Looking for MrSPOCK: Issues in Deploying a Space Application

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

Abstract

This paper describes a recent effort aimed at building a planning system to support human mission planners in the Long Term Planning of the MARS EXPRESS space mission. Specifically the paper focuses on describing the steps that brought us to develop the first release of MrSPOCK, a system that integrates various features from the planning and scheduling research efforts. In particular MrSPOCK is built on top of a core constraint-based representation centered on timelines. It uses a basic constructive procedure and an optimization cycle based on a genetic algorithm to explore the solution space driven by a multi-objective function. The system is completed by an interaction module that provides several functionalities to end users. Goal of this paper is to show how these different ingredients are combined to obtain a complete solution to a challenging problem that enable mission planners to explore alternatives.

Introduction

Over the last decades planning systems research has been deeply influenced by challenges offered by space applications. Innovations have concerned initial works on temporal planning (Vere 1983), real time control of the space shuttle (Ingrand, Georgeff, and Rao 1992), planning and execution loop, e.g., (Knight et al. 2001), the broad concept of autonomy (Muscettola et al. 1998), the allocation of Earth Observations on a satellite (Bensana, Lemaitre, and Verfaillie 1999), negotiation tools for on-ground decision making of Mars missions (Ai-Chang et al. 2004) and so on. Somehow the risk of presenting "yet another application for space" is quite high. Nevertheless the space domain is so rich of stimuli that the issues to be considered are extremely diversified and the related challenges are far from being completely addressed or closed.

In this paper we describe a work performed for ESA, (the European Space Agency) to develop a re-usable software framework for planning problems and its use to address a specific mission planning problem related to the MARS EXPRESS mission.

The paper is a report on the ingredients that brought us to develop MrSPOCK, the "MARS EXPRESS Science Plan Opportunities Coordination Kit", a new tool which combines together diversified research aspects from the planning and scheduling area.

MrSPOCK solves an interesting multi-objective optimization problem that requires the satisfaction of a number of temporal and causal constraints to produce long term plans for the MARS EXPRESS spacecraft activities. An interesting aspect of the system is the hybrid combination of a constraint-based representation that supports timeline-based planning and scheduling, an optimization algorithm that exploits such representation and an interaction front end which has multiple features. The system has been first deployed to end users during May 2008 and it is currently being refined to perfectly match the details of the daily use. Apart the fielded application it is worth highlighting the interesting leverage we obtained with respect to our previous experience in ESA projects, e.g., (Cesta et al. 2007), due to the use of the general purpose software framework based on timelines. This general framework has allowed us to capture an amount of constraints with a basic domain description language. Additionally the use of the timeline-based representation as a central concept for the user interaction front-end demonstrates again its particular suitability to capture the way of working of human planners in space domains.

The Problem

Currently, long, medium and short term planning for MARS EXPRESS is carried out through a collaborative problem solving process between a SCIENCE TEAM located at ESA-ESTEC which manages the PIs request for operating on-board payloads and the MISSION PLANNING TEAM located at ESA-ESOC which is responsible for spacecraft operational constraints (see Figure 1). These two groups of human planners iteratively refine a plan containing all activities of the mission. The process starts at the long term plan (LTP) level – three months of planning horizon – and gradually refines to obtain a Medium Term Plan (MTP) and then a set of fully instantiated activities at short term plan (STP) level – one week of planning horizon. In particular the STPs are then further refined every two days to produce final executable plans.

Broadly speaking the two groups of managers at ESOC and ESTEC share information and collaborate at each level of abstraction. At LTP level the plan is abstract and flexible and many decisions can be negotiated. As soon as the planning process moves toward the STP level, plan activities are more in charge of the Operation Team, constraints become mandatory and science requests for observations can hit against the real constraints imposed by the spacecraft (e.g., power availability, illumination constraints, maintenance windows or flight dynamics constraints). When defining a starting LTP the lack of an accurate model of the spacecraft on the SCIENCE TEAM side is one of the main cause for performing many expensive iterations between the two groups. On the other side the MISSION PLANNING TEAM has only partial information about the requested science operations for MARS EXPRESS, thus adding new sources of uncertainty to the decision process.

The general goal of MrSPOCK is to offer a pre-planning optimization tool for spacecraft operation planning able to generate a *pre-optimized skeleton LTP* subject to subsequent cooperative SCIENCE TEAM/MISSION PLANNING TEAM refinement, which can guarantee both a reduction in the time spent in the iterative refinements and an improvement of the science activity.

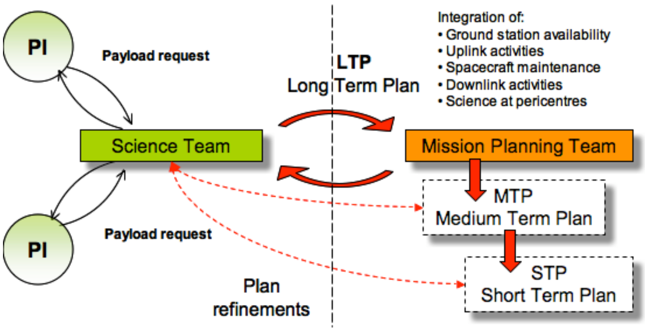

Figure 1: The complete cooperative scenario between ESOC and ESTEC. Task of MrSPOCK is to produce a skeleton Long Term Plan. Synthesis of Medium and Short Term Plans is outside the scope of the current effort but they both involve the same actors.

