An Innovative Product for Space Mission Planning
An *A Posteriori* Evaluation

Amedeo Cesta, Gabriella Cortellessa, Simone Fratini and **Angelo Oddi**
ISTC-CNR, National Research Council of Italy
Institute for Cognitive Science and Technology
Rome, Italy
{name.surname}@istc.cnr.it

Nicola Policella
ESA/ESOC
European Space Agency
European Space Operations Centre
Darmstadt, Germany
nicola.policella@esa.int

Abstract

This paper describes MEXAR2, a software tool that is currently used to synthesize the operational commands for data downlink from the on-board memory of an interplanetary space mission spacecraft to the ground stations. The tool has been in daily use by the Mission Planning Team of MARS EXPRESS at the European Space Agency since early 2005. Goal of this paper is to present a quick overview of how the planning and scheduling problem has been addressed, a complete application customized and put into context in the application environment. Then it concentrates on describing more in detail how a core solver has been enriched to create a tool that easily allows users to generate diversified plans for the same problem by handling a set of control parameters, called *heuristic modifiers*, that insert heuristic bias on the generated solutions. A set of experiments is presented that describes how such modifiers affect the solving process.

Introduction

This paper describes an a posteriori analysis of how some general ideas from Planning and Scheduling area have been combined together to create an extremely well accepted product for supporting mission planning. During 2004 we have had contacts with the Mission Planning Team of MARS EXPRESS, a deep space mission around Mars. It clearly emerged that during the first six months of spacecraft activities, the mission planners had faced serious manpower overload in addressing the Spacecraft Memory Dumping Problem (MEX-MDP). The downlink activities were synthesized mostly manually by a team of people continuously dedicated to this task. We started a specific study based on real data analysis and on experience from a previous study whose topic was specifically the dumping problem. After three months, we have started delivering to ESA increasingly accurate operational versions of a software able to cope with real problem instances, and real data files. During a subsequent period of three months the tool has been tested back to back against the previous semi-manual procedure developed within the mission planning team. Since February 2005 the new operational system MEXAR2 is in continuous use at ESA-ESOC, Darmstadt, Germany to solve MEX-MDP instances. It directly synthesizes commands to download data from on board memory to Earth. With further work authors have robustified the tool with additional functionalities and a user interface that facilitates its management.

The paper starts describing the addressed problem and the general architectural choices used to guarantee a seamless integration of the planning and scheduling system in the mission life cycle. Then, it focuses on showing how the core solver has been enriched of features for endowing users with a flexible mixed-initiative interaction with the planning tool. An experimental evaluation completes the analysis of the solver and enable some comments on the lessons learned during the whole project.

The problem

In a deep-space mission like MARS EXPRESS data transmission to Earth represents a fundamental aspect. In this domain, a space-probe continuously produces a large amount of data resulting from the activities of its payloads and from on-board device monitoring and verification tasks (the so-called *housekeeping* data). All these data are to be transferred to Earth during predefined and limited downlink sessions. Moreover, in the case of MARS EXPRESS a single pointing system is present. Therefore the space-probe either points to Mars, to performs payload operations, or points to Earth, to download the produced data. As a consequence, on-board data generally require to be first stored in a Solid State Mass Memory (SSMM) and then transferred to Earth. Therefore, the main problem consists in synthesizing sequences of spacecraft operations (*dump plans*) necessary to deliver the content of the on-board memory during the available downlink windows. This allows to save upcoming pieces of information without losing previously stored data and to optimize given objective functions.

The planned activities of the satellite are incrementally refined as soon as a specific operational period approaches. Some activities are not predicted in advance and a *short-term plan* is generated each one or two days. This two days plan is part of the set of commands uplinked to the satellite. Additionally due to differences in compression algorithms the amount of data produced by some of the activities are not exactly predictable in advance creating once in a while

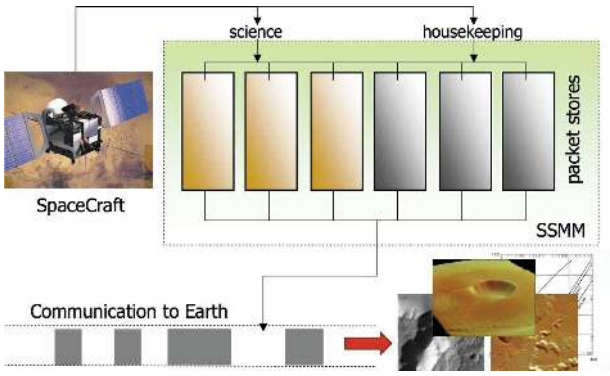

Figure 1: The on-board vs. ground segment data flow

the need to quickly recompute the short term plan before the commands uplink.

Figure 1 shows the main data flow addressed in our work: the spacecraft instruments produce data (both science and housekeeping) stored in the on-board memory subdivided into slots called *packet stores*. The goal is then to create a set of commands for downlinking such data to Earth. In general, synthesizing the plan involves managing the constraints due to the bounded on-board memory, limited number of downlink sessions, and the different data transmission rates of the communication windows.

