Autonomous Energy Management as a High Level Reasoning for Planetary Rover Problems*

D. Díaz
Universidad de Alcala
Alcala de Henares, Madrid (Spain)

A. Cesta and **A. Oddi** and **R. Rasconi**
CNR – Italian National Research Council
ISTC, Rome (Italy)

M.D. R-Moreno
Universidad de Alcala
Alcala de Henares, Madrid (Spain)

Abstract

This paper presents recent results on applying advanced autonomous reasoning capabilities for a planetary rover concept for synthesizing complete command plans that involve a wide assortment of mission requirements. Our solution exploits AI scheduling techniques to manage complex temporal and resource constraints within an integrated power-aware decision-making strategy. The main contribution of this work is threefold: (i) we propose a model of the world inspired by the Mars Sample Return (MSR) mission concept, a long-range planetary exploration scenario; (ii) we introduce a MSR-inspired scheduling problem called Power Aware Resource Constrained Mars Rover Scheduling (PARC-MRS), and we present an extension of a well-known constraint-based, resource-driven reasoner that returns rover activity plans as solutions of the PARC-MRS; finally, (iii) we conduct an exhaustive experimentation to report the quality of the generated solutions according to both feasibility and makespan optimization criteria.

Introduction

The forthcoming planetary exploration scene will call for ambitious robotic missions. Increasing the level of autonomy in those missions inevitably entails *entrusting* the rovers with higher level responsibilities, such as the *synthesis of complete mission plans* from high-level goal descriptions, *plan adaptation/modification* to address contingent situations, and even the possibility of performing opportunistic science and hazard prediction (Estlin et al. 2007).

In this work, the Mars Sample Return (MSR) mission concept (Treiman et al. 2009) is proposed as a plausible and efficient paradigm-shift to continue the exploration of the Red Planet in the near future. Roughly speaking, the MSR mission consists of placing a rover on Mars' surface, gathering scientific samples from a set of scattered and challenging sites (up to many kilometers from the landing site) within relatively short time frames, and transporting them to a specific location where an ascent vehicle will be in charge of initiating the return trip. The proposed model encapsulates a wide range of interesting features which makes it particularly challenging, as it involves: first, global path-planning, focused on "long-range navigation" planning in contrast to the classical path-planning research which addresses "local navigation" to trace safe routes between pairs of locations separated a few meters apart of each other. Second, resource management, by analyzing the *energy* production/consumption profiles of all the plan activities. Third, a wide assortment of temporal constraints, such as absolute deadlines on the experiment execution (e.g., to communicate critical experimental results via orbiting relays), or rover inactivity periods (e.g., nights or solar storms) represented as static synchronization events of finite duration.

To this aim, we introduce a MSR-inspired scheduling problem called *Power Aware Resource Constrained Mars Rover Scheduling* (PARC-MRS), and present an extension of a well-known constraint-based, resource-driven reasoner that returns rover activity plans as solutions of the PARC-MRS. Our solving process exploits advanced Artificial Intelligence (AI) P&S constraint-based, resource reasoning techniques, in particular "Precedence Constraint Posting" (PCP) (Oddi and Smith 1997; Cesta, Oddi, and Smith 2002) to reason upon a detailed model of the PARC-MRS problem instances. One of the contributions of this work is the exploitation of a known methodology to represent *consumable* resources by means of a cumulative scheme (Simonis and Cornelissens 1995), to model and solve the proposed scheduling problem. The remainder of the paper is structured as follows: we start with a detailed description of the mission scenario of reference. Next, we provide a detailed description of the extended constraint-based, resource driven reasoner with integrated power-aware decision capabilities. Following that, we conduct an exhaustive experimentation to evaluate the efficiency of our solution algorithm. Finally, a conclusion and future work section closes the paper.

The Mars Sample Return mission scenario

In this section we provide a definition of the Power-Aware Resource Constrained Mars Rover Scheduling (PARC-MRS) problem, that is grounded on the commitment to the Mars Sample Return (MSR) (Treiman et al. 2009) *reliability* and *efficiency* baseline requirements: the first requirement refers to the need to synthesize plans capable of partially absorbing the effects of possible exogenous events arising during the plan's execution, while the second refers to the goal of minimizing the plan's completion time, thus maximizing the overall science return.

The attainment of the mission's goals requires the use of the rover's set of instruments/resources whose utilization must be synchronized over time in order to guarantee the

*An extended version of this paper will be published in the IEEE Computational Intelligence Magazine - November 2013

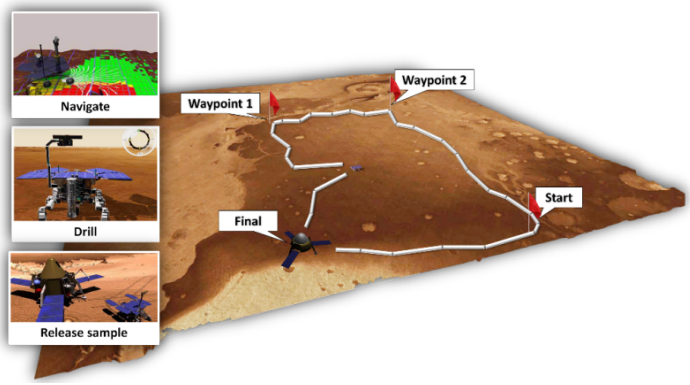

Figure 1: Mission scenario overview: (1) navigation, (2) acquiring science (drill) and (3) sample release activities

correct execution of the plan's activities. Each rover activity a_i requires a specific amount of one or more resources during its entire execution.

Specifically, the soil extraction operations require a *science acquisition asset* and a *sample cache* (SC_r), which basically consist of a drilling subsystem and a container with a capability to store and transport up to *C* standard-sized samples, respectively.

