# Evolutionary Constrained Optimization for a Jupiter Capture

Jérémie Labroquère*, Aurélie Héritier[1], Annalisa Riccardi[1], Dario Izzo[1]

Advanced Concepts Team, European Space Agency (ESA-ESTEC), Noordwijk, The Netherlands

**Abstract.** This investigation considers the optimization of multiple gravity assist capture trajectories in the Jupiter system combining the well known Differential Evolution algorithm with different classes of constraint handling techniques. The trajectories are designed to reach a desired target orbit around Jupiter with minimum fuel consumption while satisfying mission design constraints on maximum thrust level, maximum time of flight and minimum closest distance to the planet. The advanced constraints handling techniques are compared for different set of constraints on the challenging mission design problem. For each method the trade off between performance, efficiency and the structure of the feasible space is analyzed in light of the results obtained.

## 1 Introduction

The exploration of planetary moons has become a scientific interest by space agencies such as NASA or ESA. The Jupiter system particularly has been the focus for recent mission concepts such as the JUICE mission [1]. These mission scenarios consist of a tour of Jupiter's moons using multiple flybys. One of their main goal is to assess the habitability of the four Galilean moons. Satellite-aided capture is a well-known trajectory design technique that is employed to decrease the fuel usage to capture a spacecraft into orbit around a planet.

Global optimization techniques have been successfully applied to interplanetary trajectory design [2,3]. They provide automated and unbiased searches for various trajectory options. Within the last decade, several researchers have investigated automated search techniques as a new approach to interplanetary trajectory design. Abdelklalik and Gad investigate genetic algorithms to determine both the optimal flyby sequence and the optimal trajectory [4]. Recently, Englander, Conway, and Williams develop an integer genetic algorithm to determine the optimal flyby sequence and employ differential evolution for the optimal trajectory [5]. However all these approaches are limited to the inclusion of constraints as penalty factors in the definition of the fitness function.

* Private work. Formally working at German Aerospace Center (DLR), Institute of Aerodynamics and Flow Technology, Lilienthalplatz 7, D-38108 Braunschweig, Germany.

This current paper investigates an automated search procedure based on evolutionary techniques to design constrained interplanetary capture trajectories in the Jupiter system. The optimization problem is formulated as a constrained optimization problem. The trajectories are designed to target a final orbit around Jupiter with path constraints on the maximum acceleration and minimum distance to the center of the system. The system is evolved towards the minimization of the cumulative velocity increments. The purpose of this study is to investigate how different constraints handling schemes perform for different subsets of constraints, on the given trajectory optimization problem, and propose alternative techniques to the already widely used static penalty approach.

First, a modified version of the multiple gravity assist model (MGA-1DSM) for interplanetary design is presented for the formulation of the optimization problem [6]. The test case selected for this investigation is a capture trajectory in the Jupiter system using a predefined sequence of flybys at the Jupiter's moons and targeting a final orbit around Jupiter with an eccentricity constraint. The evolutionary optimization technique and the constraints handling methods selected for the study are described in the third section. The advantages and drawbacks of each method are briefly discussed while a quantitative comparative assessment, on the specific test case, is performed in the following section related to the experimental results. The summary of the results obtained and the future research directions are outlined as final conclusions of the paper.

## 2 Jupiter capture trajectory

The capture trajectory model is formulated as an optimization problem using a modified version of the MGA-1DSM model, where one deep-space maneuver is allowed between two successive flybys at the moons [7]. The initial conditions and characteristics of the spacecraft are taken from the problem statement of the Global Trajectory Optimization Competition (GTOC 6) that was organized by the Jet Propulsion Laboratory. Given a sequence of $N$ moons, the constrained optimization problem is defined as

