UNIVERSIDAD DE ALCALÁ
DEPARTAMENTO DE AUTOMÁTICA

PH.D. THESIS

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

Daniel Díaz Palacios

Ph.D. Supervisor
Dr. M.D. R-Moreno

International Advisors
Dr. Angelo Oddi, Dr. Riccardo Rasconi

MAY, 2015

To Ismael

Hardships often prepare an ordinary person for an extraordinary destiny.

(C.S. Lewis)

Acknowledgment

I want to acknowledge Mr. Michele VanWinnendael for his dedicated support throughout the course of the whole Ph.D. thesis. Special thanks to Dr. M.Dolores Rodriguez Moreno, Dr. Amedeo Cesta, Dr. Riccardo Rasconi and Dr. Angelo Oddi, because all of them made possible the results reported in this work.

Resumen Ampliado

El panorama futuro de la exploración planetaria estará caracterizado por ambiciosas misiones donde el protagonismo de la robótica inteligente no hará más que ir en aumento. El incremento de los niveles de autonomía a bordo supondrá delegar en la robótica espacial altos grados de responsabilidad en la planificación y ejecución de la misión, tales como la síntesis de planes de actividades detallados a partir de descripciones abstractas de los objetivos; la adaptación o modificación del plan en ejecución como respuesta a situaciones de contingencia; o incluso la posibilidad de realizar proyecciones de posibles riesgos futuros o la detección de objetivos científicos relevantes (ciencia oportunista).

Los riesgos y costes que suponen las misiones tripuladas continúan siendo a día de hoy inasumibles, por lo que el uso de sistemas robóticos autónomos se revela como la mejor, sino la única solución viable a corto-medio plazo. Ya en la actualidad tenemos ejemplos de misiones en las que se ha desplegado con éxito robots equipados con innovadora instrumentacin científica y avanzadas tecnologías de Inteligencia Artificial (IA), capaces de realizar actividades de alto grado de complejidad. Hoy Marte representa el escenario de referencia en el que las principales agencias aeroespaciales internacionales, en colaboración con la industria especializada, están demostrando el enorme potencial que tiene el uso de *exploradores robóticos dotados de altos grados de autonomía*.

Las distancias astronómicas que separan a un operador humano situado en la Tierra del cuerpo celeste más cercano, las posibilidades tan reducidas de comunicación directa que tiene éste con el sistema robótico, o el elevadísimo grado de incertidumbre que envuelve a las misiones interplanetarias, convierten a la tele-operación como mecanismo de control remoto en una solución poco fiable sino inviable. Pro-

visionar a los *rovers* con las capacidades suficientes para desempeñar tareas cada vez más complejas con una reducida o nula intervención humana supone enormes beneficios para el desarrollo de la misión en términos de seguridad, costes económicos y tiempo. Además y no menos importante, aumentar la autonomía a bordo se traduce en una aceleración de lo que se conoce como el retorno o utilidad científica de la misión, como consecuencia directa de la reducción de la intervención humana en el bucle de control.

El trabajo presentado en esta tesis ha sido concebido como una actividad promovida y financiada por la Agencia Espacial Europea (ESA) bajo el marco del programa *Networking and Partnering Initiative (NPI) (ESTEC-No. 2169/08/NI/PA)*. Las principales contribuciones que de este trabajo se desprenden surgen como una respuesta a la necesidad de continuar desarrollando la autonomía a bordo en la robótica espacial, evolucionando así el esquema clásico de tele-operación hacia una estrategia mixta y global de control que permita ejecutar diferentes modos de operación desde puramente manuales a totalmente autónomos.

En este contexto, hemos estudiado y desarrollado técnicas existentes de IA de *scheduling* y control de ejecución, proponiendo una solución aplicada a un modelo de misión de exploración planetaria conducida por un *rover*, consistente en una arquitectura de control autónomo avanzado que combina:

- Planificación y *scheduling* robusto basado en restricciones. El sistema de control propuesto provee herramientas de razonamiento de alto nivel que permiten sintetizar (a bordo) planes de acción que definen los pasos necesarios para la consecución de los diferentes objetivos de la misión. Las actividades que definen el plan de acción están delimitadas por un conjunto heterogéneo de restricciones complejas de naturaleza temporal y de uso de recursos, como por ejemplo plazos máximos de ejecución (deadlines), tiempos de espera entre actividades (sequence-dependent setup-times), requerimientos energéticos o de uso de instrumentos.

- Esquema de control de ejecución flexible y reactivo. La ejecución de los planes de acción se ejecutan en base al conocido esquema de control de bucle cerrado *Sense-Plan-Act (SPA)*, capaz de reaccionar a un conjunto de posibles situa-

ciones de riesgo derivadas de la alta incertidumbre del entorno. El modelo de control implementado asegura la consistencia del plan de acción a lo largo de su ciclo de ejecución mediante la coordinación de las herramientas de razonamiento avanzado, monitorización y análisis del desarrollo de la ejecución, y estrategias reactivas para la actualización y corrección del plan en ejecución en caso de diferencias (desviaciones) entre la evolución de la ejecución esperada y real.

Además, el modelo de control presentado permite desplegar un proceso de optimización continua del plan de forma paralela a la ejecución, además de la posibilidad de incorporar *ad-hoc* nuevos objetivos de misión al plan de forma dinámica. Por último, con el fin de validar las capacidades clave de la arquitectura de control presentado, se propone el desarrollo de un banco de pruebas o *testbed* integrado con el avanzado sistema de simulación de *rovers* 3DROV, una herramienta propiedad de la ESA diseñada para reproducir de forma detallada la operación de un *rover* en un escenario realista.

Abstract

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.

The work presented in this Ph.D. thesis was developed within an European Space Agency (ESA) programme[1], and was conceived as a response to the increasing interest of evolving current telerobotic capabilities towards a complete mixed-initiative strategy implementation that enables shifting control modes from purely manual to fully autonomous operations.

In this context, we study and exploit existing Artificial Intelligence (AI) scheduling and control execution techniques to propose a solution for a planetary rover-mission operation concept, which combines both robust constraint-based robot action scheduling and flexible reactive schedule execution management in the face of high executional uncertainty. First of the two major challenges entails providing high-level reasoning capabilities to enable the rover synthesizing complete command plans on the management of complex temporal and resource constraints. Second major problem deals with an efficient execution of the command plans while preserving the safety of the mission.

We present an integrated model-based execution control architecture targeted at generating and safely executing robust mission plans, by exploiting a constraint-

[1]The Ph.D. programme is under the ESA's Networking and Partnering Initiative (NPI) *Autonomy for Interplanetary Missions* (ESTEC-No. 2169/08/NI/PA)

based flexible model that allows to: represent both time and resources, with special attention to the rover power requirements; integrate plan re-scheduling and plan execution on top of the same representation, allowing to hedge against domain uncertainty by reacting to possible incoming disturbances and dynamically modifying the plan (e.g., to respond to unexpected science opportunities); and perform on-line plan optimization concurrently to plan execution.

With the aim of conducting a formal analysis on the performance of the core capabilities of the autonomous controller, we built an integrated testbed platform on top of the Planetary Robot Design Generic Visualization and Validation Tool (3DROV) planetary rover system simulator, an ESA asset that provides a realistic behavioural reproduction (i.e., from the temporal, dynamic and energy-related standpoint) of all the rover subsystems (e.g., locomotion, drilling, etc,) necessary to perform the tasks covered in our experimental model.

Contents

List of Figures

List of Tables

List of Acronyms

3DROV Planetary Robot Design Generic Visualization and Validation Tool

AI Artificial Intelligence

APSI-TRF Advanced Planning and Scheduling Initiative - Timeline Representation Framework

A&R Automation and Robotics

CSP Constraint Satisfaction Problem

EBA Enveloped Based Analysis

ESA European Space Agency

ISES Iterative Sampling Earliest Solutions

LCV Least Constrained Value

LRTN Linear Resource Temporal Network

MCS Minimal Critical Set

MRV Most Restrictive Variable

MSR Mars Science Return

OR Operational Research

PARC-MRS Power Aware Resource Constrained Mars Rover Scheduling

PCP Precedence Constraint Posting

POS Partially Ordered Schedules

P&S Planning and Scheduling

SPA Sense-Plan-Act

STNU Simple Temporal Network with Uncertainty

TCSP Temporal Consistency Satisfaction Problem

Part I

Preliminaries

Chapter 1

Introduction

In this chapter we introduce the background reference and motivations behind the work of this thesis. We start providing a general description of the basic context and main objectives pursued by this work as well as its main contributions; next we present the structure of the dissertation; we close the chapter with a list of the scientific publications produced along the course of this work.

1.1 Motivation

> *"It's one small step in the history of space flight. But it was one giant leap for computer-kind, with a state of the art artificial intelligence system being given primary command of a spacecraft. Known as Remote Agent, the software operated NASA's Deep Space 1 spacecraft and its futuristic ion engine during two experiments that started on Monday, May 17, 1999. For two days Remote Agent ran on the on-board computer of Deep Space 1, more than 60,000,000 miles (96,500,000 kilometers) from Earth. The tests were a step toward robotic explorers of the* 21^{st} *century that are less costly, more capable and more independent from ground control." – from NASA's site about Deep Space 1's Remote Agent.*

Main contributions of this thesis are rooted on the necessity of continuing infusing and promoting autonomy to space robotics. The utilization of autonomous robotic

systems is considered as extremely necessary in deep-space exploration as manned missions are currently unattainable because of the enormous hazards posed. At the present time, interplanetary exploration is deploying robots equipped with cutting-edge scientific instrumentation and innovative Artificial Intelligence (AI) technologies able of performing complex mission tasks. Mars exploration actually represents the reference scenario on which the main international space agencies in collaboration are demonstrating the enormous potential of using *robotic explorers with increasing autonomy capabilities*. Providing rovers with advance degrees of autonomy yields enormous benefits in terms of safety and mission efficiency: on the one side, the astronomic distances involved, the lack of a high-bandwidth permanent communication link and the high uncertainty permeating the whole rover's surrounding playground, make infeasible using reliable real-time teleoperation [Muscettola *et al.*, 1998]. On the other side, augmenting the *on-board* autonomy entails maximizing the overall mission science return, as the human intervention on the control-loop is not so demanded.

The state-of-the-art in "Automation and Robotics (A&R)" in mission control operation is actually represented by *smart intelligent error handling mechanisms* coupled with *basic high-level mission planning techniques*. Continuing infusing autonomy in future planetary exploration inevitably entails to close the control-loop onboard with advanced autonomous reasoning capabilities. For instance, by entrusting the upcoming robots with more high level responsibilities such as the synthesis of nominal mission plans from high-level goal descriptions, plan adaptation/modification to face with contingent situations, opportunistic science and even hazard prediction. AI machine learning or robust mission Planning and Scheduling (P&S) are two enabling technologies which will allow to improve both the vehicle's expertise in the performance of its mission activities based on the incoming sensor data; and its ability of efficiently reasoning upon complex and diverse temporal and resource constraints (e.g., energy consumption, availability of the resources, restrictive activity execution deadlines, etc.) on the decision of feasible command sequences.

1.2 Objectives

The work presented in this dissertation was developed within an ESA programme[1] conceived as a response to the increasing interest of evolving current telerobotic capabilities towards a complete mixed-initiative [Crandall & Goodrich, 2001] strategy implementation that enables shifting control modes from purely manual to fully autonomous operations.

More concretely, this work aims at exploiting existing AI scheduling techniques to propose a solution for planetary rover-mission operations which combines both:

1. Robust constraint-based robot action scheduling – First of the two major challenges here addressed is about providing high-level reasoning capabilities which allow planetary rovers synthesizing (on-board) feasible command sequences on the accomplishment of specific mission goals, while considering a wide variety of temporal and resource constraints within an extremely uncertain environment. For this reason, we will dedicate a significant part of this thesis to the study of AI scheduling techniques, specially those which enable *robust constraint-based scheduling in the face of executional uncertainty*.

2. A flexible reactive schedule execution management – The second major problem faced in this thesis is about efficiently executing the rover commands extracted from a mission baseline schedule previously synthesized, while preserving mission safety against the extraordinary uncertainty posed by the hazardous surrounding environment. In this case, the solution calls for a revision of the existing approaches and best-known architectural models in the development of autonomous control systems, with a special interest on those which allow to cope with executional contingency by exerting *advanced decision-making capabilities with flexible management strategies*.

In the light of what precedes, in this dissertation we will present the following results:

[1] The Ph.D. programme is under the ESA's Networking and Partnering Initiative (NPI) *Autonomy for Interplanetary Missions* (ESTEC-No. 2169/08/NI/PA)

- A *scheduling problem domain inspired on the Mars Science Return (MSR) mission concept*. We propose a model of the world inspired by the Mars Sample Return (MSR) mission concept, a long-range planetary exploration scenario, and introduce a scheduling problem called Power Aware Resource Constrained Mars Rover Scheduling (PARC-MRS) based on the baseline requirements of the MSR mission scenario. We proposed a study of the benchmarking problem tailored to the MSR domain, and produced a methodology to generate meaningful PARC-MRS problem instances.

- An *advanced constraint-based solving algorithm*. We present the profile-based, power-aware reasoning algorithm $ESTA^p$ for providing feasible solution plans to the so-called scheduling PARC-MRS problem, as well as a meta-heuristic for solution optimization. $ESTA^p$ provides advanced reasoning capabilities that enables 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.

- A *flexible model-based autonomous control architecture* for planetary rover mission operations. The proposed controller $CoRe^p$, which stands for (Co)ntrol (Re)sources & (p)ower, implements a single (model-based) Sense-Plan-Act (SPA) closed-loop execution scheme to safely command the robot activities considered in the context of the MSR key mission scenario, through a seamless integration of advanced decision-making capabilities (provided by $ESTA^p$) within a flexible execution process targeted at generating and safely executing mission plans. $CoRe^p$ architecture also supports (on-line) continuous plan optimization, as well as dynamic integration of new rover activities within the running plan *on-the-fly*.

- An *integrated testbed platform* built on top of an outstanding planetary rover simulation platform (3DROV) with the twofold motivation of: (a) validating the synthesized solution schedules with respect to practical cases of study; and (b) demonstrating the *degree of adaptability* of the autonomous controller when inducing executional uncertainty on the basis of the robustness and flexibility

of both the solution schedules and execution management process respectively. Therefore, an important part of this thesis is dedicated to conduct an analysis of the whole target execution capabilities, through the use of dynamic simulations recreating some representative situations as realistic cases of study.

1.3 Organization of the Thesis

We have organized the thesis along four differentiated chapters, namely Introduction, Robust Constraint-based Robot Action Scheduling, Flexible Reactive Schedule Execution Management in Harsh Environments, and Conclusions and Future work.

- Chapter 1 presents the motivation, main objectives, contents and the scientific publications in connection with this thesis.
- Chapter 2 provides a detailed survey of the trending paths and most important techniques in AI planning, scheduling and intelligent execution research literature is depicted, as well as some representative autonomous control architectures.
- Chapter 3 presents a description of the tools used for the production of the different artefacts on the scope of this thesis.
- Chapter 4 starts introducing a general overview of the scheduling problem of reference here referred as the PARC-MRS problem, as well as its formal representation as a Constraint Satisfaction Problem (CSP) for problem-solving. Then, we describe the constraint-based, resource-driven scheduling algorithm $ESTA^p$ to provide feasible solutions to the PARC-MRS problem of reference, as well as a meta-heuristic algorithm for solution optimization.
- Chapter 5 shows an empirical analysis of both the solving algorithm and the optimization framework. We start presenting the benchmark problem instance generator MSR/Gen used for the creation of the set of benchmark sets; then, we explain how the assessment is conducted; finally the experimental results closes the chapter.

- Chapter 6 starts introducing an extended, dynamic definition of the PARC-MRS problem to fit within the requirements of an advanced mission control architecture targeted at providing a reliable and efficient mission execution management process. Then, we provide a detailed description of the autonomous control architecture solution as an integrated constraint-based, power-aware control system $CoRe^p$. A couple of illustrative simulation examples close the chapter.

- Chapter 7 continues with an experimental analysis on the evaluation of an integrated testbed platform consisting of: (1) the autonomous control system presented in previous chapter $CoRe^p$; (2) an environmental module for exogenous events generation; and (3) the planetary rover simulator 3DROV. Empirical analysis consists of a couple of realistic cases of study (dynamic simulations), with the aim of validating the core capabilities of the control architecture.

- Chapter 8 summarizes the most important contributions presented along this dissertation, as well as a discussion on the possible future works.

1.4 Scientific Publications within this Ph.D. Thesis

The results presented in this dissertation has produced the following publications (some of them can be found as part of the the bibliography) in the fields of AI P&S and autonomous execution control, as contributions to international workshops, conferences as well as to an international scientific magazine:

- Daniel Díaz, Maria D. R-Moreno, Amedeo Cesta, Angelo Oddi, Riccardo Rasconi. Applying AI Action Scheduling To ESA Space Robotics. In Proceedings of the 11^{th} ESA Workshop on Advanced Space Technologies for Robotics and Automation (ASTRA 2011). ESA/ESTEC Noordwijk, Netherlands.

- Daniel Díaz, María D. R-Moreno, Amedeo Cesta, Angelo Oddi, Riccardo Rasconi. Scheduling a Single Robot in a Job-Shop Environment through Precedence Constraint Posting. In Proceedings of the 23^{th} International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems (IEA-AIE 2011). Syracuse, NY, USA.

- Daniel Díaz, María D. R-Moreno, Amedeo Cesta, Angelo Oddi, Riccardo Rasconi. Toward a CSP-based Approach for Energy Management in Rovers. In Proceedings of the 4^{th} IEEE International Conference on Space Mission Challenges for information technology (SMC-IT 2011), ISBN: 978-3-642-13160-8. Palo Alto, CA, USA.

- Miguel Doctor, Angel Moreno, Pablo Muñoz, Daniel Díaz, María D. R-Moreno. Intelligent Social Networks. In Proceedings of the 1^{st} International Workshop on Social Data Mining for Human Behaviour Analysis. Sogndal, Norway, May 2011.

- Daniel Díaz, María D. R-Moreno, Amedeo Cesta, Angelo Oddi and Riccardo Rasconi. An integrated Constraint-based, power Aware control system for Autonomous rover Mission operations. In Proceedings of the International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS'12), Turin, Italy, September 2012.

- Daniel Díaz, María D. R-Moreno, Amedeo Cesta, Angelo Oddi, Riccardo Rasconi. An Empirical Experience with 3DROV Simulator: Testing an Advanced Autonomous Controller for Rover Operations. In Proceedings of the 12^{th} Symposium on Advanced Space Technologies in Robotics and Automation (ASTRA 2013). ESA/ESTEC Noordwijk, Netherlands.

- Daniel Díaz, Amedeo Cesta, Angelo Oddi, Riccardo Rasconi and María D. R-Moreno. Autonomous Energy Management as a High Level Reasoning for Planetary Rover Problems. In Proceedings of the 5^{th} Italian Workshop on Planning and Scheduling (IPS13), Turin, Italy, December 2013.

- Daniel Díaz, Amedeo Cesta, Angelo Oddi, Riccardo Rasconi and María D. R-Moreno. Efficient Energy Management for Autonomous Control in Rover Missions. IEEE Computational Intelligence Magazine, 8(4), pp:12-24, 2013. Special Issue on Computational Intelligence for Space Systems and Operations.

Chapter 2

State of the Art

2.1 Introduction

The integration of a reactive execution with action P&S is a central topic on the design of autonomous control systems targeted at operating on real world environments. On the one hand, *reactivity* provides mobile robots with the ability of promptly adapting to possible contingencies resulting from the uncertainty permeating surrounding environment. On the other hand, *reasoning capabilities* allow the robot to autonomously synthesize action plans in situations where the environmental information is incomplete or spurious and the actions' effects cannot be determined in a deterministic way.

Most basic principles on the design of control systems are essentially rooted on (a) the complexity of the tasks or jobs the robot is targeted at performing, and (b) the surrounding characteristics that might affect the successful attainment of the plan such as the environmental uncertainty. For the sake of illustration, we can imagine a robot in charge of executing simple action plans in a very dynamic and changing environment. The resulting control system will be obliged to adapt very quickly the robot's behaviour very often. As a different example, the robot is committed to execute more complex tasks, but the surrounding uncertainty does not pose a significant threat. In this case, the autonomous controller will have more time to synthesize robust plans, so the integration of advanced reasoning capabilities would become an asset.

