Global Optimization with Hill Climbing in Earth Observation Mission Planning

Christian Wozar and Robin Steel

Telespazio VEGA Deutschland GmbH
Europaplatz 5, D-64293 Darmstadt, Germany
christian.wozar@telespazio-vega.de, robin.steel@telespazio-vega.de

Abstract

We describe an algorithm for the global optimization of image take and downlink planning of a single satellite Earth observation mission. The algorithm is based on hill climbing methods with a suitably chosen search space. It is shown how subproblems of the planning can be solved exactly, and how this enables the integration of a hierarchical constraint solving and optimization process into the hill climbing. For a test data set of 500 Earth observation requests we compare benchmark results against other methods and exact optimization results, at least for a subset of constraints. The shown algorithm is found to be superior to simple greedy searches and as powerful as simulated annealing. In special cases it can reach a solution quality close to exact solutions.

Introduction

The PRISMA mission ("Precursore IperSpettrale della Missione Operativa") of the Italian Space Agency (ASI) is devoted to the in orbit demonstration and qualification of an Italian Earth observation mission using state-of-the-art hyperspectral/panchromatic technologies. It is primarily intended for the public good, to enable scientific and institutional end users to acquire imaging data for knowledge and management of environment and natural resources. In the scope of this mission, VEGA is developing a mission planning system (MPS) for Telespazio SpA (responsible for the overall ground segment) to optimize the use of satellite resources and to enable the global optimization of the image acquisitions under a set of complex (temporally local and non-local) constraints. The planning is request and priority driven, with priorities assigned by the users to every single image take, so that the algorithmic aim is the maximization of a priority dependent target function for the oversubscribed planning problem of image request planning.

In the following we report on the performance of the implemented planning algorithm and we make (in a simplified constraint context) comparisons to exactly know solutions of the planning problem.

Planning Scope

PRISMA is a single satellite LEO mission on a Sun synchronous orbit. The driving factor for the design of the algorithm is given by the agility of the satellite that allows to take images in a roll angle range of [−18°, +18°] w.r.t. nadir. No flexibility in pitch is given, so that the possible image take timings are completely fixed by the orbit geometry and the target area.

The planning is performed using the "VEGA Planning Toolkit – Earth Observation" (VPT-E), which is a general purpose extensible planning system framework derived from the *Enhanced Kernel Library for Operational Planning and Scheduling* (Noll and Steel 2005). PRISMA specific extensions of VPT-E are responsible for the file based I/O and the concrete realization of the planning algorithm.

Objects in the Planning Process

The main object to be planned is the *User Request* (UR), which is a request for an image that has to be fulfilled in a specific time horizon (validity time, up to four weeks). URs for PRISMA can appear in two different types:

- **Standard:** A single "spot" image, specified by a target point. An area of 30×30 km is recorded (at nadir).
- **Stripmap:** An image of a strip, specified by a target point and the duration (expressed as L km, $L \in \{30, 60, \ldots, 1800\}$).

For the description in this article we only consider standard requests, although the method is directly generalizable to stripmap images.

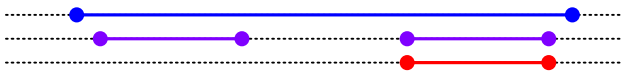

Figure 1: DTOs (purple) are contained within the UR validity time range (blue). For the latest DTO one DTA (red) is planned.

To fulfill an UR, exactly one image must be taken in a *Data Take Activity* (DTA). Associated to each single UR there are possible *Data Take Opportunities* (DTO), i.e. time ranges during which a DTA can take place, see Figure 1. So for every UR there will be, in general, several possible DTOs, which is a fact that makes the global optimiza-

tion problem exceptionally complex – an exact solution would need to consider any combination of assignments from the URs to their DTOs.

The timings of the DTOs are directly derived from the predicted orbit. An additional parameter is the roll angle that must be kept during the associated DTA, which itself lasts for the full DTO duration. In addition to the DTAs there must be a download of the recorded data that happens during a *Downlink Activity* (DLA). The timing for a DLA is constrained by the availability of the receiving station.

The MPS is in charge to make the "best" selection among the URs and DTOs, to assign DTAs to the URs, whose timing and roll angle is given by the DTO parameters. Further, DLAs must be assigned, such that the recorded data is eventually downlinked and no memory overflow situation is planned. Under nominal operation, the planning problem will be oversubscribed and only a subset of the URs can be fulfilled.

For this article we do not consider any impact of satellite unavailability periods on the planning. Although those have to be taken into account for real satellite operations, e.g. by using the *Language for Mission Planning* (Sousa and Noréus 2010) to modify and filter activities, no impact on algorithm quality and runtime performance is expected.

Constraint Setup

The operational requirements of the PRISMA mission lead to several constraints that need to be fulfilled within any planned time range. With the whole set C-1 through C-5, the constrained optimization problem becomes computationally hard and does not allow for a direct construction of the exact solution. We are therefore bound to use approximate methods to come close to the optimum of the target functions under the following constrains.

Maneuver Duration (C-1). Between any two planned DTAs it is necessary to perform a maneuver to change the roll angle. The corresponding duration depends on the roll angle difference (between 0° and 36°) and varies between 70 s and 190 s.

Earth Pointing Duration per Orbit (C-2). For any orbit (duration of approx. 97 min) it is only a maximum duration of 15 min allowed between the start of the first DTA/DLA and the end of the last DTA/DLA. This amounts to a maximum of about 13 images per orbit, assuming a minimal maneuver duration of 70 s between two image takes.

Payload Duty Cycle (C-3). For thermal reasons, only four orbits in a row are allowed for image acquisitions (DTAs). Whenever such a sequence ends (which may also happen after less than four orbits), at least four orbits without image acquisitions are required.