Navigation tasks demand a *Locomotion, Guidance, Navigation and Control (Loco & GNC) subsystem*, which provides all the functionalities that allow the rover to reliably reach a desired target. The rover energy supply is provided by a *powering subsystem* consisting of a combination of Solar Array (SA_r) as a primary power source, and a *Battery* (B_r). The battery is characterized by a maximum capacity or *saturation level* B_{max} (in Watts-hour) and by a *minimum usage threshold* B_{min} (expressed as a percentage of the maximum capacity), representing the minimum battery power level that can be reached, for safety reasons. During nominal rover operations, the power generated by the solar panels is sufficient to propel the rover and charge the batteries in the day time, while during the night the rover suspends every activity. The battery is required to sustain the execution of the soil extraction activities as well as to maintain the minimum operating temperature of the rover system during the night, but under certain conditions, it can also contribute with additional power for locomotion operations.

More formally, the PARC-MRS problem entails the synchronization of a set of resources $R = \{r_1, r_2, \ldots, r_m\}$ to perform a set of n rover activities over time $A = \{a_1, a_2, \ldots, a_n\}$. The set of activities is organized along a set of ne experiments (or job sequences) $Exp = \{Exp_1, \ldots, Exp_{ne}\}$. More concretely, the complete execution of the i-th experiment Exp_i is modeled as a tuple composed of the following ordered activities:

$$Exp_i = \langle Nav_{S,i}, Drill_i, Nav_{i,F}, Rel_i \rangle \quad (1)$$

Figure 1 illustrates the PARC-MRS problem scenario, as well as the basic activities that are executed in a typical MSR mission: the $Nav_{i,j}$ activities represent the long-range traversals for *science acquisition* or *sample delivery* between two different locations i and j (the initial location of the rover and the location of the ascent vehicle are denoted with *S* and *F*, respectively); the $Drill_i$ activities represent the deployment of the onboard science collection system (i.e., a drilling instrument) to retrieve and store a soil sample situated at the location or way-point i; finally, the Rel_i activities represent the releasing of the sample (collected at the waypoint i) at the final location, where the Mars ascent vehicle is in charge of uploading it into orbit. The ascent vehicle is equipped with a *robotic arm* (A_r) that is used to recover the soil samples collected by the rover. Every rover activity undergo complete suspension periods during the nights, which can have different durations depending on the Martian season.

A *feasible* solution $S = \{st_1, st_2, \ldots, st_n\}$ is an assignment to the start times st_i of the activities $a_i \in A$ imposing a total order among all the activities activities $a_k \in \{Drill_i, Rel_i : i = 1, 2, \ldots, n\} \subset A$, and satisfying the following set of constraints.

- *Temporal Constraints* - S is consistent with the partial ordering imposed in each sequence Exp_i. Pairs of consecutive activities in each sequence Exp_i are supposed to be contiguous, i.e., in every sequence, the end time of each previous activity coincides with the start time of the following activity. The durations of the $Drill_i$ and Rel_i activities are lower bounded by the time required to complete the science extraction and the release operations, respectively. The minimum duration of the $Nav_{i,j}$ activities depends on the nominal *traversal time* (tt_{ij}) required to travel the distance between the pair of i, j waypoints. In this work, waypoint-to-waypoint paths are considered as sequences of straight, traversable segments computed during the mission preparation phase. Finally, the completion time of some of the Exp_i sequences might be constrained by an absolute deadlines d_i.

- *Sample Cache Constraints* - the number of samples contained at all times in the cache SC_r cannot exceed the rover's maximum sample capacity capacity *C*.

- *Energy Constraints* - in our model, the execution of the power demanding activities (i.e., drilling operations and navigation) requires a certain amount of energy that has to be completely available at the beginning of the activity.

  $Nav_{i,j}$ activities demand a variable amount of energy e_{ij}, which depends on the traveling distance between the two different locations i and j, while the $Drill_i$ activities demand a constant amount of power e_i necessary to operate the drill subsystem.

  The rover *powering subsystem* imposes an additional global constraint on the set of activities A. In particular, the global production/consumption battery usage profile $B(t)$ is computed according to the hypothesis that the onboard rover solar arrays produce a continuum of energy at a monotonic rate σ_{charge} (Watts). The generated power is directly used to both propel the rover and charge the battery up to the *saturation* level B_{max}. The surplus energy, if any, is discarded as the battery cannot be charged in excess of B_{max} (saturation). As the activities a_i consume the energy instantaneously at their start times st_i, we can consider the assessment of the usage profile $B(t)$ only for $t = st_i$, with $i = 1, 2, \ldots, n$.

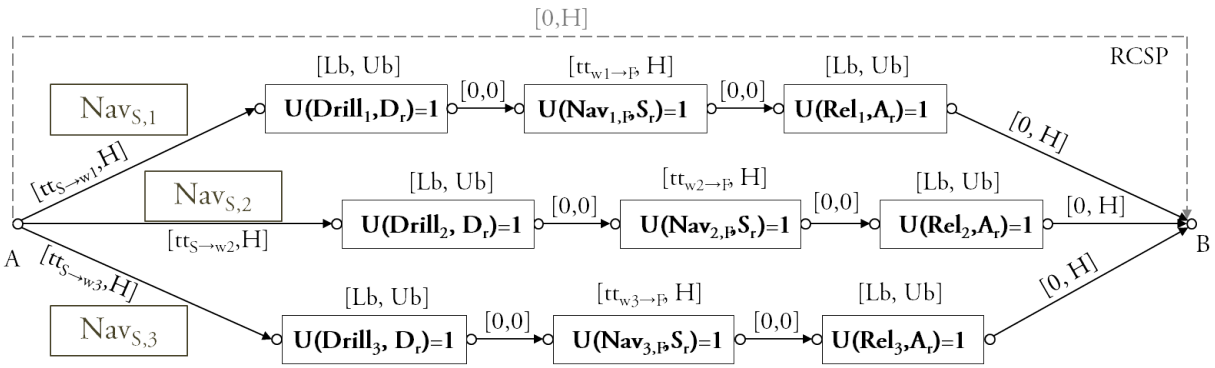

Figure 2: Activity-on-the-node graph representation of the problem model: the edges represent the precedence constraints, while the nodes (boxes) represent the activities; the resource usage information is shown within each box

