AN INTEGRATED CONSTRAINT-BASED, POWER-AWARE CONTROL SYSTEM FOR AUTONOMOUS ROVER MISSION OPERATIONS

Daniel Díaz, Maria Dolores R-Moreno

Universidad de Alcala, Alcala de Henares, Madrid (Spain)
{*ddiaz, mdolores*}*@aut.uah.es*

Amedeo Cesta, Angelo Oddi, and Riccardo Rasconi

CNR – National Research Council of Italy, Rome (Italy)
{*amedeo.cesta, angelo.oddi, riccardo.rasconi*}*@istc.cnr.it*

ABSTRACT

This paper aims at describing an integrated power-aware, model-based autonomous control architecture for planetary rover-based mission operations synthesized in the context of a Ph.D. program on the topic "Autonomy for Interplanetary missions" funded and supported by ESA. The proposed controller implements a single Sense-Act-Plan (SPA) closed-execution loop to safely command the robot activities considered in the context of a specific key mission scenario. Both highly decision-making capabilities and a flexible execution process are the two key features on which the control system is grounded. Furthermore, target execution capabilities, specially those that allow to flexibly keeping pace with temporal and power-related contingencies that might threaten the whole schedule execution attainment, are demonstrated through the integration with the ESA's 3DROV planetary rover system simulator.

Key words: Automation and Robotics; robust constraint-based action scheduling; flexible reactive schedule execution; plan execution management.

1. INTRODUCTION

Flexible and safety management of complex temporal and resource constraints involved with typical operation of planetary wheeled robotic explorers actually comprises a big challenge in deep space exploration. As for many real world problems, deciding and executing actions for a Mars rover implies (1) to deal with actions that might be executed concurrently; and (2) to be aware of a high level of uncertainty threatening action execution success. In summary, inherent uncertainty of this class of problems will inevitably invalidate even the most conservative and best-built plan during its execution. The interest for increasing autonomy and robustness on coping with uncertainty grows at the same pace as rovers becomes more skilful and missions more ambitious.

The ongoing path to address the previous constraints is actually realized by a continuous evolution of the current "Autonomous and Robotics (A&R)" techniques, more concretely those that entail major autonomy capabilities such as AI mission Planning and Scheduling (P&S) coupled with reactive execution mechanisms. In this context, the paper aims at describing an "integrated power-aware, model-based autonomous control architecture for planetary rover-based mission operations", synthesized in the context of a Ph.D. program on the topic "Autonomy for Interplanetary missions" funded and supported by ESA. The proposed control architecture basically focuses on generating plans and safely executing them under a wholesome integrated framework which enables *plan execution management* in a seamless way: it implements a single Sense-Act-Plan (SPA) closed-loop so that the real status of the execution is reliably known every time and used as a guideline for triggering reactive strategies on the face of possible contingencies. More in detail, the control architecture uses at its core an AI Planning and Scheduling (P&S) system [5] which provides highly decision-making capabilities able of handling a wide range of complex temporal and resource constraints with special attention to the management of power consumption features [6]. Furthermore, the autonomous system implements a flexible execution process that timely dispatch the rover commands by continuously tracking both the flawless rover behaviour and the changing constraints imposed by the environment. Reactive mechanisms close the execution loop by updating the internal running model and occasionally triggering re-scheduling strategies to maintain a feasible and quiescent execution if inconsistencies arise.

The remainder of the paper is organized as follows: in section 2 we describe the specific mission scenario of interest and its formalization as a twofold problem consisting on synthesizing and executing plans as an integrated execution management process. Section 3 provides a broad description of the proposed power-aware autonomous control architecture. Section 4 outlines its integration with the ESA's 3DROV planetary rover simulator system with the aim of demonstrating the target execution capabilities. Finally, some a conclusions and future work section close the paper.

2. PROBLEM FORMULATION

Consider a planetary rover-based mission inspired on the ESA's Mars Sample Return (MSR) concept [1]. In a conventional day-to-day mission readiness, a compound of scientific experiments are scheduled as a set of partially ordered sequences of activities to be executed by the (unique) rover. A typical experiment basically consists on (a) travelling to specific locations of scientific interest, (b) acquiring Martian soil samples and (c) delivering them to a final location where an ascent vehicle is in charge of initiating the return trip to bring back to Earth the collected science. More concretely, the rover aims at synthesizing and executing robust sequences of movements so that mission goals are successfully fulfilled. Figure 1 illustrates a mission scenario example where the three main mission activities are highlighted (i.e., navigation, science acquisition and delivery).