Memory Fill Level (C-4). The on-board memory has a capacity 256 Gbit, which allows to store roughly 100 spot images. Memory can be reused for new acquisitions directly after downlink of the old data. The downlink duration depends on the image size and is about a factor four longer than the image acquisition duration due to the used data rates.

Downlink Budget (C-5). The PRISMA satellite has the capability to downlink old or new data *during* the acquisition. The link budget, however, depends on the roll angle taken during the acquisitions, so that only part of a possible downlink contact may be used for downloading data. The longest contact duration that is available for download during a satellite visibility of the downlink stations forms the *Downlink Opportunity* (DLO), which depends on the planned DTAs. DLAs may only happen during a DLO.

If no DTA is on-going during a station visibility, for instance during nighttime contacts, the satellite can use the full visibility duration as DLO.

Target Function

The planning process aims at optimizing multiple objectives. These are

1. The maximization of the number of planned requests, taking into account the request priorities, and
2. The minimization of the "time to downlink", such that the data is downlinked as early as possible. Again the request priorities should be taken into account.

The corresponding target functions (which are to be maximized by the MPS) are given by

(1) $F_1 = q$ (# planned urgent requests) $+ \Sigma_i (1/P_i)$,

(2) $F_2 = \Sigma_i W_i (T_0 - t_d(i))$,

where the following definitions apply:

- i runs through all successfully planned URs (i.e. with DTAs assigned) in the sums.
- P_i is the priority level assigned to request i, where lower level indicates a higher priority. Multiple URs may have the same priority. P = 1 is reserved for *urgent requests*.
- q is an arbitrarily high number, which ensures that no urgent request is dropped in favor of a lower priority one. In practice we use q = 10,000.
- W_i is a weighting factor, dependent on the priority level and with $W_i > W_k$ for $P_i < P_k$.
- $t_d(i)$ is the start time of the downlink of request i. If only a data take has been assigned but no downlink, the end time of the current planning horizon is used.
- T_0 is the start time of the current planning horizon.

The second objective F_2, however, is only of minor importance, so that the function to be considered for quality measurements will be F_1 or, to be more specific, the three functions

(3) $F_1' =$ # planned urgent requests,

(4) $F_1'' = \Sigma_i (1/P_i)$, and

(5) R = # planned requests in total,

with the sum in F_1'' running only over non-urgent planned requests.

During the planning algorithm the target functions are therefore not considered simultaneously and *no real* multi objective optimization is performed. Rather the MPS first maximizes F_1 and afterwards (on the so obtained set of to

be planned requests) maximizes F_2. Note that two values of F_2 can only be reasonably compared if the same set of successfully planned URs (with assigned DTAs) is involved.

Planning Strategy

The Global Planning Loop – Hill Climbing

We base our planning algorithm on the iterative procedure of (perturbed) hill climbing (Russell and Norvig 2009) to arrive at an optimal solution of the constrained optimization problem to maximize the target functions under the constraints that are introduced above.

Because of the fact that a rather complex subproblem of the optimization can be solved exactly, namely the allocation of DTAs within an orbit and the payload duty cycle, the search space (a.k.a. "configuration space") for the hill climbing will be given by the possible assignments of URs to their DTOs, similar to the selection of variables in image planning for SPOT 5 (Bensana et al. 1999). Therefore, we obtain a hierarchical algorithm in which the hill climbing is used to iterate on the inputs to the (nearly) exactly solvable inner optimization loop.

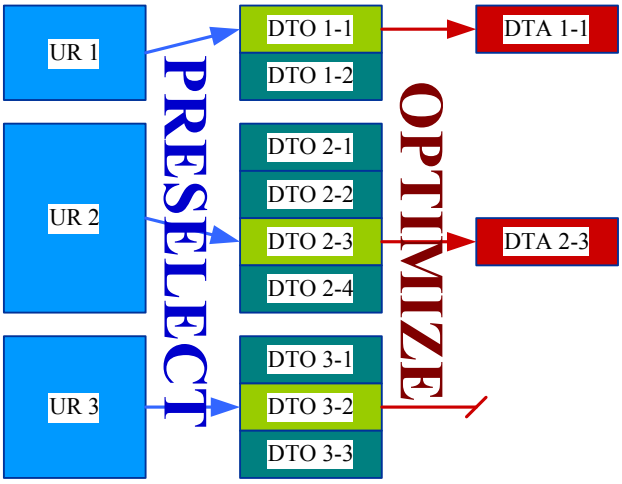

Figure 2: Global planning concept with preselection (hill climbing) and optimization (for every hill climbing step).

This is shown as *preselection* in Figure 2. I.e. for every UR we choose one DTO without taking into account any constraints. It is then subject to the constraint solving and optimization part (see below) to optimize among these preselected DTOs to fulfill all constraints with maximizing the target function F_1. This part will plan the DTAs that are chosen from the preselected DTOs together with their DLAs. Thus, hill climbing considers all DTOs on the plan, but the constraint checking (performed as part of every hill climbing step) works only on a subset of the DTOs, given by the preselection. No additional ranking (apart from the URs' priorities) is introduced by the hill climbing, which is different to methods with a priority driven iterative planning, such as Squeaky Wheel Optimization (Joslin and Clements 1999).

Of course, this procedure only arrives at an optimal solution for the preselected DTOs. To arrive at the global optimum (or at least close to it), this procedure is combined with the hill climbing among the different possible preselections.

The hill climbing procedure involves the following steps:

1. Iterate over all URs. For every UR:
   a) Compute the optimum of F_1 for a preselection of this UR to its different DTOs. This is done with keeping the preselection for DTOs for all the other URs fixed.
   b) Select the optimal DTO of this UR in the sense of maximizing F_1. (See "local optimization" below.)
   c) If there is no unique optimal DTO, then select (with a sufficient amount of randomness) one of the optimal DTOs.
2. If a new optimal F_1 is reached, introduce a small perturbation into the system.

