J Glob Optim (2007) 38:283–296
DOI 10.1007/s10898-006-9106-0

ORIGINAL PAPER

# Search space pruning and global optimisation of multiple gravity assist spacecraft trajectories

**D. Izzo · V. M. Becerra · D. R. Myatt · S. J. Nasuto · J. M. Bishop**

Received: 7 December 2005 / Accepted: 13 October 2006 / Published online: 22 November 2006


**Abstract** We introduce and describe the Multiple Gravity Assist problem, a global optimisation problem that is of great interest in the design of spacecraft and their trajectories. We discuss its formalization and we show, in one particular problem instance, the performance of selected state of the art heuristic global optimisation algorithms. A deterministic search space pruning algorithm is then developed and its polynomial time and space complexity derived. The algorithm is shown to achieve search space reductions of greater than six orders of magnitude, thus reducing significantly the complexity of the subsequent optimisation.

**Keywords** Multiple gravity assist · Space pruning · Constraint propagation · Differential evolution · Particle swarm · Genetic algorithm · GASP · Global trajectory optimisation

## 1 Introduction

Many interplanetary trajectory design problems can be stated as optimisation problems [1, 2], where one of the fundamental goals is the maximisation of the spacecraft mass available for the payload, equivalent to a minimisation of the propellant necessary on board. Consideration is usually given also to mission duration, launch

This work was partially funded under the Ariadna scheme of the European Space Agency, contract number 18138/04/NL/MV.

D. Izzo (✉)
Advanced Concepts Team, European Space Research and Technology Center,
Keplerlaan 1, 2201 AZ Noordwijk, The Netherlands
e-mail: dario.izzo@esa.int

V. M. Becerra · D. R. Myatt · S. J. Nasuto
School of System Engineering, The University of Reading, RG6 6AY, Reading, UK

J. M. Bishop
Department of Computing, Goldsmiths College, SE14 6NW, London, UK

and arrival window, velocity constraints etc. Two main typologies of trajectory optimisation problems arise in relation to interplanetary mission design depending on the type of the thruster considered for the mission. The first one appears in connection with spacecraft equipped with engines capable of thrusting for long periods of time, typically having a modest level of thrust and a high efficiency in terms of mass consumption. The problem, commonly dubbed low-thrust, is then to find the optimal control law for the thrust vector so that the mission requirements are fulfilled and some main objective is optimised. This type of problem has traditionally been approached using local search methods inherited from the well developed theory of optimal control as described for example in Bryson and Ho [3]. The second type of problem arises when the spacecraft thrust has a shorter duration with a much higher magnitude and may therefore be modelled as a sudden change in the spacecraft velocity. In this case the optimisation aims at finding the values of a finite number of parameters describing the number and the magnitude of the velocity changes and the time instant in which they are applied. In both cases, to help the spacecraft obtain the necessary momentum, it is common to exploit the gravitational pull of other celestial bodies in what is commonly referred to as a gravity assist manouvre, a swingby or a planetary kick. Such a manouvre is based on the simple fact that when a spacecraft interacts mainly with another celestial object, a small amount of the object's orbital momentum can be transferred to the spacecraft [4]. This manoeuvre was used for the first time in the 1970's, when the spacecraft Voyager used multiple gravity assist flybys of Jupiter, Saturn, Uranus and Neptune, to propel itself beyond these planets. Gravity assist manoeuvres are frequently used to reduce propellant requirements and mission duration and may or may not require some added propellant consumption.

In recent times aerospace engineers have realised the benefits of approaching trajectory designs problems from a global optimisation point of view (see for example [5] or [6]). Mainly because of the complicated relative motion between the planets the landscape of the objective function exhibits a large number of clustered minima that cause local gradient based methods to converge to solutions corresponding to local minima. We here discuss a particular formalization of an interplanetary trajectory optimisation problem that we refer to as multiple gravity assist (MGA) problem, we apply to it standard implementations of some known global optimisation solvers and we introduce a space pruning technique that may be conveniently used to improve their performances. The remainder of the paper is as follows. In Sect. 2 we describe the mathematical formalization of the MGA problem touching upon the astrodynamical principles behind its definition. In Sect. 3 we introduce a simplified version of the MGA problem more suitable for an easy integration with existing global optimisation algorithms. We then proceed in testing some population based global optimisation stochastic algorithms. The algorithms and test conditions are described in Sect. 4. In Sect. 5 we describe a pruning technique developed specifically for the MGA problem and we evaluate, in Sect. 6, that the technique has polynomial complexity in time and space. We then perform the tests exploiting the space pruning technique and show, in Sect. 7, the great advantages introduced by its use.

## 2 The MGA problem formalisation

Let us consider a sequence of $N + 2$ planets starting with the Earth and ending with the celestial body we want to transfer the spacecraft to. We introduce the vector $\mathbf{x} = [t_0, T_1, T_2, \ldots, T_{N+1}]$ where $N$ is the number of planets we want to exploit in a

gravity assist manouvre, $t_0$ is the departure epoch and $T_i$ are the durations needed to travel along the conic arc joining two consecutive planets. Given these definitions we introduce, in the framework of the patched conics approximation [4], the MGA problem in the following form:

$$\begin{array}{c} \text{find: } \mathbf{x} \in \mathcal{I} \\ \text{to maximise: } \max_j m^j(\mathbf{x}) \end{array} \tag{1}$$