In general terms, existing autonomous controller designs are contained between two opposite approaches, i.e., pure AI planning and scheduling based techniques (on the field of computer science), and hard-coded, circuit-driven schemes (used in classical control theory). AI P&S based controllers exploits internal, symbolic representations (model) of the problem domain to synthesize robot action sequences, while control theory infers preprogrammed answers from a set of possible inputs in a more straightforward fashion. Second approach is quite efficient if applied to deterministic situations, but results quite rigid and inadequate to problems involving unstructured environments where stochastic events might threaten the execution process. On the contrary, AI P&S techniques allow autonomous controllers more flexibility in the face of contingencies, and allow provisioning feasible plan solutions to complex problems. Principal drawback is that AI based controllers might become slow or ineffective, specially in tight time constraint scenarios where reflex-like responses are demanded.

Examples of mixed solutions that combine ingredients from both approaches are numerous in literature. For instance, the Teleo-Reactive (T-R) programs [Nilsson, 1994] import some control-theory ideas into computer science; or the AI sensor-based planner [Bouguerra, Karlsson, & Saffiotti, 2007] that takes advantage of run-time perceptual feedback [Nourbakhsh & Genesereth, 1996] to perform conditional planning.

Most extended execution control system designs revolve around the previous concepts, commonly referred in literature as the SPA paradigm (see figure 2.1). SPA control loop scheme assumes that mobile robots essentially perform three tasks: sense, think (plan), and act. These three tasks include communication with sensors to obtain data from the robot's environment (sense), execution of algorithms for localization and planning (plan), and driving actuators to control the robot's motion (act). Despite this control execution model works very well when applied under ideal conditions, many problems arise when applied to real world scenarios, where the feedback information used to be incomplete and spurious, and the model of the world (i.e., the internal representation) typically represents a rough approximation of the real problem scenario.

Execution monitoring capabilities are typically integrated as a basic step on the

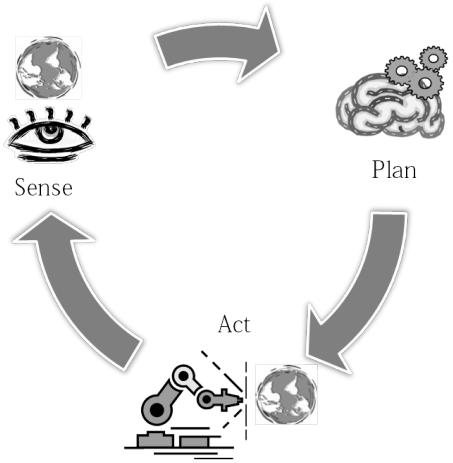

Figure 2.1: The Sense-Plan-Act control execution paradigm.

SPA execution control loop process, so that the execution outcome can be suitably verified on the basis of the sensor feedback. Monitoring actions are taking place between the execution and deliberative or planning activities by closing the gap between both steps. Monitoring actions typically consist on an analysis of the execution feedback provided by the sensors, so that the real outcome of the robot actions effects are compared against the expectations. Concretely, the following situations might arise: (a) the expected behaviour is confirmed; (b) expectations are misaligned from the real results; (c) or execution outcome simply cannot be determined because of a lack of information feedback (required observations are not always available). In case of any misalignment is detected, different plan recovering strategies might be flagged with the aim of readjusting the robot behaviour and minimizing the possible negative impact on the successful attainment of its goals. An example of a simple contingency solving strategy consists on discarding the invalidated plan in execution and setting the robot in safe mode until a new execution plan is provided. More sophisticated strategies can be considered, like providing robust plans able of absorbing possible contingency effects (i.e., plans that are synthesized by considering uncertainty as a main principle), and/or integrating flexible reaction policies that trigger different failure recovery actions depending on the anomalous situation.

Traditionally, AI P&S techniques have been successfully applied to mechanistic

problems that assume manufacturers-like conditions. Classical AI-based autonomous control systems are rooted on the conservative control theory basis which pursues an absolute and flawless governance, even in the face of non-nominal situations: contingency plans are designed in advance for every anomalous situation. But a new form of AI P&S system targeted at operating in real environments and that considers uncertainty as the main and partially controllable principle is increasingly demanded. With real problems we mean, close-to-human-intellect tasks such as fluidly plan movements to avoid obstacles to achieve a specific goal: a challenging AI P&S problem that claims more advanced capabilities than the actually provided by classical approaches, such as efficient mechanisms for managing complex and diverse temporal and resource constraints (e.g., energy demands, restrictive deadlines and resource availability, etc.).

We might find out many definitions of P&S in the context of AI in literature, and very often these two concepts are confused and interchangeably used. Indeed, P&S as AI problem solving activities are intimately related to each other as both are aimed at synthesizing sequence of tasks (as a feasible or optimal plan) on the persecution of a goal, by satisfying a set of resource requirements and temporal constraints.

Figure 2.2 aims at giving a definition of P&S so that both concepts are clearly distinguished: planning deals with the selection of the tasks or activities and their relative order (what to do), while scheduling focuses on temporally allocating the activities (when to do) while having into account different resource requirements.

Existing AI P&S techniques are intrinsically model-based, i.e., they exploit a representation of the problem domain defined through some well-established representation formalism (i.e., domain theory). Generally, the definition of a (purely) planning problem involves the following elements: (i) a description of the initial state of the world, i.e., the initial configuration of the different elements of the problem domain; (ii) a description of the goals, i.e., a specific configuration of the world that matches with the desired goal; (iii) and a set of possible actions that allow to transit from one state to other.

In the light of what precedes and other related questions, current chapter aims at providing a complete picture of the underlying challenges and existing solutions in autonomous mobile robotics along three different sections: planning techniques,

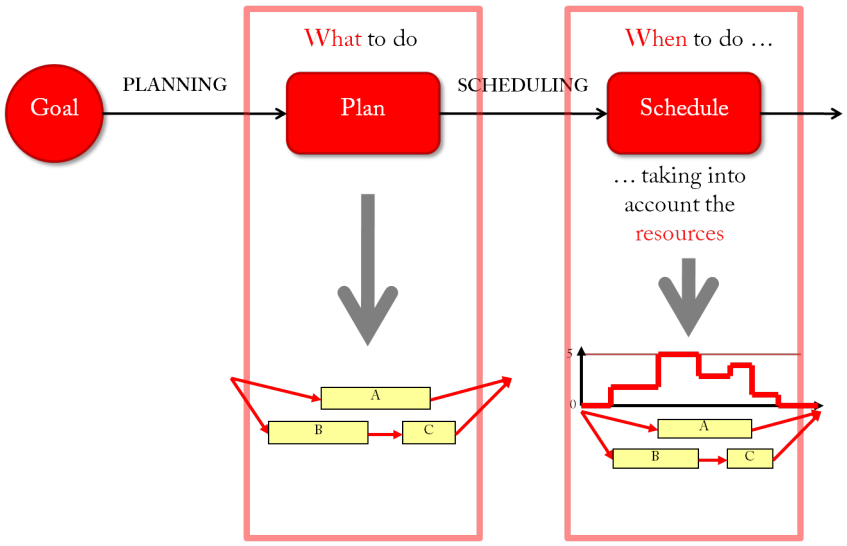

Figure 2.2: AI Planning and Scheduling definition concept scheme.

resource constrained scheduling techniques and autonomous control architectures. A last section called “experiences in applying autonomous control techniques to space robotics” presents some recent and significant projects in this field.

2.2 AI Planning Techniques

In general, we might classify the existing AI planning techniques according to the level of compliance of the following requirements:

- *Generality* – Planners should be as domain-independent as possible, by encapsulating the most general and common properties of a wide family of problems. Increasing generality usually implies reducing quality and efficiency in solution plans, so a balanced solution which makes a good trade-off represents the best solution.

- *Scalability* – An important aspect to be considered, in special when tackling with real world problems, is the ability of suitably handling with different planning problem instances despite of their sizes. In other words, planning strategy should guarantee tractability and efficiency with independence of the size of

the problems.

- *Informed search* – Exploiting key domain knowledge to guide the planning process on the search of a solution used to accelerate the solution finding, as infeasible paths are early disregarded. Planners that use informed search strategies are based on estimators or heuristics: functions that return an approximation of the cost of connecting an initial or intermediate state with a goal. Heuristic-based methods are specially recommended for those planning problems where an optimal solution is impossible or impractical because of their complexity.

- *Uncertainty management* – In an uncertain world, solution plans are likely to be invalidated in execution time. Different approaches have been proposed to limit the possible contingency effects in planning through the use of more expressive modelling languages. Some examples are probabilistic planning or conditional branching and sensing actions.

- *Temporal and resource constraints management* – Modelling languages should be expressive enough to allow representing a wide range of both: (i) temporal constraints like concurrent and continuous activity execution, mutual exclusion or precedence relations; and (ii) resource constraints like the consumption and production of finite, cumulative usage of resources.

In the next subsections, we start introducing the fundamentals of the two main models of AI planning in literature, i.e., action-oriented (STRIPS-like) and timeline-based planning. Next, we present a summary of the most relevant techniques in both deterministic (classical) planning and planning with uncertainty.

2.2.1 AI Planning paradigms

In literature, we mainly differentiate two AI planning paradigms in terms of knowledge representation for problem solving: the classical (or STRIPS-like) and timeline-based planning. Next, we summarize the most important aspects of both approaches.

2.2.1.1 Action-oriented, propositional or classical planning (STRIPS-like)

Classical approaches are grounded on pure logical formalisms, and define planning as the process of finding a sequence of actions (plan) that allows transforming an initial state of the world into a desired goal state, by applying a set of legal rules to transit between states (described by the domain theory). States are described as logical sentences, and the set of legal actions cause state transitions by adding or removing logical propositions (also known as atoms). The applicability of an action to a specific state is determined by the satisfiability of some preconditions. A planner is a problem-solver engine that takes as inputs both (a) a planning problem instance (described by its initial and goal states), and (b) a domain theory, and attempts to infer a solution (set of action sequences) through a search process on the space o states.

The classical planning approach is typically known as STRIPS-like planning. STRIPS [Fikes & Nilsson, 1971] was a pioneering language that established the fundamentals for the definition of planning problems. From the theoretical point of view, classical planning represents a well-grounded solution paradigm. But unfortunately, a good theoretical solution does not guarantee a good result in practice. Lots of work has been done to increase the flexibility of STRIPS, by adding concepts like: *durative actions* [Mausam & Weld, 2008] to allow expressing temporal planning domains; *multi-valued formulation (MDF)* [Helmert, 2006] to provide a more compact representation of the constraints defining the problem domain, by ignoring add and delete effects of inferred predicates; or *soft goals* [Gerevini & Long, 2006], that allow modelling goal preferences and distinguishing high quality plans among the many feasible plans in a solution space.

Successive STRIPS improvements have been integrated and standardized as the PDDL language (Planning Domain Definition Language), an adoption of a common formalism for describing planning domains. The latest version of PDDL (3.1) includes state variables which are neither binary (true/false) nor numeric (real-valued), but instead map to a finite domain.

2.2.1.2 Timeline-based planning

The Timeline-based planning is a more recent paradigm. The timeline-based solving process focuses on the temporal properties of a problem. A plan describes an envelope of possible temporally flexible behaviours of domain features, and these behaviours are called timelines. Temporal flexibility is an important feature of this approach because it allows to consider at planning time, possible delays in the execution of planned activities. Flexible timelines can be exploited by an executive system for robust on-line execution of plans. This solving process approach blurs the line between P&S, as it aims at synthesizing action sequences (planning) by considering the temporal constraints (scheduling) in a seamless way.

In literature, timeline-based planning is also known as Constraint-Based Temporal Planning (CBTP) [Abdeddaïm *et al.*, 2007], Constraint-based Attribute and Interval Planning (CAIP) [Frank & Jansson, 2003] or simply constraint-based planning. All of them are rooted on the following basic concepts: the problem domain is decomposed in terms of a set of primitive components (also called entities or domain attributes) that represents physical or logical subsystems described as state variables (knowledge representation); each component behaviour is modelled as a temporal function which describes its evolution over time. The domain theory describes their legal evolution or *behaviours*, and regulates the possible interactions between them by mean of causal relation and synchronization constraints (expressed by means of the Allen's interval algebra [Allen, 1983]). Each behaviour describes a different way in which the component's properties may vary in time. A timeline is a logic structure used for reasoning about the evolution of a state variable over a period of time.

The planning process as problem solving strategy is reduced to get an instantiation of feasible timelines (one for each entity) given a initial and final state, while satisfying all the synchronization constraints.

Figure 2.3 depicts a problem model decomposed into four components representing three subsystems and one resource (left), and a feasible solution plan as a set of timelines synthesizing the evolution of every entity over a temporal horizon (H) (right).

The outcome of the planning process is represented as sequences of tokens posted on every timeline, which allow to evolve from the initial states to the desired goals

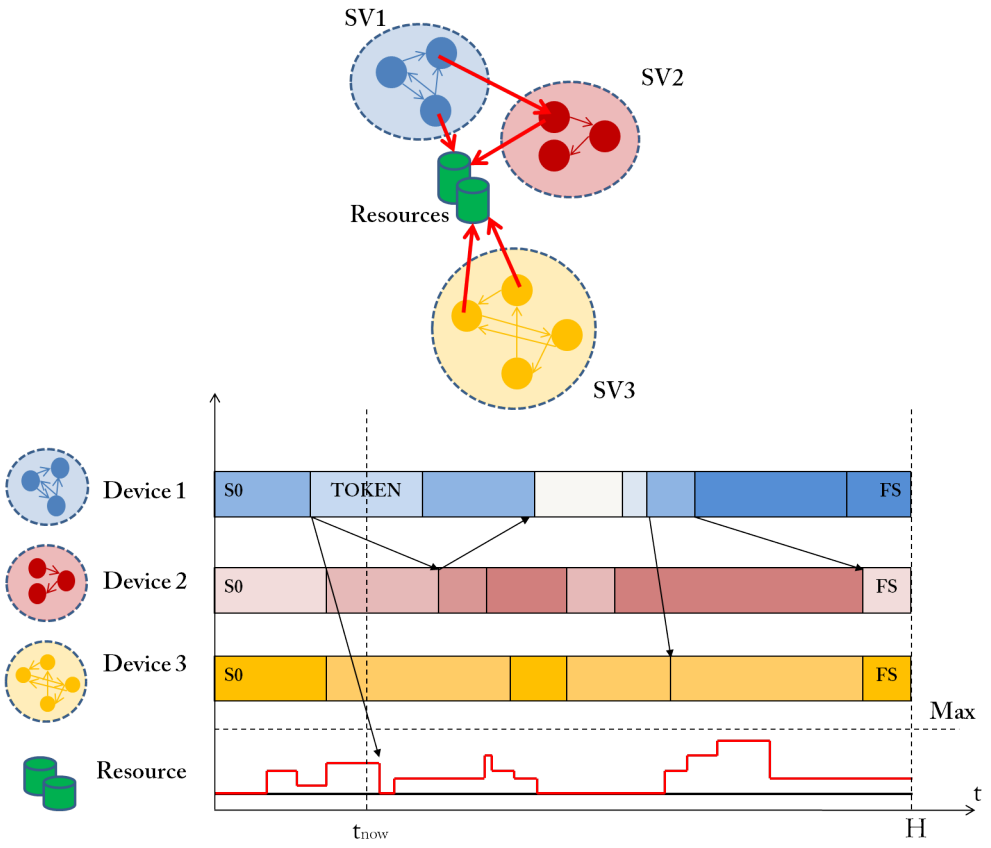

Figure 2.3: Problem decomposition into four components, i.e., two subsystems and one resource (left); solution plan as a set of instantiated timelines (right).

while satisfying the synchronization constraints (arcs). In other words, a solution plan consists of a fully instantiation of the timelines (as sequences of activities), while all the precedence and temporal constraints become satisfied.

As opposed to the classical paradigm, timeline-based approach allows modelling in a more natural way real world planning problems where temporal and resource constraints, mutual exclusion and concurrency are the key building blocks. In addition, timeline-based planning represents an ideal framework for the integration of reasoning and control execution capabilities.

Next, we present a thoroughly overview of the most important planning techniques according to the following classification schema: classical or deterministic

planning techniques, and non-deterministic or planning with uncertainty.

2.2.2 Classical Deterministic Planning Techniques

Basically, we might find two different paths in classical planning research that are actually attracting the most of the attention of the research community (specially because of their significant results on standard competitions): (i) planning as Constraint Satisfaction Problem (CSP), with GRAPHPLAN [Blum & Furst, 1995] as the most well-known and representative example; and (ii) heuristic-biased planning [Bonet & Geffner, 1999]. In general terms, most efficient planners are hybrid systems that combine the advantages of the different techniques. It is worth to mention the significant work existing on enhancing the expressiveness of the domain problem languages by including features like durative actions, concurrency or resource management.

It is worth also to mention the following relevant approaches in classical planning: Partial Order Planning (POP) aims at leaving decisions about the ordering of actions as open as possible. Despite of the fact that partial-order planning has been demonstrated to be inefficient in terms of performance (in contrast with the powerful heuristic-biased or CSP-based planners), POP planning poses some interesting features like its *least-commitment* functioning principle. In the contrary, in total order (TO) planning, solutions are generated as sequences of totally ordered actions. TO planners are either state-based or plan-based, depending if each node of the search space represents a state of the world or a partial solution (plan), respectively. Another interesting approach is the Hierarchical Task Network (HTN) planning, mainly characterized by the decomposition of the problem domain into a set of tasks hierarchically organized. Lastly, we might mention also Planning as satisfiability (SAT-based planning), a classical paradigm where problems are encoded according to a propositional logic representation, i.e., a set of propositional clauses (CNF), and the solution consists on simplifying and satisfying the logic formulae.

Next, we explain in detail the two more relevant approaches in classical planning, i.e., GRAPHPLAN-based and heuristic-biased planning.

2.2.2.1 GRAPHPLAN-based planners

The original GRAPHPLAN system [Blum & Furst, 1995] is a general-purpose planner that uses STRIPS as the domain definition language, and is based on ideas used in graph algorithms. GRAPHPLAN performs a search throughout a compact structure called *planning-graph*, a *state-on-the-node* search space graph where the nodes represent a possible state of the world expressed as a conjunction of ground predicates (also known as propositions or atoms), and the arcs represent possible transitions mapped as actions to be performed (node transitions are triggered if a set of preconditions are satisfied). Concretely, the planning-graph consists of a layered graph with two different kind of nodes, i.e., actions (odd layers) and propositions (even layers), and two kind of transitions, i.e., additive (straight arrows), and no-op or neutral (dotted arrows). Mutual exclusion relations (mutexes) are represented by means of arcs between propositions-actions within the same layer. Two actions (or propositions) are mutually exclusive on the same node or level, if both cannot be contained within the same solution plan: there is a relation of dependency (mutex) expressing the incompatibility between them, as one action (or proposition) removes the positive effects of the other, or simply hold contradictory preconditions.

A general description of the GRAPHLAN solving algorithm is as follows: given a planning-graph structure with only a single proposition level containing the initial conditions, the search process is performed by alternating two different steps. Each iteration starts by expanding the planning-graph one step up, so that a pair of action and proposition nodes or levels (expansion phase). During the first step, a certain number of mutual exclusion relationships are computed and propagated to the next level; in the second step, the planner performs a backward-chaining search starting from the last level to find a valid plan (extraction phase) with a length equal to the number of iterations performed so far. In the latter step, GRAPHPLAN either finds a feasible solution or determines that the goals are not achievable in i steps. The algorithm continues interleaving expansion and extraction phases until a solution plan is found, i.e., a plan that satisfies the problem goals. It is worth to mention that GRAPHPLAN is complete, so that if a solution plan exists then the algorithm will find it.

One interesting property of the planning-graph is that can be pruned quickly dur-

ing the construction phase of the searching process by means of the computation and propagation of the mutual exclusion relation constraints. GRAPHPLAN works under the two following principles: (i) a *strong-commitment*, i.e., a reachability analysis that allows to check in advanced if one state can be reached from other; and (ii) *disjunctive refinement*, a method that allows addressing conflicts flagged as a result of opposite multiple propositions on a same state.

We might find in literature many contributions aimed at improving the GRAPHPLAN capabilities in very different ways like: (i) by applying GRAPHPLAN to a variety of different problem domains; (ii) by introducing slight modifications on its basic search strategy for a better performance; or (iii) by exploiting GRAPHPLAN as part of other searching strategies.

Some well-known examples of GRAPHPLAN-based planners are STAN [Fox & Long, 2001], Blackbox [Fikes & Nilsson, 1971], IPP [Koehler *et al.*, 1997] and the Sensory Graphplan (SGP) [Anderson, Smith, & Weld, 1998].

2.2.2.2 Planning as a heuristic search

This kind of problem solving strategies extract some specific knowledge from the planning problem encoding as a heuristic estimator function, to perform a guided search through the state space. The typical heuristic search planner architecture consists of the following parts: the Analysis & Processing module is in charge of analysing and processing the domain theory as well as the initial state to (automatically) extract the *heuristic estimator*; the Heuristic Computation module estimates the costs to reach the final goal given a state node, by using the heuristic estimator; and the Search Module guides the searching process on the basis of the computed estimations.

