7 Global Optimization and Space Pruning for Spacecraft Trajectory Design

Dario Izzo
European Space Agency, Advanced Concepts Team, Noordwijk, NL

7.1 Introduction

Global optimization algorithms and space pruning methods represent a recent new paradigm for spacecraft trajectory design. They promise an automated and unbiased search of different trajectory options, freeing the final user from the need for caring about implementation details. In this chapter we provide a unified framework for the definition of trajectory problems as pure mathematical optimization problems highlighting their common nature. We then present the detailed definition of two popular typologies, the Multiple Gravity Assist (MGA) and the Multiple Gravity Assist with single Deep Space Manouver (MGA-1DSM). Later we describe in detail the instantiation of four particular problems proposing them as a test set to benchmark the performances of different algorithms and pruning solutions. We take inspiration from real interplanetary trajectories such as Cassini, Rosetta, and the proposed TandEM mission, considering a large search space in terms of possible launch windows and transfer times, but also from rather academic cases such as that of the First Global Trajectory Optimisation Competition (GTOC). We test four popular heuristic paradigms on these problems (differential evolution, particle swarm optimization, simulated annealing with adaptive neighborhood, and genetic algorithm) and note their poor performances both in terms of reliability and solution quality, arguing for the need to use more sophisticated approaches, for example, pruning methods, to allow finding better trajectories. We then introduce the cluster pruning method for the MGA-1DSM problem and we apply it, in combination with the simulated annealing with adaptive neighborhood algorithm, to the TandEM test problem finding a large number of good solutions and a new putative global optima.

Many of the results reported here would not have been possible without the great passion and competence of Tamas Vinko, Marco del Rey Zapatero, and Marek Rucinski, all researching, at different times, different global trajectory optimization aspects. The author also wishes to acknowledge Massimiliano Vasile who, while a research fellow with the Advanced Concepts Team at the European Space Agency, conceived the Ariadna studies on Advanced Global Optimization Tools for Mission Analysis and Design, which ignited the spark of this now incredibly rich research topic.

7.2 Notation

We will make extensive use of a notation that has become quite standard for the engineering community in the past decades. We will denote the components of multidimensional vectors with a boldface, $\mathbf{x}$. We will then refer to their components by using a subscript on a nonboldface font, x_i. An important part of the spacecraft trajectory optimization problem definition is the choice of variables that describe the spacecraft state. Here we use an abstract state vector representation $\mathbf{x}$ whenever possible. Otherwise, when explicit equations are useful, a cartesian choice is used: $\mathbf{x} = [\mathbf{r}, \mathbf{v}, m]$, where $\mathbf{r}$ denotes the spacecraft position in the inertial frame selected, $\mathbf{v}$ its velocity, and m its mass. Other representations can be more suited to particular applications. I_{sp} is the spacecraft propulsion system specific impulse and $g_0 = 9.81$ m/s. The positions and velocity of celestial bodies (planets and asteroid) will be noted with capital letters $\mathbf{R}$ and $\mathbf{V}$. In the case of multiphase problem, we will use a superscript to indicate that a certain quantity is referred to a particular phase, so that $\mathbf{x}^i$ will be the spacecraft state during the i-th phase. We also use the subscripts s and f to indicate the start or final instants of a phase, so that $\mathbf{x}_s^i = \mathbf{x}^i(t_s^i)$ will denote the spacecraft state at the beginning of the phase i.

7.3 Problem Transcription

A spacecraft trajectory optimization problem can, in general, be written in the deceptively familiar form

$$\mathcal{P} \quad \begin{array}{rll} \text{Optimize:} & \phi(t_s, t_f, \mathbf{x}_s, \mathbf{x}_f) + \int_{t_s}^{t_f} \mathcal{L}(\mathbf{x}(t), \mathbf{u}(t), t)dt & \\ \text{Subject to:} & \dot{\mathbf{x}} = \mathbf{a}(\mathbf{x}, t) + \mathbf{u}(t) & \text{dynamic con.} \\ & \mathcal{G}(t_s, t_f, \mathbf{x}_s, \mathbf{x}_f) \leq 0 & \text{boundary con.} \\ & \mathbf{u} \in \mathcal{U}(\mathbf{x}, t) & \text{propulsion / power con.} \end{array} \tag{7.1}$$

where $\mathbf{x}(t)$ is the spacecraft state (including at the least its position, velocity, and mass or equivalent quantities), $\mathbf{a}(\mathbf{x}, t)$ represent the external forces acting on the spacecraft (gravitational and non), and $\mathbf{u}(t)$ represent the control acting on the spacecraft and coming from its propulsion system. Note that the set of admissible controls has, in this general form, an explicit state dependence, a characteristic that is not accounted for in the classical optimal control theory. The characteristic that makes problem $\mathcal{P}$ unique in its kind is the extremely complicated nature of the dynamic constraints, which limits the straightforward use of numerical methods to obtain an optimal solution, and the presence of a large number of locally optimal optimal solutions that calls for the use of global optimisation techniques to effectively explore the possible solutions. As a consequence, depending on the particular trajectory problem considered, the problem is often reduced to a simpler form. This process, which we call problem transcription, is central to the spacecraft trajectory optimization process as it heavily influences the performances of the subsequent solution technique, but also the feasibility of the solution obtained, that is, the possibility

of transforming it back to an optimal solution of the full problem $\mathcal{P}$. Here we will focus our attention on the case of interplanetary mission design, and in particular on the preliminary phases of the process when the need of a fast exploration of the solution space is more important than a precise description of the trajectory. A comprehensive description of the mission design process may be found in [1]. We will make use of a number of simplifications and hypotheses that are quite commonly applied:

- The only external force considered in writing the term $\mathbf{a}(\mathbf{x}, t)$ is the gravitational attraction $\mathbf{g}(\mathbf{x})$ due to the Sun.
- The sequence of $N + 1$ celestial bodies the spacecraft interacts with is imposed a priori and not left to the optimization process to eventually find.
- The spacecraft interaction with the selected celestial bodies happens when the spacecraft state enters the body sphere of influence [2] $\mathbf{x} \in \mathcal{S}_i$, an event that is imposed as a boundary constraint. The interaction with the intermediate bodies (that is, excluding the departure and arrival) is described by a discontinuity in the spacecraft state $\Delta\mathbf{x}^i = \mathbf{x}_s^{i+1} - \mathbf{x}_f^i$, and on the time $\Delta t^i = t_s^{i+1} - t_f^i$, typically a consequence of what is commonly referred to as planetary flyby or a planet rendezvous.

These discontinuities in the state, describing trajectory phases where the spacecraft interacts with a celestial body, are subject to constraints we write in the form: $\Xi^i(\mathbf{x}_s^{i+1}, \mathbf{x}_f^i, t_s^{i+1}, t_f^i) \leq 0$ and we call phase constraints. These hypotheses formally transform problem $\mathcal{P}$ into:

$$
\mathcal{P}' \quad \begin{array}{lll}
\text{Optimize:} & \phi(t_s^1, t_f^N, \mathbf{x}_s^1, \mathbf{x}_f^N) + \sum_{i=1}^{N} \int_{t_s^i}^{t_f^i} \mathcal{L}^i(\mathbf{x}^i(t), \mathbf{u}^i(t), t)dt & \\
\text{Subject to:} & \dot{\mathbf{x}}^i = \mathbf{g}(\mathbf{x}^i) + \mathbf{u}^i(t), \forall i = 1..N & \text{dynamic con.} \\
& \mathcal{G}(t_s^1, t_f^N, \mathbf{x}_s^1, \mathbf{x}_f^N) \leq 0 & \text{boundary con.} \\
& \Phi^i(t_s^i, t_f^i, \mathbf{x}_s^i, \mathbf{x}_f^i) \leq 0, \forall i = 1..N & \text{phase boundary con.} \\
& \Xi^i(\mathbf{x}_s^{i+1}, \mathbf{x}_f^i, t_s^{i+1}, t_f^i) \leq 0, \forall i = 1..N-1 & \text{phase matching con.} \\
& \mathbf{u}^i \in \mathcal{U}(\mathbf{x}^i, t), \forall i = 1..N & \text{propulsion / power con.}
\end{array} \tag{7.2}
$$

that is a multiphase optimal control problem where the number of phases N depends on the number of planetary encounters during the overall trajectory. In explicit cartesian coordinates $\mathbf{x}^i = [\mathbf{r}^i, \mathbf{v}^i, m^i]$, and $\mathbf{u}^i = [\mathbf{0}, \frac{\mathbf{T}^i}{m^i}, 0]$, where T indicates the magnitude of the spacecraft thrust $\mathbf{T}$. In going from $\mathcal{P}$ to $\mathcal{P}'$, we have simplified the problem at the cost of introducing an aprioristic choice on the sequence of planetary encounters, a choice that also needs to be optimized and thus introduces an integer programming element into the interplanetary trajectory design process. The sequence of planetary encounters is enforced in the phase boundary constraints Φ^i. Considering the planetary sphere of influence reduced to one point $\mathcal{S}^i = \{\mathbf{x} \in \mathbb{R}^7, \mathbf{x} = [\mathbf{R}^i(t), ., ., ., .]\}$ (also