This iteration will loop over the URs until a fixed (and configurable) number of hill climbing trial steps is reached.

While most hill climbing techniques operate on a continuous search space where they may be able to use gradient driven *steepest ascent* determination to find the next solution and allow to express locality by means of geometric distances, our search space is discrete, given by the assignments of URs to their DTOs. Locality is therefore *defined* for the given problem by a configuration that differs only on the assignment of at most one UR to its DTO.

Local Optimization on One UR

To perform the iteration on the configuration space we need to step towards the next local optimum, with locality defined as above.

As part of one hill climbing trial step, exactly one UR is chosen on which the DTOs are varied. For every resulting combination of DTOs (i.e. exactly one DTO per UR) the optimization procedure is performed, taking into account the constraints and optimizations that are described below. Under this optimization not every DTO will evolve into a DTA (or DLA). The main result of one hill climbing step is therefore not the combination of DTAs but only the combination of DTOs (which implies by means of constrained optimization a set of DTAs, which can be recorded as well). The value F_1 of a combination of DTOs is then defined as the implied value F_1 of the resulting DTAs.

Note that in our implementation one fixed UR is chosen per hill climbing iteration and only the DTOs of that particular UR are varied and the best solution is chosen. Then, the iteration proceeds with the next UR according to a well-defined ordering, e.g. by an UR identifier. According to the locality definition, however, any trial configuration that differs on one arbitrary UR from the current configuration would be acceptable. A true steepest ascent algorithm would have to test *all* neighboring configurations before choosing the new optimum. Therefore, our procedure is a *coordinate ascent* hill climbing version, which, in the giv-

en use case, ensures a fast convergence without the need to probe a whole lot of configurations per iteration.

Perturbed Hill Climbing

Standard hill climbing procedures are constrained to only increase the target function. In the case that many local maxima in the search space exist, the algorithm is likely to get stuck in such a local maximum (see Figure 3). In order to step out of such a local optimum, and having a chance to reach an even better local maximum, we introduce a perturbation.

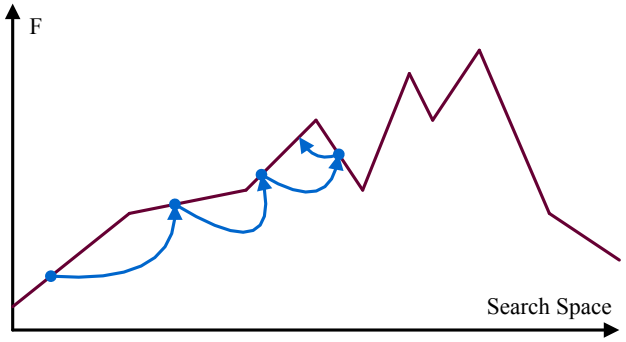

Figure 3: Hill climbing gets stuck in a local maximum.

While standard hill climbing will accept a new solution only if the target new function F_{new} is greater or equal than the last solution F_{last},

(6) $F_{new} \geq F_{last}$,

we now allow to accept a new solution if

(7) $F_{new} \geq F_{last} - G(t)$.

For this, we compute F_1 for all DTOs of the current UR and select randomly a DTO that fulfills Eq. (7). t is the number of hill climbing trial steps since the last best maximum has been found, while G(t) is the slight perturbation, which is modeled as linear function, characterized by the start value G_0 and the number of steps t_0 at which $G(t) = G_1$ (see Figure 4).

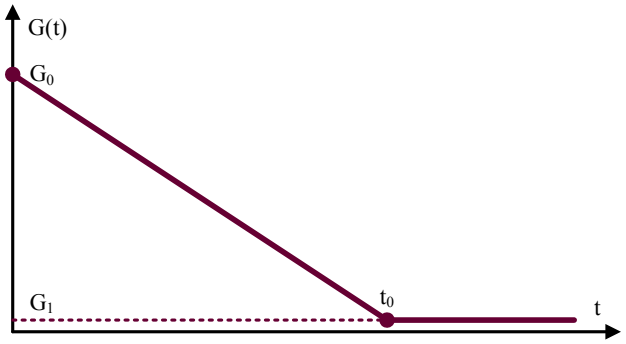

Figure 4: Perturbation function used to improve the convergence towards the global optimum.

This procedure will lead to trajectories that can leave a local optimum in favor of a better local (or even a global) maximum of the target function, see Figure 5. During the whole iteration we keep track of the maximum F_{opt} that has been observed, which is the output of the hill climbing algorithm.

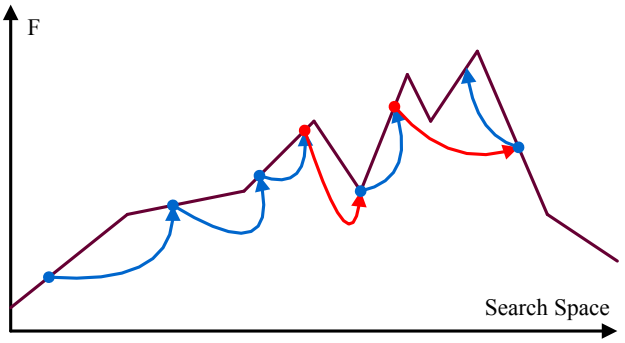

Figure 5: Hill climbing with perturbations. Steps shown in red allow for moves to better local maxima.