$$\begin{aligned} \underset{\mathrm{x}}{\text{minimize}} \quad & \sum_{i=1}^{N} \mathit{\Delta} V_i \\ \text{subject to} \quad & \mathrm{lb} \leq \mathrm{x} \leq \mathrm{ub} \\ & a = \mathrm{a}_{\text{final}},\ e = \mathrm{e}_{\text{final}},\ i = \mathrm{i}_{\text{final}} \\ & \mathit{\Delta} V_i < T_i\, \mathrm{a}_{\max},\ i = 1, \cdots, N \\ & \sum_{i=1}^{N} T_i < \mathrm{tof}_{\max} \\ & d_i > d_{\min}\ i = 2, \cdots, N \end{aligned}$$

where the variable vector $x$ is bounded between lower bounds and upper bounds, the objective function is composed of the sum of the deep space maneuvers and three types of inequality constraints and three equality constraints are considered. The first inequality constraint is an acceleration constraint that is applied for each $\Delta V_i$ to meet the thrust requirements. The second one is a limit on the maximum time of flight and the third one is a minimum constraint on the closest approach to Jupiter to avoid damaging the spacecraft due to the high level of radiations emitted by the planet. The design of capture trajectories is often dictated by specific mission requirements related to the shape and orientation of the final insertion orbit. Therefore three equality constraints are introduced to define the desired semi-major axis, eccentricity and inclination of the final orbit. Given the set of moons defined as $\Gamma = \{I, E, C, G\}$ where $I =$ Io, $E =$ Europa, $G =$ Ganymede and $C =$ Callisto, and consider a sequence of $N$ moons, denoted $\mathrm{Seq}_N$ in $\Gamma$, i.e, $\mathrm{Seq}_N \in \Gamma^N$. The objective is to find a x vector encoding an interplanetary trajectory that executes in sequence the $N$ flybys at the moons and satisfies the constraints. Given a sequence of $N$ moons, the variable vector has dimension $4N + 2$ and encodes the initial position, the flyby parameters, burn times and duration of each leg:

$$\mathrm{x} = [t_0, u, v, T_0] + \sum_{i=1}^{N-1} [\beta_i, r_{p_i}/r_{\mathrm{Planet}}, \eta_i, T_i] + [\beta_N, r_{p_N}/r_{\mathrm{Planet}}]$$

The two last variables in the x vector denoted $\beta_N$ and $r_{p_N}/r_{\mathrm{Planet}}$ describing the last flyby are added to the traditional formulation of the MGA-1DSM model. These additional variables are necessary for the computation of the equality constraints associated with the shape of the final insertion orbit around Jupiter. The spacecraft is assumed to depart from a position located at $\mathrm{R}_{init} = 1000$ JR from Jupiter center. Details on the problem statement and its mathematical formulation can be found in the problem description of the GTOC 6 competition.[1] The initial position of the spacecraft $\mathrm{r}_0$ is described in spherical coordinates as $\mathrm{r}_0 = \mathrm{R}_{init}(\cos\theta\cos\phi\hat{i} + \sin\theta\cos\phi\hat{j} + \sin\phi\hat{k})$ where the two angles, $\theta$ and $\phi$ are defined as $\theta = 2\pi u$ and $\phi = \arccos(2v - 1) - \pi/2$, respectively. The $u$ and $v$ variables are employed instead of $\theta$ and $\phi$ to get a uniform distribution over the starting sphere of radius equal to 1000 JR. The launch date is represented by $t_0$ using the Modified Julian Date 2000 (MJD2000). The total duration of the first leg is given by $T_0$. After reaching the first moon, the trajectory is propagated in a Keplerian model during $\eta_1 T_1$. A Lambert's solver is then employed to match the spacecraft position to the second moon in the sequence during $(1 - \eta_1 T_1)$. The flyby geometry at each moon is illustrated in Figure 1. The flyby is modeled as an hyperbolic path about the moons where the magnitude of the relative incoming hyperbolic velocity is equal to the magnitude of the relative outgoing hyperbolic velocity, i.e, $v_{\infty-out} = v_{\infty-in}$. The flyby angle $\delta$ describes how the spacecraft approaches the respective moon. More details on the flyby characteristics and the calculations of the b-plane angle $\beta$ can be found in [8,5].