Domain entities. To model the problem we have subdivided the basic entities that are relevant to the MEX-MDP domain into *resources* and *activities*: resources represent domain subsystems able to give services, whereas activities model tasks to be executed using resources over time. We model two different types of resources: (a) the SSMM that is used to store both science and housekeeping data. This is subdivided into *packet stores*, $\{pk_1, pk_2, \ldots, pk_{np}\}$, each one with a fixed capacity, c_i, and priority, pr_i. Each packet store can be seen as a file of given maximum size that is managed cyclically, that is, previous pieces of information will be overwritten, and then lost, if the amount of stored data overflows the packet store capacity; (b) the *Communication Channel* that represents the downlink connections to Earth for transmitting data. This resource is characterized by a set of separated communication windows $\mathcal{CW} = \{cw_1, cw_2, \ldots, cw_{nw}\}$ that identify intervals of time in which downlink connections can be established. Each element cw_j is a 3-ple $\langle r_j, s_j, e_j \rangle$, where r_j is the available data rate during the time window cw_j and s_j and e_j are respectively the start and the end-time of this window.

Activities describe *how* resources can be used. Each activity a_i is characterized by a fixed duration, d_i, and two variables, s_i and e_i, which respectively represent its start-time and end-time. Two basic types of activity are relevant to MEX-MDP, store operations st_i and memory dumps md_i. Each store operation, st_i, "instantaneously" stores, at its end-time e_i, an amount of data q_i in a defined packet store, pk_i. Through a memory dump, $md_i = \langle pk_i, s_i, e_i, q_i \rangle$, an amount q_i of stored data is transmitted from the packet store pk_i to the ground station, during the interval $[s_i, e_i)$. Two different data types are modeled using store operations: the *Payload Operation Request* (POR) and the housekeeping activities. *PORs* are scientific observations which generate a set of data distributed over the available packet stores. Housekeeping activities generate a flow of data with constant rate which is to be stored in the dedicated slots of the SSMM.

Problem definition. A single MEX-MDP instance is composed of a set of scientific observations, a set of housekeeping productions, and a time horizon $\mathcal{H} = [0, H]$. The *solution* to a MEX-MDP is a set of dump commands $S = \{md_1, md_2, \ldots, md_{nd}\}$ such that:

- At each instant of $t \in \mathcal{H}$, the amount of data stored in each packet store pk_i does not exceed the packet store capacity c_i, i.e., overwriting is not allowed;
- The whole set of data is "available" on ground within the considered temporal horizon $\mathcal{H} = [0, H]$, except an amount of residual data for each packet store pk_i lower or equal to the capacity c_i;
- Each dump activity, md_i, is executed within an assigned time window cw_j which has a constant data rate r_j. Additionally, dump commands cannot mutually overlap.

Several additional constraints deriving from operational practice are also considered by the solver. One of these constraints concerns the mentioned distinction between *housekeeping* and *science data* that have different relevance for mission planners and, for this reason, are stored in separate packet stores. The science devices allow to maintain on-board residual data, while housekeeping data have to be emptied by the end of each mission day because the status of the spacecraft should be checked continuously. A further constraint for the housekeeping packet stores requires to download them in a single dump, without preemption.

In general a solution should satisfy all the imposed constraints. A further goal is to find *high quality* solutions with respect to some specified metrics like *plan size*, *robustness*, etc. Informally, *a high quality plan* delivers all the stored data (one with no overwriting), contains the smallest number of dump activities, satisfies the priorities preferences imposed on the set of packet stores and is able to "absorb" external modifications that might arise in a dynamic execution environment.

Metrics. Four quality metrics are taken into account: the percentage of data *lost*, the *size* of a dump plan, the *robustness* of the solution, and the *weighted delivery delay*. For all of them, *the lower the value, the better the solution*. A short justification for each metric is given below.

- *Data Lost* (LOST) – Percentage of the *total input cyclic data* lost over the planning horizon. Total input data represents the sum of the initial volume of data in the packet stores and the volume of data produced by all the considered store operations.

– *Plan Size* (SIZE) – The number of dump commands in a solution S. This is an important quality for the plan because each command requires both a certain time to be uplinked to the spacecraft and a memory space[1] on-board before being executed. For these reasons mission planners strongly prefer short plans.

– *Robustness* (RBT) – In the case of the MEX-MDP problem our aim is to control the level of memory use in order to minimize data loss due to overwriting. One possibility for overwriting can occur when a greater than expected volume of data has to be stored and not enough space in the packet store is available. For this reason we define as *robust* a solution in which a specified amount of space for each packet store is preserved in order to safeguard against overwriting. The robustness of a solution is defined as the maximum value of packet store utilization. For each of them we define their utilization as the ratio between the maximum level of data in the packet store and its capacity.