where $\mathcal{I} = \mathcal{I}_0 \times \mathcal{I}_1 \times \cdots \times \mathcal{I}_{N+1}$ is an hyperrectangle in $\mathbb{R}^{N+2}$. As explained later, more than one trajectory may be associated to a given decision vector and we have therefore introduced $m^j$ as the final spacecraft mass evaluated along the $j$th trajectory compatible with $\mathbf{x}$ and fulfilling the constraints:

$$\begin{array}{l} \Delta V_0(\mathbf{x}) \leq \Delta V_0^{\max} \\ \mathbf{r}_p(\mathbf{x}) \geq \mathbf{r}_{p_{\min}} \end{array} \tag{2}$$

where the vector $\mathbf{r}_{p_{\min}}$ contains the minimum values of the pericenter radii of the various hyperbolae that define the swingbys and $\Delta V_0^{\max}$ is the maximum Earth escape velocity allowed for the spacecraft. In order to complete the mathematical description of the problem we must now define the expressions $m^j(\mathbf{x})$, $\mathbf{r}_p(\mathbf{x})$, $\Delta V_0(\mathbf{x})$.

*From the decision vector to the compatible trajectories*: Consider a generic $\mathbf{x} \in \mathcal{I}$, then the position of the $n$-th planet of the sequence at the spacecraft rendezvous is equal to its ephemeris at epoch $t_{n-1} = t_0 + T_1 + \cdots + T_{n-1}$ whenever $n>1$ and to $t_0$ whenever the planet considered is the departure planet. Each trajectory arc joining two consecutive planets needs therefore to be a solution to the boundary value problem:

$$\begin{cases} \ddot{\mathbf{r}} - \frac{\mathbf{r}}{r^3} = \mathbf{0} \\ \mathbf{r}(t_{n-1}) = \mathbf{r}_{n-1}, \mathbf{r}(t_n) = \mathbf{r}_n \end{cases} \qquad n = 1..N+1 \tag{3}$$

where $\mathbf{r}_{n-1}$, $\mathbf{r}_n$ are the position vectors of two consecutive planets in the considered sequence, where units are any $L$ for length and $\sqrt{\mu/L}$ for velocities and where we have introduced $\mu$ as the gravitational constant of the Sun. The problem stated above is commonly referred to as Lambert's problem (see Battin [7] for a first introduction). Excluding the singular cases in which $\mathbf{r}_{n-1} \times \mathbf{r}_n = \mathbf{0}$, this boundary value problem admits $2(1+2M_{n-1})$ solutions where $M_{n-1} \in [0, \infty]$ is an integer whose value depends on the boundary conditions. The solutions corresponding to $M_{n-1} \neq 0$ are called multi-revolution solutions. All the solutions to the problem stated in Eq. (3) may be located by algorithms such as that recently proposed by Prussing [8]. The whole interplanetary trajectory will be made of $N+1$ branches, where each branch is a solution to the corresponding Lambert's problem so that at the end $N_{\text{sol}} = 2^{N+1} \prod_{i=0}^{N} (1+2M_i)$ different trajectories associated with the same decision vector $\mathbf{x}$ are possible: these trajectories are said to be compatible with $\mathbf{x}$. Due to the patching of the various arcs, at the times $t_i, i = 1..N$ the velocity vector will be discontinuous having a left and a right value that we will indicate with $\mathbf{v}_i^{\text{in}}$ and $\mathbf{v}_i^{\text{out}}$. Between the trajectories compatible with $\mathbf{x}$ we first locate the feasible ones (superscript $j$), in the sense of satisfying Eq. (2), and we then evaluate the objective function $\max_j m^j(\mathbf{x})$.

*Feasibility of the compatible trajectories*: Consider one particular trajectory compatible with $\mathbf{x}$. We may determine its departure velocity using the simple definition $\Delta V_0 = \left\| \mathbf{V}_{\text{Earth}} - \mathbf{v}_0^{\text{out}} \right\|$ where $\mathbf{V}_{\text{Earth}}$ is the heliocentric velocity of the Earth at $t_0$ and $\mathbf{v}_0^{\text{out}}$ is the heliocentric velocity of the spacecraft at $t_0$. We remain with the

problem of evaluating the vector $\mathbf{r}_p$ containing the pericenter radii of the planetocentric trajectories that links together consecutive arcs. With reference to Fig. 1 we consider the $i$-th gravity assist manouvre. The values $\mathbf{v}_i^{\text{in}}$ and $\mathbf{v}_i^{\text{out}}$ are known from the Lambert's problem solution. These are respectively the heliocentric velocity at the end of the trajectory arc preceding the flyby and at the beginning of the following arc. The heliocentric velocity $\mathbf{V}_i$ of the planet (at $t_i$) is also known. Trivially the left and right value of the spacecraft velocity at $t_i$ relative to the planet may be obtained by simple application of the galilean transformation (velocities relative to the planet are indicated with the tilde) obtaining the quantities $\tilde{\mathbf{v}}_i^{\text{in}}$ and $\tilde{\mathbf{v}}_i^{\text{out}}$. Following a patched conic approach [4], the problem is then to find a planetocentric trajectory that has $\tilde{\mathbf{v}}_i^{\text{in}}$ and $\tilde{\mathbf{v}}_i^{\text{out}}$ as asymptotic velocities. Such a trajectory, in the limit of the patched conic approximation, would instantly transfer the spacecraft between two subsequent legs of the heliocentric trajectory. We make the assumption that the spacecraft will provide one single impulse and that the planetocentric trajectory is therefore made by two arcs of hyperbola patched together. We also make the hypothesis that the impulse is given at the pericenter of the incoming hyperbola arc and in a direction tangential to the trajectory itself [9]. In this case simple astrodynamic calculations show that taking as length unit any $L$ and as velocity unit $\sqrt{\mu_{\text{pla}}/L}$ (where $\mu_{\text{pla}}$ is the gravitational parameter of the planet considered), the angle between $\tilde{\mathbf{v}}_i^{\text{in}}$ and $\tilde{\mathbf{v}}_i^{\text{out}}$ is given by:

$$\alpha_i = \arcsin \frac{a_i^{\text{in}}}{a_i^{\text{in}} + r_{p_i}} + \arcsin \frac{a_i^{\text{out}}}{a_i^{\text{out}} + r_{p_i}} \tag{4}$$

where $a_i^{\text{in}} = 1/(\tilde{\mathbf{v}}_i^{\text{in}} \cdot \tilde{\mathbf{v}}_i^{\text{in}})$ and $a_i^{\text{out}} = 1/(\tilde{\mathbf{v}}_i^{\text{out}} \cdot \tilde{\mathbf{v}}_i^{\text{out}})$. This equation allows, given a trajectory compatible with $\mathbf{x}$, us to evaluate the components $r_{p_i}$ of the vector $\mathbf{r}_p$ at the computational cost of performing Newton–Raphson iterations.

*Evaluation of the objective function*: Consider a feasible trajectory compatible with $\mathbf{x}$. At the $i$th gravity assist manouvre, under the hypothesis introduced in the previous paragraph, the spacecraft engines will have to provide a velocity increment which in turn will result in some propellant consumption and therefore mass decrease. The various velocity changes during the gravity assists are given by the simple relation (non dimensional units are any $L$ and $\sqrt{\mu_i/L}$, where we have introduced the gravitational constant $\mu_i$ of the planet considered):

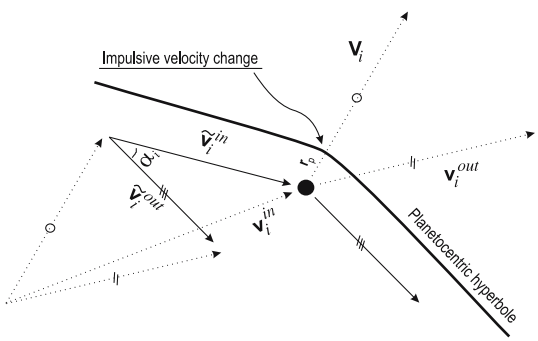


**Fig. 1** Powered flyby geometry

$$\Delta V_i = \left| \sqrt{(1/a_i^{\text{in}} + 2/r_{p_i})} - \sqrt{(1/a_i^{\text{out}} + 2/r_{p_i}))} \right|, \quad i = 1..N$$

Also at the final planet the spacecraft has to provide an additional velocity change in order to rendezvous with it: $\Delta V_{N+1} = \|\mathbf{V}_{\text{pla}} - \mathbf{v}_{N+1}^{\text{in}}\|$ where $\mathbf{V}_{\text{pla}}$ is the heliocentric velocity of the arrival planet. This last contribution to the total $\Delta V$ budget may be also written in different forms according to the level of detail one wants to describe the spacecraft operations at the planet. The relation between the various velocity increments and the final spacecraft mass is then given by the rocket equation $m_f = m_0 \exp[-\sum \Delta V_i/(I_{\text{sp}} g_0)]$, where $I_{\text{sp}}$ is the specific impulse of the engine and $g_0$ is the Earth gravitational constant at sea level and the sum is extended to all the velocity increments. Once the final spacecraft mass is evaluated along all the feasible trajectories compatible with $\mathbf{x}$, the objective function is simply the maximum value that we denoted with $\max_j m^j$. The evaluation of a given point $\mathbf{x} \in \mathcal{I}$ is therefore made by the following steps:

1. Decide $\mathbf{x}$
2. Solve $N + 1$ Lambert's problems and locate the $N\text{sol} = 2^{N+1} \prod_{i=0}^{N} (1 + 2M_i)$ trajectories compatible with $\mathbf{x}$
3. Check the feasibility of the compatible trajectories
4. Evaluate the final spacecraft mass in all the feasible and compatible cases and choose the maximum possible value

## 3 Simplifying the problem

The general definition of the MGA problem given in the previous section is complicated essentially by the fact that there is more than one trajectory compatible with each $\mathbf{x} \in \mathcal{I}$. Consequently the final spacecraft mass is not a single-valued function of the decision vector. We may simplify the problem description associating uniquely only one compatible trajectory to each point in the search space $\mathcal{I}$. This can be done by using some problem knowledge and at the risk of pruning out some interesting zones of the search space. One possibility is, for example, that of not considering multiple revolution solutions and retrograde trajectories (these are trajectories with an angular momentum $\mathbf{h}$ such that $\mathbf{h} \cdot \mathbf{h}_{\text{Earth}} < 0$). Under these hypotheses there always exists one unique solution to Lambert's problem, consequently the single valued function $m(\mathbf{x})$ may be defined. Taking into account that the functional dependence between the spacecraft mass and the sum of the different velocity increments is monotone, we rewrite the MGA problem definition in the following form:

$$\begin{aligned} \text{find:} \quad & \mathbf{x} \in \mathcal{I} \\ \text{to minimise:} \quad & J(\mathbf{x}) = \textstyle\sum_{i=1}^{N} \Delta V_i(\mathbf{x}) + \Delta V_{N+1}(\mathbf{x}) \\ \text{subject to:} \quad & \Delta V_0(\mathbf{x}) \leq \Delta V_0^{\max} \\ & \mathbf{r}_p(\mathbf{x}) \geq \mathbf{r}_{p_{\min}} \end{aligned} \tag{5}$$

A study on the complexity of the problem stated above can be found in the reports of two studies [1,2] performed by the European Space Agency, and in particular by its Advanced Concepts Team, in cooperation with the Universities of Reading (school of Systems Engineering) and Glasgow (department of Aerospace Engineering).