a common simplification), the boundary constraints Φ^i will include the following conditions that enforce the spacecraft position to be, at the beginning of each phase, equal to the departure planet position, and at the end of each phase equal to the arrival planet position

$$\Phi^i = \begin{cases} \vdots \\ \mathbf{r}_s^i = \mathbf{R}^i(t_s^i) \\ \mathbf{r}_f^i = \mathbf{R}^{i+1}(t_f^i) \\ \vdots \end{cases}, \forall i = 1..N \tag{7.3}$$

where we have introduced the sequence of $N+1$ planets via their positions $\mathbf{R}^i(t)$, $i = 1..N+1$. Under the assumption that the $\mathbf{u}^i$ are piecewise continuous functions, problem $\mathcal{P}'$ is further specified to describe low-thrust problems, whereas an impulsive modelling of the thrust describes chemical propulsion problems. In mathematical form, a chemical (impulsive) thrust can be written as $\mathbf{T}^i = \sum_j^M m^i \Delta\mathbf{V}_j^i \delta(t - \tilde{t}_j^i)$. Here $\Delta\mathbf{V}_j^i$ are the velocity increments (impulses) that fully define the thrust law together with the times $\tilde{t}_j^i$. Note that we have used the Dirac delta generalized function $\delta(t - \tilde{t}_j^i)$.

The following sections will focus entirely on chemical propulsion problems, a particular instance of problem $\mathcal{P}'$ where we basically preassign a given analytical shape to the thrust strategy that becomes thus determined unequivocally by a finite number of parameters, effectively reducing the OCP into an NLP (from infinite dimension to a finite number). In the following two sections we introduce two particular instances of chemical propulsion problems: the MGA problem and the MGA-1DSM problem.

7.4 The MGA Problem

The MGA problem definition that follows is a generalization of that given by Izzo et al. [3]. We define the general form of the MGA trajectory optimization problem as the following instance of problem $\mathcal{P}'$

$$\begin{array}{rll} \text{Optimize:} & \phi(t_s^1, t_f^N, \mathbf{x}_s^1, \mathbf{x}_f^N) & \\ \text{Subject to:} & \dot{\mathbf{x}}^i = \mathbf{g}(\mathbf{x}^i) + \mathbf{u}^i(t), \forall i = 1..N & \text{dynamic con.} \\ & \mathcal{G}(t_s^1, t_f^N, \mathbf{x}_s^1, \mathbf{x}_f^N) \leq 0 & \text{boundary con.} \\ & \Phi^i(t_s^i, t_f^i, \mathbf{x}_s^i, \mathbf{x}_f^i) \leq 0, \forall i = 1..N & \text{phase boundary con.} \\ & \Xi^i(\mathbf{x}_s^{i+1}, \mathbf{x}_f^i, t_s^{i+1}, t_f^i) \leq 0, \forall i = 1..N-1 & \text{phase matching con.} \end{array} \tag{7.4}$$

where:

- Thrusting is possible only at the beginning of each trajectory leg and at arrival and is impulsive: $\mathbf{T}^i = m^i \Delta\mathbf{V}^i \delta(t - t_s^i), \forall i = 1..N-1, \mathbf{T}^N = m^N \Delta\mathbf{V}^N \delta(t - t_s^N) + \Delta\mathbf{V}^{N+1} \delta(t - t_f^N)$.

- The phase matching constraints include $\Delta m^i = 0$, $\Delta t^i = 0$, $\forall i = 1..N-1$ and $r_p(\mathbf{v}_f^i, \mathbf{v}_s^{i+1}, t_f^i) \geq \tilde{r}_p^i$, $\Delta \mathbf{V}^i = F_1(\mathbf{v}_f^i, \mathbf{v}_s^{i+1}, t_f^i)$, $\forall i = 2..N-1$. The expression for the functions F_1 and r_p are given in appendix A and model a planetocentric hyperbolic trajectory phase where the spacecraft is allowed to thrust tangentially at the periplanet (powered flyby).
- The boundary constraints include the launcher performances $m_s^1 = G(\mathbf{v}_s^1, t_s^1)$, the departure thrust impulse definition $\Delta \mathbf{V}^1 = M(\mathbf{v}_s^1, t_s^1)$, and the arrival thrust impulse definition $\Delta \mathbf{V}^{N+1} = N(\mathbf{v}_f^N, t_f^N)$. The expression for G, M, and N are problem dependent and are part of the further problem instantiation.

From an engineering point of view, these assumptions model a spacecraft equipped with chemical propulsion engines, that is, high thrust engines that deliver their acceleration to the spacecraft in a very short time and that are able to thrust only at each planet. As noted before, we no longer have an optimal control problem (OCP) as $\mathbf{u}^i = [\mathbf{0}, \frac{\mathbf{T}^i}{m^i}, 0]$ is now fully defined by a finite number of parameters. At this stage, the variables to be optimized, that is, the decision vector, are the initial mass m_s^1 and the start and final epochs of each leg t_s^i, t_f^i. The whole dynamic $\mathbf{x}^i(t)$ during each phase i, and thus all of the constraints can be, as shown in the following, determined by these variables, allowing us to simplify the problem solving explicitly most of the constraints, starting from the dynamic constraints, and thus reducing the problem complexity and dimension.

7.4.1 Spacecraft Position and Velocity

The dynamic constraints relative to $\mathbf{r}^i$ and $\mathbf{v}^i$ may be solved explicitly if we consider them separately grouped with the appropriate phase boundary constraints appearing in Equation (7.3) as follows

$$\begin{cases} \dot{\mathbf{r}}^i = \mathbf{v}^i \\ \dot{\mathbf{v}}^i = -\dfrac{\mu}{r^{i^3}}\mathbf{r}^i \\ \mathbf{r}_s^i = \mathbf{R}^i(t_s^i), \mathbf{r}_f^i = \mathbf{R}^{i+1}(t_f^i) \end{cases} . \tag{7.5}$$

For each phase, the above two-points boundary value problem defines what is commonly known as a Lambert's problem [2]. Such a problem admits $2(1 + 2M_i)$ solutions where $M_i \in [0, \infty)$ is an integer depending on the boundary conditions in a rather complex way. Solutions to the Lambert's problem are commonly divided into posigrade orbits and retrograde orbits according to the direction of their angular momentum vector $\mathbf{h}^i = \mathbf{r}^i \wedge \mathbf{v}^i$ with respect to the ecliptic frame. Also, they are divided into multirevolution and single revolution, according to the number of times the initial position is acquired during the trajectory (one or more). To simplify the problem, we consider only single-revolution and posigrade solutions (a general formal treatment of the MGA problem not using these hypotheses can be found in Izzo et al. [3]). Under these hypotheses, Lambert's problem always admits a unique solution, and we may thus derive the relations $\mathbf{r}^i(t) = \mathbf{L}_r^i(t, t_s^i, t_f^i)$, $\mathbf{v}^i(t) = \mathbf{L}_v^i(t, t_s^i, t_f^i)$.

7.4.2 Spacecraft Mass

Also the remaining dynamic constraint, relative to the mass, can be solved explicitly. From the MGA problem definition we know the thrust law along each phase. Considering last of Equation (7.5) (relative to the spacecraft mass), we have $\dot{m}^i = -\frac{\Delta V_i}{I_{sp}g_0} m^i \delta(t - t_s^i)$. Integrating from t_s^i to t_f^i, we may write explicitly the mass as a function of time using the Heaviside step function H and explicitly using the phase matching constraints $\Delta m^i = 0$

$$m^i(t) = m_s^1 \exp\left(-\frac{\sum_{i=1}^{N} \Delta V^i H(t - t_s^i) + \Delta V^{N+1} H(t - t_f^N)}{I_{sp}g_0}\right). \tag{7.6}$$

Eliminating m_s^1 and ΔV_i in the expression above by explicitly using the various constraints F_1, G, M, N in the general MGA problem definition, we may eventually write $m^i(t) = \mathbf{L}_m^i(t, t_s^1, .., t_s^i, t_f^1, .., t_f^i)$.

7.4.3 The Final Form

In the remaining phase, matching constraint we still have, from the definition of the MGA problem, $\Delta t^i = 0 \rightarrow t_f^i = t_s^{i+1}$, which allows us to eliminate $N - 1$ further variables. Choosing as new variable notation $t_0 = t_s^1$ and $t_i = t_f^{i-1}$, $\forall i = 1..N$ we define the decision vector as $\mathbf{p} = [t_0, t_1, .., t_N]$. For each choice of the decision vector, the whole spacecraft state $\mathbf{x}^i(t)$ is known throughout phase i and the general form of the MGA problem described by Equation (7.4) is thus reduced to

$$\begin{array}{rlll} \text{Optimize:} & \phi(\mathbf{p}) & & \\ \text{Subject to:} & \mathcal{G}(\mathbf{p}) \leq 0 & \text{boundary con.} & \\ & r_p(t_{i-1}, t_i, t_{i+1}) \geq \tilde{r}_p^i & \text{phase matching con.} & \end{array} \tag{7.7}$$

where no phase boundary constraints are left as they all are explicitly satisfied. The above problem is defined here as the MGA problem and has a dimension $N + 1$.