– *Weighted Average Delivery Delay* (WDD) – It is defined as the average elapsed time over a set of stored activities. Each elapsed time is the difference between the instant when a generic store activity st_i is actually memorized on-board and its delivery time to Earth. This value is weighted through the packet store priority, such that the higher the priority, the higher WDD.

The above set of metrics allows for a comparison of different solutions for the same problem along different views. According to our experience the following *priority schema* is the most commonly used: LOST $>$ SIZE, SIZE $>$ RBT, SIZE $>$ WDD. Where $metric_1 > metric_2$ means that $metric_1$ dominates over $metric_2$. For example, with LOST $>$ SIZE, we mean that we never sacrifice data on account of shorter dump plans. Even if the last decision is up to the mission planner, these metrics can be useful to decide which solution to upload among a set of different ones.

Type of problem. It is worth noting that the MEX-MDP is mostly a planning problem because its key aspect is the synthesis of the sequence of dump commands md_i. To solve MEX-MDPs, we have introduced two levels of abstraction: (a) the first, *Data Dump Level*, assesses the amount of data to dump from each packet store during each time window; (b) the second, *Packet Level*, starting from the results of the previous level, generates the final dump commands. Such an abstraction allows us to focus on the *dominant* aspects of the problem, i.e. data quantities, packet stores capacities, and dump capability over the communication windows, without considering the problem of actually generating the dump commands. It is worth remarking that the packet level step can be automatically accomplished once a solution for the Data Dump level is computed (no failure is possible). In rest of the paper, we focus on producing solutions for the Data Dump level.

[1]Dump commands, as well as other spacecraft commands, have a dedicated and limited space on board which is distinct from regular SSMM.

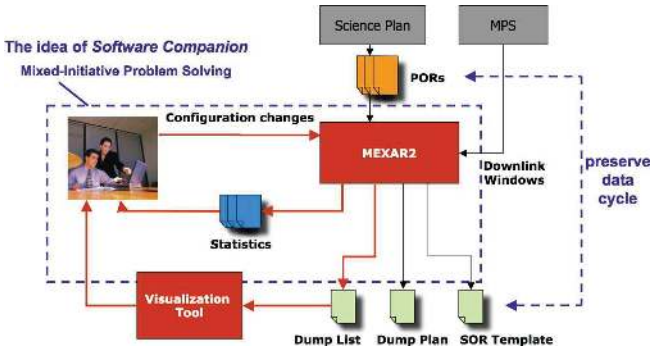

Figure 2: Dump plan synthesis based on MEXAR2

MEXAR2, a usable product for the problem

Introducing an AI decision support tool into a space mission which had been operational already for six months was not an easy task. Two key requirements drove our approach: (a) the data flow should not to be modified and (b) the responsibility of the mission planners should be preserved.

Supporting seamless integration. The current procedure for synthesizing dump plans by using MEXAR2 is shown in Fig. 2. MEXAR2 directly accepts as input the POR requests from the Science Plan, and the specification of the downlink windows from MPS (Mission Planning System). It produces the dump plan in the three formats expected by ESA people (Dump List, Dump Plan and SOR Template in the figure). This has been obtained by encapsulating the intelligent system between two software modules (see Fig. 3): the first (the *Parsing* module) that processes the input files and selects the relevant information for the symbolic model used by the solver, and the second (the *Output Generation* module) that manipulates the results produced by the system and generating the output according to external formats.

As for the responsibility issue, we have designed a highly interactive tool that enhances the capability of the users offering a change of perspective in generating a solution. Users do not directly interact in the attempt of "almost manually" producing the plan, as was done before the introduction of MEXAR2, they rather establish a dialogue with MEXAR2, having access to the model of the domain and various control features to tune the algorithms. From MEXAR2 they receive also additional information, for short referred to as Statistics in Fig. 2. This information enriches their analyses of the current solution. In general we have pursued the goal of allowing users to take more strategic decisions, and maintain control over the synthesis of *dump plans*, delegating to MEXAR2 not only the repetitive part of the work but also the proactive role of creating the dump plan. This has been achieved by exploiting MEXAR2 capability to work quickly and to consider clues given by the solving strategies. In general all the parts of the intelligent system have been

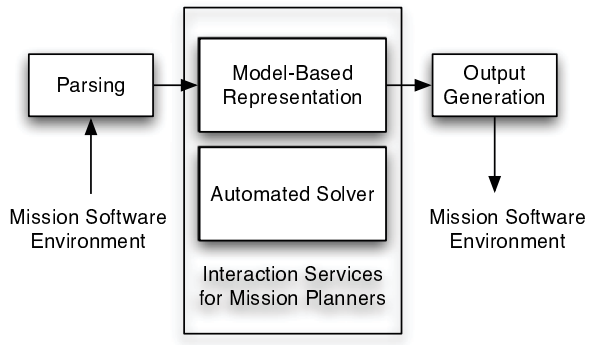

Figure 3: The general architecture underlying the approach

designed to allow a mixed-initiative problem solving activity.[2]