Finally, an *optimal solution* S^* is a feasible solution where the plan's completion time, defined as the highest end time among of all the plan's activities, is minimized.

The constraint-based solving algorithm

In this section we present in detail the profile-based, power-aware reasoning algorithm $ESTA^p$ developed in this work to provide feasible solutions for the PARC-MRS problem, as well as a meta-heuristic strategy for solution optimization.

The constraint-based PARC-MRS problem representation

We formulate the PARC-MSR scheduling problem in terms of Constraint Satisfaction Problem (CSP) (Montanari 1974). In our CSP-based formulation of the PARC-MSR a set of decision variables called *Minimal Critical Sets (MCSs)* are identified; An MCS is defined as a set of activities that simultaneously require a resource r_k with a combined capacity requirement greater than the resource's total capacity, such that the combined requirement of any subset is less than or equal to the resource capacity. From the definition of an MCS, it follows that the posting of a single precedence between some pair of activities in the MCS is sufficient to eliminate the conflict. Each MCS variable is associated with a domain of feasible values corresponding to the set of precedence constraints that can be posted to resolve the MCS (i.e., the possible orderings allowed between any pair of activities belonging to the same MCS).

Two different types of solving separation constraints are considered: *simple precedence* and *traversal time* constraints, denoted respectively as $a_i \prec a_j$ and $a_i \prec_{val} a_j$, where val is the minimum separation value that must hold between a_i and a_j. Traversal time constraints are posted between drilling and/or sample release activity pairs in order to properly model the traveling times among the different locations, while simple precedence constraints are used in all the other cases.

To support the search for a consistent assignment to the set of MCS variables, for any PARC-MRS problem instance we can define a *temporal constraint network* which maps the temporal constraints in the problem to distance constraints between appropriate time-points (i.e., the activity start times and/or end times); such temporal constraint network corresponds to the so-called Simple Temporal Problem (STP) (Dechter, Meiri, and Pearl 1991), and is formulated as a CSP (*ground-CSP*). Thus, our PARC-MSR formulation can be seen as a *meta-CSP* formulation, which utilizes the ground-CSP representation for the underlying temporal reasoning on top of which a second CSP problem is formulated that enables resource constraint reasoning.

Figure 2 presents an activity-on-the-node graph representation of the PARC-MRS problem considered here. In the graph, the nodes represent the problem activities, each characterized by (i) a pair of time points (indicating the starting and end times), (ii) a resource demand, where $U(a_i, r_k)$ represents the amount of resource r_k required by the activity a_i, and (iii) a specific (flexible) duration, i.e., expressed as a temporal interval $[lb, ub]$; the edges correspond to the precedence relation constraints between the activities, again expressed as temporal intervals. The graph contains two special time points (A and B) indicating the schedule's time *origin* and *horizon*, respectively.

The integrated power-aware, resource driven $ESTA^p$ solver

The proposed $ESTA^p$ procedure for solving instances of the PARC-MSR problem is based on the *precedence constraint posting* (PCP) approach (Smith and Cheng 1993; Cheng and Smith 1994), that consists of deciding and posting a set of temporal precedence constraints that eliminates all the resource contentions. Basically, $ESTA^p$ is a modified version of the basic profile-based schema of $ESTA$ (Cesta, Oddi, and Smith 2002) which provides *cumulative-based* resource reasoning by "iteratively leveling contention peaks" through the exploitation of a new set of dominance conditions introduced in (Oddi et al. 2011) that allow us to take into account both the simple and setup time precedence constraints within the general problem-solving strategy.

Algorithm 1 shows $ESTA^p$'s resolution process in detail. The algorithm receives as input a description of the scheduling problem according to the constraint-based specification introduced in the previous section, and iteratively performs a solving sequence composed of three steps: (i) checking the temporal consistency of the current partial solution; (ii)

Algorithm 1: $ESTA^p$ algorithm.

```
   Input: Problem, Horizon
   Output: FeasibleSolution, EmptySolution
1  <meta-CSP, ground-CSP > ← CreateCSP (Problem)
2  loop
3      if CheckConsistency (ground-CSP) then
4          // Earliest Start-time Solution extraction
5          ESS ← ExtractESS (ground-CSP)
6          // Resource profiling
7          ComputeResourceUsages (ESS)
8          // Resource contention peaks levelling
9          meta-CSP ← ComputeMCSs (ground-CSP)
10         if ConflictFree (meta-CSP) then
11             FeasibleSolution ← ExtractSolution
               (ground-CSP)
12             Return (FeasibleSolution)
13         else
14             if Unsolvable (meta-CSP) then
15                 Return (EmptySolution)
16             else
17                 MCS ← SelectMCS (meta-CSP)
18                 PrecedenceConstraint ←
                   SelectPrecedence (MCS)
19                 ground-CSP ← PostConstraint
                   (ground-CSP, PrecedenceConstraint)
20     else
21         Return (EmptySolution)
22 end-loop
```

estimating all the resources utilization throughout the current solution, i.e., by profiling the sample cache, rover and battery resources; (iii) identifying and resolving all of the resource conflicts possibly existing in the current solution. In the following sections we provide a detailed description of each of these steps.