We have formally presented the procedure to transcribe a trajectory optimization problem into an MGA problem, highlighting the hypotheses that underlie this particular transcription. Further specification of the objective function, of the boundary constraints, and of the decision vector bounds will create different instances of the MGA problem.

7.5 The MGA-1DSM Problem

The MGA-1DSM problem definition that follows generalizes a number of particular problem instances studied in previous works [4, 5, 6]. We define the general form of the MGA-1DSM trajectory optimization problem as the following instance of

problem $\mathcal{P}'$

$$\begin{array}{rll} \text{Optimize:} & \phi(t_s^1, t_f^N, \mathbf{x}_s^1, \mathbf{x}_f^N) & \\ \text{Subject to:} & \dot{\mathbf{x}}^i = \mathbf{g}(\mathbf{x}^i) + \mathbf{u}^i(t), \forall i = 1..N & \text{dynamic con.} \\ & \mathcal{G}(t_s^1, t_f^N, \mathbf{x}_s^1, \mathbf{x}_f^N) \leq 0 & \text{boundary con.} \\ & \Phi^i(t_s^i, t_f^i, \mathbf{x}_s^i, \mathbf{x}_f^i) \leq 0, \forall i = 1..N & \text{phase boundary con.} \\ & \Xi^i(\mathbf{x}_s^{i+1}, \mathbf{x}_f^i, t_s^{i+1}, t_f^i) \leq 0, \forall i = 1..N-1 & \text{phase matching con.} \end{array} \tag{7.8}$$

where:

- Thrusting is possible only at departure, at arrival, and once at some point along each phase and is impulsive: $\mathbf{T}^1 = m^1 \Delta\mathbf{V}^0 \delta(t - t_s^1) + \Delta\mathbf{V}^1 \delta(t - \tilde{t}_1)$, $\mathbf{T}^i = m^i \Delta\mathbf{V}^i \delta(t - \tilde{t}^i), \forall i = 2..N-1, \mathbf{T}^N = m^N \Delta\mathbf{V}^N \delta(t - \tilde{t}^N) + \Delta\mathbf{V}^{N+1} \delta(t - t_f^N)$.
- The phase matching constraints include $\Delta m^i = 0$, $\Delta t^i = 0$, $\forall i = 1..N-1$, $r_p(\mathbf{v}_f^i, \mathbf{v}_s^{i+1}, t_f^i) \geq \tilde{r}_p^i \ \forall i = 1..N-1$ and $\tilde{v}_{in}^i = \tilde{v}_{out}^i, \forall i = 1..N-1$, where $\tilde{v}_{in}^i = |\mathbf{v}_f^i - \mathbf{V}^{i+1}(t_f^i)|$, $\tilde{v}_{out}^i = |\mathbf{v}_s^{i+1} - \mathbf{V}^{i+1}(t_s^{i+1})|$, and the functional relationship r_p is the same as in the MGA problem and is given in appendix.
- The boundary constraints include the launcher performances $m_s^1 = G(\mathbf{v}_s^1, t_s^1)$, the departure thrust impulse definition $\Delta\mathbf{V}^0 = M(\mathbf{v}_s^1, t_s^1)$, and the arrival thrust impulse definition $\Delta\mathbf{V}^{N+1} = N(\mathbf{v}_f^N, t_f^N)$. The expression for G, M, and N are problem dependent and are part of the further problem instantiation.

These hypotheses model a spacecraft equipped with chemical propulsion engines, able to thrust at departure, at arrival, and only once during each trajectory phase and never during planetary flybys. The MGA-1DSM problem removes most of the limitation of the MGA problem and is an accurate problem transcription for many preliminary trajectory design cases. The most important remaining limitation of this problem transcription is in the fixed number of DSM allowed in each phase. As in the MGA problem, the optimal control problem is transformed into an NLP problem as the control $\mathbf{u}^i(t)$ is now fully parameterized by a discrete number of variables. At this stage, the variables to be optimised, that is, the decision vector, are the initial mass m_s^i and, for each phase, the vector $\mathbf{p}^i = [t_s^i, \mathbf{v}_s^i, \tilde{t}^i, t_f^i]$. The whole dynamic during each phase $\mathbf{x}^i(t)$ is, as shown in the following, determined analytically by these variables, allowing us to simplify the problem solving explicitly most of the constraints, starting from the dynamic constraints, and thus reducing the problem complexity and dimension.

7.5.1 Spacecraft Position and Velocity

Let us focus on the i-th phase. Given t_s^i and $\mathbf{v}_s^i$, it is possible to find $\mathbf{r}^i(t)$ and $\mathbf{v}^i(t)$ for $t \leq \tilde{t}^i$ using the known analytical solution to Kepler's problem expressed, for example, in terms of the Lagrange coefficients [2]. The initial position is also known from

the phase boundary constraints and is $\mathbf{r}_s^i = \mathbf{R}^i(t_s^i)$. From $\tilde{t}^i$ to t_f^i, we also can derive explicitly the spacecraft position and velocity by considering (as in the MGA case) the dynamic constraints separately grouped with the appropriate phase boundary constraints appearing in Equation (7.3) as follows

$$\begin{cases} \dot{\mathbf{r}}^i = \mathbf{v}^i \\ \dot{\mathbf{v}}^i = -\frac{\mu}{r^{i^3}}\mathbf{r}^i \\ \mathbf{r}^i(t_s^i) = \mathbf{r}^i(\tilde{t}^i), \mathbf{r}^i(t_f^i) = \mathbf{R}^{i+1}(t_f^i). \end{cases}$$

This is again a Lambert's problem that, under the same hypotheses assumed for the MGA case, always admits a unique solution and allows us to derive the relations $\mathbf{r}^i(t) = \mathbf{L}_r^i(t, \mathbf{p}^i)$, $\mathbf{v}^i(t) = \mathbf{L}_v^i(t, \mathbf{p}^i)$. Note that in $\tilde{t}^i$, the spacecraft velocity will be discontinuous as the value evaluated using the Lagrange coefficients will generally differ from that returned by the Lambert's problem solution. Such a discontinuity defines the velocity increment $\Delta\mathbf{V}^i, i = 1..N$ as a function of $\mathbf{p}^i$.

7.5.2 Spacecraft Mass

As was done in the case of the MGA problem, by explicitly using the phase matching constraints $\Delta m^i = 0$ and the equation for the mass, we may derive the simple expression

$$m^i(t) = m_s^1 \exp\left(-\frac{\Delta V^0 H(t - t_s^1) + \sum_{i=1}^{N} \Delta V^i H(t - t_s^i) + \Delta V^{N+1} H(t - t_f^N)}{I_{sp} g_0}\right). \tag{7.9}$$

Eliminating m_s^1 and the ΔV^i in the expression above by explicitly using the launcher performance, the departure and arrival thrust impulse definitions, we may eventually write $m^i(t) = \mathbf{L}_m^i(t, \mathbf{p}^1, .., \mathbf{p}^i)$.

7.5.3 The Final Form

All the remaining phase constraints from the definition of the simple MGA-1DSM problem may also be explicitly satisfied using suitable variable changes in the decision vector. In particular we may substitute the variables $\mathbf{v}^i$ for each phase, except the first one, with the periplanet of the planetocentric hyperbola r_p^i and the b-plane orientation β^i using the existing functional relationship

$$\mathbf{v}_s^{i+1} = F_2(\mathbf{v}_f^i, \mathbf{V}^{i+1}(t_s^{i+1}), r_p^{i+1}, \beta^{i+1}) \tag{7.10}$$

that describes explicitly the flyby dynamics satisfying explicitly the phase constraints $\tilde{v}_{in}^i = \tilde{v}_{out}^i$, $\forall i = 1..N-1$ (see Appendix A) and allows to transform the nonlinear

constraint $r_p(\mathbf{v}_f^i, \mathbf{v}_s^{i+1}, t_f^i) = r_p^i \geq \tilde{r}_p^i$ into a lower bound for the introduced decision vector variable r_p^i. Eventually eliminating further variables using the phase matching constraint $\Delta t^i = 0$, we have a decision vector $\tilde{\mathbf{p}} = [t_s^1, \mathbf{v}_s^1, \tilde{t}^1, t_f^1, r_p^2, \beta^2, \tilde{t}^2, t_f^2, ..]$. To simplify the search space structure, we introduce some variable substitutions. Instead of the heliocentric spacecraft initial velocity $\mathbf{v}_s^1$, we use the variables V_∞, u, v defined as

$$\begin{aligned}
\mathbf{v}_s^1 &= V_\infty \left(\cos(\theta)\cos(\phi)\hat{\mathbf{i}} + \sin(\theta)\cos(\phi)\hat{\mathbf{j}} + \sin(\phi)\hat{\mathbf{k}}\right) \\
\theta &= 2\pi u \\
\phi &= \arccos(2v - 1) - \pi/2 \\
\hat{\mathbf{i}} &= V^1(t_s^1)/|V^1(t_s^1)| \\
\hat{\mathbf{k}} &= R^1(t_s^1) \wedge V^1(t_s^1)/|R^1(t_s^1) \wedge V^1(t_s^1)| \\
\hat{\mathbf{j}} &= \hat{\mathbf{k}} \wedge \hat{\mathbf{i}}.
\end{aligned}$$