Existing purely heuristic planners are classified according to the following characteristics: (i) the direction of the searching process, i.e., if a progression or a regression state space is used; (ii) the informed searching algorithm used (typically best-first or hill-climbing); (iii) and the heuristic estimator (function) extracted from the problem representation.

Roughly speaking, given a specific node or state of the search space n an heuristic estimator $H(n)$ is a function that computes the costs to reach a goal state in terms

of minimum distances. Some interesting properties that define an heuristic function are the following: *optimal or perfect* (typically denoted as $H*$), if it computes the shortest path from every node of the state space to any possible goal; *admissible*, if the estimated distance is equal or less than the optimal one; *safe* if the estimated distance from a node is infinite when the goal is unreachable; *Goal-aware*, if the estimated distance from a node is zero when the node is a goal; and *consistent*, if a node n_{succ} is a successor of the node n via an operator of cost c then the estimation cost from n should be equal or less than the sum of the estimation cost from n_{succ} plus the operation cost c.

There are four different general methods in literature to extract the heuristic estimator from the problem encoding:

- Relaxation – This is the simplest method and considers less constrained versions of the problem by generally ignoring the dropping list associated to an operator (the list of negative effects).

- Critical paths – This method aims at extracting an estimation function by analysing some of the qualitative properties of the planning problem.

- Abstraction – This technique considers reduced versions of the problem by abstracting some of its domain features (not only constraints). More concretely, given an planning problem P and an abstraction of this problem $T(P)$ by means of an abstraction mapping function α, the heuristic extracted from the projection $T(P)$, denoted as $H(\alpha)$, is safe, goal-aware, admissible and consistent.

  In general, good abstractions are those that allow to extract heuristics that are informative and efficient. Sometimes the best solution consists of combining different abstractions by computing the maximum/the sum of the resulting heuristics. Three different abstraction methods are mainly distinguished in literature, i.e., pattern database heuristics (PDB-H) [Culberson & Schaeffer, 1996], merge-and-Shrink abstractions [Drger, Finkbeiner, & Podelski, 2006] and structural Patterns [Katz & Domshlak, 2008].

- Landmarks – This method considers propositional formulas that have to be-

come true at some point in every plan for the task at hand [Hoffmann, Porteous, & Sebastia, 2004]. A landmark can be partially or totally ordered, and is used to infer *inadmissible* heuristic estimators.

Some relevant heuristic search planners in literature are the following: FF [Hoffmann, 2001] (Fast-Forward) and Metric-FF [Hoffmann, 2003], STAN [Fox & Long, 2001], HSP-R [Bonet & Geffner, 1999], AltAlt [Nigenda & Kambhampati, 2003], VHPOP [Younes & Simmons, 2002], MO-GRT [Refanidis & Vlahavas, 1999] and the LAMA [Richter & Westphal, 2010] planner.

2.2.3 Non-deterministic Planning Techniques

Realistic planning problems typically involves a high degree of uncertainty. In pure classical planning a deterministic behaviour is presumed, i.e., every action produces a predictable outcome, achievable goals and a fully observable environment. As a result, solution plans are reduced to sequences of actions or activities that can be executed *blindly*: observation feedback is not necessary at all, plan execution is a trivial problem. Classical planning has demonstrated to be specially good at solving toy problems, or planning problems involving structured and predictable environments with relatively low uncertainty levels.

A considerable amount of work has been done to provide previous classical techniques with uncertainty management capabilities like the generation of conditional plans beforehand that branches off depending on the sensing feedback during the execution. In addition, more recent and realistic approaches have been proposed to handle non-deterministic domains by considering uncertainty as the main principle. In general, the existing techniques in planning under uncertainty cope with problem domains whose complexity depends on three different aspects: (i) the action effects representation (conditional branching or probabilistic results), (ii) the observability scope or the sensing capabilities (full, partial or null observability), and (iii) the expressiveness of the goal formulation, like “try to reach a certain state” (a classical reachability goal) or “maintain some specific conditions” and “guarantee a safe state”, two extended goal definition examples.

We distinguish three different approaches in planning under uncertainty in litera-

ture: probabilistic planning (MDP/POMDP), planning as model checking and uncertainty with classical techniques.

In probabilistic planning, uncertainty is handled by using probabilistic models by assigning statistical values to every action outcome, representing the fact that there are some outcomes that are more likely to occur than others. The Markov Decision Process (MDP), and its extension POMDP (Partially-Observable MDP) [Puterman, 1994], provide a mathematical framework for modelling decision-making processes in situations where action effects are stochastic at a certain extent. Generally speaking, MDP/POMDP approaches deal with non-deterministic problem domains characterized by using probabilities, full or partial observability and extended goals. More concretely, a planning problem is modelled as a stochastic system, i.e., a non-deterministic state-transition system that assigns probabilities to state transitions, modelled as *probability distribution functions*. The planning process is reduced to an optimization problem aimed at maximizing the utility function, where goals are represented by means of an *utility function* and solution plans are *execution policies* that define the action to execute on every possible state. It is worth to mention that most important drawback of probabilistic planning approaches is the *state explosion problem*. Main difference between MDP and POMPS is that the former works under the hypothesis of full observability, i.e., the whole state of the system is completely known in execution time, and the latter assumes partial observability through the introduction of the concepts of *sensing actions or observations*. Some well-known examples of MDP/POMDP-based planners are the following: *real-time iteration algorithm* [Bonet & Geffner, 2000], SARSOP [Kurniawati, Hsu, & Lee, 2008], ZMDP [Smith, Thompson, & Wettergreen, 2007], SPUDD [Hoey *et al.*, 1999] and GPT [Bonet & Geffner, 2001].

Model checking [Nau, Ghallab, & Traverso, 2004] was initially conceived as a technique for software and hardware verification, but more recently has been applied to solve planning problems with some remarkable results. We can mainly distinguish between traditional and symbolic model checking techniques. Traditional model checking approach results more suitable to address deterministic planning problem domains by using a logical representation. In the contrary, symbolic model checking allows to address more complex, non-deterministic problems as it allows to

handle with large finite-state systems in a more efficient way. In general, planning as model checking is a technique to synthesize a plan using a formal model of the problem and a logical goal specification. The planning process consists of a search through the state space of a plan that satisfies a specific goal. Concretely, planning as (symbolic) model checking allows addressing planning problem domains involving partial observability and extended goals, as MDPs-based planning does, but in a model-theoretic manner. The problem is modelled as a non-deterministic state transition system where an action might lead to different states. Reachability goals are expressed in terms of a temporal logic formulae that can be extended to represent more complex goal definitions. Solution plans are *execution policies* like in MDP, defined as an expressive control flow scheme that uses iterative and conditional statements. Model checking policies, as opposed to MDP policies, might not be necessarily defined over all the space of states but over an abstraction of them. Some examples of well-known model checking-based planners are the Model Based Planner (MBP) [Cimatti *et al.*, 2003], SimPlan [Bacchus & Kabanza, 2000], UMOP [Jensen & Veloso, 2000] or TLPlan [Bacchus & Kabanza, 2000].

Some classical planning techniques like plan-space planning, graph-based planning or planning as satisfiability have been extended to cope with non-deterministic problem domains involving some degree of uncertainty. Basically, there are two approaches that allow addressing non-deterministic planning by using classical techniques: first approach simply consists of embedding classical algorithms within a simple execution process that exploits them both to provide baseline solution plans but also to synthesize new ones if plans become invalidated during execution, by simply re-planning from the scratch; second approach extends the scope of classical planner by performing some *transformations/relaxations* on the problem domain to incorporate some of the aspects involved in planning under uncertainty (soft/extended goals, non-determinism degree and sensing capabilities). Attending to the second category we distinguish three different solutions: (i) *Conditional planning*, initially proposed in [Peot & Smith, 1992], extends classical plan-space search algorithms to consider conditional actions, i.e., actions with a mutually exclusive set of possible outcomes; (ii) *conformant planning with satisfiability (SAT) methods* represent another approach to cope with uncertainty by generalizing the expressiveness of the

logic formulation of the planning problems through the use of *existentially and universally quantified propositional variables*; (iii) graph-based techniques have been also enhanced in the same direction so that uncertainty sources are modelled by using a set of planning graphs, one for each possible initial state and non-deterministic action effect. As opposed to planning as satisfiability, planning-graph techniques allow to deal with conformant planning [Smith & Weld, 1998], and partial observability like SGP [Weld, Anderson, & Smith, 1998] and [E.-Martín, Rodríguez-Moreno, & Smith, 2014]. Some well-known examples of conditional and SAT planners are CNLP [Peot & Smith, 1992] and MAXPLAN [Majercik & Littman, 1998] respectively.

2.3 Constraint Satisfaction Scheduling Techniques

Generally speaking, scheduling aims at temporally allocating activities to resources by satisfying a set of constraints *in a cost-effective way*. More concretely, scheduling reasoning processes are targeted at generating plans where starting and end execution times of the activities are assigned (schedules) by considering a set of temporal and resource constraints like deadlines, causal relations or demands of scarce and shared resources. Some additional constraints are typically considered like the minimization of some specific cost criteria like the execution of the action plan as soon as possible (makespan minimization), or the use of the most economic resources allocation.

A resource is defined as an entity that an activity requires for its execution, for instance, memory or energy. In general, activities might require many resources for their execution, and a same resource might be shared by different activities.

According to the stratified definition of P&S given in the introduction of this chapter (see figure 2.2), planning process might precede scheduling by providing an initial partially ordered sequences of activities (action choices are made first). But this is only one of the three existing schemas to integrate AI planning and scheduling (P&S):

- *Stratified P&S*. It is also known as plan-driven approach and represents the simplest and traditional decomposition scheme. P&S are seen as two interrelated but decoupled processes, where causal reasoning (activity ordering choices)

precedes temporal and resource allocation reasoning. This approach is specially suitable for problems where both planning and scheduling activities are clearly distinguished as two loosely decoupled and differentiated processes. Some well-known stratified P&S frameworks are RealPlan [Srivastava, Kambhampati, & Do, 2001] or CRICKEY [Halsey, Long, & Fox, 2004].

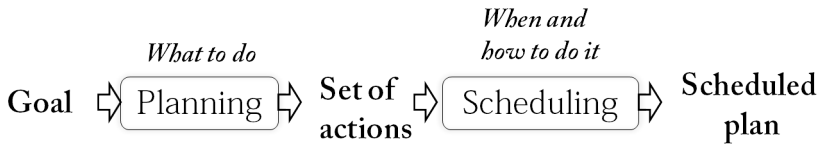

Figure 2.4: Simple Planning and Scheduling decomposition scheme (stratified P&S).

- *Interleaved P&S*. This approach is also known as resource-driven or oversubscribed scheduling: activities compete for the available resources over time. Planning activities are subordinated to scheduling decisions (resource and temporal allocation introduces planning sub-problems). This integration scheme is specially recommended to: (i) very constrained problems with a very restricted number of feasible schedule solutions, or (ii) to problem domains involving shared resources. Constraint satisfaction techniques (described later in detail), are based on this interleave P&S integration schema. Some relevant P&S frameworks are PIPSS [E.-Martin, R-Moreno, & Castaño, 2010], HSTS [Muscettola, 1993], IxTET [Ghallab & Laruelle, 1994] or the APSI framework [Cesta & Fratini, 2008], [Cesta *et al.*, 2009].

- *Composite approach*. This integration schema aims at formulating and solving the P&S problem as a whole in a seamless way. Two different techniques are mainly distinguished in literature: MILP model-based solvers [Richards, Kuwata, & How, 2003] and P&S as satisfiability (SAT). Former technique belongs to the Operational Research (OR)[1] area. SAT-based techniques convert the P&S problem into a boolean satisfiability problem as already described in previous sections.

[1] An interdisciplinary branch of mathematics that uses methods such as mathematical modelling, statistics, and algorithms to solve P&S problems as optimization problems.

It is worth to mention that OR uses mathematical models and reduces the P&S resolution process to an optimization problem. The main drawback of the OR-based techniques is the lack of modelling expressiveness: real-world problems are specially difficult to be formulated and solved by using OR models. In fact, very few realistic problem examples have been successfully addressed by using OR, so the only interesting paradigm that allows addressing P&S problems in a practical way is Constraint Satisfaction Problem (CSP) [Russell & Norvig, 2003].

CSP offers a simple and standard representation of the scheduling problem that allows to take advance of the structure of the state space by using general-purpose heuristics rather than specific knowledge estimators. In general, scheduling as CSP-based reasoners exploits a model of the scheduling problem defined in the following terms: (i) a set of decision variables $X = X_1, X_2, \cdots, X_n$; (ii) a domain D_i of possible values for each variable X_i; and a set of constraints $C = C_1, C_1, \cdots, C_q$, such that $C_1 \subseteq D$, with $D = D_1 X D_2 X \cdots X D_n$. A CSP solution consists of a feasible assignment of domain values to all variables consistent with all the constraints: scheduling as CSP consists of finding a feasible assignment of starting times and resource demands (values) to activities (variables) according to the restrictions imposed by the domain theory (constraints).

We can basically distinguish two different types of CSP solving processes in literature: (i) refinement or constructive search methods and (ii) iterative repair or local search methods. The former assigns values to variables in an incremental way, by checking in every step if any constraint is violated. Iterative repair methods essentially work in the opposite way, i.e., starting from a complete, usually random assignment of values to variables, the process evolves by resolving the existing constraint conflicts.

In both cases, either a systematic (backtracking-based) or a heuristic-based strategy might be applied. Efficient techniques typically use general purpose heuristics like Most Restrictive Variable (MRV) or Least Constrained Value (LCV) for variable and value selection, and look-ahead methods like forward-checking and constraint propagation.

In general, CSP problem solving algorithms typically consist of two interleaved steps (see Figure 4.4), i.e., *constraint propagation* (automatic inference) and *decision*

(heuristic-biased search) steps, executed according to the following iterative search process: firstly, a constraint propagation and temporal consistency checking step is performed to both detect any possible temporal constraint violation and propagate the effects of past decisions to early identify possible inconsistencies (by pruning infeasible solutions in advance); secondly, a decision step consisting on selecting the next variable and value to be assigned is executed. This process is repeated until either a conflict-free plan or an infeasible (non-consistent) solution is found.

Different strategies can be used to provide better solutions by embedding previous solving process within an optimization framework. Most simple (and inefficient) strategy is backtracking but more sophisticated schemas can be applied, like stochastic-based, meta-heuristic biased optimization procedures [Oddi & Smith, 1997].

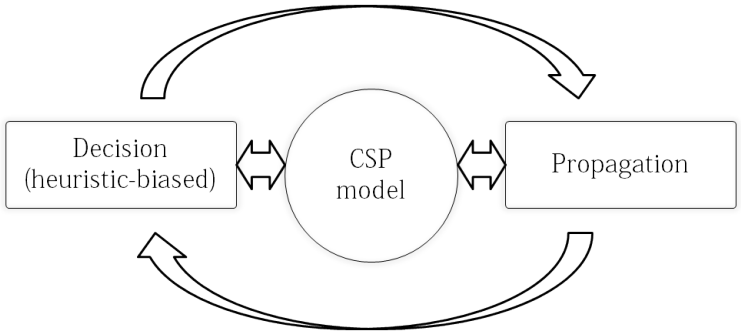

Figure 2.5: a CSP solution is generated by interleaving (automatic) propagation and deliberative steps.

Scheduling problems formulated as CSPs are reduced to assigning temporal values and resource demands (domain values) to both start and end time points of each activity or event (variables). A feasible solution consists of a feasible assignment of all the variables that satisfies a set of temporal and resource constraints.

In the next subsections we analyse more in detail the CSP core concepts from a deterministic perspective, attending to the two different constraints involved, i.e., temporal and resource constraints, and present some significant CSP-based (deterministic) techniques for purposes of illustration. Then, we present a summary of the most important approaches and examples in scheduling under uncertainty.

2.3.1 Deterministic Scheduling

In the current subsection we present the most relevant techniques in deterministic scheduling on the basis of the two underlying CSP sub problems: (i) the Temporal Consistency Satisfaction Problem (TCSP), a CSP where only temporal constraints are considered; and (ii) the resource consistency problem, a CSP generalization where both temporal and resource constraints are considered.

2.3.1.1 Temporal Consistency Satisfaction Problem (TCSP)

The CSP scheduling problems that only involve temporal constraints are typically known in literature as TCSP [Dechter, Meiri, & Pearl, 1991]. In this case, time points are the variables, and the value domain of each variable defines the permitted temporal intervals. Constraints define the boundaries of the value domains as the minimum and maximum temporal permitted values (temporal distance). Additional information can be implicitly extracted from the temporal constraints such as "precedence" or "causal relations".

Simple Temporal Problems (STPs) represent a restricted TCSP version which only admits an interval constraint for any pair of time points. In general, TCSPs problems are modelled by means of a direct constraint graph, known as Simple Temporal Network (STN) [Dechter, Meiri, & Pearl, 1991], where nodes and edges represent time points and temporal relationships respectively. Figure 2.6 shows two equivalent representation of the same STN: the time map (left) represents a more compact view of the directed acyclic graph (DAG) (right).

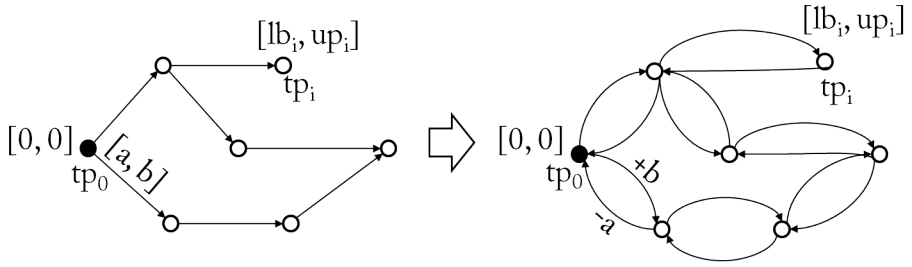

Figure 2.6: Constraint graph equivalent representations.

Given an initial temporally consistent constraint network, and a temporal con-

straint posting/removing operation schema, the *minimal network* (set of feasible solutions) is obtained by applying a *Shortest Path Algorithm* such as SSSP or APSP [Papadimitriou & Steiglitz, 1998]. Typically, a path consistency method which checks that the DAG is free of negative-cycles is applied as preprocessing step to know if a feasible solution can be extracted.

General TCSPs can be addressed by applying a *divide-and-conquer* solution scheme that decomposes the problem into different STPs to be solved by separately, so that the partial solutions obtained are combined later.

2.3.1.2 Resource Consistency Problem

Resource consistency problems are CSPs involving also resource constraints: a resource constraint determines how a given activity demands or affects the availability of a given resource. This kind of CSPs are considered as a generalization of TCPs, as they are aimed at finding a feasible temporal instantiation and resource allocation of all the decision variables (activities), by ensuring that all the temporal and resource constraints become satisfied.

We can mainly distinguish two different types of resources on literature attending to how a given activity demands and affects the availability of the resource: *reusable* and *consumable* resources. The former is also known as *renewable* or *cumulative*, and activities that use them may demand a certain quantity of resource during an interval of time. The sum of all activity demands cannot overpass the maximum level capacity as illustrated in figure 2.7, where the cumulative resource demand of three activities remains below of the maximum capacity. If the resource has unit capacity then it is known as *unary* or *discrete*, and the activities that use it have to be totally ordered since overlaps cannot occur. Examples of cumulative resources are the container of the rover used to transport items as a multi-capacity resource, and the robotic arm or the experimentation facilities as unary resource. The latter is also known as *reservoir* and it is a multi-capacity resource that can be consumed and/or produced by an activity. The battery of the rover is an example of consumable resource that is recharged by the solar cells and consumed by the robotic activities.