Optimization and Constraint Solving

In the course of finding the optimal choice of planned DTAs (and DLAs) for the given preselection of DTOs it is necessary to arrive at an optimal and consistent solution. This is done by processing the given DTOs in a specific order:

1. Temporal constraints are solved, such that a consistent and optimal solution is found, which takes into account the maneuver duration (C-1) and the maximum daylight Earth pointing durations (C-2).
2. The resulting orbits are optimized under the constraint of the payload duty cycle (C-3).
3. This result is processed, such that downlinks are planned (taking into account the attitude profile via C-5) and the memory consumption constraints are fulfilled (C-4).

Altogether this procedure keeps in every step the DTAs (and DLAs) on the plan that can still be planned out of the DTAs from the previous step (see Figure 6). Steps 1 and 2 optimize F_1 while step 3 optimizes F_2. Only if DTAs are removed in step 3, the function F_1 is considered again. Therefore, a *hierarchization* of the constraints has been made according to their interdependencies and expected impacts on the planning result.

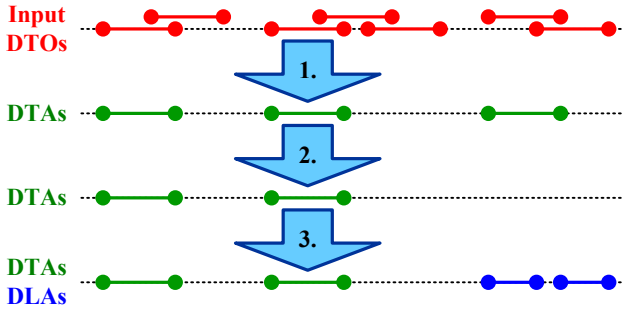

Figure 6: Constraint solving and optimization steps within one hill climbing trial.

Timing Constraints within One Orbit

The input for this timing constraint is, that for any of the URs one DTO is selected during which the DTA may be implemented. We are therefore left with a set of (in the global hill climbing part preselected) DTOs, each of which will provide the time range to implement the corresponding DTA. In this respect, implementing the corresponding DTA means that the DTA is planned with the time range and roll angle given by the corresponding DTO.

Due to constraint C-1 we need to fulfill

(8) $s_{n+1} - t_n \geq A(n+1,n)$.

A(n+1,n) computes the minimal attitude change duration from the attitude information that is stored along with the DTAs n+1 and n. s_n denotes the start time of the n^{th} DTA on the plan and t_n denotes the end time of the n^{th} DTA. DTAs are assumed to be ordered according to the start time.

In addition we need to fulfill C-2 by the following conditions:

- The maximum time from start of first acquisition to end of last acquisition, the "Acquisition Interval" (AI), within one orbit is set via a parameter T_T (nominally 15 min).
- There can be maximally one such session per orbit.

The optimization problem consists in choosing the AI and the contained DTAs in a way that F_1 is maximized. This proceeds in the following steps (assume that the DTOs are ordered according to their start time):

1. Construct the distinct maximum sets S_n of DTOs that are consistent with T_T.
2. Optimize within the every such set the choice of DTAs according to F_1.
3. Select the optimal set S^* according to F_1.

An example to illustrate the first step is given in Figure 7. There are three maximum sets and any further possible set consistent with T_T will be a subset of an already existing one.

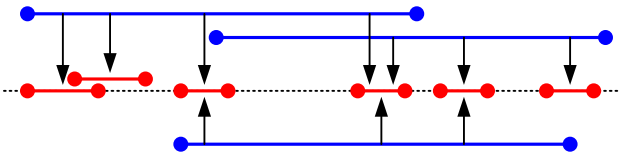

Figure 7: On the timeline, three maximum sets of DTOs (red) determined for T_T (blue) are indicated by arrows.

For every such set S_n we then solve the optimization problem of selecting the DTOs within that set that are consistent with the attitude maneuver timing (given by the roll angle change) and which maximize F_1. This can directly be done via dynamic programming (DP), see (Bellman 1954). An explicit application of DP is given below for the duty cycle constraint. Linear programming is not applicable for solving this type of constraints, because an explicit selection on the DTOs must be made.

The result of the first optimization step is an *exact solution* for the constraints C-1 and C-2 under the preselection of one DTO per UR.

Payload Duty Cycle Constraint

After the first step of the temporal acquisition planning we are left with a set of orbits and their optimal values for F_1. We then have to select among these orbits the ones that optimize the global F_1 under the payload duty cycle constraint C-3, which covers the following:

- There can be up to $N_E = 4$ consecutive daylight Earth pointing sessions.
- After such a sequence there shall be at least $N_S = 4$ consecutive orbits without DTAs.

An example of the situation before optimizing and solving for the payload duty cycle is shown in Figure 8.

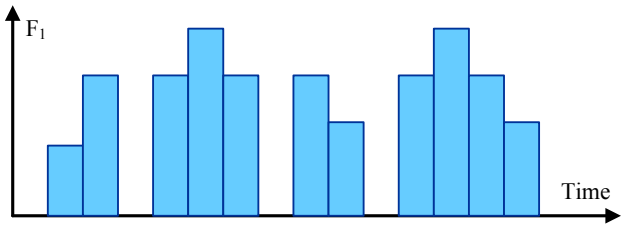

Figure 8: Orbits with associated F_1 after solving for C-1 and C-2.

The optimization of F_1 for the duty cycle constraint is again based on dynamic programming. Starting from the first orbit, we iterate over all orbits on the plan which are available (i.e. have F_1 different from zero). For every orbit we iteratively construct the optimal choice of orbits under the C-3, including orbits only up to this one. This optimization involves the following steps for a given orbit O_E:

- For every orbit O_S prior to O_E, for which O_S to O_E are building a complete sequence of up to N_E orbits with F_1 different from 0:
  - Merge the sequence [O_S, O_E] with the optimal choice of sequences for the orbits before O_S, taking into account N_S orbits without DTAs.
  - Compute the value F_1 of the merged sequence.
- Choose the optimal O_S, such that F_1 is optimized for the given sequences.
- Compare this F_1 with the already computed optimal F_1's for the previous O_E's.
- Record the optimal sequence, taking into account the previous optima, for the end orbit O_E. In case there are two possibilities with same value F_1, choose the one that has already been computed for a previous O_E.
- Go to the next O_E.