Also, instead of the absolute epochs t_f^i we use the transfer times $T^1 = t_f^1 - t_s^1$, $T^i = t_f^i - t_f^{i-1}$ and instead of $\tilde{t}^i$ we use η^i defined as $\tilde{t}^1 = t_s^1 + T^1\eta^1$, $\tilde{t}^i = t_f^{i-1} + T^i\eta^i$. The decision vector used will thus be $\tilde{\mathbf{p}}' = [t_s^1, V_\infty, u, v, \eta^1, T^1, r_p^2, \beta^2, \eta^2, T^2, ..]$. This allows us to specify as upper and lower bounds on the decision vector what would otherwise need to be nonlinear constraints. Eventually the general form of the simple MGA-1DSM problem described by Equation (7.8) is reduced to

$$\begin{aligned}
&\text{Optimize:} && \phi(\tilde{\mathbf{p}}') \\
&\text{Subject to:} && \mathcal{G}(\tilde{\mathbf{p}}') \leq 0 \quad \text{boundary con.}
\end{aligned} \tag{7.11}$$

No phase matching constraints or phase boundary constraints are left as we have explicitly satisfied all of them, reducing the search space structure to a hyper-rectangle. The remaining boundary constraints, as detailed later in the description of particular instances of this problem, express, typically, a maximum trajectory length duration or other particular mission requirements. The MGA-1DSM problem, as defined here, includes trajectories with multiple revolutions and with no deep space maneuvers in a particular phase. Thus, this particular transcription creates a continuous optimization problem while being able to describe discrete decision variables such as the number of revolutions or the use of a deep space maneuver during a trajectory phase.

7.6 Benchmark Problems

We now describe the detailed instantiation of some MGA and MGA-1DSM problems that can be used as test problems to study the performance of global optimization solvers or of space pruning techniques. All problems proposed present very large bounds to be representative of the type of trajectory optimization often required in preliminary mission design phases. For each problem we define the flyby sequence,

the objective function, the bounds on the decision vector variables, and the constraint expressions. Clearly there are a number of other factors that influence, to some smaller extent, the exact calculation of the objective function, such as the values of the different planetary gravitational constants, the planet ephemerides used (that is, the functions $\mathbf{R}^i, \mathbf{V}^i$), the planets' radii, and so on. To allow the scientific community to share a common implementation of each problem instance, the code in C++ and Matlab of each one of the problems described in detail here is available for download from the European Space Agency Global Trajectory Optimisation Problems (GTOP) database [7]. A preliminary description of some of these test problems is also given by Vinko [8]. We also report the best putative global optima known at the time of writing as taken from the GTOP database where the reader can also find exact numerical details of the solutions here reported.

7.6.1 Cassini1

This problem is a particular instance of the MGA problem as defined in Equation (7.7). It has $N = 5$ phases and hence $N+1 = 6$ celestial bodies are defined in the flyby sequence: Earth, Venus, Venus, Earth, Jupiter, and Saturn. Thus the decision vector is $\mathbf{p} = [t_0, .., t_5]$ and contains the epochs of each planetary encounter. The objective function is defined as $\phi(\mathbf{p}) = -g_0 I_{sp} \log(m_f^N / m_s^1)$, where the ratio between the final and the initial mass m_f^N / m_s^1 is given by Equation (7.6) evaluated at t_f^N. Note that, after taking the logarithm, the objective function is essentially the sum of the various velocity increments: $\phi(\mathbf{p}) = \sum \Delta V_i$. No further constraints are considered except those appearing in the generic MGA problem definition. The Cassini1 problem can be written as

$$\begin{array}{ll} \text{Minimize:} & -g_0 I_{sp} \log(m_f^N / m_s^1) = \sum \Delta V_i \\ \text{Subject to:} & r_p(t_{i-1}, t_i, t_{i+1}) \geq \tilde{r}_p^i, \forall i = 1..N-2 \quad \text{phase matching con.} \end{array} \tag{7.12}$$

The departure thrust impulse is defined as $\Delta V_1(t_0, t_1) = |\mathbf{V}^1(t_0) - \mathbf{v}_s^1|$, the arrival thrust impulse is defined as an orbital insertion as detailed in the Appendix A (r_p^{ins} and e^{ins} are given in Table 7.3). The launcher performance is in this case not relevant, as the value of the initial spacecraft mass m_s^1 does not appear anywhere in the problem definition (the objective function depends only on the velocity increments). Introducing the variable change $T_i = t_i - t_{i-1}, i = 1..5$, the exact values needed to define this test problem are given in Table 7.1. The Cassini1 problem admits a putative global optima at $\phi = 4.93$ km/s and is characterized by a number of local optima with a very strong basin of attraction, particularly noteworthy the one at $\phi = 5.30$ km/s that seems to be very difficult for many optimization techniques to overcome.

7.6.2 GTOC1

This problem is a particular instance of the MGA problem as defined in Equation (7.7). It has $N = 7$ phases and hence $N + 1 = 8$ celestial bodies (the flyby

Table 7.1. *Bounds and other parameters for the problem Cassini1*

Variable	Lower Bound	Upper Bound	units	parameter	value	units
t_0	-1000	0	(MJD2000)	$\tilde{r}_p^1$	6351.8	km
T_1	30	400	days	$\tilde{r}_p^2$	6351.8	km
T_2	100	470	days	$\tilde{r}_p^3$	6778.1	km
T_3	30	400	days	$\tilde{r}_p^4$	671492	km
T_4	400	2000	days	r_p^{ins}	108950	km
T_5	1000	6000	days	e^{ins}	0.98	

Table 7.2. *Bounds and other parameters for the problem GTOC1*

Variable	Lower Bound	Upper Bound	units	parameter	value	units
t_0	3000	10000	(MJD2000)	$\tilde{r}_p^1$	6351.8	km
T_1	14	2000	days	$\tilde{r}_p^2$	6778.1	km
T_2	14	2000	days	$\tilde{r}_p^3$	6351.8	km
T_3	14	2000	days	$\tilde{r}_p^4$	6778.1	km
T_4	14	2000	days	$\tilde{r}_p^5$	600000	km
T_5	100	9000	days	$\tilde{r}_p^6$	70000	km
T_6	366	9000	days	ΔV_{lau}	2.5	km/s
T_7	300	9000	days	m_0	1500	kg
				I_{sp}	2500	sec

sequence) are forced to be encountered by the spacecraft along its trajectory: Earth, Venus, Earth, Venus, Earth, Jupiter, Saturn, TW299. Thus, the decision vector is $\mathbf{p} = [t_0, .., t_7]$ and contains the epochs of each planetary encounter. The objective function is defined as $\phi(\mathbf{p}) = m_f^N |(\mathbf{V}^{N+1} - \mathbf{v}_f^N) \cdot \mathbf{V}^{N+1}|$ and no further constraints are considered except those appearing in the generic MGA problem definition. The GTOC1 problem can be written as

$$\begin{array}{ll} \text{Maximize:} & \phi(\mathbf{p}) = m_f^N |(\mathbf{V}^{N+1} - \mathbf{v}_f^N) \cdot \mathbf{V}^{N+1}| \\ \text{Subject to:} & r_p(t_{i-1}, t_i, t_{i+1}) \geq \tilde{r}_p^i, \forall i = 1..N-2 \quad \text{phase matching con.} \end{array} \tag{7.13}$$

where the final mass m_f^N is given by Equation (7.6) evaluated in t_f^N. The departure thrust impulse is defined as $\Delta V_1(t_0, t_1) = \max(|\mathbf{V}^1(t_0) - \mathbf{v}_s^1| - \Delta V_{lau}|, 0)$, the arrival thrust impulse is defined as $\Delta V^{N+1} = 0$. The launcher performance is defined as $m_s^1 = m_0$. Introducing the variable change $T_i = t_i - t_{i-1}, i = 1..7$, the exact values needed to define this test problem are given in Table 7.2. The GTOC1 problem admits a putative global optima at $\phi = 1,580,599$ kg km^2 /s^2 and is characterized by a large number of local optima at almost all objective function ranges.