*An example: the case of 99942 Apophis*: We here take as an example the asteroid 99942 Apophis and we consider a direct transfer to it (no gravity assist manouvres).

In this case the problem has only two dimensions and a graphical representation of the objective function is possible. We take as constraint $\Delta V_0^{\max}$=3.5 km/s and as search space $\mathcal{I} = [3600, 4800] \times [51, 500]$ (the departure epoch $t_0$ is measured using the modified Julian date 2000 (MJD 2000) and the transfer duration $T_1$ is measured in days). In Fig. 2 the set $\mathcal{I}_{5.5} = \{\mathbf{x} \in \mathcal{I} | J(\mathbf{x}) \leq 5.5\}$ is plotted. Note that units for the objective function are km/s. The plot reveals a quite irregular landscape also in this low-dimensional case.

## 4 Preliminary tests of selected global optimisers

The problem described by Eq. (5) has been coded in Matlab and C++ and a particular instance of it has been made available to the scientific community in form of a blackbox function. The problem instance made available corresponds to the planetary sequence Earth–Venus–Venus–Earth–Jupiter–Saturn used during the Cassini/Huygens mission and may be downloaded from the web site of the European Space Agency following the link from the page http://www.esa.int/gsp/ACT/mission_analysis. In this particular implementation the last velocity increment $\Delta V_{N+1}$ is evaluated by considering the spacecraft insertion into a highly elliptic orbit. Also, $\Delta V_0$ is not constrained and is included in the sum of the velocity increments that defines the objective function.

We report some preliminary results on the performances of common implementations of stochastic global optimisation algorithms. The algorithms selected have a similar computational complexity and the number of objective function evaluations has been used as a stopping criteria. We first describe briefly the various algorithms used and then how the test was performed.

*Differential Evolution—DE*: Differential evolution [10] is a novel probabilistic global optimiser which was the highest ranked Genetic-type algorithm in the First

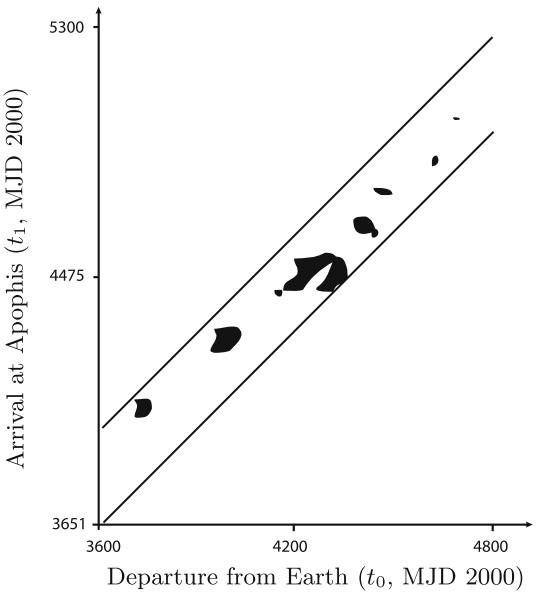


**Fig. 2** Visualization of the set $\mathcal{I}_{5.5} = \{\mathbf{x} \in \mathcal{I} | J(\mathbf{x}) \leq 5.5\}$. The search space $\mathcal{I}$ is comprised between the two oblique lines. The existence of different irregularly located valleys is evident

International Contest on Evolutionary Computation. In their paper, Rainer and Storn, considered two different crossover schemes, DE1 and DE2. Scheme DE1 will be used as the crossover operator as it was shown to perform the best on the most complex test function they examined, which was an 8 dimensional Chebychev polynomial fitting problem. The parameters used for this optimiser were: population size = 20, crossover probability $CR = 0.5$, stepsize $F = 0.8$.

*Particle Swarm Optimisation—PSO*: Particle swarm optimisation [11] was originally designed as a simulation of flocking behaviour in birds, although its potential for optimisation was recognised shortly afterwards. Each particle has a position within the search space and a velocity, both of which are initialised randomly. As iterations progress, each particle keeps tracks of the position of the best solution it has so far encountered, and also knows the globally best solution found by the entire population. The velocity is updated by two main components: the cognitive component, which attracts the particle towards its own best solution, and the social component, which attracts the particle to the best known solution. Each particle moves in the search space according to the following equation:

$$\begin{aligned}\mathbf{x}_i &= \mathbf{x}_i + \mathbf{v}_i\\ \mathbf{v}_i &= \omega\mathbf{v}_i + \eta_1 r_1(\mathbf{x}_p^* - \mathbf{x}_i) + \eta_2 r_2(\mathbf{x}_g^* - \mathbf{x}_i)\end{aligned}$$

where $\mathbf{x}_p^*$ is the personal best solution of the $i$-th particle, $\mathbf{x}_g^*$ is the globally best known solution, and $r_1, r_2$ are uniform random numbers in the interval $[0, 1]$. The parameters used for this optimiser where: population = 20, $\omega = 0.65$, $\eta_1 = 2.0$, $\eta_2 = 2.0$.

*Multiple Particle Swarm Optimisation—MPSO*: Multiple particle swarm optimisation is a novel technique based on that investigated by Blackwell [12], which showed significantly improved performance over the basic algorithm. MPSO includes aspects of niching genetic algorithms, in that each swarm evolves separately, for the main part. However, every 5 iterations the swarm membership of two randomly chosen particles is exchanged, effectively introducing new information into those swarms as to the position of other minima. The parameters used for this optimiser where: population = 20, number of swarms = 3, $\omega = 0.65$, $\eta_1 = 2.0$, $\eta_2 = 2.0$.