The result of this procedure for the given example is shown in Figure 9. Only DTAs within green orbits are taken over to the memory and downlink planning.

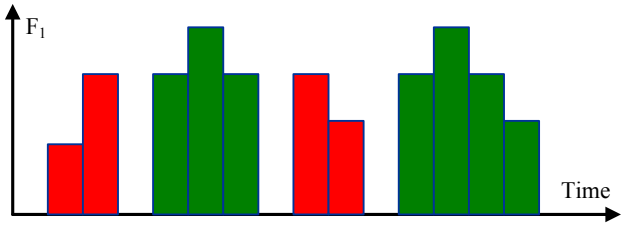

Figure 9: Optimally assigned orbit sequence for the whole planning horizon.

Memory Constraints

As a last optimization step we consider the memory constraints C-4 and C-5. For that reason the memory consumption is modeled by increasing the used memory level whenever an image take happens during a DTA. In addition the download of images will lower the amount of used memory.

The main input to the downlink planning is the roll angle profile (and orbit use profile) along the plan. Depending on the planned acquisitions there are two possible cases:

- If an orbit is not used for acquisitions it can safely be used for nighttime downlinks. Thus a corresponding downlink window will be fully available.
- If an orbit contains already some DTAs, then we have to calculate the maximum available uninterrupted downlink time by using the roll angle profile in combination with given contact time offsets for the different roll angles, as provided in a flight dynamic sequence of events file.

After the useable downlink times have been generated, we assign for every DTA a DLA. To account for the different priorities expressed in Eq. (2), we assign the DTAs in order of priority and start time to the first possible time slot.

With this assignment of DTAs and DLAs we compute the resulting memory profile and check it against the mass memory size.

If there are no constraint violations, then the solution of the downlink planning is found. Otherwise, we need to remove some DTAs from the plan to allow for a valid memory profile. This removal can be directly constructed in a way that is optimal in the sense of F_1.

Calculation of the Maximum Downlink Window. The input for the daytime downlink window consists of:

- Planned DTAs and associated roll angles,
- Contact durations and offsets for different roll angles,
- Maximum AI duration T_T.

Based on this information, we compute the maximum duration of a downlink contact by superimposing the roll angle profile onto the contact durations per roll angle. The longest uninterrupted sequence is taken.

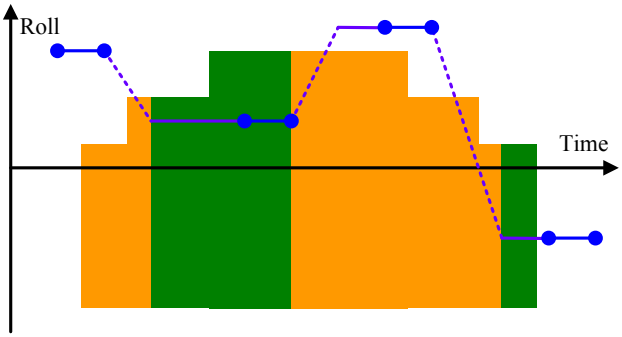

Figure 10: Possible downlink windows (green) within one contact for a given set of DTAs (blue). Only the filled areas are available for downlink.

This is shown as example in Figure 10. There, the roll angle profile (purple) and the DTAs (blue) are used to determine the possible downlink durations (green). Note that the time during the attitude maneuvers is not used here, because the maneuvers are not entirely within the roll angle limits and we cannot assume a linear roll angle vs. time relationship during the maneuvers. Because two windows are possible we choose the first one as DLO for this contact, i.e. the one that has the longer duration. Note that this choice is made, because planning for PRISMA requires to have only *one uninterrupted* link per contact.

In case the contact duration extends beyond the time range used for the DTAs, the maximum AI duration is used, such that the planned Earth pointing activities (acquisitions and downlinks) are contained within that interval. Before the first acquisition (or after the last, if applicable) the satellite is commanded to nadir pointing attitude, to maximize the downlink time.

Computation of Downlink Activities. The assignment of stored images (from DTAs) to DLAs proceeds in order of priority level. In case of same level the first acquired image is downlinked first. For every image sequence to be downlinked we assign the first possible time slot within the previously determined downlink windows, see Figure 11. This ensures an approximate (greedy) optimization of F_2 for the given set of DTAs.

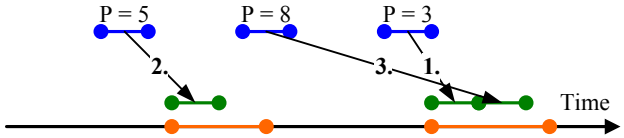

Figure 11: Order of assignment of DLAs (green) during the DLOs (orange) for a given set of DTAs (blue) with associated priority levels P.

Memory Profile and Conflict Resolution. Derived from the assigned DTAs and DLAs the memory profile is then directly constructed by increasing the memory fill level *at the beginning* of a DTA and decreasing the fill level *after* downlink.

In case the memory capacity is exceeded, this conflict must be solved. In a first step it is determined which DTAs contribute to the conflict. These are simply the ones that

occur in an interval between the time where the memory is full and the last empty state.