Research in CSP-based scheduling has been mainly focused on the development of effective heuristic-biased models that involve unary and cumulative resources,

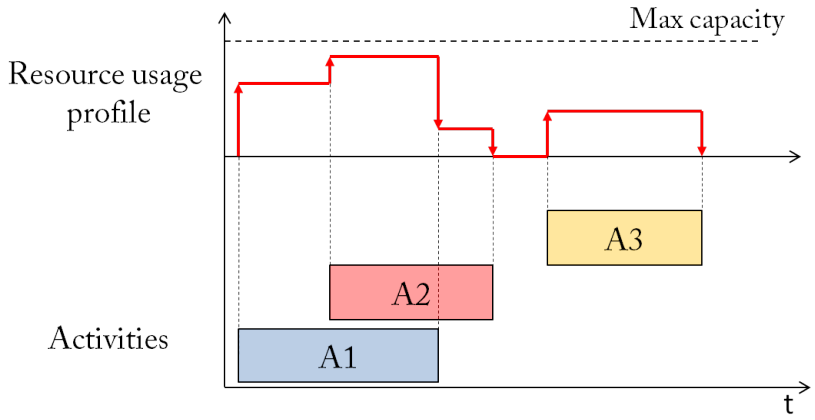

Figure 2.7: Discrete resource profile example.

such as the precedence constraint posting algorithm proposed by [Cheng & Smith, 1994] or the resource profile-driven algorithm ESTA [Cesta, Oddi, & Smith, 2002] respectively. To the best of our knowledge, the majority of the work done in CSP-based scheduling with consumable resources is referred to "resource constraints representation and propagation problems" rather than proposing strategies for problem-solving. Hence, we can find expressive formalisms for expanding the basic Simple Temporal Network (STN) [Dechter, Meiri, & Pearl, 1991] foundations to explicitly deal with resource constraints in general, by defining positive and negative resource allocations to resource productions or consumptions events respectively. Some examples are: the work described in [Muscettola, 2004], that defines an augmented STN called "Activity Network" that allows a *piecewise constant* representation of the resource usage, on top of which an envelope-based resource consistency checking mechanism is built; or the "Linear Resource Temporal Network (LRTN)" proposed by [Frank & Morris, 2007] that enables bounding the resource availability for activity networks with linear resource impact, i.e., permits a *piecewise linear* representation of the resource usage along the duration of the activities.

A different direction to cope with consumable resource constraints proposes a simplified model that aims at reducing consumable constraints to a cumulative scheme by performing some simple transformations. An example is described in [Simonis & Cornelissens, 1995] and formally defines producer and consumer events with a "clas-

sical cumulative scheme" through the following conversion: since a producer event makes some amount of resource available at a specific time instant, such event is expressed as an activity that blocks the resource from the start until the resource becomes available; in the same way, a consumer event is modelled as an activity that uses the resource at a time point and remains blocked until the end, since the resource is no longer available.

More concretely, Simonis' model assumes *stock-based* consumable resources with minimum/maximum values (such as a fuel tank or a storage warehouse), arising in flow shop or job shops environments. In other words, the model takes bounded consumable resource where the total amount of resource demanded/produced is known in advance, and globally will not exceed any level (when scheduled). For instance, within a scheduling system for crude oil transport in a pipeline network that aims at balancing the stock levels for the different sources and sinks. In this case, the whole crude oil may be enclosed within the total capacity of the whole pipeline network (if correctly distributed). To the extent of our knowledge, Simonis' model actually constitutes the only existing approach in literature to model consumable resources in the CSP context.

As previously mentioned, CSP problem solving strategies generally consist on the integration of automatic constraint propagation with (decision) search techniques:

- *Constraint propagation* – Two main families of propagation techniques are distinguished, i.e., Absolute Time Position Algorithms (ATPA) and Relative Time Position Algorithms (RTPA), depending if the absolute or relative (temporal) position of the activities are considered. Well-known examples of ATPA-based algorithms are time-tables, disjunctive constraints, edge-finding or energetic reasoning [Laborie, 2003]. The main limitation of the ATPA-based techniques is twofold: on the one hand, they do not perform constraint propagation until the activities' time windows (i.e., the executional boundaries) are relatively small; on the other hand, precedence constraints (causal relations between activities) cannot be exploited[2]. In the contrary, RTPA-based algorithms take advantage of the precedence constraints, and can perform temporal propaga-

[2]It is worth to recall that most of the partial order planners (POPs) make an extensive use of the precedence constraints

tion regardless of the size of the activities' time windows. Some representative examples are the Energy Precedence Constraint or Balance Constraint algorithms [Laborie, 2003].

- *Search process* – We basically find two well-known strategies for solving resource consistency problems, i.e., *Precedence Constraint Posting (PCP)* [Oddi & Smith, 1997] and start time assignment [Nuijten & Pape, 1998], [Sadeh, 1991]. PCP-based techniques take as decision variables the possible orderings between pairs of activities subscribing a same resource, and the solving process is reduced to post a sufficient set of precedence constraints between contending pairs of activities by satisfying the temporal and resource constraints. The solution provided by PCP-based algorithms are typically known as Partially Ordered Schedules (POS), a type of temporally flexible schedule where each activity retains a set of feasible start times, and every possible temporal solution instance is also a resource-consistent assignment. A remarkable PCP-based example is the Enveloped Based Analysis (EBA) algorithm, that computes the upper and lower bound projections of the resource usage profiles along the POS makespan. In the contrary, start time assignment techniques take as decision variables the set of time points defining the start execution instants of the activities, and the search process focuses on seeking a consistent value assignment for those temporal points. According to this solution schema, two different constraint types are considered, i.e., *binary constraints*, which represent the start-to-start temporal relations between pairs of activities, and the *maximum/minimum capacity resource constraints* of each demanded resource. The ESTA [Cesta, Oddi, & Smith, 2002] algorithm is a paradigmatic example of a start time-based approach that generates a single (fixed-time) solution by reasoning on the earliest start time projection of the schedule. Figure 2.8 summarizes the two kind of solution schedules generated by the two previous approaches: PCP-based algorithms compute the resource usage lower and upper bounds for the set of solutions (POS) (left), while start-time based approaches generate a unique resource utilization profile corresponding to the earliest start time solution (right).

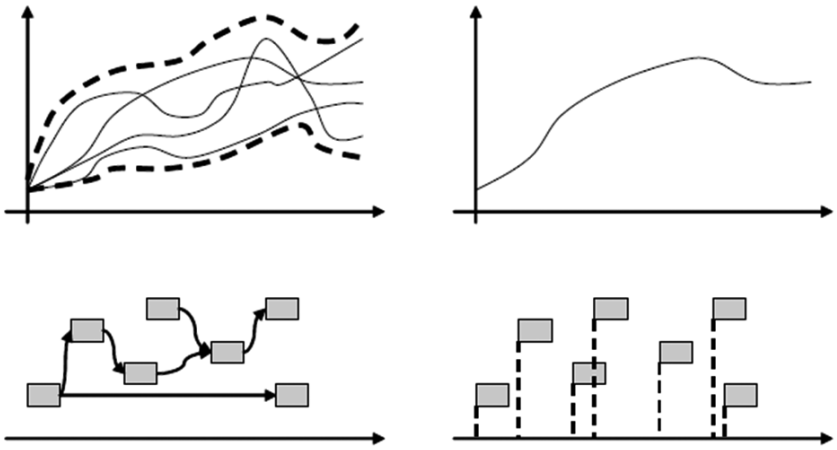

Figure 2.8: Upper and lower bound resource usage computation (left) and resource usage profile projection of an earliest start time schedule

.

It is worth to mention that start-time schedule solutions might be easily converted to partially ordered schedules (POS) by applying the so-called chaining process [Policella *et al.*, 2004]: $ESTA^c$ is an improved version of ESTA that performs a post-processing chaining step to transform the fixed-time solution into a POS. Figure 2.9 illustrates an sketched representation of the POS extraction process according to a pure PCP approach like EBA (left), and the two step procedure that combines an start-time assignment approach with a chaining post-processing step (right). The use of the resource envelope approach (left) entails an iterative reduction of the space of all the possible temporal solutions until this set contains only feasible solutions, by considering all temporal solutions at each stage and trying to select decisions which reduce as minimally as possible this set (i.e. least commitment). The approach on the right side computes a single feasible solution, represented by the point in the search space, and later generalizes the result as a POS (by applying the chaining process). In both cases, the shadow area represents all the feasible schedule solutions (POS).

Next, a summary on the most relevant concepts and techniques in scheduling under uncertainty is presented.

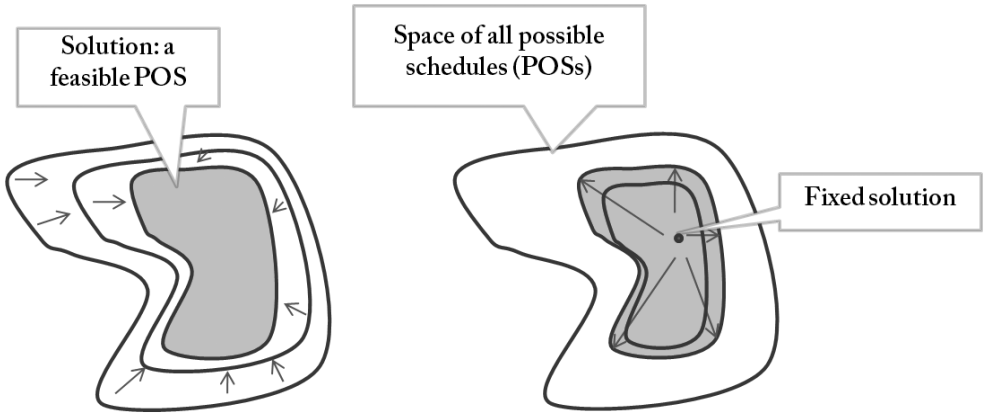

Figure 2.9: An sketched representation of a partially ordered schedule (POS) computation process.

2.3.2 Scheduling Under Uncertainty

The majority of the scheduling research is based on a deterministic view of the scheduling problem, where all the information about the surrounding environment is completely known in advanced. However, in real world problem domains, schedule activities are typically subject to a considerable amount of uncertainty that determines to a great extent the success of their execution.

In scheduling execution, we might differentiate three different sources of uncertainty as illustrated in figure 2.10), i.e., temporal, resource and causal uncertainty: (a) temporal uncertainty might provoke variations on the start times or durations of the activities; (b) resource uncertainty affects to the resource availability or resource usage; and (c) causal uncertainty refers to possible disturbances affecting the order the activities are executed.

Flexibility and *robustness* are two key concepts on scheduling under uncertainty: robustness refers to the capacity of a schedule to tolerate or absorb *in-situ* the effects of possible exogenous events (no re-scheduling action is considered); the purpose of flexibility is to provide one or more alternative solutions to adapt to the different scenarios that might occur in the presence of uncertainty in execution time.

Two different kind of methods can be applied to face with schedule execution under uncertainty:

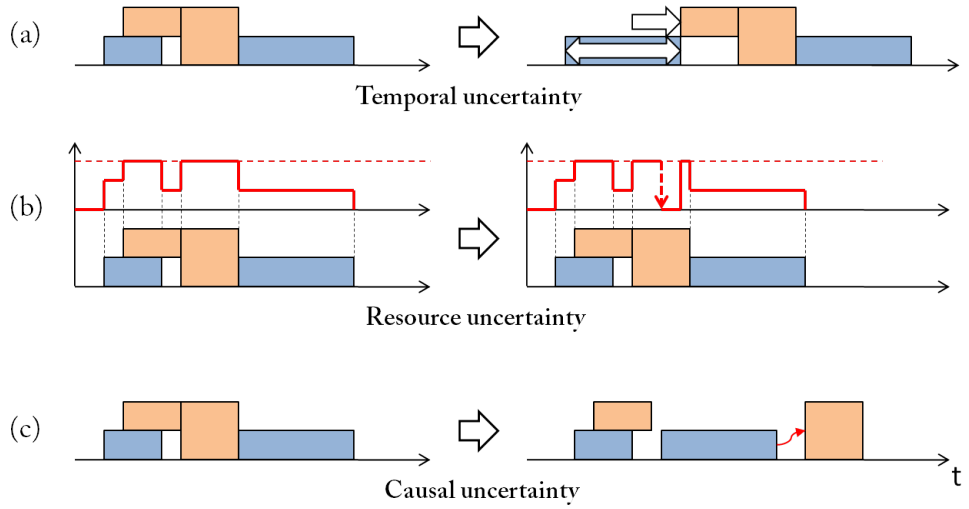

Figure 2.10: Uncertainty sources in scheduling execution: temporal, resource and causal.

- *Predictive techniques* – The underlying idea is to generate either: (i) robust fixed-time based solutions where possible contingencies' effects are considered beforehand (off-line); and (ii) flexible partially ordered solutions (POS) that allow coping with uncertainty in execution time (on-line).

- *Reactive techniques* – These methods consist of (on-line) mechanisms that are triggered in response of the possible disturbances in execution time. Reactive techniques use the scheduler to repair the invalidated schedule (local rescheduling) or provide a new solution from the scratch by considering the new constraints (global rescheduling).

Next, we present three main approaches in literature to cope with schedule under uncertainty from a predictive perspective.

2.3.2.1 Partial Ordered Schedules (POS)

As previously mentioned, an easy but effective way of making solution schedules more robust is to use a POS representation. The most interesting property of POS is that the activities' start and end times are expressed as intervals of feasible values

instead of fixed, unique values. This allows a fast adaptation to temporal uncertainty by simply drifting the activity execution intervals (by means of automatic constraint propagation). It is worth to underscoring that this POS property allows performing "fast-rescheduling", i.e., a reactive execution mechanism that allows fixing the schedule by simply performing a constraint propagation step (see figure 2.11).

Fixed start time schedules can be easily converted to robust POS by using the so-called *chaining process* [Policella *et al.*, 2004], by providing significant improvements with respect to the fixed counterpart solution schedules in terms of adaptability: the validity of the POS schedules is notably increased when disturbed with the same exogenous events in execution time.

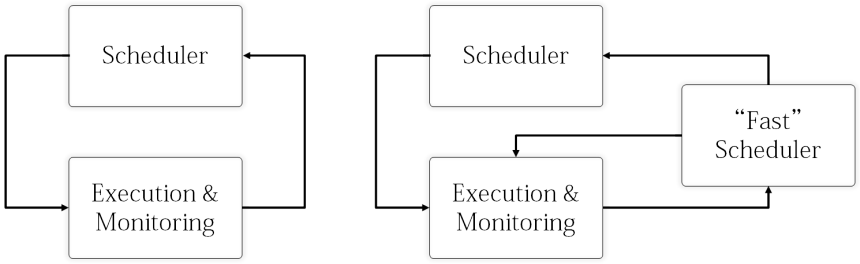

Figure 2.11: (Left) Fixed-time solution execution scheme. (Right) POS-based execution scheme

.

2.3.2.2 Simple Temporal Networks with Uncertainty (STNU)

Simple Temporal Networks (STN) have been extensively used to solve the temporal consistency problem (TCSP) in numerous deterministic scheduling problems. One of the most interesting properties of the STN representation is that it enables "efficient checking of temporal consistency" (by means of All-Pairs Shortest-Path algorithms). But this approach results inappropriate when the success of the plan execution depends on external and uncontrollable events affecting the timing of certain activities. An illustrative example of temporal uncertainty in an spacecraft operation domain is the following: instrument commanding or sensor data acquisition activities typically involves a varying amount of time that cannot be predicted beforehand, as environ-

mental uncertainty introduces delays on the activity durations that are totally out of the control of the spacecraft.

Dynamic Temporal Constraint Propagation (DTCP) problem deals with efficient temporal (and resource) constraint propagation under uncertainty, and represents the counterpart of the TCSP problem for scheduling domains involving uncertain durations that are only known at execution time.

The problem of "control of STN with uncertainty" was formally addressed in [Nicola, Morris, & Muscettola, 2001] where demonstrated that determining *dynamic controllability of STNUs* can be efficiently done in polynomial time, by converting STNUs to equivalent "minimum dispatchable networks" where local propagation (to immediate neighbours only) is sufficient to ensure a valid execution.

Simple Temporal Network with Uncertainty (STNU) is similar to an STN, so that the edges are divided into two classes, i.e., *contingent links and requirement links*. Contingent links model causal relations with uncertain durations, and their connecting timepoints (called contingent timepoints) are out of control (but limited by the contingent links boundaries). Requirement links are the classical links in STN and connect timepoints (called executable timepoints) whose temporal occurrence are under control. It is worth to note that STNU approach only copes with the temporal dimension of the scheduling problem.

In a further study [Rossi, Venable, & Yorke-Smith, 2006], dynamic controllability property was also demonstrated for soft STNUs where preference constraints are also considered.

2.3.2.3 Probabilistic Scheduling

An alternative approach to increase schedule robustness represents activity durations by using probabilistic tools: activity durations are modelled as independent random variables. For instance, in [Beck & Wilson, 2007] presents a theoretical framework for formally addressing the *job shop scheduling problem* when the durations of the activities are random variables, with the objective of finding schedule solutions which have a high probability of having a good makespan. More concretely, the solution proposed uses a Monte Carlo simulation in combination with classical, deterministic scheduling algorithms to find sub-optimal solutions, like branch-and-bound, con-

straint programming and tabu search.

A different work [Frank & Dearden, 2003] presents an study on the complexity of the problem of scheduling tasks that consume uncertain amounts of a resource with known capacity and where the tasks have uncertain utility. Concretely, it describes different probabilistic models based on both open and closed loop executional schemes (i.e., with and without considering sensing feedback) by using resource availability distributions.

It is worth to mention that the central drawback that probabilistic techniques present in general is that the performance used to be drastically reduced with large-sized problem instances.

2.4 Autonomous Control Architectures

Control execution architectures are designed to manage action plan execution processes by guaranteeing that mission goals are successfully fulfilled despite of the incoming disturbances. To achieve that, one of the most important keys is the provision of the necessary mechanisms to allow a suitable management of the complexity involved on the interactions between the control system and the surrounding environment. Some examples that allow to cope with such a complexity are the following:

- *Tractability*: complexity might be reduced through a partition of the overall system functionality into smaller and tractable pieces. A classical solution consists on modular designs that implement the different system capabilities into integrated components by following a layered decomposition scheme.

- *Expressiveness*: control execution architectures should provide expressive representation tools to model core aspects of the problem domain in a natural manner.

- *Robustness and flexibility*: in realistic problem domains, environmental uncertainty might invalidate even the most robust constructed plan. The autonomous controller should be able of identifying non-nominal, unexpected situations and provide effective contingency handling implemented either through robust,

classical fault protection and response mechanisms or by means flexible reactive executional policies.

Next, we summarize some general properties that an autonomous control system should exhibit for maximizing the success and efficiency in plan execution:

- *Action plan dispatching.* Reasoning process typically uses an abstract representation (model) of the problem domain. High-level actions defined in the plan should be translated to low-level command sequences in execution time so that the functional layer can actuate over the robot hardware accordingly. This downward translation of abstract actions to low-level commands (typically known as action dispatching) might significantly influence on the execution process.

- *Least commitment.* A least committed plan usually provides faster and more efficient responses in the face of contingent situations in execution time. This is achieved because most restrictive decisions are delegated to the lowest (reactive) levels in the execution architecture: deliberative actions like replanning are the most timely consuming tasks.

- *Conservative resource usage estimations.* A classical and simple way of increasing plan robustness is to perform pessimistic estimations on resource demands allocations. This way, plans will have scope to absorb a sudden loss of resource availability. It is worth to mention that conservative resource estimations typically lead to plans with longer makespans or temporal lengths.

Attending to the reactivity or deliberativeness of the control system, three main architectural approaches are distinguished in literature: reactive or behavioural robotic systems, deliberative or hierarchical models, and hybrid planning-execution schemes.

2.4.1 Reactive, procedural approach (non-deliberative architectures)

Reactive control architectures represent the most simple kind of control execution as they do not provide any kind of reasoning capability: the execution process basically consists of a sense-act closed loop of plan dispatching and procedural-based, contingency management activities. Pure procedural-driven control systems are typically

modelled as finite-state machines, so that its behaviour is totally deterministic such that for every possible input, a preprogrammed outcome was considered beforehand. Reactivity is implemented by following the same schema: as soon as a (registered) anomalous situation is detected, a rule-based program is triggered to provide a timely response. Control system is decomposed into a set of building blocks called *behaviours* [Brooks, 1990] so that each behaviour encapsulates a specific response to a particular situation. A behaviour typically consists of a simple sensor-motor pair (sense-act), i.e., the automatic execution of a low-level motor reflex action in response of a particular sensor input. The resulting autonomous behaviour of the reactive control systems emerges from the interaction with its environment.

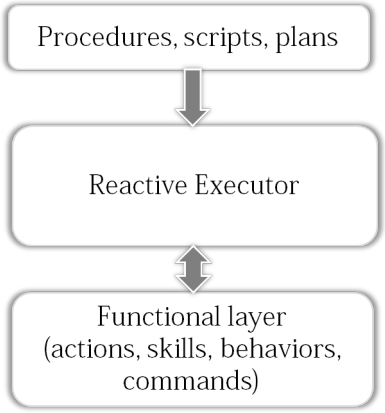

Figure 2.12: Simple reactive (no-deliberative) executive scheme.

Reactive control systems are typically designed by following a modular scheme, by partitioning and organizing the whole system functionality into a set of well-defined behaviours. This kind of control systems are specially suitable in the following situations: (i) if the problem domain cannot be accurately modelled, (ii) the uncertainty level is relatively low and controllable, or (iii) if real-time constraints are a safety-critical issue. On the contrary, procedural-biased control systems do not deal well with real-world problems with high uncertainty levels, as it is completely unrealistic to consider beforehand all possible exceptional cases.