7.6.3 Rosetta

This problem is a particular instance of the MGA-1DSM problem as defined in Equation (7.11). It has $N = 5$ phases and hence $N + 1 = 6$ celestial bodies (the

flyby sequence) are forced to be encountered by the spacecraft along its trajectory: Earth, Earth, Mars, Earth, Earth, Jupiter, Saturn, 67P/Churyumov-Gerasimenko.. Thus, the decision vector is $\tilde{\mathbf{p}} = [t_s^1, \mathbf{v}_s^1, \tilde{t}^1, t_f^1, r_p^2, \beta^2, \tilde{t}^2, t_f^2, ..]$. The objective function is defined as $\phi(\mathbf{p}) = -g_0 I_{sp} \log(m_f^N / m_s^1)$, where the ratio between the final and the initial mass m_f^N / m_s^1 is given by Equation (7.9) evaluated in t_f^N. Note that, after taking the logarithm, the objective function is essentially the sum of the various velocity increments: $\phi(\mathbf{p}') = \sum \Delta V_i$. The Rosetta problem can thus be written as

$$\text{Minimize:} \quad -g_0 I_{sp} \log(m_f^N / m_s^1) = \sum \Delta V_i \tag{7.14}$$

and is an unconstrained global optimization problem. The departure velocity increment is defined as $\Delta V_0 = 0$, the arrival velocity increment is defined as $\Delta V_{N+1}(\mathbf{p}') = |\mathbf{V}^N(t_f^N) - \mathbf{v}^N(t_f^N)|$. The launcher performance is in this case not relevant, as the value of the initial spacecraft mass m_s^1 does not appear anywhere in the problem definition (the objective function depends only on the velocity increments). The exact values needed to define this test problem are given in Table 7.3. The Rosetta problem admits a putative global optima at $\phi = 1.34$ km/s.

Table 7.3. *Lower and upper bounds defining the MGA-1DSM problems Rosetta and TandEM*

Rosetta				TandEM	
Variable	LB	UB	units	LB	UB
t_0	1460	1825	MJD2000	5475	9132
V_{inf}	3	5	km/s	2.5	4.9
u	0	1		0	1
v	0	1		0	1
T_1	300	500	days	20	2500
T_2	150	800	days	20	2500
T_3	150	800	days	20	2500
T_4	300	800	days	20	2500
T_5	700	1850	days		
η^1	0.01	0.9		0.01	0.99
η^2	0.01	0.9		0.01	0.99
η^3	0.01	0.9		0.01	0.99
η^4	0.01	0.9		0.01	0.99
η^5	0.01	0.9			
r_p^1	1.05	9	Planet Radii	1.05	10
r_p^2	1.05	9	Planet Radii	1.05	10
r_p^3	1.05	9	Planet Radii	1.05	10
r_p^4	1.05	9	Planet Radii		
β^1	$-\pi$	π	rad	$-\pi$	π
β^2	$-\pi$	π	rad	$-\pi$	π
β^3	$-\pi$	π	rad	$-\pi$	π
β^4	$-\pi$	π	rad		

7.6.4 Constrained TandEM–Atlas501–EVEEJ

This problem is a particular instance of the MGA-1DSM problem as defined in Equation (7.11). It has $N = 4$ phases and hence $N + 1 = 5$ celestial bodies (the flyby sequence) are forced to be encountered by the spacecraft along its trajectory: Earth, Venus, Earth, Earth, Jupiter. Thus the decision vector is $\tilde{\mathbf{p}} = [t_s^1, \mathbf{v}_s^1, \tilde{t}^1, t_f^1, r_p^2, \beta^2, \tilde{t}^2, t_f^2, ..]$. The objective function is defined as $\phi(\mathbf{p}) = m_f^N$, where the final mass is given by Equation (7.9) evaluated in t_f^N. We also introduce a global constraint on the total trajectory duration. The TandEM-Atlas501-EVEEJ (in the following "TandEM problem" for brevity) can thus be written as

$$\begin{aligned} &\text{Minimize:} && m_f^N \\ &\text{subject to:} && t_f^N - t_s^1 \leq t_{tot} \end{aligned} \tag{7.15}$$

and is a constrained global optimization problem where the total trajectory time is limited to $t_{tot} = 10$ years. The departure thrust impulse is defined as $\Delta V_0 = 0$, the arrival thrust impulse is defined as an orbital insertion as detailed in the Appendix ($r_p^{ins} = 80,330$ km and $e^{ins} = 0.9853$). The launcher performance is that of Atlas-501 obtained as detailed in Appendix A using the table given by NASA Launch Services (NLS) Launcher Performances. Outside the reported declinations ($\pm$ 28.5 deg), a null mass is considered. The TandEM problem admits a putative global optima at $\phi = 1437.58$ kg.

7.7 Global Optimization

The advantage of transcribing a given trajectory design problem into a well-defined optimization problem stems from the possibility of using computer algorithms to achieve a complete automation of the design. Each problem instance can in fact be coded into a black-box function expressing the functional relationship between the decision vector and a figure of merit expressing the quality of the related trajectory and its constraint violations. Derivative-free global optimization algorithms can then be applied to try finding optimal solutions. This approach to designing spacecraft trajectories is reaching a great maturity and promises a completely unbiased and automated listing of optimal trajectory options. Algorithms such as Differential Evolution, Genetic Algorithms, Particle Swarm Optimization, Simulated Annealing, and, even more recently, Monotonic Basin Hopping, just to quote a few, have all been tried with different degrees of success. It is important to be aware that all of the above algorithms are able to solve some particular instances of the trajectory problems. Simply taking a suitable problem instance, it is possible to achieve good performances in almost all cases. It is crucial to escape the temptation to pick an algorithm, perhaps introduce some modifications, and present a few pseudo-randomly selected trajectory optimizations where its performances seem good. With this respect, the test problems introduced here, in Vinko et al. [8], and in general

Table 7.4. *Performances of off-the-shelf solvers on two MGA test problems over 100 runs*

Problem	Paradigm	DE	PSO	MPSO	SA-AN	SGA
Cassini1 (min = 4.93)						
	Mean	8.57	9.47	7.05	11.67	7.09
	Std	3.29	4.20	2.39	4.236	2.30
	Min	4.93	5.33	5.43	5.12	5.44
	Max	16.71	22.90	15.53	23.44	17.98
GTOC1 (max = 1,580,599)						
	Mean	1,140,759	759,221	602,331	1,179,835	907,781
	Std	146,589	174,463	125,163	139,590	232,290
	Min	836,772	295,143	341,780	846,720	81,820
	Max	1,523,629	1,134,860	876,177	1,511,767	1,416,050

Table 7.5. *Performances of off-the-shelf solvers on two MGA-1DSM test problems over 100 runs*

Problem	Paradigm	DE	PSO	MPSO	SA-AN	SGA
Rosetta (min = 1.34)						
	Mean	7.55	10.36	10.76	4.13	9.95
	Std	1.82	2.72	1.93	0.94	3.29
	Min	3.94	5.34	7.01	2.61	4.35
	Max	13.26	15.79	15.46	6.65	17.52
TandEM (max = 1437.58)						
	Mean	216.32	144.03	108.12	625.26	78.75
	Std	80.35	135.79	50.42	254.62	80.50
	Min	95.28	22.42	33.57	60.43	3.78
	Max	460.87	862.64	321.37	1298.19	498.22

those present in the GTOP database [7] can offer a significant help in understanding the value of any proposed algorithm under the conditions that the bounds, the parameter values, and the underlying models are left untouched. They describe a fair range of quite complex interplanetary spacecraft trajectory optimization problems. Some of them have a rather academic value, like Cassini1 or GTOC1, and some are instead quite close to the type of problems mission designers solve in preliminary phases of the trajectory design process, like Rosetta or the constrained TandEM problem.

In Table 7.4 and Table 7.5, we list the performances that several standard implementation of popular heuristics paradigms achieve on the described test problems. These results can give a feel for the type of performances one can expect by applying

a given paradigm but should in no way be considered as a general comparison table between paradigms, as the tables are obtained by choosing a particular algorithmic setting that can, no doubt, be improved by tuning appropriately the various constants or by changing some implementation details. The algorithms tested are all well described in the literature and we here only briefly touch upon them:

- DE: The Differential Evolution paradigm has been found by Myatt et al. [9] to be a good solver for spacecraft trajectory optimization problems and is described in detail in the work by Storn and Price [10]. It is fully defined by the strategy adopted (several are proposed in the original paper) the population size NP, and two parameters, the weighting factor F and the crossover ratio CR.
- PSO: The Particle Swarm Optimization algorithm, in its simplest form, has been proposed by Kennedy and Eberhart [11] and is fully defined by the number of particles NP, the inertia weight ω, by the cognitive component factor η_1, and the social component factor η_2.
- MPSO: The Multiple Particle Swarm Optimization algorithm is a variation to the canonic PSO whereby multiple swarms are performing the search independently except randomly swapping every k iterations the swarm membership. It has been found to provide some advantages over PSO in [12] and is fully defined by k, the number of particles NP, the inertia weight ω, the cognitive component factor η_1, the social component factor η_2, and the number of swarms n.
- SA-AN: The Simulated Annealing with Adaptive Neighborhood paradigm [13] is a variation to the Simulated Annealing, where the sampling neighborhood for each decision vector component is adaptively changed according to the acceptance rate of new solutions. There are many ways of implementing such a paradigm; here we use the algorithm proposed by Corana [13] detailed in the Appendix B using re-annealing and that is fully defined by the starting temperature T_i, the final temperature T_f, and the annealing speed nf_{ann} defining the number of function evaluation allowed for each annealing.
- SGA: The Simple Genetic Algorithm is a basic version of a genetic algorithm [14] that uses roulette wheel selection, exponential crossover, uniform mutation, and an elitist strategy. The free parameters for this algorithm are the population size NP, the mutation probability M of each gene, and the crossover ratio CR defined in the same way as that of the DE algorithm.

