - t_{\mathrm{arr}}$ plane of the feasible points located by grid sampling in the case of the Earth-Mars mission. The actual points are are in six dimensions

Notice the difference in computations between grid sampling and the proposed pruning method, given the amount of feasible vectors found by each method. Increasing the sampling resolution to enable the grid sampling method to find more feasible points would result in even heavier computations.

B. Earth-Venus-Mars mission

To further test the proposed pruning method, we have defined an Earth-Venus-Mars mission with one deep space manoeuvre between Venus and Mars. The problem has 7 decision variables. The initial ranges for all variables are defined below:

$$\begin{aligned} t_0 &\in [3650, 7302]\,\mathrm{MJD2000} \\ T_{\mathrm{of}}^{(1)} &\in [50, 400]\,\mathrm{days} \\ T_{\mathrm{of}}^{(2)} &\in [50, 700]\,\mathrm{days} \\ r^{(2)} &\in [-0.28, 2] \\ \theta^{(2)} &\in [-\pi, \pi]\,\mathrm{rad} \\ \phi^{(2)} &\in [-\pi/8, \pi/8]\,\mathrm{rad} \\ \alpha^{(2)} &\in [0.1, 0.9] \end{aligned} \tag{18}$$

The impulsive manoeuvres were constrained as follows:

$$\begin{aligned} \Delta v_{\mathrm{dep}}^{(1)} &\leq 5\,\mathrm{km/s} \\ \Delta v_{\mathrm{ga}}^{(1)} &\leq 5\,\mathrm{km/s} \\ \Delta v_{\mathrm{DSM}}^{(2)} &\leq 2\,\mathrm{km/s} \\ \Delta v_{\mathrm{b}}^{(2)} &\leq 5\,\mathrm{km/s} \end{aligned} \tag{19}$$

An insertion manoeuvre at Mars is specified with radius of pericenter $r_p = 3950$ km, and eccentricity $e = 0.98$.

Figures 5 and 6 show the projected bounding boxes for each phase. These diagrams illustrate how an individual box in the first phase can be related to an individual box in the second phase to form a solution family. Nine solution families can be identified.

To perform the pruning for each phase, 120 random points were generated in phase 1 as described in section V ($N_1 = 120$), and the local optimizer was started from each point. Similarly, 350 random initial points were generated in phase 2 ($N_2 = 350$, 50 points per feasible $t_{\mathrm{dep}}^{(2)}$ interval, with seven initial feasible intervals $\mathcal{I}_1, \ldots, \mathcal{I}_6$). Four local optimizations from each initial feasible point in phase 2 were performed ($N_3 = 4$), to evaluate feasibility with respect to the gravity assist manoeuvre (Step 3 of Algorithm 2). After the gravity assist calculations and backward constraining, 220 feasible vectors were found in phase 1 ($Q_1 = 220$), while 65 vectors were left in phase 2 ($Q_2 = 65$). The mean shift clustering algorithm was run with bandwidth value of 230 and 270 in phases 1 and 2, respectively. A total of 389,805 calls to the Lambert solver were required for the pruning phase (650 s CPU time)

Differential evolution with a population size of 20 individuals and parameter values $F = 0.8$ and $CR = 0.8$ was employed to search for an optimal solution for each of the six solution families located. The algorithm was initially run for $N_4 = 200$ iterations for each solution family. This was done to select the most promising solution family. Afterwards, Differential Evolution was run for $N_5 = 2000$ iterations for the selected solution family. A total of 192,000 Lambert solver calls were performed during the optimization phase (215 s CPU time). The returned result gave the following values for the decision variables: $t_0 = 4469.9$, $T_{\mathrm{of}}^{(1)} = 171.7855$, $T_{\mathrm{of}}^{(2)} = 682.4994$, $r^{(2)} = 0.5371$, $\theta^{(2)} = -1.9133$

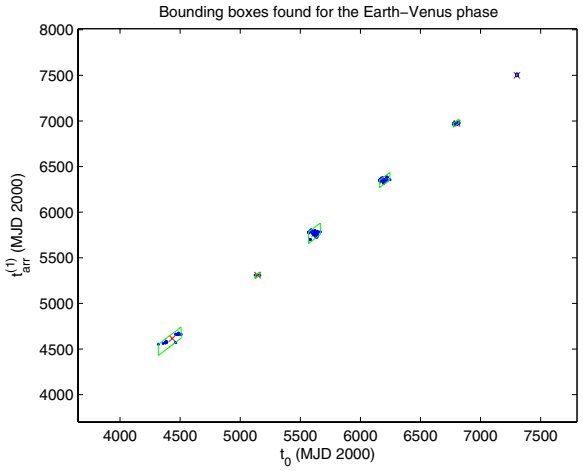

Fig. 5. Illustration on the located bounding boxes on the $t_0 - t_{\text{arr}}^{(1)}$ plane for the estimates of the feasible regions for the Earth-Venus phase. dimensions