In order to introduce the main types of activities that compose the LTP skeletal plan we distinguish among three phases of each orbit around MARS: (1) time interval around the *pericentre* (the closest orbital point to the target planet); (2) time interval around the *apocentre* (the farthest orbital point from the planet); (3) time interval *between* the pericentre and apocentre passages. During the pericentre period the spacecraft is preferably requested to point the planet thus allowing observations of the planet surface with its payloads – this is generically referred to as *Science operation*. Between pericentre and apocentre passages, the spacecraft can transmit data to Earth (*Communication*), thus pointing to Earth. This activity should occur within ground station availability windows. Additionally, *Maintenance* operations should occur around the apocentre passages.

At the LTP level the problem consist in deciding a set of slot assignments for the main activities of the spacecraft (i.e., Science, Communication, Maintenance) such that all the operational constraints are satisfied.

Operational Constraints. The problem presents different *hard* and *soft* constraints to be satisfied. Examples of soft constraints are related to the uplink windows frequency and duration. In particular it is required a four hours uplink time (δ=4) each 24 hours (T_{ud}=24). Additionally, the possibility to split a four-hour uplink window into two-hour uplink windows should be preserved. Apocentre slots for spacecraft maintenance windows must be allocated between o_{min} and o_{max} orbits apart (usually 2 and 5 are the used values). The maintenance duration of 90 minutes is to be centered around the apocentre. Communication activities are source of several temporal constraints to be considered as hard. For example: (1) the minimum/maximal durations for the X-band transmitter in the *on* state, (2) the minimum duration for the X-band transmitter in the state *off*; (3) the periods in which the X-band transmitter has to be *off* (e.g., eclipses, occultations, slewing manoeuvres and non-Earth pointing status);

Furthermore, there are different operational constraints to take into account when selecting ground stations. In particular, ground stations have different features like different dish diameters (there are 70 meters dish antennas, 35 meters and 34 meters). Usually they allow both uplink and downlink communications, but there are cases where downlink is the only possible operation. Additionally there are ground stations owned by different agencies and they should be used according to some restriction policy.

The need for a new problem management. Given these broad requirements, current practice at the MISSION PLANNING TEAM is to produce an initial skeleton plan for MARS EXPRESS by allocating over the planning horizon (which generally covers hundreds of orbits) the three different types of decisions already introduced: (a) selection of the *Maintenance* windows (centered around the apocentre events); (b) selection of the *Communication* windows among the available ground stations; (c) selection of the windows for *Science* operations around pericentre events.

An open problem for the actual working scenario is introduced by the approximation in taking decision. As already mentioned this ends up requiring a high number of interations in the negotiation process between SCIENCE TEAM and MISSION PLANNING TEAM and entails a slow convergence to an agreed shared solution.

In this context, the challenge of the current open problem is to provide an automated procedure for producing a *good* skeleton plan, i.e., a LTP that takes into account the needs of both parties, thus reducing the effort in reaching a shared solution. Overall, the generated LTP should be such that: (a) the number of (expensive) iterations between SCIENCE TEAM and MISSION PLANNING TEAM is reduced; (b) a set of objective functions are optimized, that include the maximization of data downlink operations; the number of pericentres for science operations; the number and the uniform distribution of uplink windows.

General Approach to the Problem

In moving planning and scheduling into the real world an important feature is not only producing a solution to a difficult problem but also integrating a number of features in the delivered software that contribute at creating "a complete solution to the problem". For example, a seamless connection of the solver to the actual work environment is an additional issue to consider. To foster a smooth deployment of our solution within the working environment we chose to address three distinct aspects in a combined/integrated way: (a) a timeline-based knowledge representation, (b) a specific algorithm for the problem, and (c) a user-oriented interaction front-end.

The last two aspects will be described in details later, while the rest of this section focuses on the representation issue and on the general approach to address the problem.

The choice of a core representation based on *timelines*, (i.e., temporal functions that describe key features of the domain to be controlled by the planner over time), is particularly suitable for the space domains where the need for controlling, reasoning and, in general, taking decision over time is a recurring activity. A direct manipulation of this representation is important also for end users and ensures a beneficial impact on the ease of comprehension of the proposed solutions.

Timeline-based representation has been used in a primitive way in several planners like, for example, RAX-PS/EUROPA (Jonsson et al. 2000), and ASPEN (Chien et al. 2000), it has been more formally studied in papers like (Cesta and Oddi 1996) and (Frank and Jónsson 2003), and it is also the basic building block of state-of-the-art scheduling algorithms (Cesta, Oddi, and Smith 2002).

One of our recent efforts has been devoted to study features integration from such previous research and has resulted in proposing a general purpose planning and scheduling system called OMPS – Open Multi-component Planner and Scheduler (Fratini, Pecora, and Cesta 2008). OMPS unifies timelines of different nature under the unique concept of *component*, where each component is an entity that has a set of possible temporal evolutions over an interval of time, the *horizon* over which these evolutions are defined.