In the test results presented here, each algorithm was left free to calculate for enough function evaluations allowing a statistical convergence of the results. That is, the results given in the tables refer to a fixed number of function evaluations $FEVAL$ and would not be significantly statistically different (a pair-wise Welsh test has been carried out with a confidence level of 95%) if the results were compiled after $FEVAL/2$ function evaluations. For all tests, $FEVAL = 1{,}200{,}000$ except for Cassini1 where $FEVAL = 80{,}000$. All population-based algorithms were tested with a population size $NP = 20$. The parameters for DE are $F = 0.7, CR = 0.7$, and the strategy DE/rand/1/exp was used. For PSO we used $\omega = 0.65$, $\eta_1 = \eta_2 = 2$. For

MPSO we used the same settings as for PSO and $n = 4$ swarms, that is, each swarm had five particles. For the GA we used $M = 0.2$ and $CR = 0.7$. For SA-AN we used a starting temperature $T_i = 10$ for Cassini1 and Rosetta, $T_i = 100{,}000$ for GTOC1 and $T_i = 1$ for TandEM, a final temperature of $T_f = 0.036$ for Cassini1 $T_f = 138$ for GTOC1, $T_f = 0.073$ for Rosetta, and $T_f = 0.0024$ for TandEM. All SA-AN simulations were allowed $nf_{sa-an} = 10,000$ function evaluations per annealing cycle.

7.7.1 Discussion

The results presented aim at showing that the standard implementations of the global optimization algorithms tested all have, to different degrees, quite poor reliability and performances if applied directly to the difficult interplanetary test problems proposed. The straightforward use of these algorithms seem thus to fail providing the hoped automated and unbiased approach to complex trajectory optimization. Tuning the various algorithms parameters can surely offer a performance improvement that comes, though, at the cost of further objective function evaluations and needs to be performed for each problem instance. From the results reported here and from the experience accumulated by this author in several other experiments, we can claim that DE and SA-AN are the best performing algorithms, DE being particularly efficient in MGA problems and SA-AN, whose performances are reported here for the first time, appearing to outperform all others in MGA-1DSM problems. A further performance improvement can also be obtained by letting heterogeneous versions of these algorithms run in parallel in a so-called island model[1], as recently suggested by Izzo et al. [15]. Other algorithms are available that have not been tested here, and new ones are certain to come in the future. Noteworthy is the case of the Basin Hopping algorithm [16] that has been very recently applied, to the knowledge of this author, for the first time by Bernadetta Addis, Fabio Schoen, and Marco Locatelli to a great number of interplanetary trajectory problem instances (also to the ones described here) locating reliably good solutions and beating consistently all the known global optima [7]. Another approach able to find reliably a large number of good trajectory options in reasonable computational times is that of applying global optimization algorithms on a reduced portion of the search space obtained by pruning out regions according to some predefined criteria. This way the problem instance complexity can be substantially reduced allowing different algorithms to reliably converge in short times to optimal solutions. In the next sections, we will introduce two of such techniques and we will comment on how, while allowing for an automated and efficient search, they introduce the need to perform a number of

[1]The island model is one successful paradigm to perform heuristic global optimization in a CPU network by having different sets of solutions being optimized separately in different islands, typically assigned to different CPUs, and letting some solutions stochastically move to new islands following predefined "migration" paths. A careful setup of such a system can bring to an improvement that is superlinear with the respect to the number of CPUs used and eventually to the definition of a new algorithm outperforming those operating on the single islands.

choices on the pruning criteria that need to be carefully made as not to introduce unwanted biases in the optimization process.

7.8 Space Pruning

The term space pruning refers to all techniques that allow reduction of search space focussing the optimization in smaller areas where the optimal solutions are to be found. The output of a typical pruning process is a set of hyperrectangles contained in the original search space where, according to some criteria, good solutions are expected. There are a large number of criteria that can be adopted to define such regions and that are dependent on the particular problem instance considered, here we introduce techniques that are generically applicable to a whole problem class, regardless of the instantiation details.

7.8.1 Pruning the MGA Problem: GASP

For the MGA problem an efficient pruning method is that developed by Myatt et al. [9, 3] and named Gravity Assist Space Pruning (GASP). The details of such a technique are well described in these two references and are thus not reported here. The method is based on the possibility of incrementally dividing the MGA problem in a cascade of two-dimensional problems where grid sampling is computationally efficient. Propagating back and forward pruning criteria defined on the flyby constraint satisfaction and on maximum ΔV^i allowed, it is then possible to reduce the number of sampled points to a fraction of the original space. For the Cassini1 problem, a space reduction of six-order of magnitude is reported. The polynomial complexity of the resulting algorithm has also been demonstrated both with respect to the grid size defined for each two-dimensional problem and to the overall problem dimension. While polynomial complexity as such does not necessarily lead to efficient algorithms, in the case of GASP the low exponents involved produce an incredibly fast pruning algorithm that allow to reliably solve MGA problems locating all good launch windows and the globally optimal solution contained therein. Recent attempts have been made to improve the GASP algorithm to allow a mathematical proof on the global optimality of the found solution [17] or to extend it to low-thrust trajectories [18] and to problems similar to the MGA-1DSM [19, 20, 21]. In these methods, polynomial complexity is retained, but with much larger exponents, or to the price of an excessive problem simplification that make the resulting implementations of a rather limited use.

7.8.2 Pruning the MGA-1DSM Problem: Cluster Pruning

The cluster pruning algorithm we propose here was developed by Marco del Rey Zapatero and by this author during March 2008 at the Advanced Concepts Team and is reported here for the first time. It results in the possibility to achieve full automation of the trajectory optimization process at the cost of employing appropriate computing

power. The algorithm is based on the observation that in the MGA-1DSM problem, good solutions are often clustered in a small portion of the solution space rather than being equally distributed all over it. This is quite intuitive for variables such as t_s^1, that is, the launch date, but is also equally valid for the other variables, as we will see. Each cluster of solutions corresponds to a different strategy and can be detected automatically and further explored in more depth as isolated problems by global optimization algorithms. The following pseudoalgorithm illustrates the approach in more detail:

(1) instantiate the problem with bounds LB, UB
(2) while not *convergence-criterion*
(3) perform N optimizations using the algorithm $\mathcal{A}$
(4) refine the N solutions found using a local optimization technique (optional)
(5) identify clusters of good solutions
(6) prune according to the results obtained and define new bounds LB, UB

In step 1, an MGA1-DSM problem is instantiated. In the following main loop, a global optimization algorithm is used to produce N solutions to the problem. The solutions, representing local optima of varying quality, are then optimized locally; this step is not necessary but helps in reducing the total pruning steps. The obtained trajectories are later analyzed to identify regions where the best of them lie and thus to produce reduced bounds. At each iteration, a smaller space is produced, and the N following optimizations produce increasingly better solutions. The baseline performances of the chosen algorithm $\mathcal{A}$ are crucial to the success of the process, as is the method used to produce the new bounds (or the set of new bounds in case multiple clusters are allowed to be selected).

We illustrate the detailed steps of a possible implementation of the algorithm in the particular case of the MGA-1DSM problem TandEM. We select one the best performing algorithms from Table 7.5, that is Simulated Annealing with Adaptive Neighborhood. Performing $N = 100$ optimization on the full problem, we obtain the relatively poor results reported on the same table. We evaluate the p percentile of the different objective functions returned and consider only the decision vectors $\mathbf{x}$ that are above such a value. We set the new bounds to $LB_{new_i} = min(x_i) - (UB_i - LB_i)p/100/2$ and $UB_{new_i} = max(x_i) + (UB_i - LB_i)p/100/2$ only if these are still within the old bounds.