Step 1: constraint propagation & temporal consistency checking. Within this step, the temporal constraint network (ground-CSP) underlying the problem is checked for consistency by the function `CheckConsistency(ground-CSP)` (line 3 of Algorithm 1). If the ground-CSP is found to be inconsistent, the procedure exists immediately.

Step 2: Resource usage profiles computation. In this step (lines 4-7), the algorithm extracts a solution and performs an estimation of all the resource usage profiles. At line 5, an Earliest Start-time Schedule[1] (ESS) is extracted from the partial CSP schedule solution (`extractESS(GroundCSP)` function), while at line 7 the `ComputeResourceUsages(ESS)` function returns all the resource utilization profiles on the basis of the ESS solution.

In this work, in order to reduce the consumable behavior of the battery to the cumulative scheme used by ESTA, we exploit a modified version of the model introduced by Simonis[2] (Simonis and Cornelissens 1995). Below, we describe how the estimation of the overall power respectively consumed and produced by all the activities of the schedule is computed according to our adaptation of the Simonis model.

Energy consumption profile. Figure 3 (left) illustrates an example of how energy consumptions are modeled in our framework, by introducing as many *battery consuming activities*, or *energy consumers* ($Cons_1$ and $Cons_2$) as the plan's activities that require energy (A_1 and A_2, in the figure). As shown in the figure, the energy consumers' end times are constrained to coincide with the horizon time point (i.e., the end of the schedule), while their start times are constrained to match with the start times of A_1 and A_2 (i.e., to the instants at which a specific amount of energy is required), thereby expressing the fact that each amount of energy required by a task is lost forever (unless replenished by a producer task), hence modeling the typical renewable resource behavior.

Energy production profile. The computation of the energy production profile follows a logic which is directly exemplified by Figure 3 (right). As a consequence of adopting the Simonis' model, the continuous charging rate curve (i.e., the σ_{charge} rate charging profile) is approximated by means of a sequence of small, discrete chunks of energy producers (i.e., the $Prod_i$ activities, in the figure) distributed along the complete horizon. The result is a piecewise constant representation of the energy production profile; each chunk of energy is modeled as a time-fixed activity which produces an amount of energy equal to the *nominal* quantity of power collected during the related piecewise segment minus the energy possibly lost because of saturation during the same segment. Each energy producer activity starts at the beginning of the schedule (i.e., the origin time point), and terminates at the instant at which the battery is charged with the associated energy chunk (i.e., the energy chunk is released).

Figure 4 presents an MSR problem instance composed of two job sequences (top) together with the set of the relative energy consuming activities (four consumers for each sequence), while at the bottom of the figure, the resulting overall battery usage profile is drawn (energy consumptions are depicted as red down arrows, while energy productions are depicted in green). As shown in the figure, the relation between the rover activities and their related consumers is as follows: the first consumers (i.e., those starting at t_0) refer to the consumption of the initial traversals to be performed before reaching the drill locations; the second consumer refer to the soil extraction operations ($Drill_i$); the third consumers refer to the navigation activities between the soil extraction and final location; and the last consumers of each job (i.e., those attached to the end of the Rel_i activities) are introduced to model the consumption of further possible movements starting from the final location. The energy

[1]ESS is a consistent temporal assignment where all the time points are assigned with the lower bound values of their respective feasibility intervals.

[2]The Simonis' model was originally proposed to model consumable resources following a *classical cumulative scheme*. The reference model basically copes with *stock-based* consumable resources (such as a fuel tank or a storage warehouse) in flow shop or job shop application contexts.

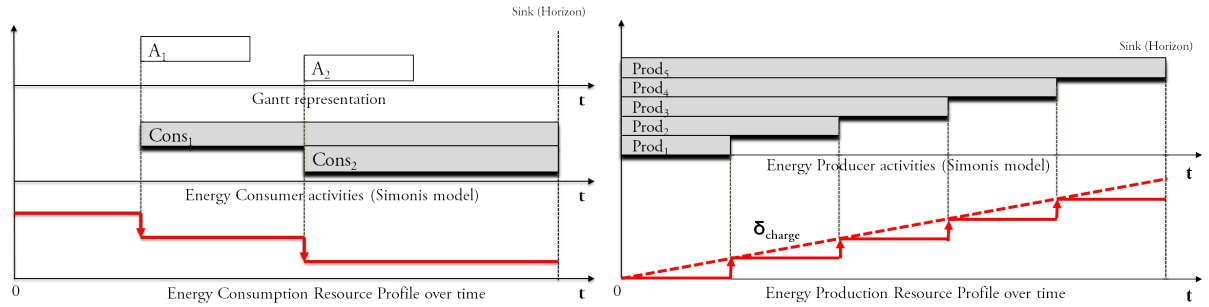

Figure 3: Energy consumption (left) and production (right) constraints representation

consumption profile is ultimately computed as the sum of all the power demands (depicted as downward red arrows) on behalf of all the consumer activities across the whole schedule's makespan.

Step 3: Resource contention peaks leveling. This step (lines 8-19 in Algorithm 1) constitutes the most important part of the solving process, as it deals with the identification and the resolution of the next resource conflict on the basis of a specific heuristic rationale. This decision process is known as "resource contention peak leveling", since it consists of (i) identifying the resource over-consumptions and (ii) *flattening* them through the imposition of new precedence constraints which temporally separate the execution of the contending activities, according to the steps below.