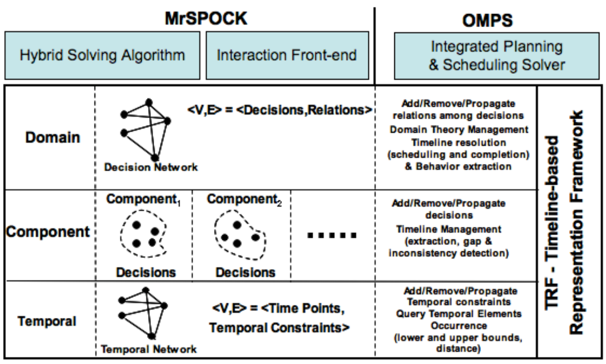

Figure 2: Connections between MrSPOCK and OMPS.

The design of MrSPOCK intersects with the development of OMPS but presents some differences derived from the need to solve a specific problem of a real world context.

Connections and distinctions between the two systems are sketched in Figure 2. MrSPOCK uses the core timeline based representation services of OMPS. Indeed, the two planning systems share a layered representation module referred to as the "Timeline-based Representation Framework" (TRF). The TRF allows representing the temporal evolution of the *components* as well as the constraints that affect their temporal evolution. Additionally in the TRF deductive systems are implemented that proactively propagate effects of external decisions on the modeled segment of the target domains. As shown in the picture, the TRF is composed of three layers: (a) the lower layer dedicated to the temporal constraint network (*Temporal Layer*), (b) the intermediate layer where entities of the domain can be represented as independent components (*Component Layer*). At present components can be both *resources* and *multi-valued state variables*, (c) the higher layer (*Domain Layer*), which allows representing the interaction among the different components to reflect the causal constraints among different parts of the represented domain. This causality is represented in the form of constraints that temporally synchronize the behaviors of different components in time instants or time intervals. In other words, these constraints force two or more components to have a particular behavior either in specific instants or in time intervals.

This last layer supports the problem solving capability by providing the possibility to represent solving *decisions*. Decision is a generic term to represent a *choice* with respect to the temporal evolution of a component timeline (e.g., decide a value of a state variable in a given time interval, or posting an ordering constraint). Decisions are the basic means for a solver to interact with the TRF.

We give now slightly more details on the three layers with reference to the MrSPOCK current implementation.

The **Temporal Layer** manages temporal information in shape of Temporal Constraint Networks (TCNs) (Dechter, Meiri, and Pearl 1991). TCNs allow representing *events*, also called *time points*, and *temporal constraints* that represent distances, separation constraints, etc. This layer is endowed with propagation algorithms to maintain the consistency of the possible value assignments to time points. The current implementation in OMPS and MrSPOCK is based on the Simple Temporal Problem (Dechter, Meiri, and Pearl 1991). The **Component Layer** provides the main entities that can be represented in the form of components. In this architecture a component has the primitives to compute the effects of *decisions* (generated by solvers or users at higher levels) over its *behaviors*. A component provides to the higher level some basic timeline-management primitives (like behavior extraction, inconsistency-detection, etc.). This layer allows expansion of the TRF representation ability since components make the architecture independent from the actual implementation of the functionalities they provide, encapsulating component-specific algorithms and hiding differences about behaviors, inconsistency detection and resolution behind a common interface. The **Domain Layer** manages *relations* among decisions maintaining the *decision network* updated. This is the level where concurrent threads represented by each component in the underlying level are put together to constitute the component-based *domain*: this level is in fact responsible for providing domain theory management functions (e.g., sub-goaling and/or unification possibilities) and to generate synchronizations among components. Figure 2 emphasizes what distinguish the *domain independent planner* OMPS from the *domain dependent planner* MrSPOCK. OMPS connects the TRF with a general purpose search algorithm able to solve planning and scheduling problems specified in terms of a multi-component domain theory (see (Fratini, Pecora, and Cesta 2008)). MrSPOCK takes advantage from the TRF representation capabilities and their associated deductive services and develops a component based representation of the domain features. It connects such a representation layer to two modules that complete the application: namely a hybrid solver and an interaction module.

A comment is worth doing with respect to the differences between OMPS and MrSPOCK. A critical point in developing an application to produce the MARS EXPRESS skeleton LTP is the consideration to be given to a great number of operational constraints that cannot been removed after four years of daily mission operation practice. In order to capture the work practice we had to cope with very specific constraints that are difficult for the general purpose solving framework but more easily to be taken into account in a domain specific solver, hence the choice of creating such solver on top of the TRF. In general it is worth underscoring that in developing application of planning and scheduling in real context the trade-off generality/specificity is a relevant one even if it is usually not mentioned in official literature. In our previous experience described in the MEXAR2 tool (Cesta et al. 2007) we have used a model-based representation based on timelines and several principles of mixed-initiative planning that are research products of our area, the whole implementation was done on-purpose for the application. In MrSPOCK the amount of the general purpose modules used in the implemented system is quite high with respect to our previous work. It is also worth mentioning that the development of a solver entirely based on OMPS would require the customization of an amount of specific knowledge in the domain description with a consequent production of a rather cumbersome domain model. Our choice has been to use TRF for clean modeling purposes while relying on a specific module for driving an efficient problem solving.