[1] `http://sophia.estec.esa.int/gtoc_portal/`

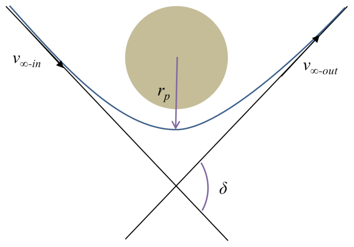

**Fig. 1.** Flyby geometry

## 3 Constrained Evolutionary Optimization

Population based evolutionary techniques are all based on a common structure: a random set of solutions encoded into a chromosome is randomly initialized, then the chromosomes evolve through proper algorithm operators to create a new potential set of solutions. The process iterates from one solution set to another until a stopping condition is met. For each chromosome $x$ a fitness function $F(x)$ is assigned as a measure of quality for the solution.

The self adaptive Differential Evolution (DE) algorithm [9] is an evolutionary technique that has already shown promising results in the design of interplanetary trajectory problems [7]. The parameters of the DE algorithm (variant rand/1/exp) [10,11], which represent the mutation parameter and the crossover constant, are encoded in the chromosome even though they do not evolve with the same operators. The possibility of having a dynamic updating rule for the algorithm parameters is particularly interesting in trajectory design problems, where the solutions in the early design phase are highly diversified.

### 3.1 Constraint handling techniques

DE and its self-adaptive variant jDE have been designed for single-objective unconstrained optimization. To solve for constrained single-objective optimization problems the evolutionary algorithm needs to be coupled with a constraint handling technique. Within the past few years several techniques have been developed to handle constrained optimization problems [12]. In this investigation a few of them have been selected based on their applicability and their different ways of dealing with the constraints.

**Death penalty** The rejection of the infeasible individuals is the most straightforward approach to handle constraints in evolutionary optimization. The fitness update rule of each individual has a penalty factor which become activated for infeasible individuals and assign to their fitness a large constant value. The process can easily get stuck if no feasible individual can be found. The Kuri variant of the method is considered [13], where the fitness function is updated as

$$F(x) = \begin{cases} f(x) & \text{if } x \text{ feasible} \\ K - \sum_{i=1}^{s} K/m & \text{otherwise} \end{cases}$$

where $m$ is the number of constraints, $s$ is the number of non violated constraints and $K$ is a large constant.

**Adaptive penalty** More advanced adaptive penalty approaches have been developed to overcome the limitations of the previously presented techniques. In particular the co-evolution method proposed in [14] is considered for this study. It makes use of two populations $P_1$ and $P_2$. The first one, $P_1$, encodes the penalty coefficients while the second one, $P_2$, encodes the optimization variables. The fitness of each individual belonging to $P_2$ is updated using the following fitness:

$$F(x) = f(x) + y_1 \sum_{i=1}^{m_I} \max[0, g_i(\mathbf{x})] + y_2 \mathrm{Nvio}_{m_I} + y_3 \sum_{i=1}^{m_E} \max[0, h_i(\mathbf{x})] + y_4 \mathrm{Nvio}_{m_E}$$

where $(y_1^j, \ldots, y_4^j)$ is the encoding of the $j$-th individual in $P_1$, $m_I$ and $m_E$ are respectively the number of inequality and equality constraints $g(x)$ and $h(x)$, $\mathrm{Nvio}_{\mathrm{m_I}}$ and $\mathrm{Nvio}_{\mathrm{m_E}}$ are respectively the number of inequality and equality violated constraints. The fitness of each individual in the population $P_1$ depends on the entire population $P_2$ that is associated to it and it is a measure of its infeasibility. The two sets of populations are evolved cooperatively towards feasibility and optimality. The main drawback of such approach is its efficiency. The adopted formulation diverges from the original formulation of Coello Coello where equality and inequality constraints were aggregated in a single term.