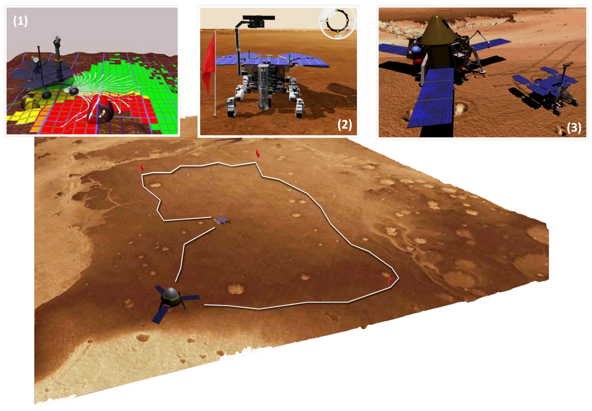


Figure 1: Mission scenario overview: (1) navigation, (2) science acquisition and (3) delivery

The successful execution of a solution schedule which synthesizes the whole rover activity within a specific makespan (e,g, one sol or Martian day), involves performing a set of experiments $E = \{Exp_1, \ldots, Exp_n\}$ while synchronizing the use of a set of resources $R = \{r_1, \ldots, r_n\}$. Each experiment Exp_i (with $1 \leq i \leq n$) consists of a sequence of activities $Exp_i = \{Nav_{Si}, Drill^i, Nav_{iF}, Rel^i_F\}$, where Nav_{ij} activity is the traversal which takes place between two different locations *i,j* (*S* and *F* are the initial and final locations respectively); $Drill^i$ activity represents the sample extraction and storage tasks (i.e., the experiment itself); and Rel^i_F activity is concerned with placing or releasing the collected science corresponding to the experiment *i* at the final location *F*.

Activities are defined by their minimum and maximum durations, by their start and due dates, as well as by the demanded quantities of one or more resources. The set R of managed resources consists of an assortment of assets of different nature and complexity:

- A **drilling subsystem (s/s)** to perform extraction operations.
- A **sample cache** as a container with capacity *C*, to store collected science for transportation. This implies that the rover might collect up to a number of *C* samples consecutively before leaving them at the ascent vehicle location.
- A **robotic arm** mounted on the ascent vehicle, which is used to take the science collected by the rover.
- A **power s/s** consisting on a battery and a set of solar arrays. The battery is characterized by its maximum capacity or *saturation level* B_{max} (in Watts per hour), and by its minimum usage threshold B_{thres} as a percentage of B_{max}. B_{thres} is used to avoid the usage of the battery below a specific value as a safeguard for unexpected situations.

Resources are finite capacity assets modelled as *cumulative* (binary or multicapacity). The amount of resource demanded by activities remains completely blocked during the entire activity execution (i.e., no preemption is allowed), and the sum of all resource demands cannot exceed the maximum or minimum usage thresholds.

The following temporal and resource constraints govern the overall rover behaviour:

- *Activity execution times*. $Drill^i$ activity has a flexible and minimum duration which corresponds to the total time needed to perform the sample collection and storage activities; Rel^i_F activity has a fixed duration equal to the time required by the ascent vehicle to unload the collected science; and the Nav_{ij} activities have flexible and minimum durations which correspond to the *traversal times* needed to cross the path joining the two different locations *i,j*. A detailed definition of the traversal time is given in the following.
- *Traversal time (tt_{ij})* is the minimum time required by the rover to move between two different locations i, j. Each path joining pairs of locations is considered as a sequence of straight segments, which were previously discovered as a result of a traversability map[1] computation process. In general, we do not assume that traversal times satisfy the triangle inequality property, since the travelling paths might be computed according to multi-objective optimization methods [7] over a three-dimensional Euclidean space.
- *Resource demands*. $Drill^i$ activity demands the drilling s/s and a specific amount of energy. Nav_{ij} activities demand a variable amount of energy e_{ij} which depends on the travelling distance between the two different locations *i,j*, as well as one storage unit of the sample cache if the traversal takes

[1]Representation of the terrain which contains information about the ease of traversal of different regions according to diverse terrain features such as hills or obstacles.

place between the experiment and final locations; and Rel_i activity uses the robotic arm of the ascent vehicle.

– *Energy production constraints*. The battery is supposed to be continuously charged at a specific constant rate σ_{charge} (unit energy/unit time) by means of the solar arrays, until the saturation level *B* is reached. Once the saturation level is reached, all further charging energy is simply discarded.

In the next section we explain our specific problem model which is embedded in the autonomous controller.

2.1. Problem modelling

The reasoning framework used in this work is based on a Constraint Satisfaction Problem [8] (CSP) description scheme: the baseline scheduling problem is represented as a special type of CSP, i.e., it is modelled in terms of a set of variables each characterized by a specific domain, and a set of (local and global) constraints that bind the legal variable assignments. According to this problem formulation, a feasible CSP solution is defined as a complete feasible assignment of domain values to all variables so that all constraints are satisfied.

Transitioning to a CSP representation scheme implies to perform the following mapping: variables are the time points representing the starting and end times of the whole set of activities; and constraints are the feasible temporal values of each time point, defined by the activity durations and temporal separations.

Additional assumptions are considered in order to embody our reference problem as a CSP:

– *Resource modelling*. Both the drilling s/s and the robotic arm are characterized as *binary resources*, while the sample cache is represented as a *cumulative (multi-capacity) resource* to allow the rover to carry multiple samples at a time.