Resource conflict detection. Firstly, a resource usage analysis is performed with the aim of determining all possible resource capacity violations (i.e., the resource contention peaks), by identifying the sets of activities that are executed concurrently (on the basis of the ESS projection) and that cause resource over-consumptions by globally requiring a resource in excess of its maximum capacity.

More concretely, a meta-CSP is computed by extracting a set of *Minimal Critical Sets (MCSs)* from each resource contention peak (`ComputeMCS(meta-CSP)` function, line 9). For example, contention peaks occurring on the battery resource are detected for all instants t_i where the overlapping of (at least) one production and one consumption activity causes the total battery capacity to fall below the B_{min} value. Figure 4 shows an example of battery contention peak spanning over a temporal interval (denoted as *critical segment*) during which the battery usage profile remains below the threshold energy level B_{min} (i.e., battery is overconsumed).

Resource conflict resolution. Subsequently, the function `SelectMCS(meta-CSP)` (line 17) is invoked to return the next MCS. Such MCS is chosen according to the *most constrained variable ordering* heuristic, so as to select the MCS candidate (the decision variable) characterized by the *smallest temporal flexibility*, i.e., a function of the degree to which constituent activities can be reciprocally shifted in time, the idea being that the less flexibility a MCS has, the more critical it is to resolve that first. Once the MCS is selected, the function `SelectPrecedence(MCS)` (line 18) is in charge of (i) selecting a pair of activities from the MCS, and (ii) deciding their relative separation ordering for MCS resolution. This decision is made following the *least constraining value ordering* heuristic guideline: the greater the flexibility is retained after inducing a precedence ordering constraint, the more desirable it is to post that constraint.

The three steps previously discussed are iterated until (i) a conflict-free solution schedule (i.e., temporal and resource feasible) is found, or (ii) a temporal inconsistency is detected. In the second case, the algorithm stops as it has reached a dead-end situation.

Providing better solutions

Both the feasibility and optimization version of the scheduling problem here addressed is NP-hard, and therefore cannot be solved in reasonable time by using systematic, non-informed search techniques. As previously stated, the solutions provided by the $ESTA^p$ algorithm are generally far from being optimal as the $ESTA^p$ procedure is only concerned with providing feasible solution schedules. Therefore, we embedded our $ESTA^p$ algorithm within an iterative sampling optimization loop, similarly to the approach used in the Iterative Sampling Earliest Solutions (ISES) (Cesta, Oddi, and Smith 2002) strategy for makespan minimization, an efficient *multi-pass approach* which performs quite well in the face of scheduling problems involving very large search spaces. More concretely, ISES is a stochastic procedure that controllably broaden the exploration of the search space without incurring in the exponential cost of classical backtracking strategies, by iterating a non-deterministic version of the $ESTA^p$'s conflict selection heuristic (called $ESTA^p_{rand}$) across solutions characterized by increasingly smaller temporal horizons.

The solution we employ in this work is a simplified version of the ISES procedure (still referred to as ISES, for simplicity reasons), and is illustrated in Algorithm 2. The procedure receives as inputs (i) a scheduling problem specification, (ii) an initial "sufficiently large" horizon value ($MaxH$), and (iii) two additional parameters to control the stop conditions, i.e., the maximum CPU time allotted for optimization ($MaxTime$), and the maximum number of permitted iterations without getting any improvement ($MaxAttempts$). Our ISES version works according to the two following basic steps: (i) an initial and deterministic invocation of $ESTA^p$ with the horizon value $MaxH$ (line 1), in order to find the first feasible solution, and (ii) the execu-

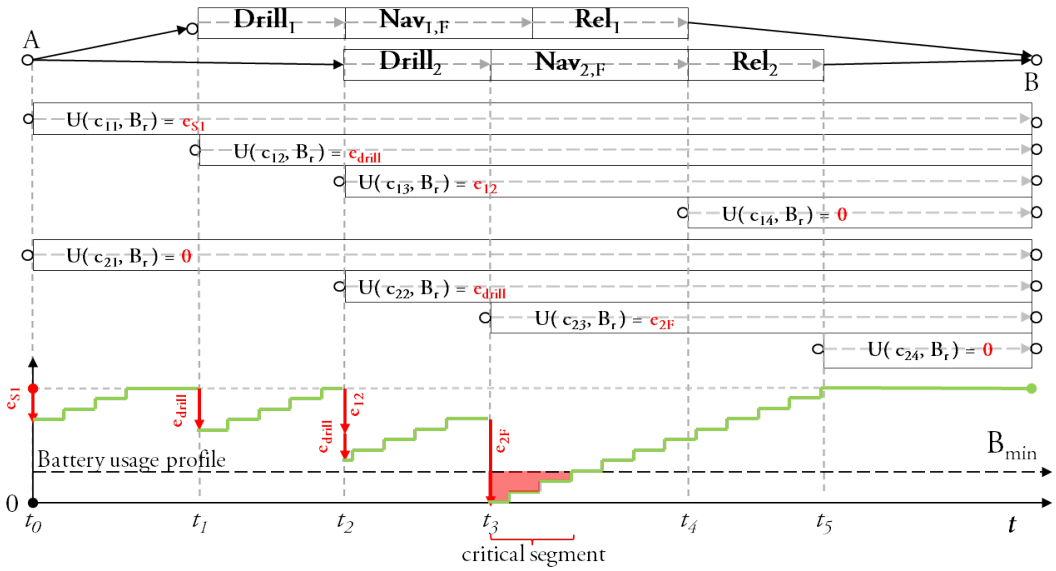

Figure 4: Example of energy profile computation: the consumption components of the profile (e_{ij}) are depicted in red, while the production components are depicted in green

Algorithm 2: The iterative sampling search framework (ISES) for solution optimization.