After the possible conflict causing DTAs are determined, the DTA with the highest priority level is removed from the plan to keep the maximum possible F_1. In case that there are multiple such DTAs, the one with the largest memory contribution is chosen. After that, the memory profile is recalculated, going again through the downlink planning steps above to account for any changes in the roll angle profile.

Special care has to be taken if the last remaining DTA within one orbit is removed. In that case, a full impact assessment on the orbit selection must be made, because an orbit sequence of C-3 might become broken.

Benchmark Results

The benchmarks are computed with an input set of 500 URs for standard spot images (30×30 km area) that are distributed all over the world (see Figure 12). These URs are supported by 2103 DTOs in a time range of 28 days.

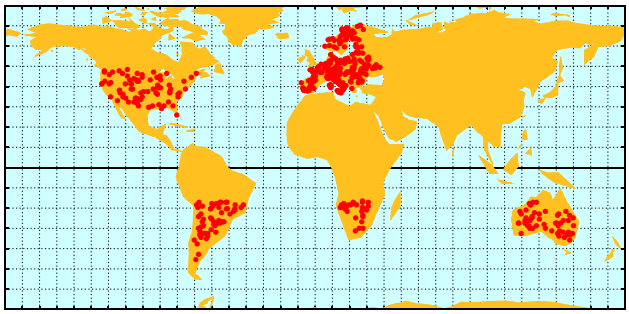

Figure 12: Target points of the test data set.

Table 1: Priority level distribution.

Priority	1	10	11	12	13	14	15	16
Count	4	141	126	51	41	46	53	38

The distribution of URs to the priority levels is listed in Table 1 while the histogram of DTOs per UR can be seen in Figure 13.

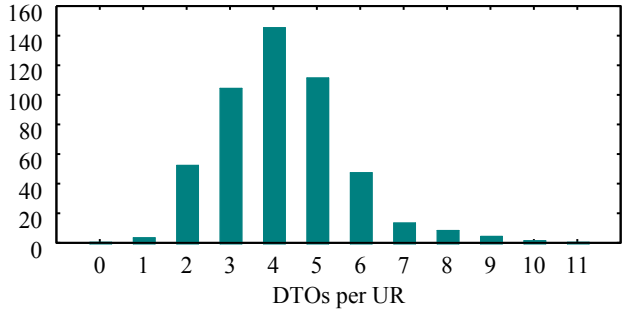

Figure 13: Histogram of number of DTOs per UR.

For the downlink planning we consider only one downlink station at Matera, so that either two or three contacts are available every single day and every single night.

Algorithm Shootout

We compare the perturbed hill climbing (PHC) on the full problem scope (including C-1 through C-5) with three algorithmic variants:

- **Standard hill climbing (SHC):** Here, the perturbation function G(t) is set to zero for all t.
- **Simulated annealing (SA):** Our version of SA draws configurations by randomly altering the DTO preselection of single URs. The new configuration is accepted with probability $Q = \exp[(F_{1,new} - F_{1,old})/Z(t)]$. Z is constructed as $Z(t) = Z_0 \times (1 - t/K)^2$, where K is the total number of steps and t is the current step number. The result of the SA is the best value of F_1 that has been found during the iteration.
- **Simple greedy search (SG):** We order the URs according to their priority. Then, starting with the most important UR, we try to add a DTA of the UR to the plan by testing all its DTOs. The earliest possible DTO is chosen and will be fixed for the UR during the processing of all subsequent URs. The result after one loop over the URs determines the value of F_1.

The parameters of the PHC are taken to be K = 20,000 iterations, $G_0 = 0.2$, $G_1 = 0.0001$, $t_0 = 20$. Further, regular perturbations are included with a height of $G_0' = G_0 \times (1 - t/K)$ every 809 iterations (which corresponds to the number of URs times the golden ratio).

For the SHC we only set the perturbations of the PHC to zero and perform the same number of iterations.

The SA is performed with K = 100,000 iterations. A manual optimization among different Z_0 reveals a best normalization factor of $Z_0 = 0.1$.

The SG has no tunable parameters, so that the results are computed in a straightforward way.

In addition to the quality measures, we also keep track of the wall time W that is required for planning. All measurements have been taken on a Intel Xeon X5560 CPU (2.8 GHz). Programs are compiled with gcc 4.3.4 and flags '–O3 –march=core2 –mfpmath=sse –msse4.2'. Note that the optimization flags lead to a speedup factor between 2.5 and 3 compared to '–O0 –g'.

Table 2: Planning quality comparison for different algorithms.

Method	PHC	SHC	SA	SG
F_1'	4	4	4	4
F_1''	21.364	20.702	21.596	16.066
R	244	236	248	181
W [sec]	49.8	51.6	59.6	1.1

As shown in Table 2 the PHC algorithm is superior to the SHC alternative while requiring about the same amount of computation time.

SA slightly outperforms the PHC in terms of solution quality. The SA results, however, depend very much on the chosen factor Z_0. With $Z_0 = 0.2$, for instance, only $F_1'' = 20.990$ and R = 241 were found. Such a strong dependency on algorithm parameters has not been observed by us in the PHC.

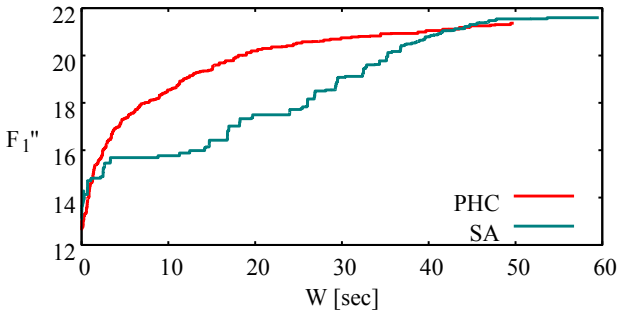

Figure 14: Wall time dependence of intermediate results for PHC and SA.