In Table 7.6 we report the results of the $N = 100$ runs of the SA-AN algorithm at each pruning step (we perform four iterations of the cluster pruning algorithm with increasing percentile levels of 80, 90, and 95. At the end of each iteration, we run a local optimizer starting from all the N trajectories found. A total of 400 locally optimal trajectories is thus computed during the process.). In Figure 7.1, as an example, we show the values of u and T_3 for these 400 solutions together with the bounds at each pruning steps. The final best solution can be further improved by performing iteratively local optimization starting from a close neighborhood. We thus find a new putative global optimum at $m_f^N = 1{,}476.01$ kg improving the known

Table 7.6. *Stochastic pruning for the TandEM-Atlas501-6 MGA–1DSM problem*

Problem	Pruning iterations	0	1 ($p = 80$)	2 ($p = 90$)	3 ($p = 95$)
TandEM-Atlas501-6 (max)	Mean	625.26	838.94	1060.74	1445.45
	Std	254.62	269.31	262.08	38.133
	Min	60.43	249.37	341.2	1370.52
	Max	1298.19	1381.38	1475.22	1475.71

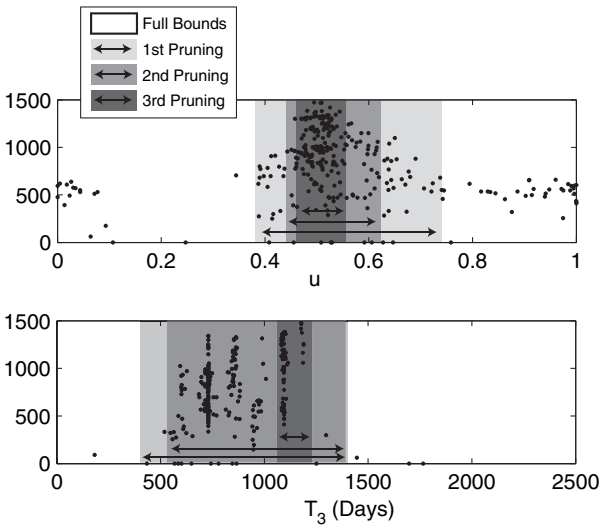
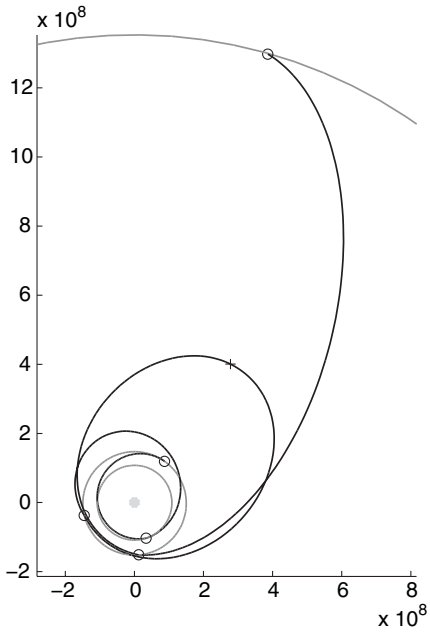


Figure 7.1. Plot of u and T_3 against the final mass for all the 400 computed trajectories. The bounds reduction obtained by pruning is also visualized. Note the Earth-Earth transfer times resonances clearly visible as clusters in the T_3 graph.

best solution reported in the GTOP database [7]. The corresponding trajectory is visualized in Figure 7.2.

At the end of the process, we have not only a putative globally best trajectory, but a large number of other trajectories distributed in those parts of the search space where good solutions are likely to be found and thus a thorough representation of the solution space that is necessary in the preliminary phases of the trajectory design process when requirements change quite often (for example, bounds and constraints) and it is not possible to run a new optimization at each time. It is noteworthy that the global best solution employs only one deep space maneuver, that is, between the two consecutive Earth flybys, but this strategy was not imposed a priori; rather it is a result of the optimization process. In the best solution found, no multiple revolutions are present. While these can in principle increase the final mass, the constraint on the total flight duration in this case drives the optimization process toward trajectories that do not make use of multiple revolutions as not to lose time. Once again, this is not imposed upfront, but it is a result of the automated optimization process. Releasing the constraint on the total flight duration, some tests



Spacecraft	
Daparture mass	2085.44 kg
Arrival mass	1476.03 kg
I_{sp}	312 s
Departure	
Epoch	15/11/2021
V_∞	3.34 km/s
Declination	3.1 deg
Cruise	
Venus fly-by	30/04/2022
Earth fly-by	04/04/2023
DSM Epoch	13/01/2025
DSM ΔV	167 m/s
Earth fly-by	25/06/2026
Arrival	
Epoch	03/07/2031
V_∞	0.676 km/s
Total flight time	9.63 years

Figure 7.2. Putative globally optimal trajectory found in the TandEM problem using cluster pruning.

revealed that the optimal solutions indeed exploit multiple revolutions to increase the final mass.

7.9 Concluding Remarks

Global optimization meta-heuristics are useful in automatically finding and selecting good trajectory options between the often-many possibilities one has in the preliminary phases of mission design. Their use and efficiency are established for chemical propulsion problems of high complexity (that is, large launch windows and multiple flybys) whenever approaches more sophisticated than the straightforward use of standard algorithms are adopted. It seems likely that future research results will aim at proving the use of these techniques for the automated computation of low-thrust trajectories as well. Preliminary results in this sense are already available and pointing to an increased need of computational resources. Under the assumption that the available computing power will keep increasing at the same pace in the next decade, it thus seems possible to argue that a completely automated trajectory design process, at least in the case of patched two-body problems, may sooner or later be able to replace the current design methods relying substantially on expert knowledge, similar to how we no longer have to perform a full function study to obtain its graph.

Appendix 7A

Definition of r_p (v_{in}, v_{out}, t) and F_1 (v_{in}, v_{out}, t)

The functional expression relating the incoming and outgoing velocities during a given planetary flyby at epoch t is described here. In the following, the spacecraft is allowed to have a tangential and impulsive velocity change at the periplanet so that its trajectory will consist of two hyperbolas patched at the pericenter. Introducing the relative velocities $\tilde{\mathbf{v}}_{in} = |\mathbf{v}_{in} - \mathbf{V}(t)|$, $\tilde{\mathbf{v}}_{out} = |\mathbf{v}_{out} - \mathbf{V}(t)|$, simple astrodynamic calculations show that taking as length unit any L and as velocity unit $\sqrt{\mu_{pla}/L}$ (where μ_{pla} is the gravitational parameter of the planet considered), the angle between $\tilde{\mathbf{v}}_{in}$ and $\tilde{\mathbf{v}}_{out}$ is given by

$$\alpha_i = \arcsin \frac{a_{in}}{a_{in} + r_p} + \arcsin \frac{a_{out}}{a_{out} + r_p} \tag{7.16}$$

where $a_{in} = 1/(\tilde{\mathbf{v}}_{in} \cdot \tilde{\mathbf{v}}_{in})$ and $a_{out} = 1/(\tilde{\mathbf{v}}_{out} \cdot \tilde{\mathbf{v}}_{out})$. Inverting this equation for r_p, we define the function $r_p = r_p(\mathbf{v}_{in}, \mathbf{v}_{out}, t)$. The velocity increment necessary at the pericenter, that is, the function F, is given by the simple relation

$$\Delta V = \left| \sqrt{(1/a_{in} + 2/r_p)} - \sqrt{(1/a_{out} + 2/r_p))} \right| = F(\mathbf{v}_{in}, \mathbf{v}_{out}, t).$$

Note that in a MGA-1DSM problem as $\tilde{\mathbf{v}}_{in} = \tilde{\mathbf{v}}_{out}$, there is no velocity increment needed ($F = 0$), and the definition of r_p can be obtained by explicit inversion of the first equation.

Definition of the Arrival Thrust Impulse $N(\mathbf{v}, t)$ as an Orbit Insertion

In case when at the arrival planet the spacecraft is inserted into an elliptic planetocentric orbit having an assigned pericenter r_p^{ins} and eccentricity e^{ins}, the tangential velocity increment needed at the pericenter is determined by the spacecraft arrival velocity $\mathbf{v}$ at epoch t. Taking as length unit r_p^{ins} and as velocity unit $\sqrt{\mu_{pla}/r_p^{ins}}$ (where μ_{pla} is the gravitational parameter of the planet considered), we have, from simple astrodynamics

$$\begin{aligned} v_{p-} &= \sqrt{\tilde{v}^2 + 2} \\ v_{p+} &= \sqrt{1 + e^{ins}} \\ \Delta V &= |v_{p-} - v_{p+}| = N(\mathbf{v}, t). \end{aligned}$$

Definition of the Launcher Performances $m = G(\mathbf{v}, t)$

When we want to model a particular launcher, its performances are often given in terms of a table relating the mass that can be delivered by the launcher to a given value of the hyperbolic escape velocity $\tilde{\mathbf{v}}$ and at a certain declination δ (this last being

referred to the equatorial reference system). These two quantities may be evaluated from the heliocentric departure velocity $\mathbf{v}$ and the epoch t as

$$\begin{aligned}\tilde{\mathbf{v}} &= \mathbf{v} - \mathbf{V}(t)\\ \tilde{\mathbf{v}}_{equ} &= \mathbf{R}\tilde{\mathbf{v}}\\ \sin\delta &= \frac{\tilde{\mathbf{v}}_{equ}\cdot\mathbf{k}}{\tilde{\mathbf{v}}_{equ}\cdot\mathbf{k}}\end{aligned}$$

where $\mathbf{R}$ is the rotation matrix that allows a change of reference system to the equatorial planetocentric. Then m can be found by interpolation.