**Immune system** The technique emulates the biological behavior of an immune system making use of two populations and two optimization strategies [15]. In biological immune systems the antigenic molecules are recognized and then eliminated by the antibodies. Two populations are employed: one representing the antigenes and one representing the antibodies. The fitness of an individual is determined by its ability to recognize antigenes. Hence the population of antibodies, generated after the immune system simulation, is able to recognize antigenes in the population and to evolve towards immunity and, therefore, feasibility.

**Repair Methods** Repair methods are hybrid techniques that combine heuristic strategies with local searches. The infeasible individuals of the population are repaired to get closer to the feasible region. The repairing method considered in this investigation is a variation of the algorithm proposed in [16]. The infeasible individuals are repaired by means of a gradient descent algorithm that aims at minimizing the sum of the constraint violations. The outer optimization loop minimizes a penalized formulation of the original problem. This has been added to the original algorithm to preserve the repaired solutions reinjected into the population.

## 4 Experiments

The test case selected in this investigation considers 4 flybys in the Jupiter system. The predefined sequence of moons selected is $\mathrm{Seq}_4 = \{C, G, G, G\}$. This capture sequence corresponds to the one employed in previous work [7]. The values for the lower and upper bounds for the chromosome x, are defined in Table 1. The bounds of the inequality constraints introduced in Section 2 are reported in

**Table 1.** Bounds on the variables vector

| Variables | lb | ub |
|---|---|---|
| $t_0$ [MJD 2000] | 7305 | 11323 |
| $u$ [-] | 0 | 1 |
| $v$ [-] | 0 | 1 |
| $T_0$ [days] | 180 | 210 |
| $T_1$ [days] | 0.1 | 10 |
| $T_2$ [days] | 3 | 80 |
| $T_3$ [days] | 3 | 40 |
| $\beta_i$ (i = 1..3) [rad] | -2$\pi$ | 2$\pi$ |
| $r_{p_i}/r_{Planet}$ (i = 1..3) [-] | 50 | 2000 |
| $\eta_i$ (i = 1..3) [-] | 0 | 1 |
| $\beta_4$ [rad | -2$\pi$ | 2$\pi$ |
| $r_{p_4}/r_{Planet}$ [-] | 50 | 2000 |

**Table 2.** Bounds on the constraints for i=0..3 and j=1..3

| Constraints | lb | ub |
|---|---|---|
| $\Delta V_i/T_i$ [m/$s^2$] | $-\infty$ | $5 \cdot 1e - 05$ |
| $\sum_{i=0}^{3} T_i$ [days] | $-\infty$ | 328.725 |
| $d_j$ [JR] | 2 | $\infty$ |
| $e$ [-] | 0.7-0.02 | 0.7+0.02 |

Table 2. The maximum acceleration is set by considering a thrust of 0.1 Newton and a spacecraft of 2000 kg, the maximum time of flight is constrained to 0.9 years and the minimum distance to the center of the system is constrained to 2 Jupiter radius. Concerning the equality constraints related to the shape of the final orbit, an eccentricity constraint is introduced as a proof of concept for this test case where $e_{final} = 0.7$. A high eccentricity is desirable, for example for the exploration of Jupiter and its environment. In particular the region between Callisto and Ganymede is interesting for magnetospheric/plasma physics science [17]. The implementation of the evolutionary and constraints handling techniques presented in Section 3 are made available as part of the open source scientific library PaGMO[2], and its python front-end PyGMO[3]. The framework also provides a generic interface to multiple well known optimization libraries. In particular, the local technique used in the experiments for the repair algorithm is the Nelder and Mead simplex algorithm from the GSL library [4]. The parameters involved in the constraints handling schemes have been tuned benchmarking a variety of constrained optimization problems taken from the 2006 IEEE Congress on Evolutionary Computation (CEC) competition, and have been kept the same in the different scenarios.