We further compare the intermediate results during the optimization for PHC and SA, see Figure 14. The PHC has a much smoother convergence towards its optimum than SA and show a steeper increase of the target function at early planning times. Therefore the PHC should be preferred whenever a limitation of available planning time may lead to a premature stop of optimization, such as in urgent replanning situations close to an uplink contact. This observation is similar to the comparison of (deterministic) Squeaky Wheel Optimization with (stochastic) genetic algorithms (Barbulescu et al. 2004), where the stochastic method becomes superior to the deterministic one for a large number of iterations.

The SG is inferior to all the other algorithmic options. We conclude that a certain amount of (pseudo) randomness, as given in the PHC and SA, is beneficial to the algorithm quality.

Planning Horizon Dependency

The full problem description is given on a time scale of four weeks. We analyze the effects of planning this time scale in subsequent blocks of size B = 14, 7, 4, and 2 days. I.e. we start with the full set of URs on the first block and only the so far unplanned URs are considered in the second block, etc. Boundary conditions are taken from the previously planned block. The algorithm to be used is PHC as described above. Note that the operational concept for the PRISMA mission nominally foresees a DTA planning horizon of only one day.

Table 3: Planning quality with different block sizes.

B	28	14	7	4	2
F_1'	4	4	4	4	4
F_1''	21.364	20.395	19.789	19.311	19.178
R	244	237	229	223	221

The results of Table 3 show that for a fixed problem setting it is highly beneficial to plan the complete time range as a whole, given that enough computation power is available. The improved planning quality for larger blocks indicates that *global* optimization is effective, i.e. the algorithm successfully considers globally optimal solutions.

Memory Size Dependency

In all benchmark results above there are effectively no constraints implied by the memory capacity. Therefore we measure the effects on the planned DTAs per priority level if the memory capacity is lowered to less than ten spot images in the 28 day scenario. Firstly, we have considered the full set of constraints. In that case the PHC algorithm allows to reach a high level of planned requests (>90% of maximum) down to a capacity of three images, see Figure 15. The value of F_1'' drops at exactly the same rate, indicating that the planned URs of the individual priority levels are reduced by the very same factor when the memory capacity is lowered. The results further show that a high level of service can be maintained, even if part of the memory becomes unavailable (as it might happen on a long duration mission).

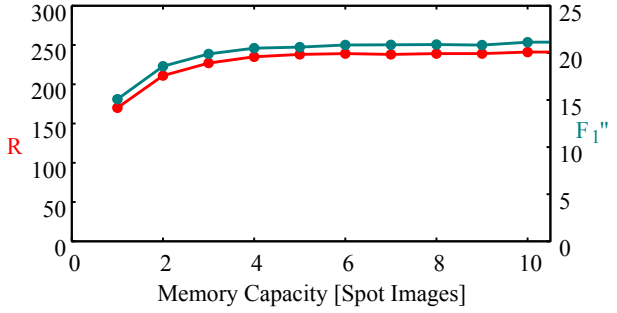

Figure 15: Planned URs and value of F_1'' for varying memory capacity.

In a second run we have removed the constraint C-3. In that case we find that the planned requests of the individual priority levels drop at different rates, see the values of Q[P] in Figure 16, which denote the ratio of the contribution to F_1'' from the priority P at specified memory capacity to the value at maximum memory size. We conclude that the payload duty cycle introduces a binding between different priority levels, because DTAs from different orbits are related irrespective of their priority.

Comparison to (Nearly) Exact Results

By considering only the constraint C-1 with a fixed maneuver duration of 70 seconds it is possible to obtain via brute force optimization an exact result of F_1 for a subset of 247 URs. In addition, we have invested about two days of CPU time into stochastic optimization methods on the remaining URs to generate an (approximate) reference value for the full set of 500 URs with the simplified constraint. These results are $F_1' = 4$, $F_1'' = 33.482$, and $R = 394$. After only about 5 CPU seconds, the PHC method with the parameters given above reaches already $F_1'' = 33.042$ (98.7% of the reference), plans all urgent requests, and reaches $R = 387$ (98.2% of the reference). Investing about 47 CPU seconds ($K = 200{,}000$) further increases these values to $F_1'' = 33.281$ and $R = 390$. For comparison, SHC comes pretty close to these results with $F_1'' = 33.228$ and $R = 390$ using $K = 200{,}000$.

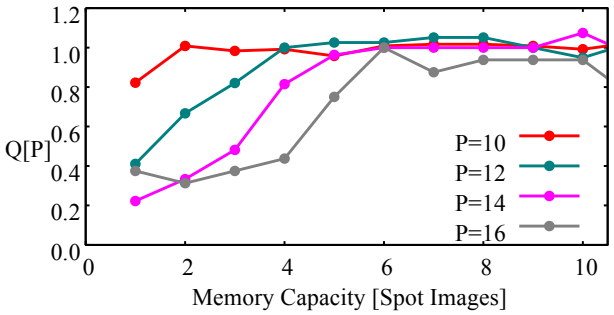

Figure 16: Priority dependence of the contribution to F_1" at different memory capacity.

Algorithmic Extensions

The algorithm as described above is very well suited to extensions. This allows for easy refinement and adopting to different planning requirements.

Boundary Condition Handling

Any planning process is affected by the boundary conditions, which dictate the constraints to be fulfilled at the beginning of the planning horizon. These boundary conditions are taken into account at the single constraint level. E.g. the maneuver planning between two DTAs considers the last DTA *before* the planning horizon and ensures that enough time is given to perform a roll maneuver to reach the first DTA *within* the planning horizon, while only the memory constraint optimization considers the memory fill level at the planning horizon start.