Definition of $\mathbf{v}_{out} = F_2(\mathbf{v}_{in}, \mathbf{V}, r_p, \beta)$

In a patched conic approximation, the spacecraft heliocentric velocity after a flyby can be related to its incoming heliocentric velocity $\mathbf{v}_{in}$, the planet velocity $\mathbf{V}$, the planetocentric hyperbola periplanet r_p, and its plane orientation β. The following equations, implementing standard astrodynamic calculations, describe such a relation

$$\begin{aligned}\tilde{\mathbf{v}}_{in} &= \mathbf{v}_{in} - \mathbf{V}\\ e &= 1 + r_p/\mu_{pla}\tilde{v}_{in}^2\\ \delta &= 2\arcsin(1/e)\\ \hat{\mathbf{v}}_{out} &= \cos(\delta)\hat{\mathbf{i}} + \cos(\beta)\sin(\delta)\hat{\mathbf{j}} + \sin(\beta)\sin(\delta)\hat{\mathbf{k}}\\ \tilde{\mathbf{v}}_{out} &= \tilde{v}_{in}\hat{\mathbf{v}}_{out}\\ \mathbf{v}_{out} &= \mathbf{V} + \tilde{\mathbf{v}}_{out}\end{aligned}$$

where the unit vectors are $\hat{\mathbf{i}} = \frac{\tilde{\mathbf{v}}_{in}}{\tilde{v}_{in}}$, $\hat{\mathbf{j}} = \frac{\hat{\mathbf{i}}\wedge\mathbf{V}}{|\hat{\mathbf{i}}\wedge\mathbf{V}|}$, and $\hat{\mathbf{k}} = \hat{\mathbf{i}}\wedge\hat{\mathbf{j}}$. Note that in this representation, the incoming and outgoing hyperbolic velocities, $\tilde{v}_{in}$, $\tilde{v}_{out}$, are of equal magnitude (no ΔV is modeled during the hyperbola) and form an angle δ. The second angle, β, determines the position of the outgoing hyperbolic velocity on the cone with axis $\tilde{v}_{in}$ and aperture δ.

Appendix 7B

The particular implementation of the Simulated Annealing algorithm with Adaptive Neighborhood (that we call SA-AN) is given in the following pseudocode describing a single annealing cycle.

1: Select a point $\mathbf{x}_0$
2: for i=1:n_o
3: for j=1:n_t
4: for k=1:n_r

5: for l=1:D
6: alter x_l component adding a random $\delta \in [-r_l, r_l]$
7: accept or refuse according to Metropolis criteria
8: adjust each component of the neighborhood **r** using Corana's method [13] (the acceptance rate is evaluated separately for each component)
9: adjust the temperature using $T = \alpha T$

At the end of the annealing process, the temperature will be $T_f = T_i \alpha^{n_o}$. The total number of function evaluation per annealing cycle has to be set and is $nf_{ann} = n_o n_t n_r D$. We use $n_t = 1$ and $n_r = 20$ so that we adjust the neighborhoods and the temperature simultaneously and we have enough points to evaluate the acceptance rate (here, 20 for each component). The starting neighborhood is set to be equal to $r_l = UB_l - LB_l$. We then perform re-annealing (that is, we restart the algorithm from the point returned by a previous run) up to when we reach the total number of function evaluations $FEVAL$. The algorithm free parameters are T_f, T_i defining the cooling schedule, and nf_{ann} defining the annealing cycle speed.

REFERENCES

[1] Wertz, J., Larson, W., Kirkpatrick, D., and Klungle, D. (1999) *Space Mission Analysis and Design*, Microcosm Press.
[2] Battin, R. (1999) *An Introduction to the Mathematics and Methods of Astrodynamics*, AIAA.
[3] Izzo, D., Becerra, V., Myatt, D., Nasuto, S., and Bishop, J. (2007) Search Space Pruning and Global Optimisation of Multiple Gravity Assist Spacecraft Trajectories, *Journal of Global Optimization*, **38**, No. 2, 283–296.
[4] Vasile, M., and De Pascale, P. (2006) Preliminary Design of Multiple Gravity-Assist Trajectories, *Journal of Spacecraft and Rockets*, **43**, No. 4, 794–805.
[5] Biesbroek, R., and Ancarola, B. (2002) *Optimization of Launcher Performance and Interplanetary Trajectories for Pre-Assessment Studies*. IAF abstracts, 34th COSPAR Scientific Assembly, The Second World Space Congress, held 10-19 October, in Houston, TX, USA., pA-6-07IAF.
[6] Izzo, D. (2006) Advances in Global Optimisation for Space Trajectory Design. *Proceedings of the international symposium on space technology and science*, **25**, p. 563.
[7] Izzo, D., Vinkó, T., and Del Rey Zapatero, M. (2007) GTOP Database: Global Trajectory Optimisation Problems and Solutions. *Web resource*, http://www.esa.int/gsp/ACT/inf/op/globopt.htm
[8] Vinkó, T., Izzo, D., and Bombardelli, C. (2007) *Benchmarking Different Global Optimisation Techniques for Preliminary Space Trajectory Design*. Paper IAC-07-A1.3.01, 58th International Astronautical Congress, Hyderabad, India.
[9] Myatt, D., Becerra, V., Nasuto, S., and Bishop, J. (2004) *Advanced Global Optimisation Tools for Mission Analysis and Design*. Tech. Rep. 03-4101a, European Space Agency, the Advanced Concepts Team, available online at www.esa.int/act
[10] Storn, R., and Price, K. (1997) Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *Journal of Global Optimization*, **11**, No. 4, 341–359.
[11] Kennedy, J., and Eberhart, R. (1995) Particle Swarm optimization. *Proceedings, IEEE International Conference on Neural Networks*, **4**.
[12] Blackwell, T., and Branke, J. (2004) Multi-Swarm Optimization in Dynamic Environments. *Lecture notes in computer science*, **3005**, 489–500.

[13] Corana, A., Marchesi, M., Martini, C., and Ridella, S. (1987) Minimizing Multimodal Functions of Continuous Variables with the Simulated Annealing Algorithm. *ACM Transactions on Mathematical Software (TOMS)*, **13**, No. 3, 262–280.
[14] Holland, J. (1992) Genetic Algorithms Computer Programs That "Evolve" in Ways That Resemble Natural Selection Can Solve Complex Problems Even Their Creators Do Not Fully Understand. *Scientific American*, **267**, 1992, 66–72.
[15] Izzo, D., Rucinski, M., and Ampatzis, C. (2009) *Parallel Global Optimisation Meta-Heuristics using an Asynchronous Island-model*. IEEE Congress on Evolutionary Computation (IEEE CEC 2009), Trondheim, Norway, May 18-21.
[16] Wales, D., and Doye, J. (1997) Global Optimization by Basin-Hopping and the Lowest Energy Structures of Lennard-Jones Clusters Containing up to 110 Atoms. *Journal of Physical Chemistry*, **101**, No. 28, 5111–5116.
[17] Armellin, R., Di Lizia, P., Topputo, F., and Zazzera, F. (2008) *Gravity Assist Space Pruning Based on Differential Algebra*. New Trends in Astrodynamics and Applications V, Milano, June 30th–July 2nd.
[18] Schutze, O., Vasile, M., Junge, O., Dellnitz, M., and Izzo, D. (2009) Designing Optimal Low-Thrust Gravity-Assist Trajectories Using Space Pruning and a Multi-Objective Approach. *Engineering Optimization*, **41**, 155–181.
[19] Vasile, M., Ceriotti, M., Radice, G., Becerra, V., Nasuto, S., and Anderson, J. (2007) *Global Trajectory Optimisation: Can We Prune the Solution Space When Considering Deep Space Manoeuvres?* Tech. Rep. 06-4101c, European Space Agency, the Advanced Concepts Team, Available online at www.esa.int/act
[20] Zazzera, F., Lavagna, M., Armellin, R., Di Lizia, P., Topputo, F., and Bertz, M. (2007) *Global Trajectory Optimisation: Can We Prune the Solution Space When Considering Deep Space Manoeuvres?* Tech. Rep. 06-4101b, European Space Agency, the Advanced Concepts Team, Available online at www.esa.int/act
[21] Olympio, J., and Marmorat, J. (2007) *Global Trajectory Optimisation: Can We Prune the Solution Space When Considering Deep Space Manoeuvres?* Tech. Rep. 06-4101a, European Space Agency, the Advanced Concepts Team, available online at www.esa.int/act