The MrSPOCK Domain Model

MrSPOCK uses the TRF domain modeling capabilities to capture the main entities of the Long Term Plan domain within the MARS EXPRESS mission.

In order to describe the components we used to model the problem it is important to introduce two different types of them (1) *Controllable Components*, whose temporal behavior is decided by the solver. They define the search space for the problem, and their timelines ultimately represent the problem solution; (2) *Uncontrollable Components* the evolution of which is exogenous to the solver. They represent values imposed over time which can only be observed; they can be seen as additional/external data and constraints for the problem.

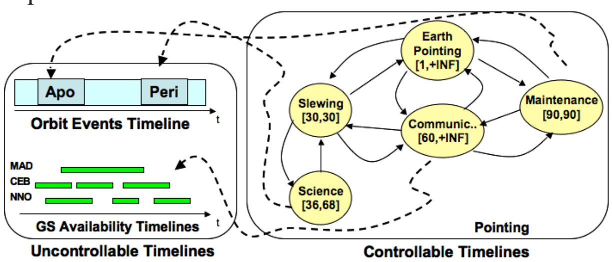

Figure 3: Domain components and their causal synchronizations.

Figure 3 shows how the MARS EXPRESS LTP domain is captured in the current release of MrSPOCK. In particular in this case we only use the multi-valued state variable component type. [1]

[1]We have delivered two versions of the solution. A preliminary one, mildly described in (Fratini, Pecora, and Cesta 2008), included a modeling of the battery discharge that has been obtained with a modified resource component, taking advantage of the heterogeneity of components in TRF. At present the model agreed upon with the users requires state-variables only.

A single *controllable* state variable models the spacecraft's pointing mode (*Pointing*), which specifies the temporal occurrence of *Science* and *Maintenance* operations as well as the spacecraft's *Communication* to Earth. The values that can be taken by this state variable, their durations (represented as a pair $[min, max]$) and the allowed transitions among the possible states are synthesized by the automaton shown in the right side of Figure 3.

As uncontrollable variables we represent ground stations (GS) availability and the occurrence of the key orbit events (Apocentre and Pericentre). The temporal occurrences of pericentres and apocentres are shown in Figure 3 ("Apo" and "Peri" values on the timeline, left/top part of the picture) and are defined in time according to an orbit event file decided by the flight dynamics team. The other state variable maintains the visibility information of three ground stations ("MAD","CEB" and "NNO" timelines left/bottom part of the figure). The allowed values of these state variables are: {`Available(?rate,?ul_dl,?antennas)`, `Unavailable()`}, where the `?rates` parameter indicates the bitrate at which communication can occur, `?ul_dl` indicates whether the station is available for upload, download or both, and the `?antennas` parameter indicates which dish is available for transmission.

Any valid plan needs temporal synchronizations among the pointing timeline and the uncontrollable variables. These synchronization constraints are represented as dotted arrows in the figure: *Science* operations must occur during Pericentres, *Maintenance* operations must occur during Apocentres and *Communication* must occur during ground station visibility windows. As mentioned, in addition to those synchronization constraints, the Pointing timeline must respect the transitions among values specified by the automaton and the minimal and maximal duration specified for each value (in the automaton as well). How all these constraints can be naturally represented in terms of TRF features is omitted for the sake of space. The implementation of these constraints (causal constraints like in planning) in the problem solution is similar to sub-goaling activity in a timeline-based planner (e.g., (Jonsson et al. 2000)).

A solution is obtained when a set of consistent timelines for the controllable component are defined and all the operational constraints are satisfied. A distinctive aspect of MrSPOCK is the direction we have taken to build a problem solver once obtained the timeline representation: instead of using a generic search engine (for example the planning and scheduling integrated search of OMPS) we have built a specialized solver that dialogues directly with the problem representation in the TRF. In this way we exploits the TRF constraint engines for propagating several types of constraints, while using specialized search engines partly general partly tailored to the problem. In particular, the module called "Hybrid Solving Algorithm" in Figure 2 integrates a greedy one pass constructive search procedure with a generic optimization cycle that uses a genetic algorithm approach as discussed next. One of the interesting achievements in our current work is the hybridization of a timeline based general purpose approach with a wrapping module that implements a genetic optimization search. It is worth underscoring again how the TRF is endowed with propagation algorithms hence it is not just a bookkeeping data structure rather it has an active role as is current practice of constraint satisfaction engines. In creating a complete architecture we situate MrSPOCK at an intermediate stage between generic timeline-based planners and the domain specific timeline-based solvers experience described for example in (Cesta et al. 2007).

MrSPOCK Solver

Our first solver built on top of the representation has been a greedy one pass constructive search that, scanning the temporal horizon from left to right, decides allocations for *Science*, *Communication* and *Maintenance* satisfying the detailed constraints. This initial solution, when validated with the users, has been particularly useful to gain further confidence with the problem but was leaving open one important issue: the requirement from the users to explore the solution space optimizing according to different possible objectives. We formalized an objective function to minimize defined as follows:

$$f(S) = \alpha f_{sc}(S) + \beta f_{dw}(S) + \gamma f_{up}(S) + \epsilon f_{ta}(S) \quad (1)$$

where α, β, γ and ϵ are non negative constant real values.