– *Precedence relations*. Two different separation constraints are considered: *simple* and *setup time (bounded)* precedence constraints. In particular, pairs of activities $\langle Drill^i, Nav_{iF}\rangle$ and $\langle Nav_{iF}, Rel^i_F\rangle$ pairs are joined by simple (tight) precedence constraints. As the former navigation activity (Nav_{Si}) does not make use of the sample cache resource, we decided to model it as a setup time constraint bounded by the *traversal time* needed to cross the path joining the starting and experiment locations.

– *Mutual exclusion constraints*. Pairs of activities which use different resources and belong to different experiments, e.g., $Drill^i$ and Rel^j_F such that $i \neq j$, are mutually exclusive.

Figure 2 illustrates a problem example with three experiments. It shows: (a) a slightly overloaded PERT chart in that we are not only trying to convey the causal relationships among the activities but also their precise (absolute) temporal allocations (top); and (b) the resource profiles corresponding to the drilling s/s, the robotic arm and the sample cache resource respectively (bottom). The part concerning to the power production/consumption is further explained in detail. Two additional time points (i.e., start *Start* and *Sink*) are added to represent the initial and latest feasible instants on the schedule. The maximum distance between both time points is equal to the time horizon.

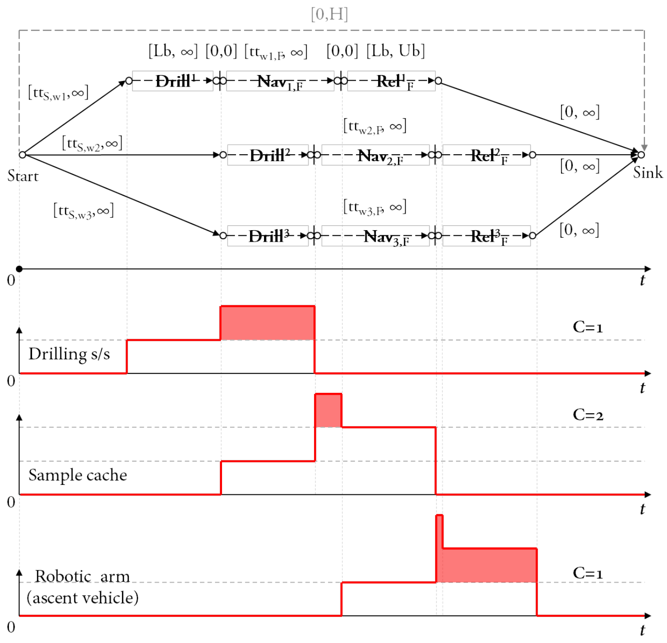


Figure 2: Problem example representation with three experiments where the capacity and usage constraints of the drilling s/s, the robotic arm and the sample cache are highlighted.

Power demands and production constraints are modelled on the basis of the reference model proposed by Simonis [13], which basically consists on representing consumable resources (such as a battery) in the shape of cumulative resources by applying a simple transformation rules. More concretely, the battery resource processes the two following sets of activities:

– *Energy consumers.* A new set of battery consuming activities are injected within the overall model so that their end times match with the sink time point, i.e., the end of the schedule; and starting times match with the instants on which a specific amount of energy is required, i.e., when the rover collects science (e_{drill}) and travels between two different locations (e_{ij}). It is worth to mention that the power consumption profile is conservative in that the entire amount of energy needed to completely perform an activity has to be available at the beginning of the activity execution.

– *Energy producers.* The continuous charging rate curve (i.e., charging profile) is modelled as a sequence of small, discrete chunks of energy distributed along the complete horizon with a specific resolution: the result is a piecewise segment representation of the energy production profile. Each chunk of energy is modelled as a (fixed) activity which demands an amount of energy equal to the quantity of power collected at each piecewise segment. Energy producer activity are constrained to start at the beginning of the schedule, and finish at the instant on which the battery is charged with the related energy chunk.

The intersection of both resulting power consumption and power production profiles represents an estimation of the battery usage along the complete schedule makespan as seen in figure 3. The computation of the overall battery profile (i.e., the charging values at each instant) is performed through the following recursion formula:

$$Val_i = Min\{Val_{i-1} + SP_{i-1,i}, B_{max}\} - SC_i$$

where the unknown members are:

Val_i is the battery charging level at each instant.

Val_{i-1} is the battery charging level at the previous instant.

$SP_{i-1,i}$ is the amount of energy collected between two consecutive instants.

SC_i is the amount of energy consumed at each instant.

We assume that the battery is fully charged at the beginning ($Val_0 = B_{max}$).

3. THE AUTONOMOUS CONTROL SYSTEM

The complexity involved in both synthesizing feasible solutions and in their safe execution demands a solution which combines advanced reasoning capabilities and flexible management strategies. As seen in the previous section, providing feasible solutions for execution means to deal with a wide set of decision constraints ranging from path planning to pure scheduling choices such as multi-capacity resource allocation. Additionally, safely executing plans entails to equip the rover with flexible execution mechanisms to allow both *plan execution monitoring* and *reactive strategies* to cope with possible contingent situations.