## 4.1 Problems definition

Within gravity assist maneuvers, the desirable trajectories are the ones that do not require deep space maneuvers to reach a certain orbit. These specific

[2] `https://github.com/esa/pagmo/wiki`
[3] `http://esa.github.io/pygmo/`
[4] `http://www.gnu.org/software/gsl/`

trajectories are called *ballistic trajectories* and are defined such as $\sum_{i=0}^{N} \Delta V_i = 0$. These solutions, are very challenging to obtain within the feasible region because of the highly multimodal landscape of the fitness space. In the following, multiple problems definitions are considered with an increased complexity in terms of constraints satisfaction, see Table 3:

– **Case 1: Ballistic capture trajectory around Jupiter.** This case finds ballistic solutions under inequality constraints on the minimum distance to Jupiter and the maximum time of flight. The parameters of the final orbit around Jupiter, such as eccentricity, inclination and semi-major axis are free.
– **Case 2: Ballistic capture trajectory around Jupiter targeting a desired final eccentricity.** An extra equality constraint targeting a desired final eccentricity is added to the problem presented in case 1. The inclination and semi-major axis of the final orbit around Jupiter are free.
– **Case 3: Ballistic capture trajectory around Jupiter targeting a desired final eccentricity with constraints on maximum acceleration.** A maximum acceleration constraint on each trajectory leg is added to the problem definition of case 2.

**Table 3.** Details of the 3 test problems. $\rho = |F|/|S|$ is the estimated ratio between the feasible region and the search space, LI and NI are respectively the number of linear and nonlinear inequality constraints, LE and NE are respectively the number of linear and nonlinear equality constraints. $a$ is the number of active constraints at optimality.

| Problem | Constraints | $\rho$ | LI | NI | LE | NE | a |
|---|---|---|---|---|---|---|---|
| **Case 1** | minimum distance to Jupiter, maximum time of flight | 35.2097% | 0 | 4 | 0 | 0 | 0 |
| **Case 2** | minimum distance to Jupiter, maximum time of flight, final eccentricity | 0.0000% | 0 | 5 | 0 | 1 | 1 |
| **Case 3** | minimum distance to Jupiter, maximum time of flight, final eccentricity, maximum acceleration | 0.0000% | 0 | 9 | 0 | 1 | 5 |

## 4.2 Results

All experiments have been sequentially run on a 1.8GHz i5 dual core processor. For each experiment, 250 runs are performed with a population size of 50 individuals. At each run, 400 evolutions for each algorithm are considered. Each algorithm contains 5000 internal generations to reach a maximum number of 1$e$8 cost function evaluations. To keep the experiments computationally tractable within a reasonable time, the CPU time is limited to 2700s (45 minutes). The tolerances are set to 1$e$-8 for both the cost function convergence and the algorithms internal tolerances. A special treatment is introduced for the case 3 by sequentially activating the acceleration constraints linked to each leg. This has been added because the constraint on the trajectory leg between Callisto and Ganymede is hard to be satisfied due to the limited time of flight allowed in the problem definition. Figure 2 illustrates the convergence of the different constraints handling techniques towards ballistic solutions, with respect to the required CPU time. Table 4 reports the probability of finding a ballistic solution. Some observations on the results obtained can be stated:

– **Case 1**: The first test case shows comparable trend for the death penalty and repair techniques in terms of performance (probability of convergence) and efficiency (CPU time). The immune system has the best convergence performance but the worst convergence rate. Finally the co-evolution is the method that requires the highest CPU time to achieve convergence.
– **Case 2**: The behavior of the techniques is similar to the above case, except for the immune system that, even though it is able to reach feasibility, is not able to converge to the global optimum (ballistic solutions).
– **Case 3**: Only the co-evolution method is able to reach feasibility and global optimality. The required CPU time is comparable to the one of case 2.