1. $f_{sc}(S) = 1 - NP(S)/NP_{max}(S)$ where $NP(S)$ and $NP_{max}(S)$ are respectively the number of pericentre events associated to a science operation and the total number of pericentre events within the problem planning horizon $\mathcal{H}$. $f_{sc}(S)$ measures the fitness with respect to the science opportunity.
2. $f_{dw}(S) = 1 - DV(S)/DV_{max}(S)$ where $DV(S)$ and $DV_{max}(S)$ are respectively the volume of data which can be down-linked by the set of communication operations included in the solution and the maximum volume of data which can be downlinked within the given planning horizon $\mathcal{H}$. $f_{dw}(S)$ measures the fitness with respect to the downlink opportunity.
3. $f_{up}(S) = LA_{ud}/T_{ud}$ represents a measure of the uplink *smoothness*, that is a measure of the uniform distribution of the uplink operations over $\mathcal{H}$. Where LA_{ud} is the standard deviation of the set of distance values δ_k computed for each pair of subsequent pair of uplink operations.
4. $f_{ta}(S) = TA_{ud}/T_{ud}$ represents a measure of the "uplink tardiness", i.e., the violation degree of the maximum constraint imposed on each pair of communication operations with duration greater or equal to a given δ constant.

Given the multiobjective nature of the user optimization requirements and the need to generate robust solutions in a phase of the mission that can be modified in different directions we have considered the possibility to build an optimization procedure based on Genetic Algorithms (GA) a population-based optimization procedure inspired from the study of population genetics. A GA uses a *population* of possible solutions, which are subject to modifications aimed at the determination of the optimal solution. Every possible solution is encoded into a *chromosome* – a summarized representation of an individual or a solution – and positions in the chromosome are called genes. The value a gene takes is called an *allele* (or *allelic value*). Given an initial set of feasible solutions (the initial population), individuals are selected according to their fitness. The fitness of the N_p individuals is made explicit by means of a *fitness function* which is related to the *objective function* of the problem. After selection, individuals are randomly crossbred allowing the *recombination* of genetic material with probability p_c. The resulting individuals can then be *mutated* with a specific mutation probability p_m. The new population so obtained undergoes again a process of natural selection which favors the survival of the fittest individuals (the best solutions), and provides the basis for a new evolutionary cycle (this is iterated for N_g generations). The key idea of the integration of GA with the TRF representation relies on an intermediate algorithmic layer that is a simplified version of the initial greedy heuristics. Broadly speaking the key aspect of our hybrid solution relies on current solution encoding in terms of chromosomes that can be manipulated by a classical GA. Externally the GA environment uses classical operators for Selection, Recombination and Mutation. Then the chromosome is decoded and used is as seed of a sketchy real solution that is completed by the light version of the greedy algorithm which works on the TRF representation. In this way the GA leads the way for the optimization but the greedy part of the solver still maintains responsibility to create a ground complete solution that satisfies the set of problem constraints not natively described in the domain model. Such an encoding/decoding phase is another interesting original contribution of MrSPOCK.

Encoding and Decoding Chromosomes

Given a solution S for a pre-planning problem instance, it is *encoded* into a chromosome ch by just reporting the position over time of the science and maintenance operations. In particular, a solution S is encoded by a vector of integer values *0-1* of size $|E|$, where $|E|$ is the size of the set of reference events E. The sequence of allelic values (0 or 1) respectively represents the position of science or maintenance operations according to the position of the corresponding reference events $e_i \in E$. In particular, for each event representing an apocentre (pericentre) the value 1 indicates the allocation of a maintenance (science) operation around the event, the value 0 indicates a free event.

A chromosome ch is decoded into a solution S by a constraint-based procedure that exploits the TRF features. A procedure *DecodeChromosome* scans the problem horizon from left to right and generates the total ordered sequence of operations (the solution) $S = \{op_1, op_2, \ldots, op_{no}\}$ that are translated into a temporal occurrence of proper values for the pointing state variable. It takes as input an instance of the problem (uncontrollable state variables describing the temporal position of apocentres and pericentres and the ground station visibility windows), a chromosome ch and a reference operation op_0.

According to a reference events E – we include only apocentre and pericentre events – over the horizon $\mathcal{H} = [0, H]$ three different type of decisions de_i are considered:

1. Around apocentre events – in this case it possible to select between two type of operations: maintenance (`mn`) or communication `cm(j)` with a ground station j. Maintenance operations are decided according to the chromosome and they are considered as mandatory decisions. When no maintenance is decided a communication operation is selected if a ground station j is available on the basis of the so-called ground stations *de-overlapping* strategy sketched below;

2. Between an apocentre and a pericentre event – in this case only communication operations are possible, which can be joined to the last operation op_{k-1} inserted in the solution and are decided on the basis of the *de-overlapping* strategy;
3. Around pericentre events – in this case it possible to select between two type of operations: science (`sc`) or communication `cm(j)` with a ground station j. Science operations are decided according to chromosome, however this kind of decisions are not mandatory. In fact, a communication operation is selected when the decision of a science operation cannot generate a tardiness value between two consecutive communication operations with duration greater or equal to the given threshold δ.