Our proposed solution consists on a constraint-based, power-aware control system which implements a sense-plan-act (SPA) closed-loop execution scheme: the controller exploits advanced reasoning capabilities at its core,

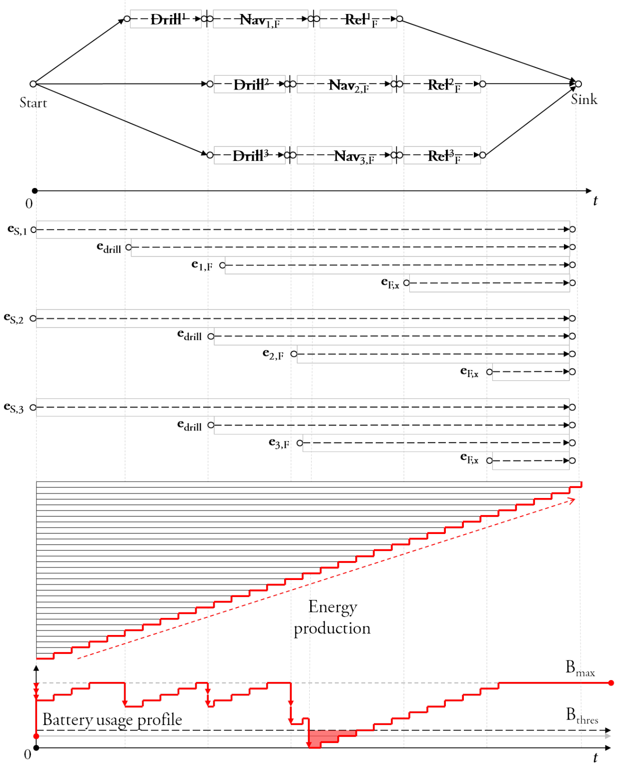


Figure 3: The battery usage profile is computed as an intersection of the power consumption (top) and charging (bottom) profiles.

so that the consistency of the entire plan execution is continuously guaranteed by analysing the execution outcome and by triggering reactive strategies in case misalignments are detected (between the expected and the observed execution results).

More concretely, the autonomous controller is in charge of performing the following steps within the SPA closed-loop execution process, until the whole schedule is completely dispatched:

1. Synthesize feasible schedule solutions and timely dispatch them.

2. Read rover status parameters such as position, orientation, command completion confirmation as well as battery state of charge (SoC).

3. Detect possible misalignments (in terms of resource usage violations or activity execution delays) between the planned and the real rover behaviour.

4. Provide (on-line) alternative schedule solutions to face with the detected contingent situations.

Figure 4 illustrates the main functional building blocks of the autonomous controller.

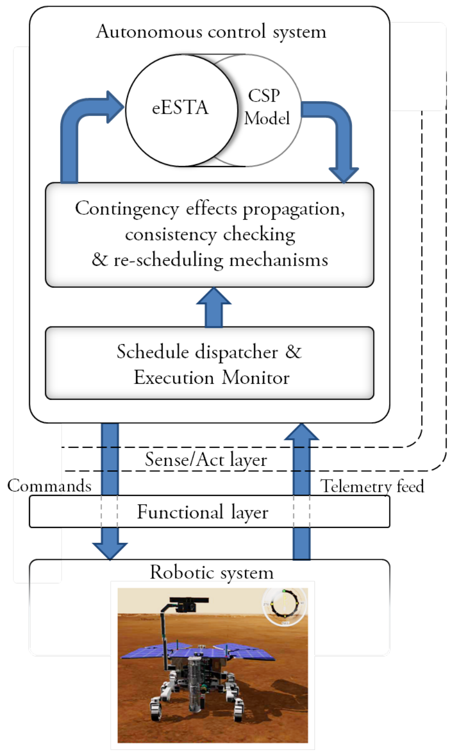

Autonomous control system
eESTA
CSP Model
Contingency effects propagation, consistency checking & re-scheduling mechanisms
Schedule dispatcher & Execution Monitor
Sense/Act layer
Commands
Telemetry feed
Functional layer
Robotic system

Figure 4: Conceptual schema of the autonomous control system.

In the next subsection we explain more in detail the previously mentioned schedule execution management capabilities.

3.1. Constraint-based reasoning

The schedule execution process starts with the synthesis of a complete baseline plan which enables the rover to comply with the mission targets while satisfying temporal and resource constraints within a given temporal horizon. An extended version of the constraint-based solving algorithm eESTA presented in [5] is embedded at the core at the autonomous controller, with the twofold aim of (a) providing the initial schedules and (b) repairing them on the face of the occurred contingencies. Very briefly, eESTA is a resource-driven, heuristic-biased solving algorithm able of reasoning upon a wide variety of (a) temporal and resource constraints such as sequence-dependent setup-times or dynamic resource demands, and (b) resources such as complex consumable resources (e.g. a battery). More concretely, the main eESTA characteristics are summarized as follows:

- The underlying *problem-solving schema* basically combines the *precedence constraint posting* heuristic guideline used by the reference solving algorithm ESTA [4], with an adaptation of the *dominance conditions* of the SP-PCP (Shortest Path-based Precedence Constraint Posting) algorithm proposed in [9, 10]. Both compounding ingredients allows eESTA to explicitly deal with multi-capacity cumulative resources while considering setup-time precedence relations respectively.