Input: Problem, MaxH, MaxTime, MaxAttempts
Output: S_{best}

1 $S_{best} \leftarrow ESTA^p$ (MaxH)
2 **while** ($\neg$ `StopCondition` (MaxTime, MaxAttempts)) **do**
3 Sol $\leftarrow ESTA^p_{rand}$ (`Mk` (S_{best}))
4 **if** (`Mk` (Sol)<`Mk` (S_{best})) **then**
5 $S_{best} \leftarrow$ Sol

tion of an optimization loop, in the shape of successive calls to $ESTA^p_{rand}$ where at each iteration the temporal horizon is reduced to the best solution makespan $\texttt{Mk}(S_{best})$ found so far (line 3), thus forcing the algorithm towards solutions of increasingly smaller makespans. The algorithm returns the best solution encountered when either of the two stop conditions previously described is met.

Experimental analysis

In this section we conduct an experimentation analysis which aims at assessing both: (a) the efficiency of our solving algorithm $ESTA^p$; and (b) the efficacy of an optimization framework based on the ISES strategy.

The MSR benchmark problem sets

Due to the lack of comparative scheduling problem instances of reference in literature which suitably match the characteristics of our MSR problem description, we decided to generate a problem library by using our own benchmark instance generator MSR/Gen[3] for the class of problems here referred to as PARC-MRS.

The benchmark library used in this work has been instantiated by using seed templates whose baseline parameters were carefully selected from specifications characterizing recent real-world rover-based missions. More concretely, we have based our benchmark production on one of the rover models contained within the ESA's 3DROV (Poulakis et al. 2008) simulator, an advanced planetary robot design, visualization, and validation tool. The 3DROV's rover model actually represents a prototype of one of the possible configurations of the ExoMars rover, a planned Mars mission to search for possible biosignatures of Martian life (van Winnendael, Baglioni, and Vago 2005).

The MSR benchmark library used in the experimental phase of this work consists of three different benchmark sets (containing 40 problem instances each) where each set is composed of instances respectively characterized by 20, 25 and 30 experiment sequences (referred to as MSR_{40}–20, MSR_{40}–25 and MSR_{40}–30, respectively)[4].

Experimental results

The empirical analysis has been organized in two different parts, relatively to the feasibility and the optimization assessments respectively. The former analysis conveys the results related to the execution of the deterministic $ESTA^p$ algorithm on the computation of a feasible schedule solution, while the second analysis focuses on the outcome produced by the optimization framework (ISES) on the attempt

[3]The MSR/Gen Java code can be downloaded from the following link: *atc1.aut.uah.es/~mdolores/PARC_MSR*

[4]The complete MSR benchmark library, as well as a self-contained description of the specific format of each problem instance and seed templates can be downloaded at the following link: *atc1.aut.uah.es/~mdolores/PARC_MSR*

to improve the results obtained from the feasibility analysis. In either case, we solved the benchmark instance sets previously presented under three different environmental conditions, depending on the particular period of the year the mission takes place, i.e., *summer, winter and mid-season*. The idea is to study the performances of the solar arrays and battery-powered rover relatively to the problem at hand, under different conditions of available daylight. More concretely, a martian day is slightly longer than 24 terrestrial hours (here considered exactly 24 hours for simplicity reasons) and, depending on the particular season and latitude of the rover's area of operations, daytime periods might vary from approximately 16 hours (i.e., the nights lasting 8 hours) to 8 hours (i.e., the nights lasting 16 hours). In our study, we also considered a "mid-season" situation where each day is equally divided in 12 hours of daylight and 12 hours of night. In the model, we use the simplifying assumption that the day/night transitions occur instantaneously. Regardless of the season, feasible solution plans must guarantee the rover's capability to retain the energy required to keep the rover subsystems sufficiently warm during the night inactivity cycles. It should be noted that for obvious reasons, such heating power can only be supplied by the onboard battery.

As explained in the previous section, 3 different benchmark sets are used in the experimental campaign, labeled in agreement with the notation MSR_{40}–x–yh, where x denotes the number of jobs of each instance of the set ($x \in 20, 25, 30$), and y refers to the duration, expressed in hours, of the night periods ($y \in 8, 12, 16$, referring to summer, mid-season and winter light conditions, respectively). Every problem instance contains four experiment sequences characterized by deadline constraints (therefore defined *critical*), two of which are forced to be executed at some random instant before the 10th day of mission, while the other two are forced to be completed before the 20th day of mission.

Table 1 collects the results of both the feasibility (feasibility assessment section) and optimization assessment (optimization assessment section), for each previous benchmark set. All the reported figures are computed by averaging the data obtained from the 40 instances belonging to every set. The results shown in each column of the table have the following meaning:

- $Mksp^{avg}(mins)$ is the average solution makespan length (expressed in minutes).
- $CPU^{avg}(secs)$ is the average CPU computation time (expressed in seconds).
- $Cache^{avg}(\%)$ represents the rover's average sample cache usage (expressed in percentage[5]) along the whole plan's horizon.
- $Bat^{avg}(\%)$ is the battery resource usage (expressed in percentage[5]) along the whole plan's horizon.
- $\Delta^{avg}_{LWU}(\%)$ conveys the average improvement ratio (expressed in percentage[6]) between the makespan lengths related to the initial and optimized solutions, respectively.
- $\#Iter(avg)$ is the average number of iterations performed by ISES while attempting at improving the initial solution within an estimated maximum time window of 10 minutes (or after 200 consecutive attempts if no makespan improvement is obtained).

A maximum CPU time of 10 minutes has been allotted for each optimization run. In both assessments we considered an initial mission horizon of 138 Sols (Martian days). Finally, the current experimentation has been executed on an Intel(R) Core(TM)2 Quad CPU Q8200 @2.33Ghz machine, with 4Gb RAM.