The plot of a representative solution, for each of the test cases, is reported in Figure 3. For all constraints handling techniques, the solutions converge to an area of the search space where the orbits have very similar shapes. In order to achieve a greater variety in the set of final orbits, within ballistic solutions, also the sequence of moons needs to be enlarged and optimized.

To summarize, all the constraints handling techniques, but the immune system, were able to find feasible and ballistic solutions for the cases 1 and 2.

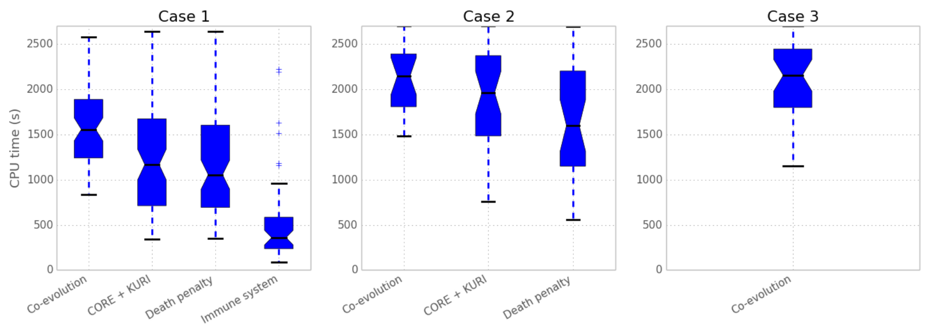

**Fig. 2.** Convergence performance, in terms of CPU time, towards ballistic solutions, of each constraints handling techniques.

**Table 4.** Probability of convergence to a ballistic solution over 250 runs.

| Constraints handling technique | Convergence probability case 1 | Convergence probability case 2 | Convergence probability case 3 |
|---|---|---|---|
| Co-evolution | 0.272 | 0.1 | 0.152 |
| CORE + Kuri | 0.324 | 0.132 | 0 |
| Death penalty | 0.320 | 0.148 | 0 |
| Immune system | 0.192 | 0 | 0 |

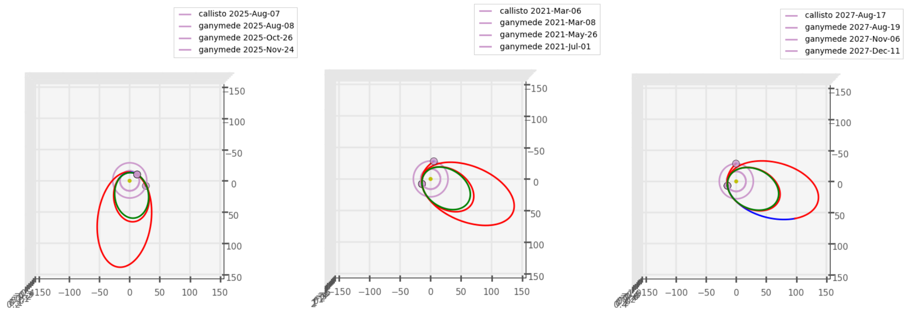

**Fig. 3.** Selected trajectories for the case 1, case 2 and case 3.

For the case 3, only the co-evolution could find solutions. The repair method CORE+Kuri and death penalty constraints handling technique are very similar in term of computational performance and probability of convergence for the first two cases, which highlights, that the repairing process does not have a large effect. Indeed if the constrained problem has a wide feasible region, easily reachable in the early stage of the evolution, the two techniques become comparable. They both fail in finding feasible solutions in the last test case. The adaptability of the penalty approach embedded in the co-evolutionary algorithm is the only strategy, between the selected ones, able to converge to feasibility and global optimality in each of the test case. As expected its main drawback, as illustrated in the first two cases, is its efficiency.