- Further enhancements extend the original algorithm's capabilities with the aim of coping with new and more complexity elements introduced within our problem of reference. The most remarkable contribution was providing the scheduling algorithm with specific mechanisms to deal with the particularities arising from the integration of our specific battery model representation within its cumulative-based scheme [6].

The knowledge encapsulated within the eESTA reasoning guidelines allows the algorithm to iteratively balance the *resource contention peaks* (i.e., resource overcommitments) by posting new precedence constraints between activities until a conflict-free schedule (or a temporal inconsistency) is found. Figure 5 illustrates one possible solution for the previous problem example: it consists of a feasible (temporal) distribution of the activities related to the three experiments that also satisfies all the resource usage constraints. Extracting the rover behaviour (i.e., the commands sequence) is a trivial issue. In this example, the rover (1) moves to the first experiment location and drills to extract the first sample; then, (2) it navigates towards the second experiment location and does the same task; afterwards, (3) the rover goes to the final location to release both collected samples; and finally, (4) the rover travels to the last experiment location to complete the pending tasks.

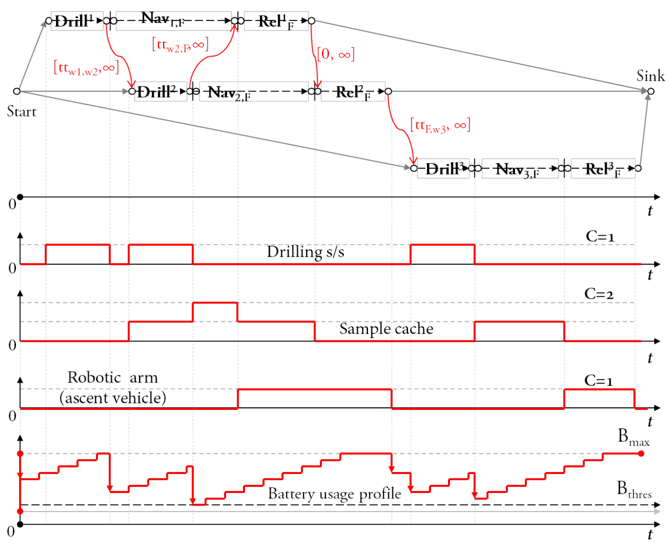


Figure 5: Feasible solution example: a set of solving constraints were posted to solve all resource conflicts.

3.2. Monitoring

While rover position and orientation information is used to determine if the rover was delayed on the completion of some activity execution, battery SoC readings are constantly interpreted to check if the battery is being overconsumed.

Temporal contingencies are detected when the rover is late in arriving at a specific place or while performing a soil extraction/storage activity. Contingencies related to the battery usage are detected when an overconsumption occurs, i.e., the battery SoC exhibits an unforeseen loss of energy along a specific time interval. For instance, if the rover is performing sample collection activities, the controller checks that the SoC readings evolve according to the estimated production/consumption curve contained within its model; if the SoC readings do not match with the expected values during a specific time period, so that the curve experiences a (downwards) *anomalous deviation*, the controller determines that the battery is being overconsumed.

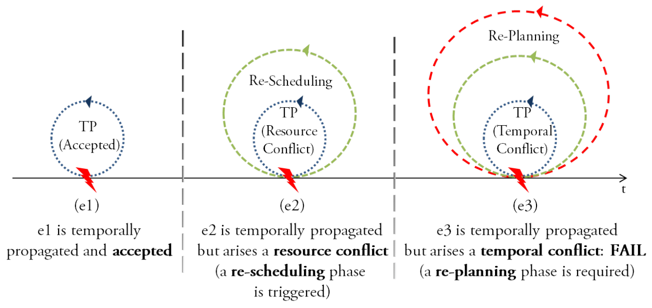
3.3. Contingency solving

Reactive mechanisms allow the controller to (a) project the effects of the detected contingencies by aligning (in time and resource usage) its internal model with the real situation, and (b) reason upon the new possibly introduced inconsistencies by *adjusting the running schedule through a re-scheduling cycle*.

Projecting the effects of contingencies entails injecting additional (temporal or resource) constraints, as well as propagating their temporal effects along the schedule part which is pending to be executed. If a temporal inconsistency is detected, for instance while the rover is travelling between two different locations, the controller injects in the current problem's model a temporal constraint which delays the next activity to be executed (i.e., an experiment or a sample release activity). Otherwise, if a power contingency is detected, the controller reflects such overconsumption by proportionally increasing the amount of energy demanded by the power consumer activity which is actually oversubscribing the battery resource.

The propagation step is then followed by a global consistency checking process which might result in the three following ways (see figure 6):

- Neither temporal nor resource conflicts have arisen. In this case, the running schedule solution has been robust enough to absorb the propagation effects. Hence, execution continues as normally.
- A resource conflict is introduced. The controller triggers a re-scheduling step to attempt to solve the new resource overconsumptions by running eESTA through the pending part of the schedule to be executed. If the scheduler succeeds at synthesizing a new solution, this might simply consists on an timely stretched schedule or on a solution where the activities are reshuffled. Then, the execution continues executing the repaired solution.
- A temporal conflict is introduced. If a temporal contingency introduced a delay which violates some temporal constraint such as the time horizon, then no solution can be obtained by simple rescheduling and the execution is aborted (re-planning actions are required).