Generic software architecture. Fig. 3 shows also a complete blow up of MEXAR2 software components. The role Parsing and Output Generation modules have been already described. The rest of the system involves three modules that provide all the core functionalities: (a) a domain modeling part (Model-Based Representation in the figure), (b) an algorithmic module (Automated Solver), (c) an interaction module (Interaction Services) that allows mission planners to access both previous modules. The Parsing and Output Generation modules directly interact with the model-based representation that acts as the key module for the whole approach, e.g., also the solver directly extract and store all the relevant information from/to the modeling part.

Interaction services. An in depth description of the MEXAR2 Interaction Module is not in the scope of this paper and we just remind its main role. The module has been designed with the aim of (a) reproducing the usual problem solving cycle of real users, collecting in a unique tool all features that guarantee their traditional problem solving style; (b) allowing to exploit the domain expert abilities and foster her involvement and contribution to the problem solving. In general we can say that the interaction services support the mixed-initiative loop between user and system underscored in Fig. 2. On a given problem it is possible to set up the solver parameters (see later in the paper), inspect statistics from the current solution, graphically visualize specific aspects, tune the parameters and ask for further solutions. Additionally a solution database maintain a set of solutions for the same problem thus allowing exploration of alternatives.

Internal representation

The modeling core grounds on the temporal evolution of key components, plus the ability to capture relevant domain constraints. This approach to the solution based on "timeline

[2]The term mixed-initiative is stretched in different directions by different researchers. We use it here to stress the ability of MEXAR2 to include the user in the decisional loop.

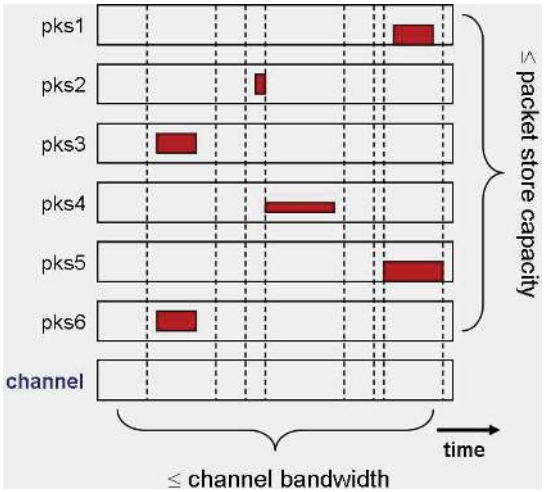

Figure 4: A computational model for MEX-MDP

synthesis" for problem components has roots in solid work in space domain such as, for example, RAX-PS/EUROPA (Jonsson *et al.* 2000), and ASPEN (Chien *et al.* 2000).

Model-based representation with timelines. The timeline based approach focuses on problem features relevant to the MEX-MDP (see Fig. 4). In particular it considers the temporal evolution of specific system components, namely the packet stores, $\{pk_i\}$, and the transmission channel. The problem reduces to decide temporal functions representing amount of data manipulated over time by these key components, such that the constraints given by the packet stores capacity and the channel bandwidth are satisfied.

In our first approach to the MEX-MDP problem we specialized a Constraint Programming approach. We first refined the model subdividing the temporal horizon in contiguous intervals (called *windows*) such that instantaneous memory operations may happen only at the edges of these windows. Decision variables are then associated to each interval on the packet stores and represent the volume of data dumped within each time slot. Hence we added constraint propagation rules to cope with global constraints (see again Fig. 4) for the limited capacity of packet stores and the channel availability and transmission rates. A first solver was then based on iterative sampling (Oddi *et al.* 2005). In further work (Oddi & Policella 2007), we introduced a Max-Flow reduction of such an internal representation that enabled the use of standard Max-Flow algorithms as solvers. This approach is currently operationalized within MEXAR2.

The use of Max-Flow supports different flexible aspects particularly useful for MEX-MDP. For example, it finds, whenever exists, a consistent solution given any initial situation (or dump allocation), e.g., the initial situation can be empty (no dumps decided yet) or not (some dumps already allocated). This property is important for example to enable the user to impose some dump operations (e.g., a user can force dump operations from a given packet store in order to have its data available at a specific time instant). Hence it is

possible to use the algorithm to complete the timelines when a partial solution exists.