From the observation of the obtained results, we can infer the following conclusions. Relatively to the results returned by $ESTA^p$, it can be observed that for all the three benchmark sets, the average makespans ($Mksp^{avg}(mins)$ column, feasibility assessment) follow an increasing trend with the shortening of the daylight periods, thus confirming our expectations about the significant impact of the seasonal conditions on the solution quality (the results show that in some cases the plan's duration can be as much as doubled). Still relatively to the makespan, we can appreciate the significant improvement rates provided by $ISES$ ($Mksp^{avg}(mins)$ column, optimization assessment), ranging from a 35.6% improvement for the MSR_{40}–20–$8h$ instances, to a 8.5% improvement for the MSR_{40}–30–$16h$ instances. It should be however observed that, in the latter case, only an average of ≈ 3 optimization iterations have been possible within the allotted time of 10 minutes (see $\#Iter(avg)$ column).

Still relatively to the makespan improvement averages, it can be observed that the deteriorating seasonal lighting conditions severely affect the optimization quality ($\Delta^{avg}_{LWU}(\%)$ column), as the room for "compacting" the plan's activities decreases for reasons related to both the augmented rover periods of quiescence, and the higher amount of battery power that must be charged before the rover goes off-duty. In fact, this power (which might otherwise be used to perform a number of pre-dusk activities that have to be inevitably postponed to the following day) must be saved to guarantee the equipment's proper heating during the longer nights.

With regards to the average battery power utilization (see both $Bat^{avg}(\%)$ columns), we observed a rather regular trend which confirmed that the shorter the martian days' duration, the higher is the battery average power demand. While this result may seem quite straightforward (e.g., more battery power is required to safely "survive" the longer nights), the fact that approximately the same amount of power is used for both the baseline and makespan-optimized solutions is puzzling.

One possible explanation may be directly derived from the formula used for the Bat^{avg} assessment, as we can see

[5] $Res^{avg} = \frac{1}{n*maxCap} \sum_{i=1}^{n} \frac{\int_0^{mk_i} f_i(t)\mathrm{d}t}{mk_i} \times 100$, where n is the number of problem instances, $maxCap$ is the resource maximum capacity, mk_i is the solution's makespan of each instance, and $f_i(t)$ is the curve representing the resource utilization profile along the complete makespan.

[6] $\Delta^{avg}_{LWU} = \frac{1}{n} \sum_{i=1}^{n} \frac{mk_i - mk_i^0}{mk_i^0} \times 100$, where mk_i corresponds to the makespan length of the optimized solution provided by $ISES$, and mk_i^0 is the the makespan length of the initial solution provided by $ESTA^p$

	ESTA^p (feasibility assessment)				ISES (optimization assessment)				
Benchmark	$Mksp^{avg}$	CPU^{avg}	$Cache^{avg}$	Bat^{avg}	$Mksp^{avg}$	Δ^{avg}_{LWU}	$Cache^{avg}$	Bat^{avg}	$\#Iter(avg)$
MSR_{40}–20–8h	17027.2	50.367	28.575	8.864	12620.325	35.628	30.473	8.679	14.575
MSR_{40}–20–12h	19336.232	55.609	33.062	21.511	15990.45	26.399	33.866	21.395	12.375
MSR_{40}–20–16h	30144.4	59.440	23.535	39.771	25729.3	17.239	25.795	39.672	11.3
MSR_{40}–25–8h	23826.228	108.793	30.458	8.91	20870.925	21.805	33.954	8.861	7.85
MSR_{40}–25–12h	29249.686	121.651	28.153	21.598	26293.85	17.907	28.481	21.523	6.05
MSR_{40}–25–16h	41558.232	163.706	22.062	39.825	37720.55	11.785	24.801	39.783	4.725
MSR_{40}–30–8h	30605.825	181.842	32.674	8.889	26643.7	15.588	35.213	08.857	5.775
MSR_{40}–30–12h	36516.125	207.214	30.166	21.639	33517.45	9.248	30.995	21.608	3.975
MSR_{40}–30–16h	52707.65	272.359	22.435	37.185	48680.2	8.569	22.819	38.034	2.925

Table 1: Experimental results corresponding to the feasibility and optimization assessments

that while shorter plans should require less battery power (e.g., the distance traveled are shorter), the Bat^{avg} value is inversely proportional to the makespan (i.e., an optimized makespan increases the Bat^{avg} value). Despite all of the above, the very strict correspondence of values in all the cases remains however to be fully explained.

Finally, the average rover cache utilization data (both $Cache^{avg}(\%)$ columns) deserve some attention. Looking at the $Cache^{avg}(\%)$ columns, a decreasing utilization of the cache can be observed as the seasonal situation move from the summer to the winter daylight conditions. This can be noticed for all the MSR_{40}–25–$*$ and the MSR_{40}–30–$*$ benchmark sets, and the same behavior applies to both the feasibility and the optimization assessment data (even though it can be observed that in the makespan-optimized solutions the average cache utilization tends to increase). This circumstance is easily explained as a direct consequence of the longer times necessary to complete the same missions under less favorable power charging conditions (i.e., longer plan makespans entail a less efficient cache utilization). Yet, it can also be observed that in the MSR_{40}–25–$*$ case, the previous regular trend is not followed: as the lighting conditions worsen, there is an "counterintuitive" behavior where the average cache utilization seems to increase, before definitely falling to the expected values. This "anomaly" on the general trend might be explained with the influence of the maximum time windows on the execution of some job sequences, which may cause the rover to decide not to release all of the acquired samples at the AV location before heading for a new experiment's location, in order to satisfy some experiment-related deadline constraint. It is straightforward that in all such circumstances, the cache utilization tends to increase as the cache itself remains occupied by the unreleased samples. The reason this phenomenon becomes evident only with the smaller instances (i.e., those composed of 20 experiment sequences) is related to the fact that, since each problem instance always has 4 sequences characterized by a deadline (regardless of its size), the presence of such deadlines become more relevant for the instances where the constrained/unconstrained sequences ratio increases.