Re-Planning
Re-Scheduling
TP (Accepted)
TP (Resource Conflict)
TP (Temporal Conflict)
t
(e1)
(e2)
(e3)
e1 is temporally propagated and **accepted**
e2 is temporally propagated but arises a **resource conflict** (a **re-scheduling** phase is triggered)
e3 is temporally propagated but arises a **temporal conflict**: **FAIL** (a **re-planning** phase is required)

Figure 6: The three possible situations which might occur when the effects of an exogenous event e_i are injected.

In next section we illustrate in detail the previous execution management process, by means of a simple scenario example running on an integrated testbed platform with the ESA's 3DROV [11] planetary rover simulator.

4. TESTING THE CONTROLLER WITH 3DROV SIMULATOR

The main difficulty of designing and testing a planetary rover mission frequently stems from the lack of enough knowledge and reproduction assets about the target environment. The "Planetary Robot Design, Generic Visualization and Validation Tool" [11] (3DROV) was created with the objective of providing ESA's Automation and Robotics (A&R) section[2] with a software framework to effectively support the complete development of planetary robot systems *within realistic mission contexts*.

4.1. The integrated testbed platform

One of the many interesting capabilities of the 3DROV simulation platform is to serve as a test bench for onboard controllers and ground station operation modules. We created an integrated testbed platform on top of the 3DROV generic controller with a twofold motivation: (a) to check if the synthesized solution schedules actually

[2]The ESA's Automation and Robotics (A&R) group is the responsible for carrying out with the creation and maintenance of such industrial technology base for the automation and remote control of space based operations.

represent realistic estimations in time and resource usage when applied to practical cases of study; and (b) to demonstrate both the *robustness of the solution schedules* and the *flexibility of the execution management process* when introducing uncertainty.

Figure 7 shows the different modules which constitutes the integrated platform:

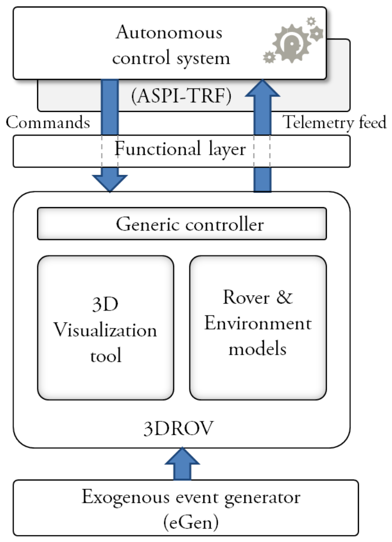


Figure 7: The integrated testbed platform with 3DROV simulator.

- Our autonomous control system uses the APSI Timeline Representation Framework (APSI-TRF) [2, 3] as the scheduling software development environment to implement and deploy the whole execution control system.
- We created a middle module between the controller and the 3DROV simulator as a functional layer to implement a TCP/IP-based communication and mapping the high-level locomotion commands and telemetry queries to the specific low level format defined in 3DROV.
- 3DROV simulator system has a modular and distributed architecture which we organized here around three integrated subsystems for simplicity: the *generic controller* module encapsulates a set of specific libraries[3] and interfacing assets which allows to control in very detail all the rover operations; the *rover and environment models* module provides information about a variety of rover and environment related features such as rover dynamics, kinematics, power, thermal, or terrain and atmospheric characteristics; the 3D visualization component is the front-end of the 3DROV simulator, and allows to track in real-time the evolution of the simulation execution through a virtual 3D representation of the complete mission scenario.
- The exogenous event generator (eGen) module was created to stress the simulation execution by injecting proper disturbances such as provoking temporal delays or power overconsumptions. Thus, we manage to analyse the controller response capacity against external threats in a controlled manner.

[3]3DROV implements a generic A&R control system based on the ESA's A&R standardised development concepts and guidelines captured within the Control Development Methodology (CDM) [12] framework.

4.2. Case study

Table 1 summarizes the main characteristics of the test problem example:

Test problem description features		
Attribute	*Value*	*Observations*
Num. Experiments	3 experiments	First experiment has a completion deadline (time-window communication requirement)
Sample cache cap.	2 samples	Max. num. of soil samples handled concurrently
Linear & rot. speed	0.4 m/sec	Rover moves by performing straight, soft moves and standing rotations
Max. battery cap.	600 Wattsh	Battery starts 90% charged with a min. usage threshold of the 50%
Power consumption	200 Watts (T) & 133 Watts (E&S)	T: travelling, E&S: soil extraction and storage
Power production	15.02 Watts	Linear, constant rate (in average)

Table 1: Test problem example.

It is worth mentioning that the steepness of the time scale and power consumption/production rates are intentionally augmented to (a) collapse a full working day (sol) into a few minutes of simulation, and (b) demonstrate the main controller capabilities with more clarity.