Some of the most representative examples of purely reactive control architectures are presented next:

- *Subsumption architecture* [Brooks, 1986]. It is probably the best-known reactive agent architecture example. The underlying design principle of the subsumption architecture is rooted on the following ideas: the complexity of behavior is due to a combination of embodiment and situatedness. Embodiment is the degree to which a robot can affect its environment, and situatedness is the ability of the robot to sense its environment. As the sensors of a robot become more sophisticated, or even merely greater in number and variety, the robot will become capable of producing more and more complicated behaviours. In this context, an interesting example is an environment in which various qualities that can be sensed are distributed unevenly. In the subsumption architecture, each individual behaviour is implemented as a sense-act rule that basically maps perceptual input to motor actions. A master-program is in charge of orchestrating the execution of the different behaviours in a concurrent manner. Complexity is addressed by organizing the set of behaviours according to a hierarchical, layered scheme, so that lower levels contain high priority behaviours and higher layers more abstract behaviours.

- The *Agent Network Architecture (ANA)* [Maes, 1991] is another well-known example of reactive controller that defines the overall system behaviour in terms of a set of *key competence modules* similarly to the subsumption architecture. Each module consists of the two following elements: (i) preconditions and postconditions defined in an STRIPS-like manner; and (ii) an activation level, i.e. an indicator that dynamically ranks the the relevance of the module over the overall agent behaviour. Modules are organized and connected along a network-shaped structure where nodes (modules) are activated accordingly to provide a specific behaviour. Module behaviours might result either in a motor actuation or in an increment of the activation level of its successor modules.

It is worth to mention the symbolic reactive controllers as a special type of procedural control execution that introduces a simple (still procedural-biased) "decision-making" step in between the sense-act loop, to infer action outcomes from the sensory inputs by using a symbolic representation of the domain model.

2.4.2 Three Layer Architectures (deliberative architectures)

Deliberative execution control architectures are also known as three-layer (3T) systems because their principal feature is that a model-based reasoning step is interleaved on the execution control loop between the sensing and acting steps, and are typically organized hierarchically on three separate layers. Reasoning step (planning and scheduling process) is performed by using a symbolic representation of the world (model).

The control execution process is implemented as a *Sense-Plan-Act (SPA) open-loop cycle*: the autonomous control architecture concept basically consists of an executive enhanced with deliberative capabilities that performs the following basic steps: (i) gathering sensing data, (ii) analysing the data against the model (plan), and (iii) actuating in consequence. Figure 2.13 illustrates the SPA open-loop control scheme concept, so that S_i and S_{i-1} are the current and the previous environment states, i.e., two consecutive states of the environment before and after acting over it respectively.

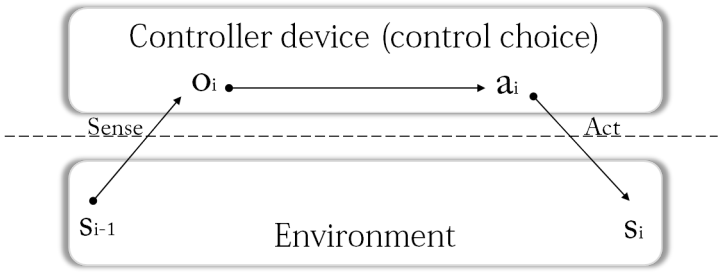

Figure 2.13: Sense-Plan-Act (SPA) open-control loop scheme.

Probably the first and most well-known three-tier architecture example that we might find in literature is the agent called *Shakey the Robot* [Nilsson, 1984]. Shakey implements a control execution process that can be considered as partly reactive and partly deliberative, as it is organized in three differentiated blocks implementing the sense, act and a sort of (program/rule-driven) reasoning steps: the lowest level of the architecture is aimed at providing fast, reactive behaviours to handle timely critical situations where a reflex-like response is required (i.e., stop when touching sensors detect an obstacle). Intermediate layer provides more complex behaviours by using the low level layer as basic building blocks. Lastly, the higher level of the archi-

tecture is targeted at generating plans consisting of a sequence of intermediate level programs. It is worth mentioning that the top layer does not actually implement an AI P&S reasoning process but rule-based programs.

In general, deliberative reasoning control architectures are characterized by the following aspects:

- They are designed following a vertical (hierarchical) scheme where the whole functionality is partitioned and organized in separate levels.
- The control communication flow implemented is completely deterministic and follows either a unique direction from top to down (classical 3T concept) where decisions are translated into actions; or a bi-directional scheme if sensing data is transmitted back to the top (deliberative) levels.
- The higher levels in the hierarchy work by exploiting the functionality implemented on the lower levels.
- The temporal planning scope is different on each level of the hierarchy, so that the lower levels are more timely constrained (they typically implement fast behavioural responses).
- The deliberative layers provide a reasoning process that relies on a symbolic representation of the problem domain (model).

Generally, the classical deliberative 3T architecture (see Figure 2.14) integrates P&S and execution as follows: the bottom block is the functional layer or interface between the control software and the hardware. The higher block is the P&S system, that synthesizes plans as sequences of activities that allow to achieve a set of goals by considering both temporal and resource constraints. Finally the middle layer is the execution system, designed to dispatch low level action sequences or commands (translated from the high-level activities defined in the plan into motor actions) to the bottom level, eventually monitor the execution course and process back the sensing feedback.

Classical (pure) deliberative architectures [Albus *et al.*, 1987] have demonstrated to be very inefficient, particularly in unpredictable scenarios involving environmental

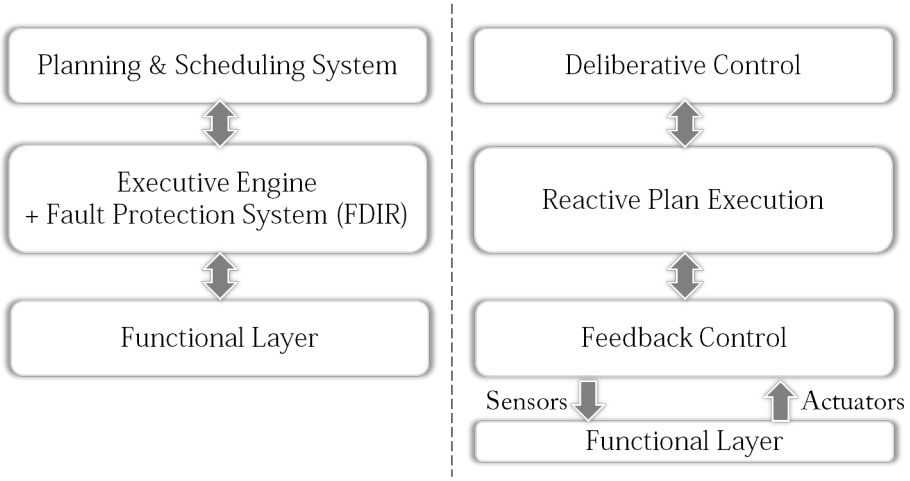

Figure 2.14: (Left) Classical 3T Autonomous Control Architecture. (Right) A modern view of the 3T scheme where the deliberative-reactive integration is emphasized.

uncertainty. Their major drawback is that synthesized plans are invalidated very often during execution as a result of the environmental changes coupled with the low (or totally lack of) reactivity. The solution to this problem consists of closing the SPA execution control loop by providing reactive capabilities [Gat, 1998]: a feedback mechanism creates an ascending communication flow that transmits sensing data to the deliberative levels where is suitably processed. Reactive mechanisms might be implemented in two different ways to face with the detected executional inconsistencies: (a) by using fast but short-sighted methods that provide reflex-like responses, or (b) by transmitting the inconsistencies to the upper (deliberative) levels where a more complex answer is provided to correct the execution deviations. Typical recovery actions consist of repairing or adapting the former plan to the new circumstances, re-planning from the scratch, or simply giving up the original goals.

Main limitation of modern 3T architectures that integrate reactive mechanism to face with uncertainty is their design principle: whole functionality is encapsulated into a vertical, layered scheme, as the controller provides inefficient responses when deliberativeness is required. In addition, each layer is typically implemented by using different technologies that make difficult the integration and validation of the control system as a whole, by entailing for instance functional redundancy that decreases the

reliability of the system.

Some relevant examples of successful deliberative execution architectures are following depicted:

- The Reactive Action Packages (RAP) [Firby, 1987] is actually one of the most representative 3T architectures that follows a modern approach, i.e., that integrates reasoning capabilities with reactive methods. It is decomposed into three different layers namely planner, executive and control system. The planner returns sketchy and high-level sequences of tasks to achieve a specific mission or goal. The execution module timely dispatches the high-level tasks provided by the planner by expanding them into low-level, more detailed actions. Finally, the control layer translates these detailed actions into low-level commands which directly operate the robot's sensors and actuators. Every task is decomposed into methods, and a method is mapped either to an atomic action or to a list of sub-tasks to be expanded. If a task execution fails, RAP reacts by providing alternative methods. It is worth mentioning the ability of RAP to autonomously detect and manage new goals or interesting opportunities, or even foreseeing future contingencies, by integrating this information into the plan in execution.

- ATLANTIS [Gat, 1991] is a 3T heterogeneous and asynchronous architecture which can perform multiple activities in real time by considering noisy and partially observable environments, by integrating purely symbolic planning with reactive capabilities. ATLANTIS consists of a reactive controller, a sequencer and a deliberator. The reactive controller is located at the lower block of the architecture and is in charge of executing and managing collections of atomic actions. The sequencer initiates and finishes sequences of atomic actions, and manages possible execution failures by providing some reactive mechanisms. The deliberator is concerned with planning and world modelling. These initial capabilities were extended in later work with monitoring routines by enabling *cognizant failures*[3] to provide more efficient responses.

[3]The notion of cognizant failure refers to the robot's ability to realize about a task that has not been completely executed

- Tripodal Schematic Control Architecture [Kim & Chung, 2006]. This architecture follows a similar scheme than the previous ones, as it consists of a deliberative, sequencer and reactive layer. The deliberative (upper) layer purpose is twofold: (a) is the interface between the user and the control system; and (b) synthesizes action plans. The sequencer (middle) layer implements a set of asynchronous and non-real-time components that are grouped depending of if they are involved in performing simple execution supervision (monitoring) and low-level configuration tasks, or in extracting advanced information from raw sensory data. Lastly, the reactive layer interfaces with the hardware and implements real-time functionality. The overall execution process is governed by the behaviour coordinator. Two different communication schemas are implemented, i.e., synchronous/asynchronous and pull/push. The former allows to establish a communication channel between two components where the sender remains blocked waiting for an acknowledgement message from the receiver; pull/push mode allows to send status information between two components.

2.4.3 Model-based, planning-on-the-loop methods (hybrid architectures)

The third and most recent architecture design paradigm is the hybrid execution control, and they are mainly characterized by a seamless integration of the automated planning and reactive capabilities in the SPA control execution loop: AI (model-based) planning reasoning techniques are exploited at the core of the control execution process. Figure 2.15 illustrates two different architectural organizations of reactive, planning-centred control systems: the architecture of the left integrates a fast planner for action dispatching in short-time horizon reasoning with a more deliberative planner for complex, time-consuming reasoning; the solution on the right represents an evolution of the previous example, so as planning complexity is partitioned along the whole execution process by following a divide-and-conquer scheme.

Hybrid control architectures actually get the advantages of both the pure deliberative and reactive approaches previously presented, as they allow providing an adjustable reactive process while providing model-based, high decision-making capabilities. Generally speaking, the control execution process starts with a planning phase that provides a baseline solution to accomplish mission goals, and a reactive,

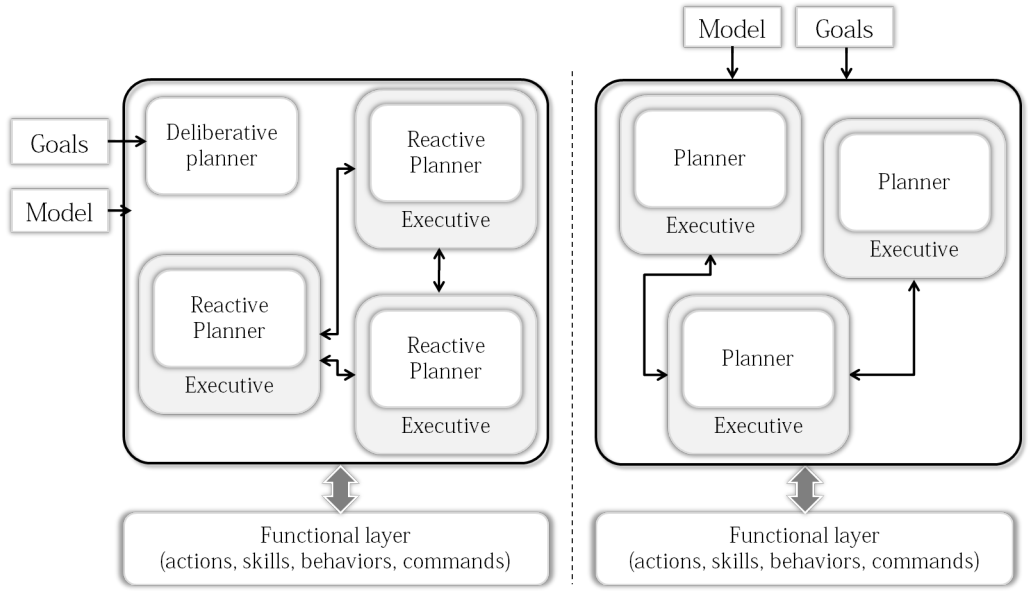

Figure 2.15: Two different reactive planner-based organizations.

planning-centred execution control process dispatches the plan by exploiting planning as an atomic activity by closing every SPA control cycle.

It is worth mentioning that planning centric architectures are typically implemented by using modern planning approaches such as timeline-based planning, which is grounded on the principles of the Constraint-Based Temporal Planning (CBTP) [Frank & Jansson, 2003]. A detailed explanation of the techniques used by the hybrid systems is presented further.

In practice, the main drawback presented by planning-centric architectures is a possible execution latency caused by the iterative calls to the planner on each execution cycle. If planning complexity is not suitably scaled, the execution control process might be delayed or even stuck because of an inefficient response of the planner. Providing advanced planning decisions within a single execution cycle represents a challenging requirement that cannot be always satisfied because of the involved computational complexity. Alternative solutions propose to distribute the control execution process along a set of synchronized agents that provide a specific and bounded level of responsiveness and deliberativeness. For the sake of clarity, we describe next some well-known examples that uses a divide-and-conquer scheme to address the

planning complexity decomposition problem:

- ARMADiCo [Mariela, 2008] is a multi-agent based and general purpose architecture for mobile robots that is organized along the following modules: (i) a *social component* is in charge of the interaction with other agents like robots or humans; (ii) a *deliberative component*, that provides reasoning capabilities; (iii) a *reactive component* that is targeted at addressing possible environmental contingencies; and (iv) a *perception/actuators component*, that deals with the interaction with the physical world by providing the necessary functionality to manipulate the robot hardware. Figure 2.16 shows the different components of the architecture and agents: sensor/actuator's agents are designed to manage the robot actuators and sensor readings; reactive capabilities are implemented as a set of behavioural agents, so that each agent encapsulates a single or atomic behaviour like *go to a point, avoid obstacles, go through a narrow space, etc.*; social capabilities are encapsulated within a single interface agent; finally, a set of heterogeneous agents designed to support the overall multi-agent system are grouped as back agents. It is worth mentioning that, each of the agents can be recursively decomposed as a multi-agent subsystem. For example, each mission planning agent is decomposed in collaborative agents that deal with some specific mission activity planning.

  Agents communication is implemented by means of a distributed coordination protocol that regulates the robot resources usage, by resolving possible resource over-subscription situations (i.e., if a resource is overused by a set of contending agents). Concretely, resource over-subscription is solved as follows: each agent manages an utility function that computes its contribution benefit to the overall system in relative terms, so that the resource is allocated to the most ranked agents. This decentralized communication scheme entails that deliberative decisions provided by every agent are based on both their internal state (individual dimension) but also on the state of the overall system (collective dimension).

- IDEA (Intelligent Distributed Execution Architecture) [N. Muscettola & Plaunt, 2002] is a relevant example aimed at integrating planning and execution under

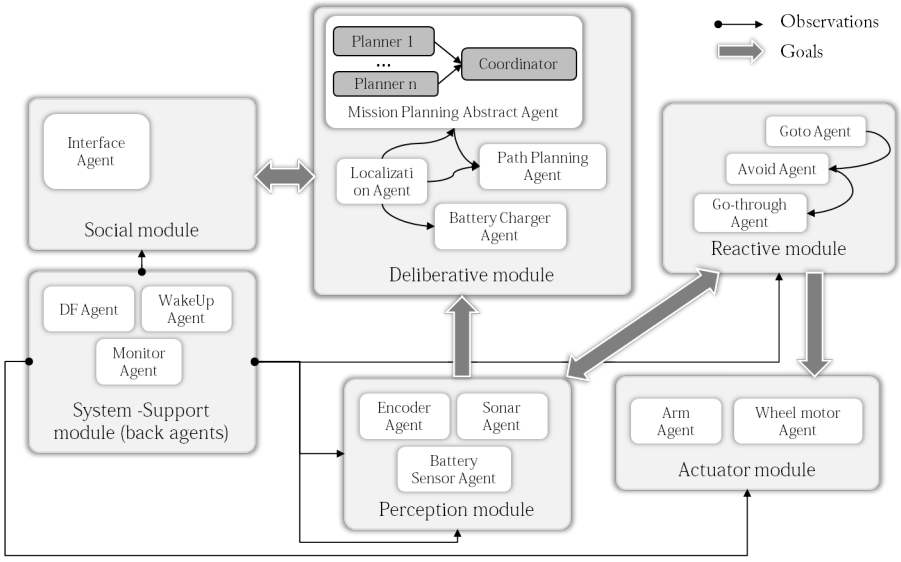

Figure 2.16: A multi-agent-based architecture example.

a seamless framework, by providing reasoning capabilities at the core of the execution engine. The most fundamental principle of IDEA is the decomposition of the complexity of the autonomous control system along a collection of agents suitably organized according to a well-founded topology. Every agent implements either a *reactive or deliberative concrete behaviour*, and interact each other by using a unique same modelling language, regardless of the functional level every agent operates at. IDEA guarantees, in theory, the accomplishment of reaction times if the model is suitably designed.

Concretely, every agent is responsible of the management of a set of timelines, i.e., logic structures which represent the evolution of a subsystem over time that can be executed in parallel. A plan is defined as a set of timelines, grouped and distributed along the different agents. A timeline is represented as a sequence of tokens or atomic actions that are dispatched sequentially.

Figure 2.17 illustrates the basic organization of the robot autonomous behaviour through four main components: (i) the domain model encapsulates the most relevant features of the domain theory as a set of objects (agents) interacting each other; (ii) the plan database stores and maintains the plan in execution;

(iii) the plan executor, aimed at dispatching the plan actions step by step; and (iv) the reactive planner, that suitably responds to incoming events or data in a timely fashion.

Furthermore, IDEA allows the integration of different planners and uses them either in a reactive or deliberative/proactive way, for instance, to anticipate to potential dangerous situations in the future (proactive planning) or simply to ask for the next planning input/goal (reactive planning).

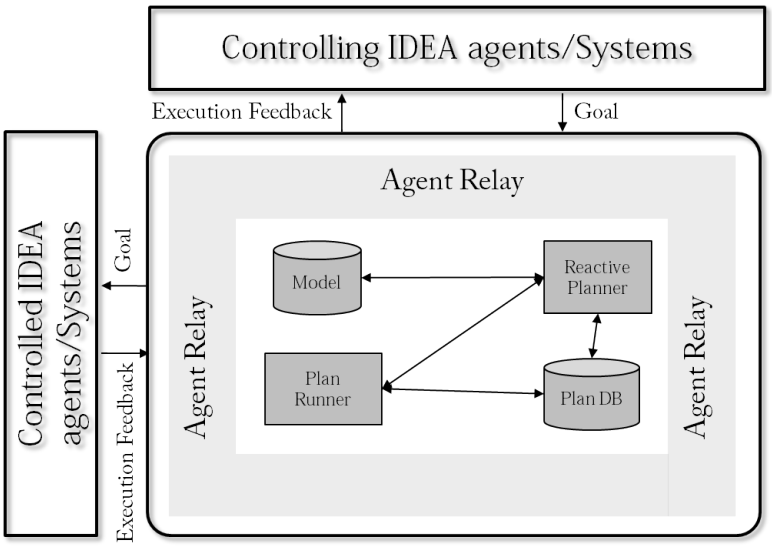

Figure 2.17: General structure of an IDEA agent.