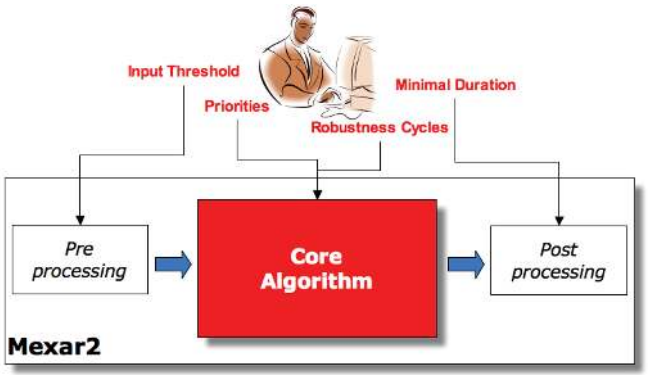

Figure 5: Sketch of the solving cycle

The automated solver

The solving algorithm is organized as sketched in Fig. 5. The core algorithm is sound and complete. Additionally, a set of heuristic *modifiers* are used to influence the core algorithm to obtain different solutions w.r.t. specific quality measures. We first introduce the algorithm then dedicate attention to the modifiers.

Core Algorithm. Algorithm 1 gives the high level picture. The algorithm is composed of two steps. Goal of the first step (Data Dump Level) is to produce a data allocation over the communication channel: for each packet store and for each time window, the amount of data to download is decided. The second step (Packet Level), generates instead the down-link commands starting from the allocation produced in the first step.

Algorithm 1: Generation of down-link commands

Input: problem P
Output: Down-link commands Cmd

```
// Data Dump Level
// Produce a data allocation
// over the communication channel
```

while S *incomplete* **do**
  S ← `HousekeepingAllocation(P)`
  S ← `MaxFlowAllocation(P, S)`
  if *No more HK allocation* **then**
    break

```
// Packet Level
// Generate data commands
```

Cmd ← `GenerateDownLink(S)`

return Cmd

The key aspect is the generation of the data dumps. In Algorithm 1 this is obtained by the *while* loop – which is entered with an initially empty solution S. Each iteration of the loop involves the two procedures:

– `HousekeepingAllocation` that aims at finding a consistent solution for the housekeeping packet stores. Indeed these packet stores contain important data relevant for the probe's safety and, as said before, neither residual data nor preemption is allowed. The `HousekeepingAllocation` is essentially a backtracking search that, at each step, generates a possible dump plan for the housekeeping packet stores.

– `MaxFlowAllocation` that, given a partial solution produced by the previous step, generates the data dumps from the remaining (science) packet stores. This procedure is based on a reduction of the problem P (included the solution of the previous step, S) to the Max-Flow: the former problem has a solution when the maximum flow of the latter equates the total amount of data to dump.

These two steps are iterated until a complete solution is found or no further alternatives exist. It is worth noting that in any case at the end of the while loop we have a solution. Indeed, as we are facing the real problem, we always need to return a solution. Even if the algorithm is not able to find one that completely avoids overwriting (i.e., it is not able to find a complete solution), in any case it returns, for any downlink session, the data flow from each packet store (giving in this case the highest priority to housekeeping data). Eventually the solution produced in the first stage, S, is used to generate the final list of dump commands.

Heuristic modifiers. A key issue is to maintain user responsibility over the plan choices. For this reason a deep analysis has been dedicated to design a flexible solver that allows solution space exploration by acting on the "modifier knobs". In particular, as sketched in Fig. 5, four heuristic modifiers are used in order to guide the search towards solutions of different quality: the Input Threshold acts before the use of the Core Algorithm during a pre-processing phase, the Priority and Robustness modifiers affect the used Max-Flow, while Minimal Duration is taken into account in a post-processing step. Further details on the modifiers follow:

Input threshold – Each scientific observation, the POR, produces a sequence of data records, called *zd-records*, which represent the store operations input to the solving algorithm. Potentially the number of *zd-records* impacts both the Max-Flow size and the number of dump operations. To keep such number low, many small data records are grouped into a single one to influence the algorithm to dump them with a single command. The threshold value T_i bounds the max cumulative volume of the grouped data record. For each packet store a threshold can be set to a given percentage of its capacity c_i.[3]

Priorities – The priority pr_i associated to the packet stores

[3]By setting a threshold to a value T_i, a sequence of many small data records targeted on the same packet store are grouped into a single cumulative record of size $T_i \cdot c$ and stored at the end of the grouped sequence.

pk_i. The allocation in both `HousekeepingAllocation` and `MaxFlowAllocation` can be done taking into account the priority values of the housekeeping and science packet stores. In addition, the priority is considered by the `GenerateDownLink` procedure. In fact, the input of this procedure consists of a set of data dumps for each time window. These dumps can be ordered according to priority.

Robustness Cycles – The `MaxFlowAllocation` procedure accepts as a further parameter the number of cycles to iterate the search for more robust solutions. In this case, we have an iterative max-flow search that tries to *flat* possible peaks in the usage of the on-board memory (Oddi & Policella 2007).

Minimal Duration – Another parameter of a certain interest is the minimal allowed duration ϵ_d for a dump command. In order to meet this requirement, the output solution of `MaxFlowAllocation` is post-processed in order to remove possible short commands. The idea is quite simple, the resulting flow network is transformed in another consistent one by iteratively applying local adjustments. This parameter complements the threshold modifier acting as a post-processing phase (Fig. 5).