We considered two different contingent situations during the simulation execution. In both situations, the controller manages to successfully resove them. The evolution of both contingent situations are described in detail next:

Figure 8 shows the rover commands timeline extracted from the baseline solution schedule when the execution starts, as well as the battery usage profiles before and after the first contingency occurrence, respectively. It is worth to note also that the baseline solution maximizes the use of the sample cache (samples related to WP_1 and WP_2 are collected consecutively without releasing the cache between samplings).

- The first contingency occurs when the rover is moving between the starting position S and the location of the first experiment WP_1. The rover crosses a

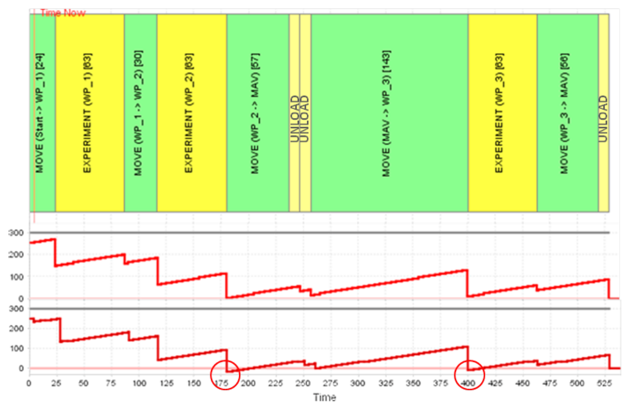


Figure 8: Commands timeline when the execution starts, and battery usage profiles before and after the first contingency arrival respectively.

soft soil area and gets stuck for *4* seconds starting from the time instant $t = 4$, and the the expected traversal time is stretched accordingly. The situation is solved when the rover escapes from the slippery area by applying a smooth and progressive acceleration; this operation consumes more power than initially expected.

Consequently, the autonomous controller detects misalignments both in the temporal pace of the execution and in the battery usage (i.e., a power overconsumption is detected) by comparing both the evolution of the expected position and the SoC values with the telemetry readings.

Contingency effects are reflected on the internal model of the controller by injecting and propagating two additional (temporal and resource usage) constraints. Then, the controller detects two inconsistencies in the battery usage profile at the instants $t = 177$ and $t = 400$ respectively. As a result, a re-scheduling step is triggered by providing an alternative solution consisting on a partial reshuffling and delay of the pending activities to be executed.

Figure 9 illustrates the resulting timeline of the execution commands after the first re-scheduling step, as well as the battery usage profiles before and after the second contingency propagation.

– Second anomalous situation takes place when the rover is travelling between the first experiment WP_1 and final F locations. The rover is deprived of incoming power for *3* seconds: the power collection efficiency of the solar panels is appreciably reduced during this period of time because of shadowing on the solar arrays (e.g. due to cloudy weather or to the proximity of a geological feature such as a hill).

As before, the controller detects misalignments between the expected and real evolution of the SoC values, and reflects such misalignment within its internal model by injecting and propagating an additional power usage constraint. As a result, another re-scheduling step is triggered as a response to a violation of the maximum power limit (new overconsumption peaks are detected around the time instants $t = 380$ and $t = 560$). An alternative solution consisting on a simply delay of the pending activities is successfully provided.

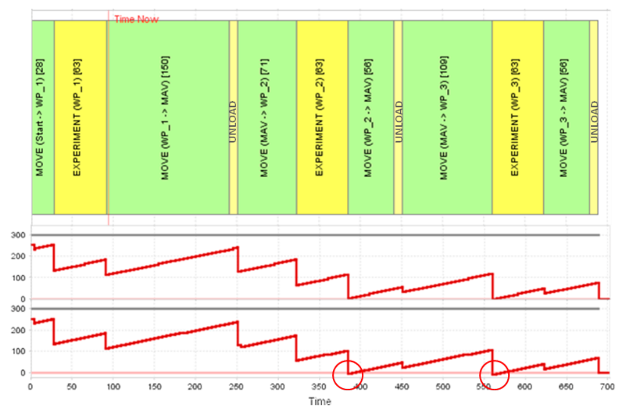


Figure 9: Commands timeline and battery usage profiles transition corresponding to the first contingency resolution.

Figure 10 shows the evolution of both the command timeline and the battery usage profile resulting from the resolution of the second contingent situation.

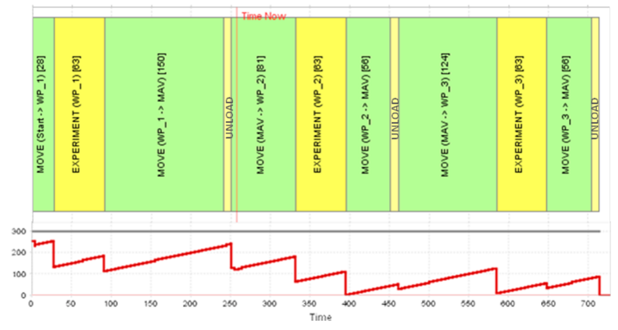


Figure 10: Commands imeline and battery usage profile corresponding to the second contingency resolution.