The most relevant IDEA implementation is the EUROPA planner [Jónsson *et al.*, 2000], but other examples that use the same IDEA's underlying principles are the Structured Reactive Controllers [Beetz, 2001], Kirk [Kim, Williams, & Abramson, 2001] or Titan [Williams *et al.*, 2003].

- TREX (Teleo-Reactive EXecutive) [McGann *et al.*, 2008a] was initially developed for oceanographic research purposes with Autonomous Underwater Vehicles (AUVs) [McGann *et al.*, 2008b], and it is basically an extension of the same IDEA principles: an hybrid control architecture that provides a goal and event-driven autonomous behaviour in a unified framework based on a temporal reasoning scheme.

TREX models the world as a set of state variables in a similar way to IDEA, by capturing its evolution in a timeline-like structure. The overall planning and execution process is partitioned and distributed along a set of agents, now referred as deliberative reactors[4] or simply reactors. Every reactor implements a complete and coordinate sense-plan-act (SPA) closed-loop (see Figure 2.18) by means of a set of timelines. Reactors are responsible of the management of their own (internal) timelines, and can observe the evolution of other external timelines (managed by others) and react in consequence.

A key difference between TREX and IDEA is the communication mechanism for state exchanging and agent synchronization, that promotes a better scalability and applicability to different problem domains [McGann *et al.*, 2009]. As illustrated in figure 2.18, communication relay flows in two different directions as goals (outcoming) and observations (incoming). In addition, TREX provides a improve planning interface specification that allows to integrate external planners like EUROPA in a more flexible and efficient way.

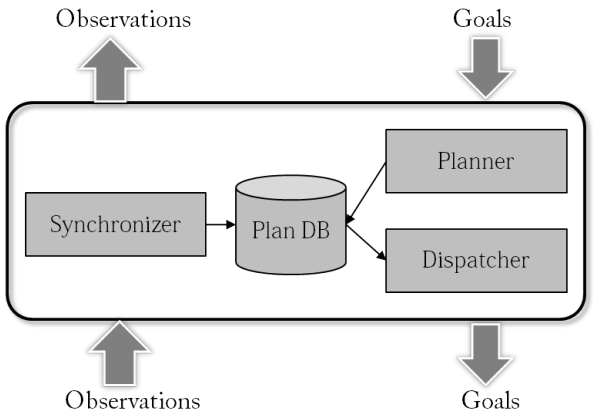

Figure 2.18: Internal structure of a Deliberative Reactor.

More concretely, a reactor is defined by means of the following main components: (i) a plan database that contains the plan as a set of timelines, as well as a set of additional data structures and algorithms for maintaining the overall plan

[4]A Reactor or a Deliberative Reactor is a special case of the more general idea of a Teleo-Reactor [Nilsson, 1994]

integrity and consistency; (ii) a synchronizer, that updates the plan database with the new observations; (iii) the planner, targeted at both monitoring the plan under execution and providing alternative responses to possible flaws or inconsistencies (arising from new goals or constraints); and (iv) the dispatcher, that executes the tokens accordingly. It is worth mentioning that both the plan database and the planner components are provided by EUROPA.

Figure 2.19 shows the TREX architecture layout for the MBARI AUV project, where the overall functionality is decomposed into four different reactors (left), i.e., mission manager, science operator, navigator and executive. On the right side we can observe with more detail the set of timelines managed by the mission manager, navigator and executive reactors (both internal and external), as well as the concerned data flows.

- VOMAS [Hsu & Liu, 2007] is another remarkable example of hybrid multi-agent architecture. Its design layout consists of three basic agents, i.e., the virtual operator (VO), the robot agent (RA) and the user agent (UA). The VO agent implements deliberative control and hardware-independent functions; RA agent encapsulates reactive control and hardware-dependent functions; and UA agent is the interface between the user and the VO agent. The control execution process starts with the transmission of a new mission task by the user through the UA (user context), so that a new VO agent is instantiated ad-hoc to manage the new task. Then, the new VO agent is passed to the robot context so that the VO and the RA agents cooperate to execute the task by using the agent communication language (ACL). If the robot requires to change the task in execution (e.g., because of an contingency), it simply replaces the VO associated to the task in execution with a different one. The RA uses a simple behaviour-based scheme to implement the different reactive behaviours. Two real applications using VOMAS architecture can be found in literature: a wheelchair robot and a formation control (multi-robot) system. The former application, RA agent uses DAMN [Rosenblatt, 1995], a purely behaviour-driven controller that implements the following behaviours: obstacle avoidance, path-following motion and goal seeking. Second example uses a RA agent based on

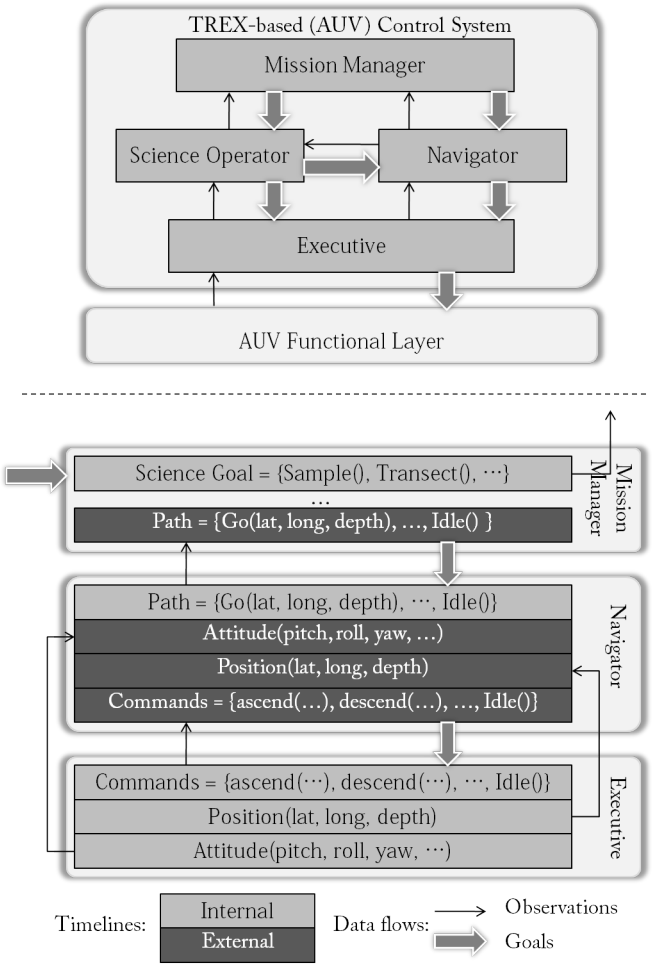

Figure 2.19: TREX architecture organization of the MBARI AUV project.

a similar behaviour-driven controller (namely motor-schema) that implements multi-robot behaviours like robot avoidance or predecessor following.

2.5 Experiences on Autonomous Control Techniques to Space Robotics

In the current section, we take an outlook to some relevant approaches in autonomous control within space domain applications. The aim is to justify and emphasize the

main contributions here presented by giving a clear picture of recent work in literature on the promotion of applied autonomy in the context of space robotics, with special attention to those enabling on-board mission task-level planning and execution in dynamic and uncertain environments.

Robotic systems with primitive degrees of autonomy, like fault detection, diagnosis and eventual recovery actions (FDIR), have been used since the beginning of space exploration, but only most recent planetary surface exploration rovers have autonomously performed relatively simple operations involving *navigation* and *instrument placement* tasks (see [Mishkin *et al.*, 1998a], [Maimone, Leger, & Biesiadecki, 2007]).

Human operators continue playing a central role on the mission execution control loop, where on-board autonomy is still used in a very conservative fashion. We might find out a few real experiences in on-board autonomy applied to space exploration. The validation of the Remote Agent Experiment [Nayak *et al.*, 1999] (RAX), where the Deep Space 1 flight software deployed autonomous mission plan generation and execution on-board the spacecraft. The *Autonomous Science Experiment (ASE)* [Chien *et al.*, 2005] onboard autonomous flight software was deployed on the Earth Observing One (EO-1) spacecraft [Chien *et al.*, 2004] to enable autonomous science by exploiting the onboard *Continuous Activity Scheduling Planning Execution and Replanning (CASPER)* [Sherwood *et al.*, 2001] planner, generating mission operation plans and providing rapid responses to a wide range of operation scenarios. The *Mixed-initiative Activity Plan GENerator (MAPGEN)* [Bresina *et al.*, 2005], an automated system which exploits constraint-based planning, scheduling, and temporal reasoning techniques, was used to support the Mars Exploration Rover (MER) operations team on the rover activity planning tasks. Another example is represented by the *Mars-Express Scheduling Architecture (Mexar2)* [Cesta *et al.*, 2007], an AI-based tool currently in use on the Mars-Express mission to plan data downlinks from the spacecraft to Earth.

With the exception of the previous examples, we have to refer to field demonstrations (in Earth) of advanced autonomy.

2.5.1 LITA: towards autonomous science for future Mars exploration

Researchers of the Carnegie Melon University (CMU), in collaboration with NASA, conducted three different field experimentations in the context of the *Limits of Life in Atacama (LITA)* project [Wettergreen *et al.*, 2005b], where a mobile robot was targeted at characterizing the presence and distribution of microbiological life on the desert of Atacama (Chile). The project was chartered at exploring technological solutions to multiply the science data return possible in future planetary missions, while at the same time conducting useful research into the extremes of life on Earth. Concretely, mission requirements of the LITA field investigation were rooted on the demonstration of key concepts in *on-board supervisory control* by considering analogous operational conditions to a Mars mission, where a rover is in charge of autonomously visiting multiple, widely-dispersed (in a kilometre-scale) science targets, in a single command cycle.

First field campaign [Wettergreen *et al.*, 2005b] used a solar-powered robot originally designed for Sun-synchronous navigation in polar latitudes [D.Wettergreen *et al.*, 2002], to test re-planning capabilities in the context of rover operations and plan stability, with special emphasis on bringing the gap between local and global navigation. More concretely, during this first experimental season the rover deployed a number of technologies including: TEMPEST [Tompkins, Stentz, & Whittaker, 2004], a mission-level path planner which provides plans and re-plans comprising locomotion, solar charging and hibernation activities; local navigation for waypoint-to-waypoint route traversals and obstacle avoidance; health (resource) monitoring to detect abnormal state conditions; and a rudimentary mission executive for traverse command dispatching.

During the second and third field seasons [Wettergreen *et al.*, 2005a], [Wettergreen *et al.*, 2008], autonomous control architecture evolved from a previous setting mainly focused on addressing obstacle avoidance, resource monitoring and demonstrating some initial concepts on mission-level planning, to an autonomous operation where both navigational and scientific activities [Smith *et al.*, 2007] are considered. Concretely, the newest rover version was equipped with the Rover Executive (RE) [Baskaran *et al.*, 2006], an execution system which uses the "Intelligent Distributed Execution Architecture" (IDEA) [Aschwanden *et al.*, 2006]. RE was mainly

aimed at both: (i) requesting and executing a nominal plan generated on the fly on the accomplishment of high-level goals like commanding the rover to move to a specific location, approach to an area of interest and gather scientific data, or deploy a plow to extract some sub-surface element; and (ii) handling faults by coordinating a recovery process from off-nominal conditions. Recovery actions generally consist of simply ceasing the rover activity until the fault gets cleared, but also might involved re-planning actions. It is worth to say that although RE also supports opportunistic science (i.e., the nominal plan might be modified in order to accommodate on the fly new science actions), this feature was not actually deployed during the field experimentation.

To the best of our knowledge, both the mission concept underlying LITA and its approach to cope with autonomous control is probably the closest existing (and best documented) example to the work here presented. With the aim of highlighting the relevance of our contributions, next we mention some of the main differentiating aspects in both solutions:

– In LITA, the general design approach follows a "bottom-up" strategy where the autonomous capabilities of the control architecture are integrated according to an process which incrementally addresses the different sub-problems like path-planning, instrument placing, mission-level planning, execution monitoring, contingency solving, etc. In contrast, our solution is rooted on a "top-bottom" approach which makes a wholesome integration of advanced reasoning capabilities within a flexible control execution process, and exploits a model of the world that encapsulates all the main problem ingredients according to a unified and seamless scheme.

– While our solution is founded on a suitable and efficient use of all the mission resources with special emphasis on the energy requirements, in LITA we found some limitations on this aspect. As far as we know, battery charging periods are not explicitly modelled, so that the autonomous controller does not perform well on coping with multi-day continuous autonomous operation where dawn, dusk and night-time activities might force projecting extended battery charging and low-power hibernation periods.

2.5.2 Back to Moon: supervised operation with utility rovers

In preparation for returning humans to the Moon in a short-term future, there is also a significant amount of work promoted by NASA on the development of autonomous exploration systems called "utility robots" for supporting humans on their lunar surface operations. Generally, utility robots are targeted at performing highly repetitive and long-duration tasks, such as site-mapping or science reconnaissance, both in a teleoperated and supervised (i.e., autonomously) modes.

In 2007, two K10 rovers were equipped with a variety of science instruments and deployed to map several sites at Haughton Crater in Devon Island (Canada) [Fong *et al.*, 2008], as part of the annual Haughton-Mars Project (HMP) field season. During the three-week field test, rovers performed more than 200 hundreds of survey operations consisting on driving kilometres between sites of interest to collect imagery data. Ten percent of these operations were conducted on a supervised mode where locomotion, localization, navigation and panorama acquisition activities were performed autonomously by using facilities provided by the *Coupled Layer Architecture for Robot Autonomy* [Nesnas *et al.*, 2003] (CLARAty) planning and execution framework, like the "Universal Executive" for plan dispatching and contingency handling.

In a similar context, the CMU (in collaboration with NASA) conducted in 2008 two field tests [Wettergreen *et al.*, 2009] with Scarab, a prototype rover conceived for lunar missions to survey resources in polar craters at lunar analogue sites, in Washington and Hawaii respectively. Concretely, the experimental results reported mobility and navigation capabilities on the execution of autonomous kilometre-scale traverses in darkness to inspect different locations, drill and analyse soil samples. Scarab executed a serialized mission plan (manually) defined beforehand, and combines laser range scanners with autonomous navigation algorithms to build models of the surrounding terrain, detect obstacles and then determine efficient and safe paths.

The main concern of the previous campaigns was mostly on the deployment of advanced autonomous control modes, as they mainly focus on the demonstration of technology concepts like autonomous (kilometer-scale) navigation and instrument utilization under challenging terrain and light conditions.

2.5.3 The LAAS autonomous control architecture

The Laboratory for Analysis and Architecture of Systems (LAAS), a research group associated to the Centre National de la Recherche Scientifique (CNRS) in Toulouse (France), has been conducting a significant research work for years, in the development of a framework to integrate deliberative planning with control execution capabilities. According to the results presented in [Ingrand *et al.*, 2007], the LAAS autonomous control architecture was integrated and tested on-board of the robots *Lama* and *Dala*, two experimental rovers fully equipped with advanced sensing and processing capabilities (like 3D odometry, visual motion estimation, stereovision and panoramic imaging), with the aim of demonstrating advanced planning, scheduling and control execution capabilities. LAAS integrates a temporal planner and an execution controller that exhibit plan repair and replanning capabilities to tackle with executional uncertainties. In the context of a rover planetary mission scenario, the planner has demonstrated action plan synthesis that include navigation, science activities (moving and operating instruments), communication with Earth and with an orbiter or a lander, while managing a wide set of resources (like power, memory, etc.) and temporal constraints (communication visibility windows, rendezvous, etc).

Concretely, the LAAS architecture design principle implements a generic control execution framework (based on the classical Sense-Plan-Act loop model) according to a layered decomposition of the robot system in three differentiated levels:

- The *functional layer* embeds all the basic built-in robot action and perception capacities for enabling the rover to perform different modes of autonomous navigation depending on the complexity of the terrain geometry (flat or easy terrain and 3D modes) and the involved distances (long-range traverses and target reaching[5] modes). The selection of the navigation mode to be applied depends on the given goal and environment context and it is decided by the *navigation planner*.

- The *decisional layer* implements the abilities of producing and supervising (controlling) the execution of the task plan through the integration of both

[5]In target reaching mode, the robot makes an accurate approach to a designated science target that is on the rover line of sight

the IXTET planner [Ingrand *et al.*, 2007] and the OpenPRS [Ingrand *et al.*, 1996] procedural executor. On the one hand, IXTET performs plan synthesis and plan adaptation (in case of contingencies) while considering temporal and multi-capacity resource constraints. On the other hand, plan execution control is implemented through two different processes, i.e., temporal and procedural control. Temporal execution control is handled by IXTET by temporally dispatching the plan actions. OpenPRS procedural executor is chartered at: (i) expands the action dispatching by transforming the high level actions into lower level sets of low-level commands (procedures) to be processed by the functional layer; (ii) monitoring the execution outcome; and (iii) reacting against contingencies by either recovering from a set of specific failures (by triggering recovery procedures), or by reporting the inconsistencies to the IXTET planner. It is worth to mention that the IXTET plan recovery strategy attempts first to repair the plan in execution so that if the plan cannot be repaired, a replanning from the scratch process is executed.

- The *execution control* layer is the interface between the decisional and the functional layers, and it is a fault protection system basically aimed at preventing the rover to enter in faulty states which might lead to fatal consequences. For instance, to prevent the rover performing incompatible actions like reorienting the antenna while a communication cycle is taking place.

LAAS architecture represents a relevant example in applied autonomous control as it actually demonstrated high-level autonomy capabilities through a simple case of study. Despite LAAS integrates reasoning capabilities that consider complex resource constraints, reported results do not provide insights on how *renewable* resources (like battery or data storage resources) are managed during plan synthesis in terms of resource production/consumption estimations.

In contrast to the wholesome integration schema of the reasoning and control execution capabilities in a seamless and unique process presented in this work, LAAS architectural design is rooted on a modular approach that implements the robotic system capabilities in a set of integrated modules organized along a layered scheme. For instance, the control execution process is implemented through two different modules

that use different data and models.

Moreover, there are also many significant studies relatively to the promotion of autonomous technology in robotic missions. For instance, Wood *et al* [Estlin *et al.*, 2005] present an integration between the *Closed-Loop Execution and Recovery (CLEaR)* system and the *OASIS* onboard science system which supports onboard decision making capabilities for mission plan generation, execution & monitoring, dynamic re-planning, optimization, and even opportunistic science. The European Space Agency (ESA) is also fostering similar works in the context of the forthcoming ExoMars [van Winnendael, Baglioni, & Vago, 2005] rover mission. In this direction, the ESA's study projects GOAC [Ceballos *et al.*, 2011] and IRONCAP [Steel *et al.*, 2012] aimed at, respectively, demonstrating key concepts towards fully autonomous onboard operations for rover-based missions, and defining the basis for developing a prototype system to support the whole science and engineering onground planning activities of an interplanetary rover.

2.6 Summary

In the current chapter we provided a detailed overview of the underlying challenges and existing solutions in autonomous mobile robotics along three different sections: planning techniques, resource constrained scheduling techniques and autonomous control architectures. A last section called "experiences in autonomous control techniques to space robotics" closes the chapter with some recent and relevant projects that exhibit advanced autonomy in the context of space robotics.

Chapter 3

Tools Used in this Thesis

3.1 Introduction

In this chapter we explain in detail the two main tools used to the development of the different software artefacts in this dissertation: the Advanced Planning and Scheduling Initiative - Timeline Representation Framework (APSI-TRF) and the planetary rover system simulator 3DROV.

3.2 Scheduling Software Development Environment (APSI-TRF)

Design and implementation of advanced Planning and Scheduling software for space applications is an activity involving a certain amount of developing effort and risk. For instance, the software may fail to meet operational requirements (performance issues), and/or may fail to capture all the essential aspects of the problem (modelling issues). The ESA Advanced Planning and Scheduling Initiative aims at the design and deployment of a software platform, called APSI Timeline Representation Framework, shortly APSI-TRF, for supporting planning and scheduling space applications design. The aim of the framework is to provide help to developers to cope with both software deployment efforts and modelling risks.

Main features of the APSI-TRF framework are summarized as follows:

- The APSI-TRF simplifies the design and developing effort by providing a library of basic planning and scheduling domain independent solvers, and robustifies the interaction among the specific solvers implemented on top of the framework by providing a uniform representation of the solution database and defining a common inter-module cooperation and coordination interface.

- Modelling risks are reduced because the use of a framework that standardizes and simplifies the process of application deployment fosters a rapid prototyping cycle, which directly helps the users to take into account their own feedback during the application design.

The philosophy underlying the APSI-TRF is inspired by classical Control Theory, in that the problem is modelled by identifying a set of relevant features whose temporal evolutions need to be controlled to obtain a desired behaviour. In the APSI-TRF such relevant features are called components and are the primitive entities for knowledge modelling. They represent logical or physical subsystems whose properties may vary in time; therefore, control decisions can be taken on components to define their evolution.