The core of the procedure *DecodeChromosome* is a loop which iteratively select an operation op_k and post it into the current solution S, translating this decision into appropriate temporal occurrence of timeline values. The loop continues until there are no more operations possible. It is worth underscoring that the reference events considered in the chromosome simply act as temporal reference for some decisions to be taken (operation to be planned or not in those intervals) but the number of operations that are actually planned can vary in different produced plans.

Furthermore, a solution is completed by specifying the decisions concerning a *de-overlapping* strategy that assigns the communication operation with mode `cm(j)` and the selection of the related ground station j. Basically, the *de-overlapping* strategy fills the gaps between maintenance and science operations with communication operations. It takes into account both a set of temporal constraints and the objective function $f(S)$.

It is worth noting as our idea to use a chromosome with references to only two kind of operations – science and maintenance – is based on the observation that communication represent a kind of *default* operation for the satellite. In fact, when no maintenance or science operation is executed, communication is the only possible option. So, the chromosome implicitly influences when to perform communication operations.

MrSPOCK user-interaction front-end

In designing the interaction services for MrSPOCK we initially had available some basic services used as visualization functionalities for the OMPS system. They were mostly dedicated to support system developers in inspecting how part of the internal model are manipulated by the solving algorithm. Indeed a system developer is mostly interested in quite low level and internal details quite far away from the point of view of end-users. For this reason the best choice would be the one of designing an interaction from scratch dedicated to mission planners. Indeed the current version of the MrSPOCK interaction module somehow uses features from both these perspectives. Some of the features are completely new and dedicated to the problem, other features are adapted or evolved from those for a timeline based system. This is for two reasons: (a) because the temporal representation for the timeline is quite close to the way of taking decisions in the space domain; (b) because we had the additional goal of gaining the users' trust on the underlying approach as being not only general but also re-usable in other ESA missions. In this light we have dedicated attention to make the approach based on defining a domain model and in solving the problem by reasoning on this model more transparent and visible to the user. Somehow even if the user interface of MrSPOCK it is not also a suitable interface for the application developer, nevertheless it contains features that bring to forefront aspect of the underlying domain modeling and in general of the timeline based approach.

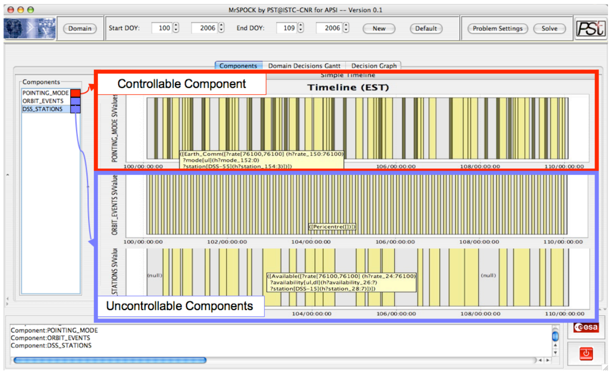

Figure 4: Basic layout

Main interaction features. The basic layout for MrSPOCK is shown in Figure 4. It is composed of a toolbar with the main commands to build instances of problem and to call and configure the solver, a message bar for the main dialogues, while the rest of the interface is mainly reserved to the timeline view. This central part describes both the uncontrollables (GS availability and Orbit Events) and the controllable (Pointing mode) components. In particular the Figure 4 shows the interface after a run of the solver and the pointing mode component presents a possible allocation of the main activities of the spacecraft (Science, Maintenance, Communication). The choice of centering the interaction on the concept of components which evolve over time allowed us taking advantage of the users' ability on reasoning over timelines to be completed and refined. Showing timelines, even in a preliminary version of the interface, resulted very useful to set up a context for the users and to facilitate our dialog with them since the early stages of the project. The preliminary version of the interface allowed us to easily check the validity of our model for the problem, bridging the gap between us and the users.

Our second step in the development of the interaction has been to select few focal concepts to meet users' expectations on the open problem, in particular we focused on: (a) the need to explore alternative solutions, (b) the ability to control some parameters to favor an optimization criteria or another, (c) the easy visualization of the solution.

Figure 5 presents a sketch of aspects that directly cope with these requirements. The main outcome of the GA run is gathered in a solution table that gives an immediate view of the fitness values specified according to the different contributors: Science and Downlink efficiency, Uplink Smoothness and Tardiness. We have given the user the possibility to act on the parameters that influence the different fitness and to inspect the effects of this manipulation on the single fitness component (same table). Additionally a graphical version of

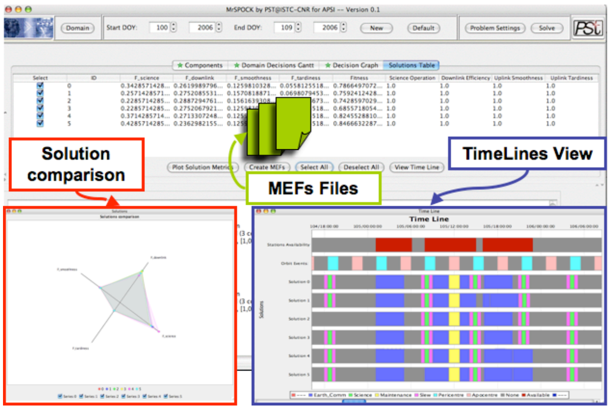

Figure 5: Examples of interaction for end users

the optimization values offer an alternative and cumulative view (left bottom of the figure) that allows to easily see the comparisons of alternative solutions.