Searching the solution space acting on modifiers. We conclude this section briefly summarizing the main effects of the setting of the previous four heuristic modifiers on the metrics LOST, SIZE, RBT and WDD.

Input threshold – Setting a T_i has the effect of grouping *zd-records* thus *reducing the number of store operations* by increasing the size of the represented *zd-records*. There are two main effects on the output of the solving algorithms. First, the *size* of the dump plans tends to be reduced, because the duration of the dump commands increases. Second, the MEXAR2 *computational time* is reduced, because the size of the associated flow network (that also depends on the number of store operations) is reduced.

Even if the setting of this parameter cannot guarantee that the duration of all the dumps is greater than the minimal duration ϵ_d, the use of thresholds in the packet stores is a practical mechanism to tune the input data and change the behavior of the solving algorithm. It easy to apply the following interactive procedure for reducing the *size* of a dump plan and avoiding plan *fragmentation* cycling the following three steps:

1. Spot the packet stores with "short" commands;
2. Set a threshold value for each spotted packet store;
3. Generate a new dump plan.

It is worth noting a side effect in the use of the threshold that should be taken into account. The use of "high value" of the thresholds can create a "large" delay between the instant of time when a *zd-record* is generated and the instant of time when its volume of data is effectively considered as an input for generating a dump plan. In some cases, this delay can generate *virtual* peaks of data, with a consequent data loss in the generated solution. This loss is not due to a lack of dump capacity, but to *coarse* sampling policy adopted for the cyclic input data.

Priorities – Dump operation from different packet stores can be anticipated or delayed setting their priority values. The WDD metric can be used to compare the output dump plans with respect to the imposed priorities: the smaller the value, the more a plan is coherent with the priorities values. In addition, priority values play an important role when the problem is *over-constrained*, that is, when no solution exists and part of the data is lost (this is an extreme situation in the MARS EXPRESS domain, but possible). As already said, the solving procedure must return in every case a solution, which *sacrifices* some data. In this case it is possible for an user to *advice* which data to sacrifice by setting a low value of packet store priority.

Robustness – The number of robustification cycles operated by the solver impacts solution's RBT. Improving the solution robustness is desirable to avoid plan regeneration and/or data losses. However, a side effect is a tendency of the dump plan to increase in *size*. This is because the dump operations tend to be split to perform a finer control of the volumes of data stored in the packet stores.

Minimal duration – Setting this parameters enables a post processing step which removes dump commands with duration lower than ϵ_d. The counterpart of this choice is an increase of the metric WDD, because some dump operations have to be delayed in order to meet the duration requirement.

MEXAR2 evaluation

Here an evaluation is presented that considers two main aspects: the performance of the automated solver and the main operative results of the overall tool that come from users feedback.

Evaluating the problem solver

The main concern of mission planners is to avoid as much as possible potential overwriting (i.e. data loss). Therefore, as introduced above, the volume of data *lost* can be considered as the driving *quality measure* for evaluating solutions. A second important aspect that characterizes a dump plan is represented by the plan *size*. Indeed, the time needed to upload the dump commands to the spacecraft (during the subsequent phase called uplink), is proportional to the size of the plan. For this reason a short plan is to be preferred to a longer one. As additional quality measure we consider plan *robustness*.

A complete report on the evaluation of all these aspects is out of the scope of this paper. Hence, this section focuses only on the following three parameters: the data *lost*, the plan *size* and the plan *robustness*. Hence, we do not consider the *priority*, and as a consequence WDD, within our experimental evaluation. In fact, the more commonly used operative schema is to force a higher priority of the *housekeeping*

data over science, without imposing any priority among science data.

Test data and settings. We report the system performance on a benchmark set which refers to real data from the mission from 28 mission days from 12 March to 8 April 2007. It is worth noting that the proposed benchmark involves a very critical value for the ratio *VG/DC* where *VG* is the *total Volume of Generated data* (housekeeping plus science) and *DC* is the total *Dump Capacity* of the communication channel. In the considered interval, *VG/DC* is equal to 0.88. This is a real critical benchmark and a likely candidate for data loss at execution time. In fact according to our experience, the value of the ratio *VG/DC* is usually between 0.2 and 0.8. For the sake of completeness, in the used benchmark the total number of stores [4] equals to 4064 and the total number of PORs equals to 358.

Quantitative Performance. We report here a set of quantitative evaluations using different input settings for MEXAR2. The tests were executed on a AMD Athlon64 3500+, 2 GByte RAM, under Windows XP Professional. Table 1 reports the following data:

Setting – The used parameter settings (the choice of the 3-ple $\langle$threshold T_i; number of robustness cycles rb; minimal duration $\epsilon_d\rangle$).