*Genetic Algorithm—GA*: The name genetic algorithm often generate confusion as it is associated to a wide class of global optimisation algorithms. In this test we used a basic version of the algorithm with real encoding using only a single-point crossover and uniform mutation. Every $n$ iterations the best individual found so far was reinserted in the population. An outline of the algorithm follows:

1. Initialize randomly the population of NP individuals in $\mathcal{I}$
2. Repeat
3. Select NP individuals to be mated based on their fitness
4. Apply single-point crossover operator with probability CR
5. Apply mutation operator with probability MU
6. Every $n$ iterations re-insert the best individual found so far
7. Until convergence

The parameters used for this optimiser where: NP = 20, $n$ = 5, CR= 0.65, MU= 0.1.

For DE the choice made are standard except for the population size, for PSO, MPSO and GA more arbitrary choice had to be made based on our personal experience with this problem. In all cases the boundary conditions were enforced by randomly placing the unfeasible components in the feasible interval.

*Test Case Description*: The search space considered in the test case was: $\mathcal{I} = [-1000, 0] \times [30, 400] \times [100, 470] \times [30, 400] \times [400, 2000] \times [100, 6000]$, where the modified Julian date 2000 is used to express the different epochs. The number of function evaluations allowed for each algorithm was 20000 after which the simulations were stopped. Each algorithm was run a total of 40 times and the best result was recorded together with the worst result, the average and the standard deviation. The results are shown in Table 1. For this case further experimentation has shown that the global optimum in this problem is less than 5 km/s, and hence it is certain that none of the algorithms have converged to the global optimum within the given number of objective function evaluations allowed.

## 5 GASP—gravity assist space pruning

Consider once again the MGA problem as stated in Eq. (5). Introduce the map:

$$f : \mathbf{x} = [t_0, T_1, \ldots, T_{N+1}] \to \mathbf{X} = [t_0, t_1, \ldots, t_N, t_{N+1}]$$

defined by the simple relation $t_i = t_0 + \sum_{j=1}^{i} T_j$, $i = 0, \cdots, N+1$. Applying the same transformation to the search space $\mathcal{I}$ we obtain a new search space that we denote with $\mathcal{I}^* = f(\mathcal{I})$. We may now consider a new statement of the MGA problem:

$$\begin{aligned} \text{find:} \quad & \mathbf{X} \in \mathcal{I}^* \\ \text{to minimise:} \quad & J(\mathbf{X}) = \textstyle\sum_{i=1}^{N} \Delta V_i(\mathbf{X}) + \Delta V_{N+1}(\mathbf{X}) \\ \text{subject to:} \quad & \Delta V_0(\mathbf{X}) \leq \Delta V_0^{\max} \\ & \mathbf{r}_p(\mathbf{X}) \geq \mathbf{r}_{p_{\min}} \end{aligned} \tag{6}$$

At the cost of dealing with a search space that is not hyper rectangular, we have formulated the MGA problem in terms of absolute times which brings the advantage of simplifying the relation between the planets positions and the decision vector (the planet positions depend now on only one component of $\mathbf{X}$). Besides, this formulation improves the chances of crossover operations in genetic strategies to generate good individuals. The optimisation method we here investigate in detail is grid sampling. Grid sampling is usually a very inefficient optimiser, particularly in high dimensionalities. However, for only 1 and 2 dimensions grid sampling is computationally tractable, as long as the objective function is reasonably smooth and the exact optimum is not required. Therefore, the objective function for a single interplanetary transfer may be grid sampled at an appropriate resolution in the departure time vs arrival time domain efficiently, although in this case most other optimisation methods would yield better results in terms of objective function evaluations. However, the grid sampled version will require many less ephemeris calculations, as the same positions/velocities need not be recalculated for a given departure or arrival time. If the

**Table 1** Results of the preliminary test on a particular MGA problem instance

| Algorithm | Minimum | Maximum | Average | Standard deviation |
|---|---|---|---|---|
| DE | 5.012 | 16.8957 | 10.270 | 3.51 |
| PSO | 5.596 | 22.127 | 11.916 | 4.43 |
| MPSO | 5.549 | 18.933 | 9.843 | 3.67 |
| GA | 5.755 | 21.199 | 8.679 | 2.96 |

About 20000 function evaluations have been allowed, units are in km/s

2D search space was discretised into $k$ cells in each dimension, only $2k$ ephemeris calculations are required for the entire sampling, and $k^2$ Lambert problem solutions. By comparison, two ephemeris calculations are required by each objective function evaluation in a standard optimiser. Clearly as soon as gravity assists are involved the increased dimension of the problem makes grid sampling appear to be unfeasible. Fortunately, the particular structure of the MGA problem allows to consider grid sampling also for higher dimensions as we will now show.