5. CONCLUSIONS AND FUTURE WORK

This paper described an integrated power-aware, model-based autonomous control architecture with application in planetary rover-based mission operations. The controller implements a smart execution management process based on a single Sense-Act-Plan (SPA) closed-loop scheme, so that the real plan execution outcome is reliably known and analysed to trigger reactive strategies, in the case possible contingencies threaten the whole plan's attainment. The proposed architecture exploits at its core an AI P&S system which provides advanced decision-making capabilities to reason upon a wide range of complex temporal and resource constraints, with special attention to the management of power consumption features. Additionally, we demonstrated the interplay between the deliberative and reactive capabilities of the autonomous controller by means of a simple scenario example running on an integrated testbed platform with the ESA's 3DROV planetary rover simulator.

Further work might be oriented to improving the robustness of the plan execution process, more concretely the contingency solving strategy. The current implementation provides one attempt at repairing an invalidated running schedule, and if no solution is found the rover remains stuck waiting for human intervention. An interesting enhancement is to allow the controller to spend more time at the expenses of finding a feasible solution if the re-scheduler fails (or just to find a better solution) by iterating eESTA algorithm within an optimization framework.

ACKNOWLEDGMENT. Daniel Diaz is supported by the European Space Agency (ESA) under the Networking and Partnering Initiative (NPI) *Autonomy for Interplanetary Missions* (ESTEC-No. 2169/08/NI/PA). CNR authors are partially supported by MIUR under the PRIN project 20089M932N (funds 2008). The second UAH author is supported by Castilla-La Mancha project PEII09-0266-6640. Special thanks to Michele Van Winnendael and Konstantinos Kapellos for their dedicated support throughout the whole course of the study.

REFERENCES

[1] P. Baglioni, R. Fisackerly, B. Gardini, G. Giafiglio, A.L. Pradier, A. Santovincenzo, J.L. Vago, and M. Van Winnendael. The Mars Exploration Plans of ESA (The ExoMars Mission and the Preparatory Activities for an International Mars Sample Return Mission). In *IEEE Robotics & Automation Magazine, Vol. 13, No.2, pp. 83-89.* 2006.

[2] A. Cesta and S. Fratini. The Timeline Representation Framework as a Planning and Scheduling Software Development Environment. In *PlanSIG-08. Proceedings of the 27th Workshop of the UK Planning and Scheduling Special Interest Group, Edinburgh, UK, December 11-12*, 2008.

[3] Amedeo Cesta, Gabriella Cortellessa, Simone Fratini, and Angelo Oddi. Developing an End-to-End Planning Application from a Timeline Representation Framework. In *IAAI-09. Proceedings of the Twenty-First Conference on Innovative Applications of Artificial Intelligence, July 14-16, 2009, Pasadena, California, USA*, 2009.

[4] Amedeo Cesta, Angelo Oddi, and Stephen F. Smith. A Constraint-Based Method for Project Scheduling with Time Windows. *J. Heuristics*, 8(1):109–136, 2002.

[5] D. Diaz, M.D. R-Moreno, A. Cesta, A. Oddi, and R. Rasconi. Scheduling a Single Robot in a Job-Shop Environment throug Precedence Constraint Posting. In *Proceedings of the 24th International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems (IEA/AIE 2011), Syracuse, NY, USA*, 2011.

[6] D. Diaz, M.D. R-Moreno, A. Cesta, A. Oddi, and R. Rasconi. Toward a CSP-based Approach for Energy Management in Rovers. In *In Proceedings of the 4th IEEE International Conference on Space Mission Challenges for information technology (SMC-IT 2011), ISBN: 978-3-642-13160-8. Palo Alto, CA, USA*, 2011.

[7] Juan Pablo Gonzalez, Bryan Nagy, and Anthony (Tony) Stentz. The geometric path planner for navigating unmanned vehicles in dynamic environments. In *Proceedings ANS 1st Joint Emergency Preparedness and Response and Robotic and Remote Systems*, February 2006.

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

[9] A. Oddi, R. Rasconi, A. Cesta, and S.F. Smith. Iterative-Sampling Search for Job Shop Scheduling with Setup Times. In *COPLAS-09. Proc. of the Workshop on Constraint Satisfaction Techniques for Planning and Scheduling Problems at ICAPS*, 2009.

[10] A. Oddi, R. Rasconi, A. Cesta, and S.F. Smith. Solving Job Shop Scheduling with Setup Times Through Constraint-based Iterative Sampling: an Experimental Analysis. *Ann. Math. Artif. Intell.*, 62(3-4):371–402, 2011.

[11] P. Poulakis, L. Joudrier, S. Wailliez, and K. Kapellos. 3DROV: A Planetary Rover System Design, Simulation and Verification Tool. In *Proceedings of the 10th International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS-08)*, 2008.

[12] Peter Putz and A. Elfving. Control techniques 2, automation and robotics control development methodology definition report. Technical Report ESA CT2/CDR/DO, 1992.

[13] Helmut Simonis and Trijntje Cornelissens. Modelling producer/consumer constraints. In *Proceedings of the First International Conference on Principles and Practice of Constraint Programming*, pages 449–462, London, UK, 1995. Springer-Verlag.