1. The first row in the table called REF obtained with no threshold, $rb = 0$ and $\epsilon_d = 0$, is used as a reference to show the effects of the heuristic modifiers. By inspecting the dump plan obtained with REF we have identified two packet stores whose plan presents commands with small durations (less than 30 seconds): ASPERA (AS) and MARSIS (MI). Therefore we have chosen these two packet stores as the focus to create further groups of settings.
2. An *explorative* group of 9 settings, represented in the central part of Table 1, is obtained by changing a single modifier (T_i, rb or ϵ_d) with respect to REF. In particular, for the threshold T_i we adopt the values: $\{T_{AS} = 10\%, T_{MI} = 4\%$, and the combined variation of $T_{AS} = 10\%$ and $T_{MI} = 4\%\}$. For rb we have chosen values $1, 2, 4$ cycles. For ϵ_d values $50, 75, 100$ (seconds) have been considered.
3. Two potential *operative* settings (with labels OP1 and OP2) have been also created to test the combined use of all the modifiers. In particular: OP1 = $\{T_{AS} = 10\%$ and $T_{MI} = 4\%$, $rb = 4$ and $\epsilon_d = 100$ seconds$\}$ and OP2 = $\{T_{AS} = 10\%, rb = 4$ and $\epsilon_d = 100$ seconds$\}$.

Lost – The percentage of data lost with respect to the total volume of data stored.

[4]This represents the total number of store activities st_i considered by the system as input data and represents the *size* of the input data, which can be directly related to the complexity of the solving algorithm, that is, its computational time and memory requirement.

Setting		Lost	Size	RBT	Cpu
REF		0.18	40.1 (111)	49.9 (99.4)	12.5
$T_{AS} = 10\%$		0.08	34.6 (90)	48.9 (99.95)	9.7
$T_{MI} = 4\%$		0.11	34.9 (70)	48.9 (99.5)	5.9
$T_{AS} = 10\%$, $T_{MI} = 4\%$		0.0	30.2 (46)	49.2 (99.9)	4.5
rb	1	0.11	35.2 (91)	44.5 (97.5)	20.6
	2	0.11	35.4 (105)	43.9 (96.8)	27.5
	4	0.07	35.4 (104)	43.5 (93.5)	44.2
ϵ_d	50	0.18	35.2 (90)	49.2 (99.6)	15.9
	75	0.18	34.9 (82)	49.2 (99.7)	16.0
	100	0.18	34.6 (81)	49.2 (99.7)	15.7
OP1		0.0	29.1 (46)	49.2 (99.6)	5.5
OP2		0.05	31.1 (81)	48.1 (96.2)	35.2

Table 1: MEXAR2 performance

Size – The average daily plan size and the maximal size of a daily dump plan (written among brackets).

RBT – To show robustness of plans we represent the average use of a packet store, and among brackets, the maximal value of its use over the planning horizon (28 days). In particular, we only consider the packet store related to the High Resolution Stereo Camera (HRSC). This packet is chosen for two reasons: (a) within the current benchmark HRSC is the most probable candidate to generate uncertain data volume, (b) its data production occupies more that 40% of the whole on-board memory.

Cpu – The total computation time to find a dump plan (in seconds).

Table 1 groups together the experimental results. The first row shows the performance of REF, we just note a small loss of data (0.18%) then we will use these row for the comparisons that follow. We first explore the use of the threshold T_i modifier. Data on the *Size* column confirms that the use of thresholds is a practical mechanisms that MEXAR2 provides to the mission planners. In particular, we see that the main achievement is the reduction of the maximal size of a daily dump plan from 90 to 46 commands and a reduction of the Cpu time for producing it from 9.7 seconds to 4.5 seconds. The experimental data on the robustness (the rows in the table with label rb) confirms the possibility of removing data peaks close to maximal packet store capacity. In fact, the maximal usage value is lowered from 99.4 of REF to 97.5 with a single robustness cycle to 93.5 after 4 cycles. However, we pay the price of increasing the Cpu time from 12.5 (REF) till 44.2 seconds, and we also note that the size of the plan tends to increase augmenting rb. The experimental data on the parameter ϵ_d shows as the post-process on the output dump plans is effective on the REF data. However the threshold T_i modifier obtains better performance on this benchmark. Finally, the two rows OP1 and OP2 show how different compromises among the modifiers values may improve the overall performance. For example, we note how in OP1 the volume of data lost (indeed always very low) is reduced to zero. In conclusion, we observe that the possibility

for the user to tune the modifiers T_i, rb and ϵ_d represents a key aspect for allowing mission planners to explore the solution space. This capability is very effective as soon as the mission planner maps his own experience acting on the parameters values. It is worth underscoring that this capability captures the very core problem of mixed-initiative problem solving: the system should be able to accept hints from the user experience which are difficult to be encoded in the core algorithm.

Main operative results

In evaluating the main results obtained with MEXAR2 we consider two perspectives, namely the benefits for mission planners and the advantages for science management.