Consider the MGA problem formalization given by Eq. (6) and focus the attention on the functional dependence of the objective function and of the constraints upon the decision vector $\mathbf{X}$. The function $\Delta V_0(\mathbf{X})$ depends only on the first two components of the decision vector: $\Delta V_0(\mathbf{X}) = \Delta V_0(t_0, t_1)$, a two-dimensional sampling of $f(\mathcal{I}_0 \times \mathcal{I}_1)$ therefore allows us to evaluate this constraint. The first gravity assist velocity increment $\Delta V_1$ depends on the incoming and outcoming spacecraft velocities. But the incoming velocity is known from the previous two dimensional sampling, and a further two dimensional sampling of $f(\mathcal{I}_1 \times \mathcal{I}_2)$ allows us to evaluate $\Delta V_1$. The same applies for $r_{p_1}$. We arrive at the final planet where $\Delta V_{N+1}$ only depends on the incoming velocity already sampled to evaluate $\Delta V_N$. This particular structure of the problem allows us to sample the objective function in a cascade of $N+1$ two-dimensional spaces rather than in a single $N+2$ dimensional space. Consider the space $\mathcal{I} = \mathcal{I}_0 \times \mathcal{I}_1 \times \cdots \times \mathcal{I}_{N+1}$ defined in terms of the decision vector $\mathbf{x}$. Grid sampling the spaces $f(\mathcal{I}_0 \times \mathcal{I}_1), f(\mathcal{I}_1 \times \mathcal{I}_2), \ldots, f(\mathcal{I}_N \times \mathcal{I}_{N+1})$ still allow to explore the whole $f(\mathcal{I})$. In these two dimensional spaces pruning can be done very effectively. Consider as an example an Earth–Mars transfer with $\mathcal{I} = [-1200, 600] \times [25, 515]$. Grid sampling with a resolution of 10 days the space $f(\mathcal{I})$ reveals that only 12.5% of this search space has a $\Delta V_0$ of less than 5 km/s. As a consequence, in gravity assist and multiple gravity assist cases starting with an Earth–Mars transfer in these bounds, at least 87.5% of the overall search space corresponds to unfeasible solutions if $\Delta V_0^{\max} = 5$ km/s. Even allowing an enormous $\Delta V_0^{\max}$ of 10 km/s, only 33% of the search space becomes valid.

Based on these simple ideas we present an algorithm called GASP (gravity assist space pruning) and designed to efficiently detect and prune infeasible parts of the space, leaving several sets of box bounds with vastly smaller volume. These reduced box bounds may then be searched efficiently using a standard global optimisation method. We will now list a number of criteria that allow us to prune the space exploiting the problem structure outlined above.

*Departure $\Delta V_0$ constraining*: The maximum allowable $\Delta V_0$ is the first main pruning criterion of the GASP algorithm. It works on the sampled space $f(\mathcal{I}_0 \times \mathcal{I}_1)$ pruning out all those points corresponding to trajectories having unfeasible departure $\Delta V_0$.

*Forward constraining*: Applying the $\Delta V_0$ constraining alone significantly reduces the search space volume in an MGA problem. As a consequence many values of the arrival time $t_1$ in $f(\mathcal{I}_0 \times \mathcal{I}_1)$ become not feasible departure times in $f(\mathcal{I}_1 \times \mathcal{I}_2)$. This observation is the key principle on which the GASP algorithm is based: if no feasible trajectories arrive at a planet on a given date because they have been pruned according to the various criteria introduced, then there will be no departures from that planet on that date and all the corresponding points will also be pruned. Consider as an example the MGA problem Earth–Venus–Earth in $\mathcal{I}_0 = [3600, 5400], \mathcal{I}_1 = [14, 284], \mathcal{I}_2 = [14, 284]$. In Fig. 3 the two-dimensional spaces corresponding to the Earth–Venus phase $f(\mathcal{I}_0 \times \mathcal{I}_1)$ and to the Venus–Earth phase $f(\mathcal{I}_1 \times \mathcal{I}_2)$ are shown together with an example of pruning caused by the forward propagation of a departure $\Delta V$ constraint of 10 km/s.

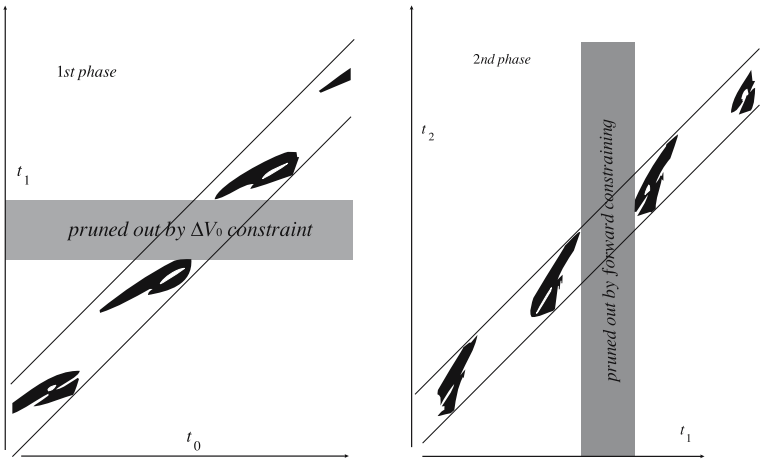


**Fig. 3** Pruning of the space $f(\mathcal{I}_0 \times \mathcal{I}_1)$ and forward propagation to the space $f(\mathcal{I}_1 \times \mathcal{I}_2)$. The effects of the breaking manouvre constraint is also shown in the 2nd phase

*Gravity assist maximum thrust constraint*: The gravity assist thrust constraint prunes out the trajectories having a difference between incoming and outgoing velocities during a gravity assist larger than some threshold, $A_v$. This threshold is set separately for each gravity assist. The following is then performed for every arrival time at a planet:

1. Calculate the bounds on the possible incoming velocity, $v^i_{\min}$ and $v^i_{\max}$.
2. Invalidate any outgoing trajectories that do not have outgoing velocities in the range $[v^i_{\min} - A_v - L_v, v^i_{\max} + A_v + L_v]$, where $L_v$ is an appropriate tolerance based on the Lipschitzian constant of the current phase plot (the tolerance may be chosen heuristically and it does not have a major influence on the results).
3. Calculate the modified bounds on outgoing velocity, $v^f_{\min}$ and $v^f_{\max}$.
4. Invalidate any incoming trajectories with velocities outside the range $[v^f_{\min} - A_v - L_v, v^f_{\max} + A_v + L_v]$.