Replanning Strategies

The PRISMA mission does not pose any strict requirements for replanning scenarios. Therefore, the basic strategy is given by *replanning from scratch*, i.e. the whole timeline is planned again with the same input data.

There are, however, more advanced strategies possible. Any new to be planned URs on an already planned (and uplinked) time range may be considered with their full set of DTOs, while only the UR/DTO combinations are kept that have been present as DTAs on the given time range. The hill climbing is then started with these DTOs only, and a quick and efficient replanning is achieved in the case that the new URs are very important. Such a procedure implies a replanning that discards any old DTAs that are in conflict with the new requests.

In case more flexibility is required in the replanning, DTAs may be discarded *and shifted* just by considering *all* DTOs of the already planned URs, but choosing as start configuration for the hill climbing procedure the previously planned DTAs.

Parallelization

We have implemented the presented algorithm in a non-threaded fashion. Nevertheless, there is room for parallelization within the hill climbing iterations. For every iteration solutions for multiple DTOs of the same UR need to be computed. This point can be easily parallelized, for instance by using OpenMP that has widely implemented compiler support. In our test scenario we gain a speedup factor of 1.5 by using two CPU cores.

Using a parallelized version of the algorithm, a larger CPU gain can be made by using multiple cores if instead of coordinate ascent hill climbing the full neighborhood of a given DTO selection is computed in a hill climbing step. In that case, the total required CPU time per hill climbing step scales with the number of URs.

Target Function Optimizations

The hill climbing loop does not directly depend on the target function that needs to be optimized. Therefore, the freedom to define the target function can be used do prefer early DTOs of urgent URs by means of putting a penalty on F_1 for later DTOs.

Limitations of the Algorithm

Downlink Planning

Although the planning of the timings within an orbit and the planning of the optimal sequence of orbits can be done in a mathematically exact way, the planning of the downlink sequence is only an approximation to the optimal solution in terms F_2. This behavior is a tradeoff between solution quality and planning speed, because in a situation where memory capacity becomes a limiting factor, the planning slows down severely.

Constraint Interaction

The conflict resolution in the planning of DLAs does not cover that a removed DTA may have an impact on the global planning of DTAs in the previous steps. E.g. if a DTA from one orbit is removed, this orbit may never have been selected, or the sequence of DTAs within that orbit would have been determined differently. So the missing feedback into the previous stages of the sequential handling of the different constraint types leads to non-optimal solutions, which is a general problem in complex planning problems for oversubscribed resources, for instance in the planning of Landsat 7 (Potter and Gasch 1998)

However, the resulting effects are expected to be very small and may be partially cured by the *global* hill climbing procedure.

Conclusion

The perturbed hill climbing algorithm presented in this article has proven to be powerful both in terms of planning quality and performance.

The main ingredients are the (nearly) exact treatment of a some constraint types and the mapping of the hill climb-

ing search space to the allocation of URs to the DTOs *and not to the DTAs*. Therefore, the shown procedure may be considered as an algorithmic option whenever an exact (or nearly exact) solution is known for a subproblem of the optimization and different choices are possible for the assignment of planning requests to planned time slots. Due to the constraint handling, a valid solution is produced at *every* iteration, and the algorithm may be stopped before the expected number of steps is reached.

Extensions to this algorithm are possible to fulfill an extended set of planning requirements in different operational scenarios, such as replanning of an already planned time range. This flexibility allows for variations of the algorithm to be used in a wider class of missions with increased flexibility and agility of the platform, thus opening ways to improve local search based solutions (Lemaître et al. 2000) by performing global optimization if multiple DTOs per request are given. For instance, VEGA is currently developing, based on the VPT-E framework, a planning system for another Earth observation mission that makes heavy use of the shown results. The planning there is driven by a strict ordering according to priority levels and the platform allows for full roll, pitch, and yaw flexibility along and between the image takes. The algorithmic flexibility allows us to trade result quality for a very fast result generation, so that "last minute" insertions of very urgent requests are possible just before an uplink contact.

Acknowledgments

We would like to thank Giuseppe Corrao for discussions during the design of the algorithm. This work is derived from VEGA's development of a mission planning system for Telespazio SpA.

References

Noll, J., and Steel, R. 2005. EKLOPS: An Adaptive Approach to a Mission Planning System. In *Aerospace Conference, IEEE 2005.*

Sousa, B., and Noréus, E. 2010. Use and abuse of the Language for Mission Planning by Venus Express. AIAA-2010-2101. In *SpaceOps 2010.*

Bellman, R. 1954. The theory of dynamic programming. *Bull. Amer. Math. Soc.* 60: 503-515.

Russell, S., and Norvig, P. eds. 2009. *Artificial Intelligence: A Modern Approach.* Prentice Hall.

Bensana, E., Lemaître, M., and Verfaillie, G. 1999. Earth Observation Satellite Management. *Constraints 4(3): 293-299.*

Joslin, D. E., and Clements, D. P. 1999. "Squeaky Wheel" Optimization. *Journal of Artificial Intelligence Research 10 (1999) 353-373.*

Barbulescu, L., Whitley, L. D., and Howe, A. E. 2004. Leap Before You Look: An Effective Strategy in an Oversubscribed Scheduling Problem. In *Proceedings of the 19th National Conference on Artificial Intelligence, 2004.*

Potter, J.W., and Gasch, J. 1998. A Photo Album of Earth : Scheduling Landsat 7 Mission Daily Activities, In *Proceedings of the International Symposium on Space Mission Operations and Ground Data Systems, 1998.*

Lemaître, M. et al. 2000. How to Manage the New Generation of Agile Earth Observing Satellites? In *Proceedings of the International Symposium on AI, Robotics, and Automation in Space, 2000.*