With respect to MPS a considerable advantage stems in the mixed-initiative style of the tool. The role of software which supported the decision making in previous practice was really secondary and relegated to constraint checking tasks. On the contrary MEXAR2 is both proactive with respect to plan synthesis and, at the same time, fosters human intervention and control during problem solving. In general data dump generation can now be performed quicker and with less effort. Additionally, the quality of plans exceeds that provided by the tools previously used. The problem of uncertainty in data production is addressed very satisfactorily from the user standpoint and the time saved for producing plans has been estimated to 50%.

The ability of MEXAR2 to quickly generate plans over multiple days allows mission planners to consider alternative solutions in order to avoid data losses. It is worth reminding that before MEXAR2 the mission planners were hardly able to solve the daily problem, e.g., deciding the dumps for one day to the following. As shown in the present evaluation the hard problem on 28 days is now explored with multiple runs of few seconds each, hence the tool guarantees the possibility of performing optimizations.

For example, problematic intervals on a longer time horizon can be easily detected and alternative allocations of involved Payload Operation Requests can be negotiated with the payload scientists thus minimizing overwrites.

The science community benefits from MEXAR2 as well: data from observations are available earlier and data losses are minimized. As already said potential overwrites can be quickly detected and fed back to the science community for PORs update. Thanks to MEXAR2 the use of the downlink channel is optimized and more data can be downlinked, thus increasing science return from the mission.

Conclusions

This paper has described the main outcome of the MEXAR2 project that started from a research study for a specific space mission problem. The problem required a lot of daily human effort. Our work ended up producing a complete tool, MEXAR2, that provides high value solutions for the problem and, additionally, allows the users to concentrate on high level decision making tasks.

The results of this work represents a new example of problem solving tool based on AI planning and scheduling technology that is successfully being used in the operational phase of a space program. Among the previous examples are (Jonsson *et al.* 2000; Smith, Engelhardt, & Mutz 2002; Ai-Chang *et al.* 2004; S. Chien *et al.* 2005). As said in the intro MEXAR2 has been in daily continuous use for two years. As shown in this retrospective evaluation it is able to efficiently solve very complex instances of MEX-MDPs and provides the ability to explore solution space by integrating the user expertise and authority.

Acknowledgements. This work has been sponsored by the European Space Agency (ESA-ESOC) under contract No.18893/05/D/HK(SC). The authors would like to thank the MARS EXPRESS mission control team at ESA-ESOC for the valuable feedback on their work and in particular Erhard Rabenau, Jonathan Schulster, Michel Denis and Alessandro Donati for their continuous assistance.

References

Ai-Chang, M.; Bresina, J.; Charest, L.; Chase, A.; Hsu, J.; Jonsson, A.; Kanefsky, B.; Morris, P.; Rajan, K.; Yglesias, J.; Chafin, B.; Dias, W.; and Maldague, P. 2004. MAPGEN: Mixed-Initiative Planning and Scheduling for the Mars Exploration Rover Mission. *IEEE Intelligent Systems* 19(1):8–12.

Chien, S.; Rabideau, G.; Knight, R.; Sherwood, R.; Engelhardt, B.; Mutz, D.; Estlin, T.; Smith, B.; Fisher, F.; Barrett, T.; Stebbins, G.; and Tran, D. 2000. ASPEN - Automating Space Mission Operations using Automated Planning and Scheduling. In *Proceedings of the 6^{th} International Conference on Space Operations, SpaceOps 2000*.

Jonsson, A.; Morris, P.; Muscettola, N.; Rajan, K.; and Smith, B. 2000. Planning in Interplanetary Space: Theory and Practice. In *Proceedings of the Fifth Int. Conf. on Artificial Intelligence Planning and Scheduling (AIPS-00)*.

Oddi, A., and Policella, N. 2007. Improving Robustness of Spacecraft Downlink Schedules. *IEEE Transactions on Systems, Man, and Cybernetics–Part C: Applications and Reviews* 37(5).

Oddi, A.; Policella, N.; Cesta, A.; and Cortellessa, G. 2005. Constraint-based Random Search for Solving Spacecraft Downlink Scheduling Problems. In Kendall, G.; Burke, E.; Petrovic, S.; and Gendreau, M., eds., *Multidisciplinary Scheduling: Theory and Application*. Springer. 133–162.

S. Chien, S.; Cichy, B.; Davies, A.; Tran, D.; Rabideau, G.; Castano, R.; Sherwood, R.; Mandl, D.; Frye, S.; Shulman, S.; Jones, J.; and Grosvenor, S. 2005. An Autonomous Earth-Observing Sensorweb. *IEEE Intelligent Systems* 20(3):16–24.

Smith, B. D.; Engelhardt, B. E.; and Mutz, D. H. 2002. The RADARSAT-MAMM Automated Mission Planner. *AI Magazine* 23(2):25–36.