*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 at a planet as follows, assuming $i$ valid incoming trajectories and $j$ valid outgoing trajectories:

1. For all $i$ incoming trajectories
2. For all $j$ incoming trajectories
3. If the swingby is valid for the current incoming
4. and outgoing trajectory, mark both incoming and outgoing trajectory as valid.
5. End
6. End
7. Invalidate all trajectories not marked as valid

The swingby angle of each sampled manouvre is decreased by an appropriate Lipschitzian tolerance $\theta_L$, in order to compensate for the effects of the grid sampling of the search space.

*Breaking Manouvre Constraint*: As well as the departure $\Delta V_0$ constraint, it is logical to add a constraint on the maximum braking manoeuvre that a spacecraft can perform and prune out trajectories with an exceedingly high propellant demand.

*Backward Constraining*: Clearly if a departure date in $f(\mathcal{I}_i \times \mathcal{I}_{i+1})$ becomes unfeasible because of pruning, also the relative arrival date in $f(\mathcal{I}_{i-1} \times \mathcal{I}_i)$ has to be pruned out.

All in all GASP algorithm may be summarized by the following steps:

1. Perform grid sampling of the spaces $f(\mathcal{I}_i \times \mathcal{I}_{i+1})$
2. Apply departure $\Delta V_0$ constraint
3. Forward constraining through all phases
4. For each phase
5. Invalidate infeasible departure dates
6. Apply gravity assist maximum thrust constraint to the next phase
7. Apply gravity assist angular constraint to the next phase
8. End
9. Apply breaking manouvre constrain
10. Backward constraining through all phases
11. For each phase
12. Invalidate arrival dates based on departure dates.
13. End

## 6 Time and space complexity of the space pruning

This section determines the time and space complexity of the GASP algorithm, that is the number of operations necessary to complete the algorithm and its memory needs. It will be shown that GASP scales quadratically in space and quartically in time. For simplicity, the following analysis assumed that the initial launch window and all phase times are the same.

### 6.1 Space complexity

Consider a launch window discretised into $k$ bins and a mission phase time also discretised into $k$ bins. For the first phase $k^2$ Lambert problems must be sampled. The next phase will need to sample $(k + k)k = 2k^2$, as the number of possible times that the planet may be arrived at is doubled (minimum launch date, minimum phase time to maximum launch date, maximum phase time). The third phase will require $3k^2$ Lambert function evaluations, and the $n^{\text{th}}$ phase $nk^2$. This gives the series $O(n) = k^2 + 2k^2 + 3k^2 + \cdots + nk^2\, O(n) = k^2(1 + 2 + 3 + \cdots + n)\, O(n) = k^2 \frac{n(1+n)}{2}$. Therefore, the amount of space required for $n$ phases is only of the order $O(n^2)$, rather than $O(k^n)$ for full grid sampling.

Similarly, it is clear that the space complexity with respect to the resolution $k$, is also of the order $O(k^2)$.

### 6.2 Time complexity

The memory space required 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 GASP algorithm must also be of the order $O(n^2)$.

*Departure velocity $\Delta V_0$ constraint complexity*: The launch energy constraint is only applied in the first phase, and hence is independent of the number of swingbys. The time complexity is $O(k^2)$ with respect to resolution.

*Gravity assist thrust constraint complexity*: The time complexity of applying the gravity assist thrust constraint is $O(n^2)$ with respect to dimensionality (number of phases), due to the inevitable increase in size of later phase plots to encompass all possible arrival dates. The first phase requires of the order of $2k \times (k + 3k)$ operations in order to perform the constraining of outgoing velocity from incoming velocity (the back constraining may be ignored at this point). The second phase requires of the order of $3k \times (2k + 4k)$ operations. In general, the $n^{\text{th}}$ phase requires of the order of $2n^2k^3$ operations. Therefore, the total number of operations over all phases is $2k^2[2^2 + 3^2 + 4^2 + \ldots + n^2] = 2k^2\frac{n(n+1)(2n+1)}{3}$. Therefore, applying this constraint yields cubic time complexity in dimensionality and quadratic complexity in resolution.

*Gravity assist angular constraint complexity*: The maximum number of swingby models that must be calculated for the first phase is close to $k \times 2k \times 3k = 6k^3$. For the second swingby, this is $2k \times 3k \times 4k = 24k^3$. In general, for $n$ phases, the upper bound on the number of swingby calculations, $\alpha$, is

$$\begin{aligned}\alpha = {} & 3 \times 2 \times 1 \times k^3 + 4 \times 3 \times 2 \times k^3 \\ & + 5 \times 4 \times 3 \times k^3 + \cdots + (n+2)(n+1)nk^3\end{aligned}$$

The total number of these operations must therefore be $\alpha = k^3 \sum_{j=1}^{n}(j+2)(j+1)j = k^3\frac{n(n+1)(n+2)(n+3)}{4}$. Therefore, the overall time complexity with respect to resolution is $O(k^3)$, while the time complexity with respect to dimensionality is $O(n^4)$. Therefore, the gravity assist angular constraint is the most computationally expensive and hence is applied after GA thrust constraint in order to minimise the number of swingby models that must be calculated.