Conclusions and Future Work

In this paper we presented last results on delivering advanced autonomous reasoning capabilities to robotic planetary exploration. In our current work, we were inspired by the requirements of a particular rover-based Mars exploration mission, namely, the Mars Sample Return (MSR) mission concept. One of the contribution of this work is to integrate the most significant MSR mission requirements into a scheduling problem model, the Power Aware Resource Constrained Mars Rover Scheduling (PARC-MRS) problem. Following the proposed model, we presented a scheduling algorithm aimed at synthesizing complete plan sequences that span the whole mission horizon by reasoning upon a wide set of realistic mission requirements. More concretely, the reasoner we propose focuses on a number of results belonging to previous research, and provides an extension of a well-known constraint-based, resource-driven procedure which exploits power-aware reasoning capabilities within an integrated resolution strategy, where a wide variety of complex temporal and resource constraints are considered, with special attention paid to the energy requirements. Indeed, one of the main contributions of this work is the successful exploitation of a well known methodology to represent renewable resources by means of a classical cumulative scheme, to model and solve the PARC-MRS problem. We also conducted an experimentation assessment to evaluate the efficiency of our solution algorithm, as well as the effectiveness of an optimization schema in providing minimum-makespan solutions.

The contents of this paper describe an ongoing work. More activities are currently being carried out in many directions, like the refinement of the terrain model to take into account characteristics such as slope, compactness, roughness, etc., in view of a fully dynamic utilization of the scheduling engine in a simulated *Sense-Plan-Act* loop execution context. It was outside the scope of this paper to present the preliminary results of such experimentation. The interested reader may refer to (Díaz et al. 2012) for more information on the current state of activities.

Acknowledgments

Daniel Diaz is supported by the European Space Agency (ESA) under the Networking and Partnering Initiative (NPI) *Autonomy for Interplanetary Missions* (ref. 2169/08/NI/PA). The authors are grateful for all the support obtained through ESA-ESTEC, specially from its ESA's technical officer Mr. Michel Van Winnendael. Last author is funded by the CDTI project COLSUVH.

References

Cesta, A.; Oddi, A.; and Smith, S. F. 2002. A constraint-based method for project scheduling with time windows. *J. Heuristics* 8(1):109–136.

Cheng, C., and Smith, S. 1994. Generating Feasible Schedules under Complex Metric Constraints. In *12th National Conference on AI (AAAI-94)*.

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

Diaz, D.; R-Moreno, M.; Cesta, A.; Oddi, A.; and Rasconi, R. 2011. Scheduling a Single Robot in a Job-Shop Environment through Precedence Constraint Posting. In *IEA/AIE, Syracuse, NY, USA*.

Díaz, D.; R-Moreno, M.; Cesta, A.; Rasconi, R.; and Oddi, A. 2012. An Intergrated Constraint-based, Power Aware Control System for Autonomous Rover Mission Operations. In *i-SAIRAS*, article n. 10A–4.

Estlin, T.; Gaines, D.; Chouinard, C.; Castano, R.; Bornstein, B.; Judd, M.; Nesnas, I.; and Anderson, R. 2007. Increased Mars Rover Autonomy using AI Planning, Scheduling and Execution. In *Robotics and Automation, IEEE International Conference*, 4911–4918.

Montanari, U. 1974. Networks of Constraints: Fundamental Properties and Applications to Picture Processing. *Information Sciences* 7:95–132.

Oddi, A., and Smith, S. 1997. Stochastic Procedures for Generating Feasible Schedules. In *14th National Conference on AI (AAAI-97)*, 308–314.

Oddi, A.; Rasconi, R.; Cesta, A.; and Smith, S. F. 2011. Solving job shop scheduling with setup times through constraint-based iterative sampling: an experimental analysis. *Annals of Mathematics and Artificial Intelligence* 62(3-4):371–402.

Poulakis, P.; Joudrier, L.; Wailliez, S.; and Kapellos, K. 2008. 3DROV: A Planetary Rover System Design, Simulation and Verification Tool. In *i-SAIRAS*.

Simonis, H., and Cornelissens, T. 1995. Modelling Producer/Consumer Constraints. In *First International Conference on Principles and Practice of Constraint Programming*, 449–462. London, UK: Springer-Verlag.

Smith, S., and Cheng, C. 1993. Slack-Based Heuristics for Constraint Satisfaction Scheduling. In *11th National Conference on AI (AAAI-93)*.

Treiman, A. H.; Wadhwa, M.; Shearer, C. K.; McPherson, G. J.; Papike, J. J.; Wasserburg, G. J.; Floss, C.; Rutherford, M. J.; Flynn, G. J.; Papanastassiou, D.; Westphal, A.; Neal, C.; Jones, J. H.; Harvey, R. H.; and Schwenzer, S. 2009. Groundbreaking Sample Return from Mars: The Next Giant Leap in Understanding the Red Planet. In *Planetary Science Decadal Survey*.

van Winnendael, M.; Baglioni, P.; and Vago, J. 2005. Development of the ESA ExoMars Rover. In *i-SAIRAS*.

Wood, E. G. 2002. Multi Mission Power Analysis Tool. In *IT Symposium, Pasadena, CA, USA*.