## 5 Conclusion and prospects

In this paper an automatic procedure, using evolutionary constrained optimization techniques, for interplanetary trajectory design has been introduced. The trajectories are designed to reach a desired target orbit around Jupiter with minimum fuel consumption while satisfying mission design constraints on maximum thrust level, maximum time of flight and minimum closest distance to the planet. To optimize these trajectories, four constraints handling techniques have been introduced: the Kuri variant of the death penalty, the CORE+Kuri repairing method, the co-evolution and the immune system. All the techniques but the immune system could find feasible ballistic solutions within a respectable time. The immune system can't reach ballistic solutions as soon as the equality constraint on the final orbit eccentricity is targeted. The co-evolution technique is the only one able to find such solutions when maximum thrust level constraints are added to the problem. As a proof of concept only an eccentricity constraint is considered in this investigation. However as future work, additional equality constraints on semi-major axis and inclination can be included in the problem definition. Moreover another interesting aspect would be to optimize the sequence and number of moons to achieve a ballistic capture at Jupiter with insertion into any desired final orbit shape.

## References

1. The JUICE Science Study Team: Juice exploring the emergence of habitable worlds around gas giants. ESA/SRE **18** (December 2011)
2. Izzo, D., Becerra, V.M., Myatt, D.R., Nasuto, S.J., Bishop, J.M.: Search space pruning and global optimisation of multiple gravity assist spacecraft trajectories. Journal of Global Optimization **38**(2) (2007) 283–296
3. Deb, K., Padhye, N., Neema, G.: Interplanetary trajectory optimization with swing-bys using evolutionary multi-objective optimization. Advances in Computation and Intelligence **4683** (2007) 26–35
4. Abdelkhalik, O., Gad, A.: Dynamic-size multi-population genetic optimization for multi-gravity-assist trajectories. Journal of Guidance, Control, and Dynamics **35**(2) (2012) 520–529
5. Englander, J.A., Conway, B.A., Williams, T.: Automated mission planning via evolutionary algorithms. Journal of Guidance, Control, and Dynamics **35**(6) (November-December 2012) DOI: 10.2514/1.54101.
6. Izzo, D.: Global optimization and space pruning for spacecraft trajectory design, spacecraft trajectory optimization. Cambridge University Press (2010) 178–199
7. Izzo, D., Simões, L.F., Märtens, M., de Croon, G.C., Héritier, A., Yam, C.H.: Search for a grand tour of the jupiter galilean moons. In: GECCO 2013: Proceeding of the fifteenth annual conference on Genetic and evolutionary computation conference. (2013) 1301–1308
8. Vinkó, T., Izzo, D.: Global optimisation heuristics and test problems for preliminary spacecraft trajectory design. ESA TR GOHTPPSTD (2008)
9. Brest, J., Greiner, S., Boskovic, B., Mernik, M., Zumer, V.: Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. Evolutionary Computation, IEEE Transactions on **10**(6) (2006) 646–657
10. Storn, R., Price, K.: Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization **11**(4) (1997) 341–359
11. Storn, R., Price, K.: Differential Evolution- A Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces. Technical report (1995)
12. Coello Coello, C.A.: Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. Computer Methods in Applied Mechanics and Engineering **191**(11-12) (2002) 1245—-1287
13. Morales, A.K., Quezada, C.V.: A universal eclectic genetic algorithm for constrained optimization. In: 6th, Intelligent techniques and soft computing European congress. (1998) 518–524
14. Coello Coello, C.A.: Use of a self-adaptive penalty approach for engineering optimization problems. Computers in Industry **41**(2) (2000) 113–127
15. Hajela, P., Lee, J.: Constrained genetic search via schema adaptation: an immune network solution. Structural Optimization **12**(1) (1996) 11–15
16. Belur, S.V.: CORE: Constrained optimization by random evolution. In Koza, J.R., ed.: Late Breaking Papers at the 1997 Genetic Programming Conference, Stanford University, CA, USA, Stanford Bookstore (13–16 July 1997) 280–286
17. Campagnola, S., Kawakatsu, Y.: Jupiter magnetospheric orbiter trajectory design: Reaching high inclination in the jovian system. 22nd International Symposium on Space Flight Dynamics (February-March 2011)