The APSI-TRF allows representing a number of planning and scheduling concepts in the form of timelines. The current APSI-TRF release provides two types of components which allow quite an amount of modelling power. Specifically, planning problems are modelled using components known as multi-valued state variables [Muscettola, 1993], coupled with resource components like those commonly used in constraint-based schedulers [Cesta, Oddi, & Smith, 2002]. Besides problem solving capabilities, the APSI-TRF also provides a domain definition language (called DDL.3) to specify both the components and the relevant physical constraints that influence their possible temporal evolutions (e.g., possible state transitions over time of a component, synchronization/coordination constraints among different components, maximum capacity of resources, etc.), as well as a problem definition language (called PDL) to specify planning and scheduling problems.

APSI-TRF's architecture design is organized along three different levels (see figure 3.1): a *Time/Parameters* layer, a *Component* layer and a *Domain* layer. The TRF design reflects the approach induced by the component based approach. The planning

domain is modelled as a set of concurrent threads (the timelines) and the problem is to synthesize a set of decisions to obtain a desired behaviour and to synchronize the threads. Hence the APSI-TRF structure, where a common lower level represents the information shared among the timelines, temporal information and parameter information, a middle level that represents the extension point where the modeller plugs the components, and an upper level that provides a unified, shared representation of the plan.

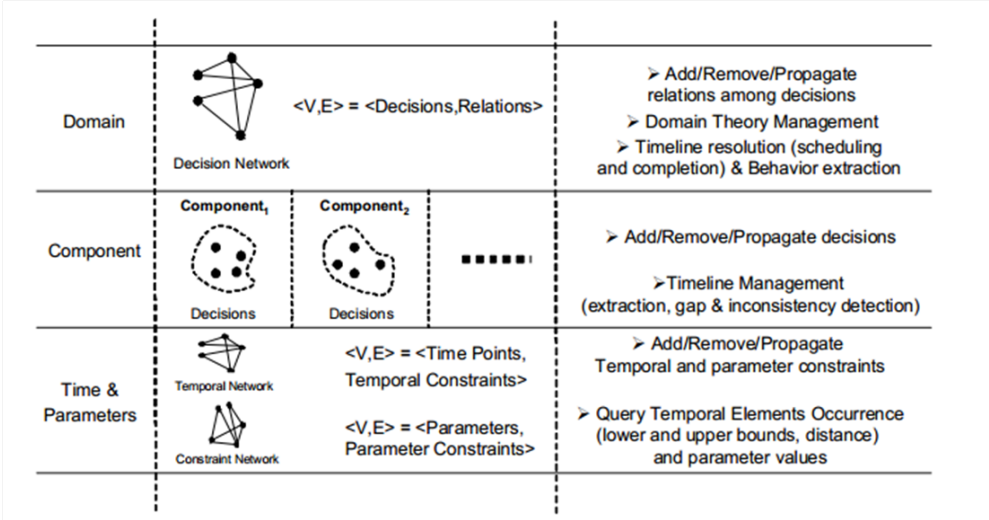

Figure 3.1: The layered implementation of the TRF.

3.3 Planetary Rover System Simulator (3DROV)

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" [Poulakis *et al.*, 2008] (3DROV), an advanced software simulation platform created to the aim of providing ESA's A&R section[1] with a supporting tool for the

[1]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.

complete development process of planetary robot systems within *realistic mission scenarios*.

One of the many interesting features of 3DROV is that it provides the necessary elements to accurately reproduce both the robotic systems and the environmental surrounding aspects involved within a planetary mission operation context. More concretely, main 3DROV features are summarized as follows:

- The 3DROV system provides an end-to-end simulation capability, that is, the simulation tool is able of recreating a mission scenario from early stage of its definition until its completion. This is achieved by offering the capability of incorporating models of scientific instruments (and the rover itself) interacting with the virtual environment to simulate the daily scientific outputs of a given scenario.

- 3DROV adopts the ESA SIMSAT 4.0 simulation framework for spacecraft and ground segment simulations and training, in order to take advantage of the already existing technology. Furthermore, the models developed for 3DROV complies with standardized model interfaces such as the "Simulation Modeling Portability 2.0 (SMP 2.0)" standard. This standard was introduced by ESA to aid the portability of simulation models developed in the context of the European space industry.

- The rover physical models include mechanical, power and thermal subsystems features as well as a dynamically consistent contact element which allows specifying the interaction between rover subsystems and environment. The novel port-based methodology [Poulakis *et al.*, 2008] has been chosen for the rover physics-based models specification and implementation, which offers an intuitive energetic model structure both within and between the different domains.

- 3DROV supports different levels accuracy of its internal models representation. The system allows the user to reduce the accuracy of its models with the aim of accelerating the simulation execution, for instance, to reproduce a mission-level simulation where the details of the rover physical subsystems behaviour is not of importance.

It is worth to mention that 3DROV system can also be used as a test-bench for on-board controllers and ground station modules. As a result, 3DROV was designed by following a modular scheme (see Figure 3.2) with well-defined interfaces to allow an easy integration with external modules like a testbench. The simulation framework is basically organized along five integrated building blocks:

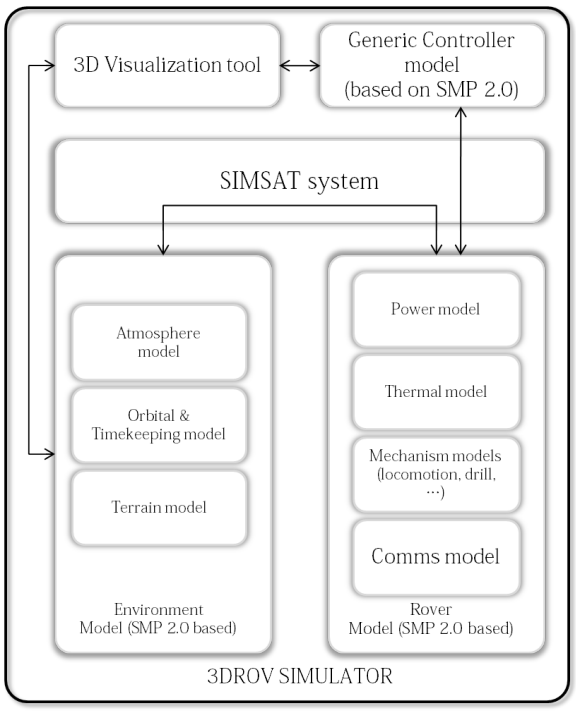

Figure 3.2: The 3DROV simulator architecture scheme.

- The "Environment Model subsystem" provides an accurate reproduction of the environment in which the rover operates. It includes terrain and atmospheric models, ephemeris, a rover positioning and timekeeping system. Terrain models includes information about the topography, soil mechanics, optical and other specific properties required for the operation of the different rover instruments.

- The "Rover Models" recreates with accuracy a variety of rover and environmental features such as the rover dynamics, kinematics, soil terrain interac-

tions, power, thermal, communications, scientific instruments and sensors.

- The ”Generic Controller subsystem” is the on-board autonomous software that controls all the rover operations. It implements a A&R control system in compliance with the Functional Reference Model (FRM) of the CDM framework.
- The ”3D Visualization component” represents the front-end of the 3DROV simulator framework and was developed with the two-fold aim of (a) providing (a real-time) visualization of the simulation execution, and (b) assisting the control station with the preparation of the A&R system activities.
- The ”Control Station subsystem” was designed in compliance with the A-DREAMS ground control system, and serves as the virtual rover's ground control station. It provides the necessary elements to define mission plans in terms of rover activities, send the telecommands to the rover, and receive/display the on-board telemetry.

3.4 Summary

In the current chapter we presented the basic ingredients used for the development of the software artefacts: (i) the ESA Advanced Planning and Scheduling Initiative aims at the design and deployment of a software platform, called APSI-TRF, for supporting planning and scheduling space applications design; and (ii) the 3DROV tool, an advanced software simulation platform created to the aim of providing ESA's A&R section with a supporting tool for the complete development process of planetary robot systems within *realistic mission scenarios*.

Part II

Robust Constraint-based Robot Action Scheduling

Chapter 4

The Constraint-based Solving Algorithm

In this part of the dissertation we start with a discussion about the necessity of infusing AI techniques to space robotics in interplanetary missions. Next, we explain the MSR mission concept as a plausible scenario where advanced reasoning capabilities can be applied to the exploration of Mars in the near future. Then we introduce a MSR-inspired scheduling problem PARC-MRS as a formalization of the most significant MSR mission requirements into a comprehensive scheduling problem model. Finally, we present in detail the profile-based, power-aware reasoning algorithm $ESTA^p$ developed with the aim of providing feasible solutions for the PARC-MRS problem, as well as a meta-heuristic strategy for solution optimization.

4.1 Introduction

Traditionally, AI P&S techniques has been successfully applied to *mechanistic problems* that assume manufacturers-like conditions. Classical AI-based autonomous control systems are rooted on the conservative control theory basis which pursues an absolute and flawless governance, even in the face of non-nominal situations: contingency plans are designed in advance for every anomalous situation. But a new form of AI P&S systems that considers *uncertainty* as the main and partially controllable principle is increasingly demanded, since *real problems* are becoming scientifically

interesting. With real problems we mean, close-to-human-intellect tasks such as *fluidly plan movements to avoid obstacles to achieve a specific goal*: a challenging AI P&S problem that claims more advanced capabilities than the actually provided by classical approaches, such as efficient mechanisms for managing complex and diverse temporal and resource constraints (e.g., energy demands, restrictive deadlines and resource availability, etc.).

Space exploration is actually self-proclaimed as a scientific discipline that is driving such paradigm shift, by disrupting old-fashioned deterministic rules that considered everything guessed in advance, in expenses of a more flexible and robust autonomous control systems.

Particularly, ESA is currently bumping into different real problems that intrinsically hold uncertainty at its core definition: the increasing interest in evolving current telerobotics capabilities towards a complete mixed-initiative [Crandall & Goodrich, 2001] strategy implementation that enables shifting control modes from purely manual to fully autonomous mission operations. Intelligent error handling mechanisms coupled with basic high level mission planning techniques currently represent the state-of-the-art in "A&R". Continuing to promote autonomy for future space missions certainly entails enormous benefits such as the reduction of operational costs, the enablement of opportunistic science, or the increase of mission quality in terms of safety, science return, reliability and flexibility. Very often, for some space missions such as deep-space probes, ground intervention may be slow (at the best cases) due to propagation signal delays and low bandwidth, or even impossible because of the occlusion of the line-of-sight communication when orbiting remote planets [Muscettola *et al.*, 1998].

In order to formally cope with the infusion of autonomy in space mission operation, ESA has established a reference decomposition scheme (ECSS Mission Execution Autonomy Levels[1]) (see table 4.1) that sorts out the different flavours in automation applied to space mission operation in four well-founded levels (ranging from *dummy* or manually controlled to fully autonomous robots).

It is basically a "functional authority" decomposition scheme ruled by a mixed-

[1]Contained within the ECSS-E-70-1 Space Segment Operability Standard document (available at www.ecss.nl)

Level	Description	Functions
E1	Mission execution under ground control; limited on-board capability for safety issues	Real-time control for nominal operations. Execution of time-tagged commands for safety issues
E2	Execution of pre-planned, ground-defined, mission operations on-board	Capability to store time-based commands in an on-board scheduler
E3	Execution of adaptive mission operations on-board	Event-based autonomous operations. Execution of on-board operations control procedures
E4	Execution of goal-oriented mission operations on-board	Goal-oriented mission re-planning

Figure 4.1: ECSS Mission Execution Autonomy Levels.

initiative principle and adjustable autonomy-based methods, that enables different degrees of cooperation between the human operators and the autonomous system (typically on-board). Since automation refers to the fully or partial replacement of a function previously carried by the human operators, it can be seen (the automation itself) as a continuum spectrum of which levels vary according to different criteria [Inagaki & Itoh, 1996] such as human-intervention or shared responsibility degrees.

For instance, a close-to-manual operational mode (E1 level in the ECSS table) entrusts very few decision-making responsibilities on the autonomous system (typically restricted to detect critical system failures and set up system in standby), meanwhile in a highly-automated operational mode, the human is relegated as a mere supervisor within the mission execution loop (see figure 4.2).

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].

Today's technology is mature enough to effectively pursue many of the previous objectives, and many examples of applying automated reasoning to planetary explorations actually exist.

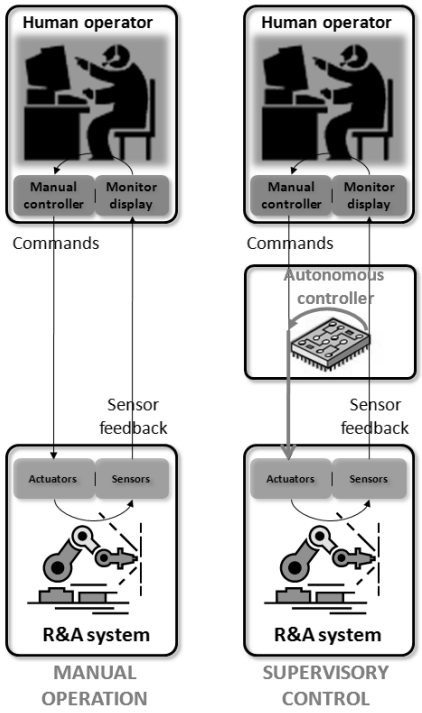

Figure 4.2: Adjustable autonomy-based execution control scheme: (left) manual operation; (right) mixed-initiative control.

In this work, the Mars Science 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, in the next section we introduce a MSR-inspired scheduling problem called *Power Aware Resource Constrained Mars Rover Scheduling* (PARC-MRS).

4.2 The Power-Aware Resource Constrained Mars Rover Scheduling (PARC-MRS) Problem

In this section we provide a definition of the PARC-MRS problem, that is grounded on the commitment to the 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 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 relevant features (see also Section 5) of the locomotion system are: (i) a nominal consumption rate σ_{loc} (in Watts); (ii) a maximum translational speed ν (in m/sec.); (iii) a maximum steering speed ω (in degrees per second); and (iv) a maximum traversable slope or tilt angle ϕ (in degrees).

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

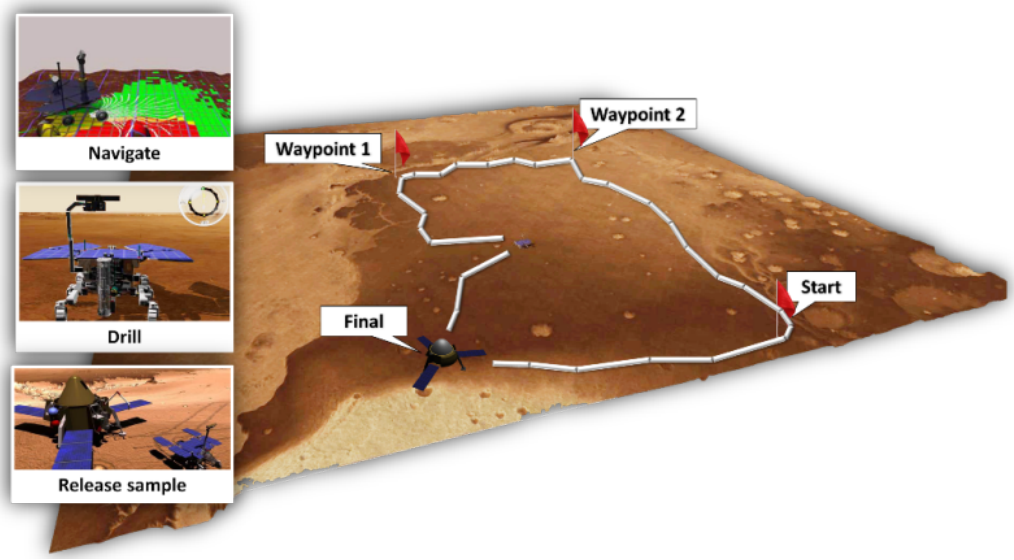

Figure 4.3: Mission scenario overview: (1) navigation, (2) acquiring science (drill) and (3) sample release activities

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:

$$Exp_i = \langle Nav_{S,i}, Drill_i, Nav_{i,F}, Rel_i \rangle \tag{4.1}$$

Figure 4.3 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 way-point 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 $i = 1, 2, \ldots, n$. Let $B_0 = B(0)$ and $B_i = B(st_i)$, the computation of the B_i values is performed according to the formula:

$$B_i = Min\{B_{i-1} + SP_{i-1,i}, B_{max}\} - SC_i \tag{4.2}$$

  where B_i and B_{i-1} are the battery charge levels at two consecutive consumption instants st_i and st_{i-1} respectively, $SP_{i-1,i} = \sigma_{charge}(st_i - st_{i-1})$ is the amount of energy generated by the solar arrays in the interval $[st_{i-1}, st_i)$, finally SC_i is the amount of energy consumed at the instant st_i. The inequality $B_i \geq B_{min}$ must be satisfied for each $i = 1, 2, \ldots, n$.

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.

4.3 The Constraint-based PARC-MRS Problem Representation

We formulate the PARC-MSR scheduling problem in terms of CSP [Montanari, 1974]. In our CSP-based formulation of the PARC-MSR a set of decision variables called *Minimal Critical Set (MCS)* 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, & 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 4.4 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. Note that according to our model, all the activities belonging to the same job are linked with tight $[0, 0]$ separation constraints, i.e., constraining the end time of the previous activity to coincide with the start time of the next. Note also that since the first navigation activities involved in the execution of each sequence (i.e., $Nav_{S,i}$) do not use any resource (i.e., the sample cache resource is always empty when moving from the rover's start location S), we decided to simply model them through separation constraints weighted with the traversal times required to travel the distances ($[tt_{S \to w_i}, H]$) between S and the soil extraction locations.

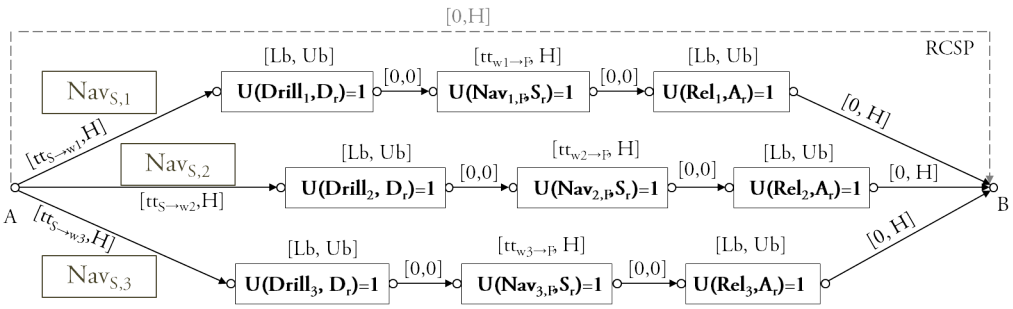

Figure 4.4: Activity-on-the-node graph representation of the problem model: the edges represent the precedence constraints, while the nodes (boxes) represent the activities; the resource usage information is shown within each box

4.4 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 PCP approach ([Smith & Cheng, 1993] and Che1994), 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, & 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 section 4.3 and iteratively performs a solving sequence composed of three steps: (i) checking the temporal consistency of the current partial solution; (ii) 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.

4.4.1 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.