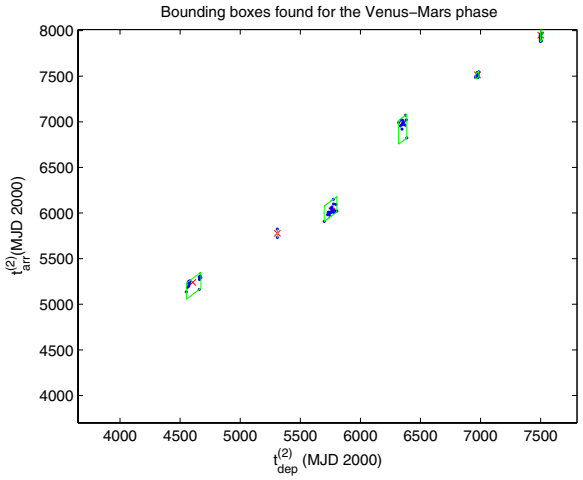

Fig. 6. Projection on the $t_{\text{dep}}^{(2)} - t_{\text{arr}}^{(2)}$ plane of the located bounding boxes for the estimates of the feasible regions for the Venus-Mars phase. The actual bounding boxes are in six dimensions

rad, $\phi^{(2)} = -0.0073$ rad, $\alpha^{(2)} = 0.5037$. The resulting impulsive manoeuvres were: $\Delta v_{\text{dep}}^{(1)} = 2.9743$ km/s, $\Delta v_{\text{ga}}^{(1)} = 8.547 \times 10^{-5}$ km/s, $\Delta v_{\text{DSM}}^{(2)} = 0.4729$ km/s, $\Delta v_b^{(2)} = 2.0158$ km/s, giving a total Δv value of 5.4630 km/s. Figure 7 shows the corresponding spacecraft trajectory projected on the plane defined by the Earth's rotation.

Notice that the tolerance of the Lambert solver was relaxed at 10^{-6} for the pruning phase, and tightened at 10^{-14} for the Differential Evolution optimization phase. This was done in order to save computation time, as only an estimate of the feasible regions is found through the pruning method, and the regions found are not very sensitive to the tolerance value.

VIII. Conclusions

A method has been presented for the design of optimal multiple gravity assist trajectories with deep space manoeuvres. A pruning method which considers the sequential nature

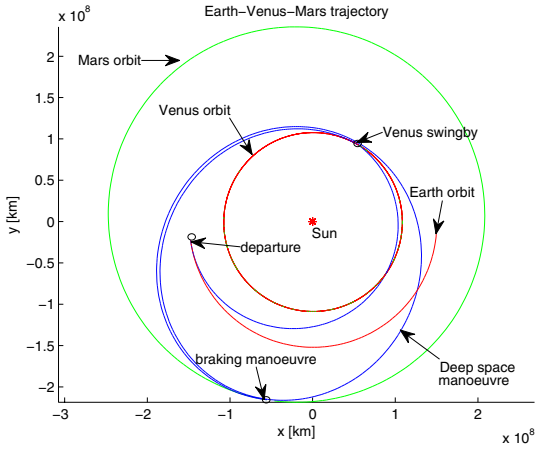

Fig. 7. Representation of the best trajectory found when Differential Evolution was applied to the pruned search space

of the problem has been described. The method locates feasible vectors using local optimization and applies a clustering algorithm to find reduced bounding boxes which can be used in a subsequent optimization step. Since multiple local minima remain within the pruned search space, the use of a global optimization method, such as Differential Evolution, has been suggested for finding solutions which are likely to be close to the global optimum. Two case studies have been presented based involving missions with deep space manoeuvres.

Acknowledgment

This work has been funded by the European Space Agency under project Ariandna 06/4101.

References

[1] J.T. Betts. *Practical methods for optimal control using nonlinear programming*. SIAM, Philadelphia, 2001.

[2] M. Vasile, L. Summerer, and P. De Pascale. Design of earth–mars transfer trajectories using evolutionary branching technique. *Acta Astronautica*, 56:705–720, 2005.

[3] D. Izzo, V. M. Becerra, D. R. Myatt, S. J. Nasuto, and J. M. Bishop. Search space pruning and global optimisation of multiple gravity assist spacecraft trajectories. *Journal of Global Optimisation*, 38:283–296, 2007.

[4] R.H. Battin. *An introduction to the mathematics and methods of astrodynamics*. AIAA, Reston, 1999.

[5] D.A. Vallado. *Fundamentals of Astrodynamics and Applications*. Kluwer Academic Publishers, London, 2001.

[6] W.E. Wiesel. *Spacecraft Dynamics*. Irwin McGraw-Hill, Boston, Mass., 1997.

[7] D. Comaniciu and P. Meer. Mean shfit: a robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:603–619, 2002.

[8] Z. Michalewicz and M. Schoenauer. Evolutionary algorithms for constrained parameter optimization problems. *Evolutionary Computation*, 4(1):1–32, 1996.

[9] R. Storn and K. Price. Differential Evolution—a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimisation*, 11:341–359, 1997.

[10] The Mathworks. *Optimization Toolbox for Use with MATLAB*. The Mathworks, Natick, MA, 2000.