The connection with the existing legacy of the mission planning at ESA has been preserved by providing the users with the possibility to generate the files containing all the activities for the spacecraft in the format required (MEFs file in figure) directly from the MrSPOCK environment.

Exploiting the central concept of the timeline shared between users and system developers, an additional graphic service has been built for the users which consists in the comparison of the pointing mode timelines corresponding to alternative solutions (see bottom right of the figure). This additional graphical view guaranteed a twofold beneficial effect. On the system developer side we were able to quickly check the validity of our solving approach since the overall view highlights features of the different solutions and consequently the solving choices. On the users' side they were able to compare and reason on their choices using this environment as a means to perform "what -if" analysis. Also in this case the choice of centering the interaction on the timeline comparison, appeared particularly successful. It is possible to speculate that in space domain the idea of taking decision over time is a quite "natural concept" which facilitate the choice of the main shared concept in term of what to show to the user at first glance.

A further aspect in the user interface is dedicated to the work done to show to the users the underlying domain model. This effort is motivated by the goal of showing the user aspects connected to the reusability of this technology within different contexts and space missions. Examples of this interaction the high level textual form domain description, an inspection of the single state variables, a graphical view of the automaton regulating the internal state transitions of the pointing mode component. This is somehow both irrelevant for the core application and also very simple but, together with other representations not shown here for lack of space, have obtained the effect of making explicit the generality of the underlying TRF representation module.

An experimental analysis

To offer a quantitative measure of the effectiveness of the current work, we present here an initial set of experimental results that evaluates the performance of both the basic greedy constructive heuristic and the genetic optimization algorithm. The greedy procedure is a one-pass constructive algorithm that uses a set of domain specific heuristics in a greedy left to right fashion. The difference between the greedy procedure and the chromosome decoding procedure employed for the genetic algorithm is that while the former performs forward checking sub-procedures, the latter is completely driven by the chromosome. The evaluation concerns the synthesis of a mission plan for a planning horizons of 10 days. The problem instances were generated on the basis of the real data of the ongoing interplanetary mission obtained from the ESA. All the proposed experiments run on a Pentium 4 processor at 3.4 Ghz, 1.5GB Ram, under Windows XP.

Experimental setup. First, we have defined three reference problems within the period from April 2006 to May 2006. Table 1 compares the fitness values obtained with a single run of the greedy procedure with the values obtained by optimization runs of the genetic algorithm decoding procedures (best and worst obtained values).

Problem	f_{gry}	f_{min}	f_{max}
P_1	0.77	0.63	0.88
P_2	0.71	0.6	0.78
P_3	0.82	0.46	0.85

Table 1: Performance of the greedy heuristic

As the table shows the greedy algorithm solution fitness (f_{gry}) is always between the worst (f_{max}) and the best (f_{min}) solution obtained with the genetic optimization. Although the greedy procedure is more computational expensive, we are comparing in the table one run of the greedy procedure with a multi-run genetic optimization process. But the best solution obtained after the optimization process proved to be always better than the one obtained with the greedy procedure. Two additional problems have been used for testing the optimization process, obtained by changing the parameters o_{min} and o_{max} for the maintenance operation (i.e., minimal and maximal allowed orbit distance between two maintenance operations), and the parameters T_{ud} (duration of non-preemptable uplink communication) and δ (distance between two uplink operations) for the communication operations. In particular, in Table 2 we indicate the parameters used within a problem P_i by using the following convention: $P_i(T_{ud}/\delta, o_{min}..o_{max})$ (allowed patterns for uplink communication are 2h every 12h or 4h each 24h). About the genetic algorithm we have chosen the following settings: each generation is composed of a number of $N_p = 20$ individuals, each computation run over a number of generation $N_g = 20$, the probability of recombination is set to $p_c = 0.8$ and the mutation probabilities is $p_m = 0.05$.

Problem	f_{start}	f_{end}	$\Delta^{gen}_{\%}$
$P_1(24/4, 3..5)$	0.71	0.63	11.2
$P_2(24/4, 3..5)$	0.6	0.50	16.6
$P_2^{(1)}(12/2, 2..6)$	0.82	0.69	16.6
$P_2^{(2)}(24/4, 2..6)$	0.6	0.53	11.6
$P_3(24/4, 3..5)$	0.52	0.46	11.5

Table 2: Performance of genetic optimization

Results. Table 2 shows the performance of the proposed genetic algorithm over the five problems P_1, P_2, $P_2^{(1)}$, $P_2^{(2)}$ and P_3 with the corresponding parameters indicated between brackets.[2] In particular, we consider the case where the coefficients α, β, γ and ϵ are set to 1. The two columns f_{start} and f_{end} are respectively the best solution found by the genetic algorithm at the first generation and the best solution found up to the last generation. $\Delta_{\%}^{gen}$ is the percentage improvement of the value f_{end} over f_{start}. Each run of the genetic algorithm takes approximately 1200 seconds. We observe a constant improvement of the genetic algorithm on the best solution obtained at the first generation. The genetic search is able to further improve over the initial iterative sampling performance. A last observation is about the performance on the two problems $P_2^{(1)}(12/2, 2..6)$ and $P_2^{(1)}(24/4, 2..6)$, in particular on the instance $P_2^{(1)}(12/2, 2..6)$. In such case, we decrease the absolute performance of the algorithm because we pay an higher price for the tardiness given the more strict constraint $T_{ud} = 12/\delta = 2$.