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

## 7 Testing GASP

Consider the same problem instance we have studied in Sect. 4. We performed again the same tests taking as individuals for the initial populations $f^{-1}(\mathbf{X}_i)$ where the $\mathbf{X}_i$ are randomly sampled in the best window located by GASP (in terms of sampled objective function). Running GASP in this case took less than one second on a Pentium IV 2.8 Ghz personal computer and brought a search space reduction of a factor 139,000 with the following pruning parameters considered $\Delta V_0^{\max} = 8$ km/s, the $A_v = 1$ km/s (equal for all gravity assists) and $\Delta V_{\text{arr}} < 8$ km/s. Grid sampling was performed with an interval of ten days on each of the variables and 23915 swingby calculation were performed in the process. Table 2 reports the outcome of 40 tests performed on each of the considered optimisation techniques. Note that GASP is not run before each of the optimisation, but just once at the beginning of the tests. The benefits of having applied the space pruning are clear by comparing this with Table 1. Not only GASP improved drastically the convergence of all the optimisers, but it also allowed Differential Evolution to locate a new optimum. For completeness the best trajectory, using a 2/1 resonant Venus fly-by, is visualized in Fig. 4.

**Table 2** Results of the preliminary test on the MGA problem instance initializing the populations in the best window located by GASP

| Algorithm | Minimum | Maximum | Average | Standard deviation |
|---|---|---|---|---|
| DE | 4.944 | 5.372 | 5.302 | 0.06 |
| PSO | 5.305 | 6.089 | 5.403 | 0.13 |
| MPSO | 5.303 | 5.347 | 5.310 | 0.0099 |
| GA | 5.401 | 6.074 | 5.693 | 0.14 |

About 20000 function evaluations have been allowed, units are in km/s

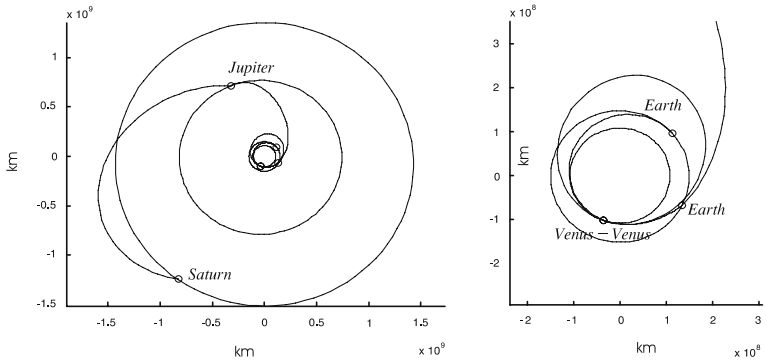


**Fig. 4** Visualization of the best found optima for the test case

## 8 Conclusions

We have formally introduced the Multiple Gravity Assist global optimisation problem and shown how the structure of its objective function and of its constraints allows to consider the multidimensional search space as a cascade of two-dimensional spaces. Based on this observation we introduced a space pruning algorithm GASP that drastically improves the performances of subsequent global optimisation techniques. The complexity of the pruning algorithm is polynomial both in time and in space. In the test performed GASP allowed search space reductions of several orders of magnitudes and was able to locate efficiently all the interesting parts of the search space.

## References

1. Myatt, D.R., Becerra, V.M., Nasuto, S.J., Bishop, J.M.: Advanced global optimisation for mission analysis and design. Final Report. Ariadna id: 03/4101. Contract Number: 18138/04/NL/MV, 2004. Available: http://www.esa.int/gsp/ACT/doc/ACT-RPT-03-4101-ARIADNA-GlobalOptimisationReading.pdf
2. Di Lizia, P., Radice, G.: Advanced global optimisation for mission analysis and Design. Final Report. Ariadna id: 03/4101. Contract Number: 18139/04/NL/MV, 2004. Available: http://www.esa.int/gsp/ACT/doc/ACT-RPT-03-4101-ARIADNA-GlobalOptimisationGlasgow. pdf
3. Bryson, A.E., Ho, Y.: Applied Optimal Control, pp. 1–89. Taylor & Francis, New York (1989)
4. Brown, C.D.: Spacecraft Mission Design, ch. 6, AIAA, Reston (1998)
5. Yokoyama, N., Suzuki, S.: Modified genetic algorithm for constrained trajectory optimization. J. Guidance Control Dyn. **28**(1), 139–144 (2005)
6. Vasile, M., Summerer, L., De Pascale, P.: Design of earth–mars transfer trajectories using evolutionary-branching technique. Acta Astronautica **56**(8), 705–720 (2005)

7. Battin, H.B.: An Introduction to the Mathematics and Methods of Astrodynamics, ch. 7, AIAA, Reston (1999)
8. Prussing, J.E.: A class of optimal two-impulse rendezvous using multiple-revolution Lambert solutions. J. Astronaut. Sci. **48**(2–4), 131–148 (2000)
9. Gobetz, F.B.: Optimal transfer between hyperbolic asymptotes. AIAA J. **1**(9), (1963)
10. Storn, R., Price, K.: Differential evolution — a simple and efficient heuristic for global optimization over continuous spaces. J. Global Optim. **11**, 341–359, (1997)
11. Kennedy, J., Eberhart, R.: Particle swarm optimization. Proc. IEEE Int. Conf. on Neural Networks, 1942–1948, (1995)
12. Blackwell, T.M., Branke, J.: Multi-swarm optimisation of the moving peaks function. Proc. Appl. Evolut. Comp. EuroWorkshops 2004, pp. 489–500, Springer-Verlag (2004)