Conclusions

In this paper we have described an effort to apply a timeline based planning to an application domain. The resulting system, named MrSPOCK, is an example of rapid prototyping and easy user-interaction, designed to address a specific problem in the context of an ESA mission. To solve the Long Term Planning problem in MARS EXPRESS we have introduced a genetic approach on top of a quite detailed constraint-based representation. The hybrid algorithm presents a trade-off between the need to optimize and the need for a strong control of the different constraints in the problem.

We have proposed a hybridization between a temporal planner and the meta-heuristic genetic procedure with a distinction of roles between the two. The genetic algorithm works on the chromosome which reasons on min/max distances between maintenance and the science allocations over time. The temporal planner does the temporal reasoning on the detailed plan and is able to complete the partial plan produced by the chromosome evolution. It is also worth saying that the final solution is a detailed temporal structure that can be annotated, explored, queried and modified by an end user though an interaction module designed on purpose. Two releases of the planning system for the optimization of pre-plans have been delivered to the users. The problem features have been captured correctly. Performances are in line with the expectations of the users. We are now in a phase of fine tuning of the performance of the whole system to improve the quality and capture some additional detailed aspects. This back-to-back phase with the users shows how the iterative prototyping approach has been functional for the success of the application deployment into the operational environment.

It is worth saying that within the same ESA project a further application is under development, which uses the same TRF functionalities. Preliminary results (Verfaillie and Pralet 2008) show how the rapid prototyping on top of the TRF allows again to quickly and correctly capture the problem as well as to find specialized solutions.

[2] We observe that the two problems $P_2^{(1)}$ and $P_2^{(2)}$ coincide with P_2 except in the values of the parameters o_{min} , o_{max}, T_{ud} and δ.

Acknowledgments. Authors are partially supported by European Space Agency (ESA) within the Advanced Planning and Scheduling Initiative (APSI). APSI partners are VEGA GmbH, ONERA, University of Milan and ISTC-CNR. Thanks to Federico Pecora and Riccardo Rasconi for interactions and joint work on several planning and scheduling topics. Thanks to the SPARK reviewers whose comments showed us problems in a previous version of this paper.

References

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

Bensana, E.; Lemaitre, M.; and Verfaillie, G. 1999. Benchmark problems: Earth observation satellite management. *Constraints* 4(3):293–299.

Cesta, A., and Oddi, A. 1996. DDL.1: A Formal Description of a Constraint Representation Language for Physical Domains. In M.Ghallab, M., and Milani, A., eds., *New Directions in AI Planning*. IOS Press.

Cesta, A.; Cortellessa, G.; Denis, M.; Donati, A.; Fratini, S.; Oddi, A.; Policella, N.; Rabenau, E.; and Schulster, J. 2007. MEXAR2: AI Solves Mission Planner Problems. *IEEE Intelligent Systems* 22(4):12–19.

Cesta, A.; Oddi, A.; and Smith, S. F. 2002. A Constraint-based method for Project Scheduling with Time Windows. *Journal of Heuristics* 8(1):109–136.

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 *SpaceOps 00. Proceedings of the International Conference on Space Operations*.

Dechter, R.; Meiri, I.; and Pearl, J. 1991. Temporal constraint networks. *Artificial Intelligence* 49(1-3):61–95.

Frank, J., and Jónsson, A. 2003. Constraint-based attribute and interval planning. *Constraints* 8(4):339–364.

Fratini, S.; Pecora, F.; and Cesta, A. 2008. Unifying Planning and Scheduling as Timelines in a Component-Based Perspective. *Archives of Control Sciences* 18(2):5–45.

Ingrand, F. F.; Georgeff, M. P.; and Rao, A. S. 1992. An architecture for real-time reasoning and system control. *IEEE Expert* 7(6):33–44.

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

Knight, R.; Rabideau, G.; Chien, S.; Engelhardt, B.; and Sherwood, R. 2001. Casper: Space Exploration through Continuous Planning. *IEEE Intelligent Systems* 16(5):70–75.

Muscettola, N.; Nayak, P.; Pell, B.; and Williams, B. 1998. Remote Agent: To Boldly Go Where No AI System Has Gone Before. *Artificial Intelligence* 103(1-2):5–48.

Vere, S. A. 1983. Planning in Time: Windows and Durations for Activities and Goals. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 5(3):246–276.

Verfaillie, G., and Pralet, C. 2008. Modeling and solving the INTEGRAL mission long-term planning problem. Project presentation at ESA-ESOC, July 23, 2008.