4.4.2 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[2] (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 behaviour of the battery to the

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

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
```

cumulative scheme, which ESTA uses, we exploit a modified version of the model introduced by Simonis. 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 [Simonis & Cornelissens, 1995]. It is worth mentioning that, although our abstraction is simpler than other existing models such as the *Multi Mission Power Analysis Tool (MMPAT)* [Wood, 2002], it encapsulates the most significant elements of the power subsystem considered within our MSR domain of interest and reveals to be suitable for the test-cases here addressed. 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 4.5 (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 4.5 (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).

The computation of the global production/consumption battery usage profile is performed according to the formula:

$$B_i = Min\{B_{i-1} + (SP_i - \alpha), B_{max}\} - SC_i \tag{4.3}$$

where B_i and B_{i-1} are the battery charge levels at two consecutive instants i and $i-1$ respectively (which can be consumption or/and production instants), SP_i is the amount of energy collected at the instant i, α is the amount of energy possibly lost because of the saturation in the last production step, and SC_i is the amount of energy

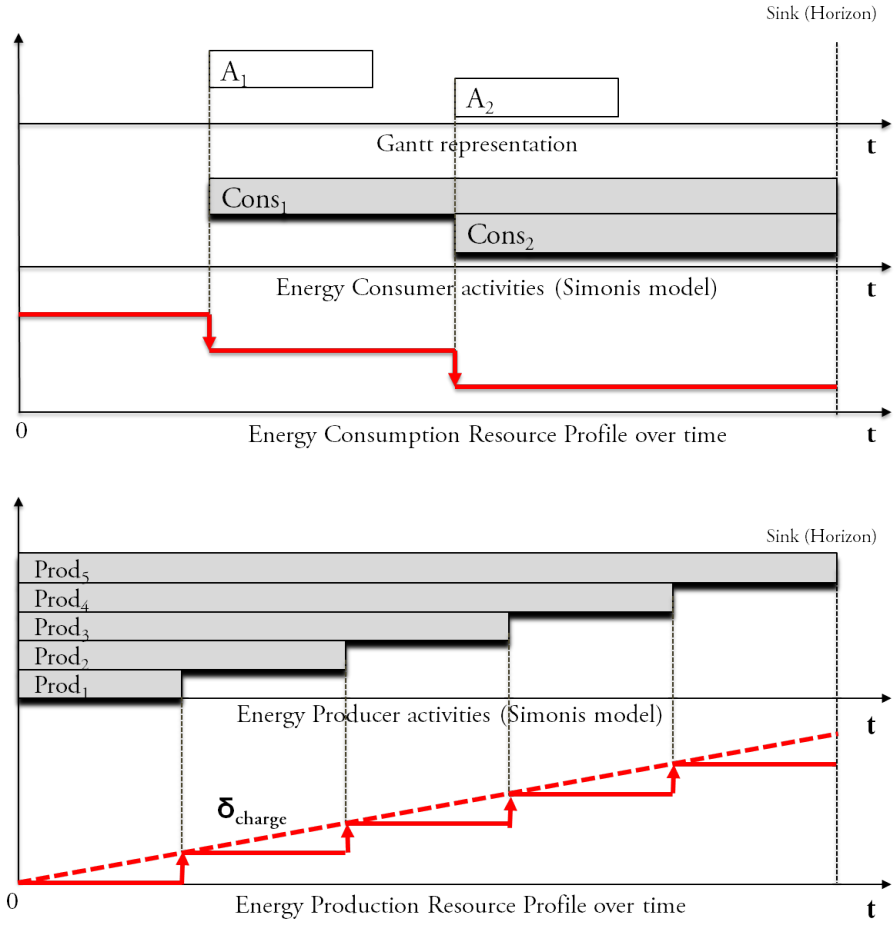

Figure 4.5: Energy consumption (top) and production (bottom) constraints representation

consumed at the instant i. Note that whenever the battery charging level B_i falls below the minimum usage threshold B_{min}, an *energy conflict* situation occurs which will have to be solved by the scheduling process. In all our test-cases, it is assumed that the battery is always fully charged at $t = 0$, i.e., $B_0 = B_{max}$.

Figure 4.6 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 en-

ergy 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 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.

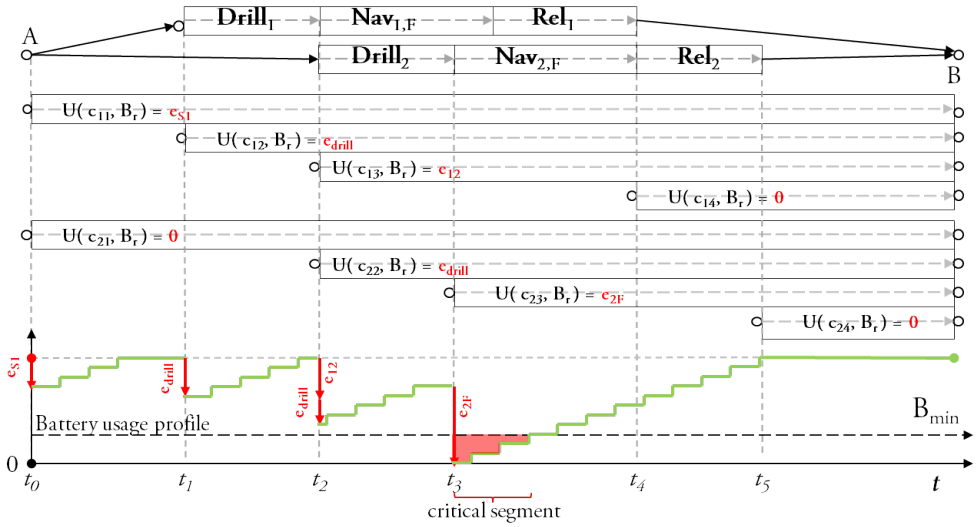

Figure 4.6: Example of energy profile computation: the consumption components of the profile (e_{ij}) are depicted in red, while the production components are depicted in green

4.4.3 Step 3: Resource Contention Peaks Levelling

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 4.6 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 careful reader should note that due to the stepwise segmentation of the power production profile, possible overloads on the imposition of precedence constraints during the battery contention peaks resolution might occur. In order to overcome with such limitations, conflicts are solved in only one step by analysing the (continuous) realistic power production profile such that the separation constraints are weighted with a time value equal to the time required to recover all the (over)demanded energy by the subscribing activities.

Finally, it is worth pointing out that due to the specific characteristics of our MSR scheduling model, the impositions of the temporal constraints during the solving process might lead to particular deadlock combinations (or *crossed constraint situations*, initially described in [Díaz *et al.*, 2011a]), and hence originate unsolvable MCSs. Figure 4.7 exemplifies the problem by providing three deadlock instances originating from different solving constraint combinations in a simple problem instance composed of two jobs. As the figure shows, in all the three cases (a), (b) and (c), the imposed constraints (thick arrows) define such an ordering between the $Drill_i$ and the Rel_j activities that the two navigation activities $Nav_{i,F}$ and $Nav_{j,F}$ can no longer be separated.

Obviously, this circumstance represents a deadlock if the maximum capacity C of the sample cache is equal to 1, as this condition forbids any two navigation activities to be scheduled concurrently. By visually inspecting Figure 4.7, it can be noted that a possible occurrence of a deadlock situation is related to the existence of a constraint *cycle* among the problem sequences (e.g., in all cases, the first constraint links sequence i with sequence j and the second constraint links sequence j with sequence i, thus forming a cycle of length 2). Among all combinations, the type of ordering relations that can possibly originate cycles (i.e., the *necessary* condition for a deadlock), are the $Drill_i \prec Drill_j$, the $Drill_i \prec Rel_j$, and the $Rel_i \prec Rel_j$ constraints.

In order to narrow down the negative effects of this phenomenon, we have extended the original heuristic with a specific look-ahead analysis to enhance *crossed-constraint awareness*. The look-ahead analysis basically introduces a modification of

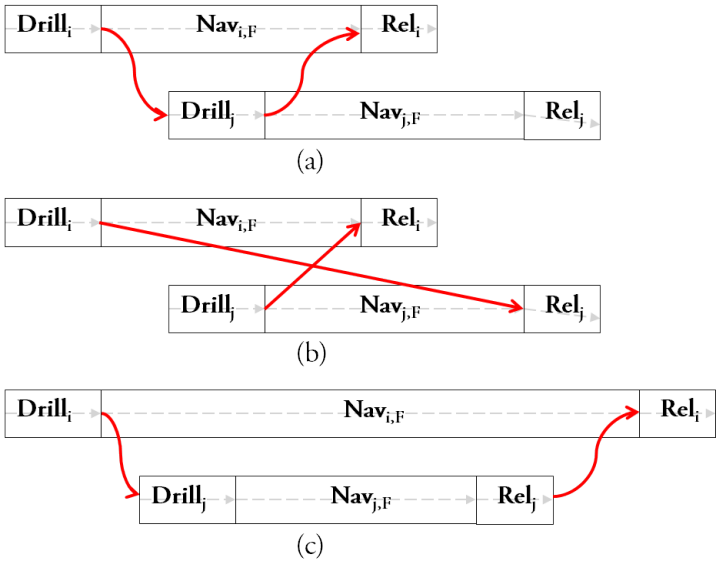

Figure 4.7: Three different examples of deadlock situations with two rover experiments, and maximum battery capacity $C = 1$

the MCS selection procedure executed at line 17 of Algorithm 1, by forward-checking if the resolution of the selected MCS (through the imposition of the related solving constraint) is going to form a cycle of length n greater than the maximum battery capacity C. In a positive case, the selected MCS is discarded and a new candidate MCS is chosen and analysed; this process is repeated until an MCS is selected whose resolution does not cause the onset of a "hazardous" cycle.

The steps discussed in Sections 4.4.1, 4.4.2, and 4.4.3 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.

4.5 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 pro-

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

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

```
1 S_best ← ESTA^p (MaxH)
2 while (¬ StopCondition (MaxTime, MaxAttempts)) do
3     Sol ← ESTA^p_rand (Mk (S_best))
4     if (Mk (Sol)<Mk (S_best)) then
5         S_best ← Sol
```

vided 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, & 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 execution 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 in-

creasingly smaller makespans. The algorithm returns the best solution encountered when either of the two stop conditions previously described is met.

4.6 Summary

In this chapter of the dissertation we presented the problem domain context as a particular rover-based Mars exploration mission, namely, the MSR mission concept. 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.

As a first important contribution, we integrated the most significant MSR mission requirements into a scheduling problem model, the so-called PARC-MRS problem. 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.

As a second main contribution, 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, it is worth to mention the successful exploitation of a well known methodology to represent re-

newable resources by means of a classical cumulative scheme, as a core concept to model and solve the PARC-MRS problem.

Chapter 5

Experimental Analysis

In this chapter 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. We start by introducing the specific benchmark problem generator used to create a set of meaningful and realistic problem instances; thereafter, we present the empirical analysis and results.

5.1 The Benchmark Problem Generator MSR/Gen

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[1] for the class of problems here referred to as PARC-MRS. The MSR problem generator was developed with the twofold aim of (i) creating well-founded benchmark problem instances with particular reference to the MSR domain addressed in this work, (ii) and establishing an initial stable reference to be exploited by other authors aimed at addressing similar problems.

The MSR/Gen takes a set of seed parameters as input provided by the user, and returns a set of benchmark problems. The input seed parameters contains the following information: the number of problem instances; the number of experiments for

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

each problem instance; the capacity of the sample cache resource; the dimensions of the terrain area; the temporal constraints (i.e., the nominal values of the durations of the soil extraction and release sample activities respectively, the number of experiments constrained by maximum time windows or *deadlines*); the energy constraints (i.e., power demand values both for the soil extraction and navigation activities — the latter expressed per distance unit— under nominal circumstances, the maximum battery capacity B_{max} and its minimum usage threshold B_{min}, the battery charging rate σ_{charge}, and the amount of power consumed by the thermal s/s to heat the rover's instruments during the night standby periods); and the rover locomotion capabilities (i.e., the angular ω and linear ν speed, as well as the maximum traversable slope angle ϕ, which are considered for both (i) the generation of the experiment waypoint locations as a set of 3D coordinates $\langle x, y, z \rangle$, and (ii) during the estimation of the energy e_{ij} required by the rover to navigate between each pair of the previous waypoint locations).

Additionally, a list of deviation factors $\langle d_1, d_2, \ldots, d_n \rangle$ (%) are given in order to allow the generation of different problem instances. More concretely, these factors allow us to controllably introduce some degree of randomness in the parameter value generation; more specifically, given a seed parameter value s and the related deviation factor d, the final parameter value will be randomly selected in the interval $[s * (1 - d), s * (1 + d)]$.

5.2 The MSR Benchmark Problem Sets

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, & 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)[2].

It should be noted that these problems represent a rather challenging testbed because of the size of the problem instance themselves and the complexity of the temporal, multi-capacity resource and energy constraints involved.

Table 5.1 contains the specifications about all the seed parameters that have been used to generate each benchmark instance set. The seed values with a deviation factor equal to 0 (last column in the table) will not undergo random modifications (i.e., they remain constant in all the generated instances). It is worth highlighting again that the energy required to both performing the experiments and heating the rover s/s during the night is entirely provided by the battery, while the energy required by the locomotion s/s is almost entirely provided by the solar arrays (the battery contributes to the navigation activities by a mere 17%).

5.3 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 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

[2]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*

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.

Table 5.1: Seed parameter values for the MSR_{40}–$20/25/30$ benchmark sets

Attribute	**Section**	**Seed value (s)**			**Dev. (d)**
		MSR_{30}–20	MSR_{30}–25	MSR_{30}–30	
Terrain dimensions	Header	$5.0\times5.0km^2$	$6.5\times6.5km^2$	$6.5\times6.5km^2$	0
Maximum sample cache capacity	Header	6	6	6	0
Soil extraction operation duration (lb)	Temp. cons.	133 mins.	133 mins.	133 mins.	0.1
Unload sample operation duration (lb)	Temp. cons.	10 mins.	10 mins.	10 mins.	0
No. experiments with deadlines	Temp. cons.	4	4	4	0
Maximum battery capacity (B_{max})	Energy cons.	500 Wh	500 Wh	500 Wh	0
Minimum usage threshold (B_{min})	Energy cons.	5%	5%	5%	0.1
Power consumption (Navigation) (σ_{loc})	Energy cons.	30 W	30 W	30 W	0.1
Battery contribution to propel the rover	Energy cons.	5 W	5 W	5 W	0
Power consumption (Sample collection)	Energy cons.	40 W	40 W	40 W	0.1
Power consumption (Overnight)	Energy cons.	20 W	20 W	20 W	0
Nominal power production (σ_{charge})	Energy cons.	600 W	600 W	600 W	0
Rover linear speed (ν)	Locomotion	0.066 m/sec.	0.066 m/sec.	0.066 m/sec.	0.4
Rover angular speed (ω)	Locomotion	14.9 deg/sec.	14.9 deg/sec.	14.9 deg/sec.	0.1
Maximum traversable slope (ϕ)	Locomotion	25 deg.	25 deg.	25 deg.	0.5

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). As reported in the 5th row of Table 5.1, 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 7.1 and 7.2 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[3]) along the whole plan's horizon.
- $Bat^{avg}(\%)$ is the battery resource usage (expressed in percentage[4]refnote) along the whole plan's horizon.
- $\Delta^{avg}_{LWU}(\%)$ conveys the average improvement ratio (expressed in percentage[5]) 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).

[3] $Res^{avg} = \frac{1}{n*maxCap}\sum_{i=1}^{n}\frac{\int_0^{mk_i} f_i(t)\mathrm{d}t}{mk_i} \times 100$, 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.

[5] $\Delta^{avg}_{LWU} = \frac{1}{n}\sum_{i=1}^{n}\frac{mk_i - mk_i^0}{mk_i^0} \times 100$, 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 $ESTA^p$

Table 5.2: Experimental results corresponding to the feasibility assessments

	ESTA^p (feasibility assessment)			
Benchmark	$Mksp^{avg}$	CPU^{avg}	$Cache^{avg}$	Bat^{avg}
MSR_{40}–20–8h	17027.2	50.367	28.575	8.864
MSR_{40}–20–12h	19336.232	55.609	33.062	21.511
MSR_{40}–20–16h	30144.4	59.440	23.535	39.771
MSR_{40}–25–8h	23826.228	108.793	30.458	8.91
MSR_{40}–25–12h	29249.686	121.651	28.153	21.598
MSR_{40}–25–16h	41558.232	163.706	22.062	39.825
MSR_{40}–30–8h	30605.825	181.842	32.674	8.889
MSR_{40}–30–12h	36516.125	207.214	30.166	21.639
MSR_{40}–30–16h	52707.65	272.359	22.435	37.185

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.

Table 5.3: Experimental results corresponding to the optimization assessments

	ISES (optimization assessment)				
Benchmark	$Mksp^{avg}$	Δ^{avg}_{LWU}	$Cache^{avg}$	Bat^{avg}	$\#Iter(avg)$
MSR_{40}–20–$8h$	12620.325	35.628	30.473	8.679	14.575
MSR_{40}–20–$12h$	15990.45	26.399	33.866	21.395	12.375
MSR_{40}–20–$16h$	25729.3	17.239	25.795	39.672	11.3
MSR_{40}–25–$8h$	20870.925	21.805	33.954	8.861	7.85
MSR_{40}–25–$12h$	26293.85	17.907	28.481	21.523	6.05
MSR_{40}–25–$16h$	37720.55	11.785	24.801	39.783	4.725
MSR_{40}–30–$8h$	26643.7	15.588	35.213	08.857	5.775
MSR_{40}–30–$12h$	33517.45	9.248	30.995	21.608	3.975
MSR_{40}–30–$16h$	48680.2	8.569	22.819	38.034	2.925

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

bility 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.

5.4 Summary

In this part of the dissertation we 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.

We proposed a study of the benchmarking problem tailored to the MSR domain, and produced a methodology to generate meaningful PARC-MRS problem instances.

As a result of this study, we have created a benchmark library using baseline parameter values which were carefully selected from recent real-world mission specifications.

More concretely, we have based our benchmark production on one of the rover models offered by the ESA's 3DROV simulator, an advanced and realistic planetary

robot design, visualization, and validation tool.

Subsequently, we carried out an empirical evaluation, and demonstrated our solving algorithm's performances in providing good quality solutions, by efficiently taking into account all the most significant problem constraints, in particular those related to energy management.

We also demonstrated that significant solution quality improvement can be provided by our makespan-optimization framework in a relatively small amount of time.

Part III

Flexible Reactive Schedule Execution Management under Uncertainty

Chapter 6

Power-aware, Continuous Mission Scheduling and Execution

In this part of the thesis, we start introducing the current scenario in space exploration on the application of automation and robotics in space exploration. Next, we present a model-based control architecture targeted at generating and executing planetary rover mission plans inspired on the MSR mission concept.

6.1 Introduction

The current scenario in space exploration is characterized by the use of automation and robotics. The latest and most outstanding example is the NASA's Mars Science Laboratory (MSL) mission and its rover Curiosity[1], aimed at assessing the planet habitability, i.e., discovering hints about whether Mars ever housed small life forms. Curiosity represents the synthesis of the NASA's vast experience on the deployment of mobile robots on the Mars' surface, gained along many successful missions such as the Mars Pathfinder probe in 1997 [Mishkin *et al.*, 1998b], basically in charge of demonstrating the technology necessary to deliver a lander and a free-ranging robotic rover (Sojourner) to the surface of the Red planet; or the twin Mars Exploration Rovers (MER) vehicles [Maimone, Leger, & Biesiadecki, 2007] –Spirit

[1]Mars Science Laboratory mission website, http://marsprogram.jpl.nasa.gov/msl/, Jet Propulsion Laboratory (JPL), visited September 2014.

and Opportunity– launched at 2003, involved with searching for and characterizing a wide range of rocks and soil samples that might hold clues to past water activity.

In the near future, top space agencies' roadmap actually passes through continuing fostering robotics able of maximizing science data return while keeping pace with unexpected events arising during mission execution. For instance, ESA in collaboration with other partners, is envisioning rover-based missions such as the MSR mission concept [Baglioni *et al.*, 2006]: a lightweight rover-based mission aimed at acquiring Martian materials from known locations on Mars to be later delivered back to Earth for a further analysis. Future mission baseline requirements, like in MSR, will be surely grounded on the investigation of multiple widely-distributed science targets in a single command or working cycle (e.g. one Martian Sol).

To successfully attain this end, the solution relies on continuing promoting autonomy *on-board*, since purely manual control becomes an excessively inefficient option [Muscettola *et al.*, 1998], if not infeasible, if considered the major constraints in remote operation: the astronomic distances involved between Earth and celestial bodies, the limited existing communication opportunities and the extremely harsh environmental conditions. Next generation of autonomous robotic systems developed for accomplishing future challenging missions to the Moon, Mars and beyond, will be enabled to make and run critical decisions with a little or no human intervention.

According to the European Cooperation for Space Standardization[2] (ECSS), autonomy in space robotics is defined in terms of four autonomy levels (ranging from E1 to E4), which basically ranges from manual *tele-operation* (E1 Autonomy Level) –where a real time control from ground is implemented for nominal operations, and just a limited critical and system-level tasks involving fault detection, diagnosis and eventual recovery actions (typically known as FDIR [Wander & Frstner, 2013]) are performed on-board–, to a fully *autonomous* operation (E4 Autonomy Level), where the robot is equipped with high-level reasoning capabilities that guarantee the execution of goal-oriented mission operations on-board.

Figure 6.1 depicts the well-known multi-layered architecture used to describe the robot autonomous reasoning capabilities.

Generally, autonomous decision-making features are realized by persistently ex-

[2]ECSS-E-70-1 Space Segment Operability Standard document (available at www.ecss.nl)

ecuting a SPA loop [Murphy, 2000] where the task dispatching (Act phase) is a result of a previous reasoning process (Plan phase) based on a continuous perceptions acquired from the environment (Sense phase). The architecture in Figure 6.1 is organized as a stack where every level, each retaining different responsibilities, implements a SPA loop dispatching commands to (and receiving feedback information from) the underlying layer. Starting from the bottom (i.e., pure *reactivity*) and moving towards the top layers (i.e., model-based goal-oriented planning) requires the exploitation more and more complex reasoning capabilities.

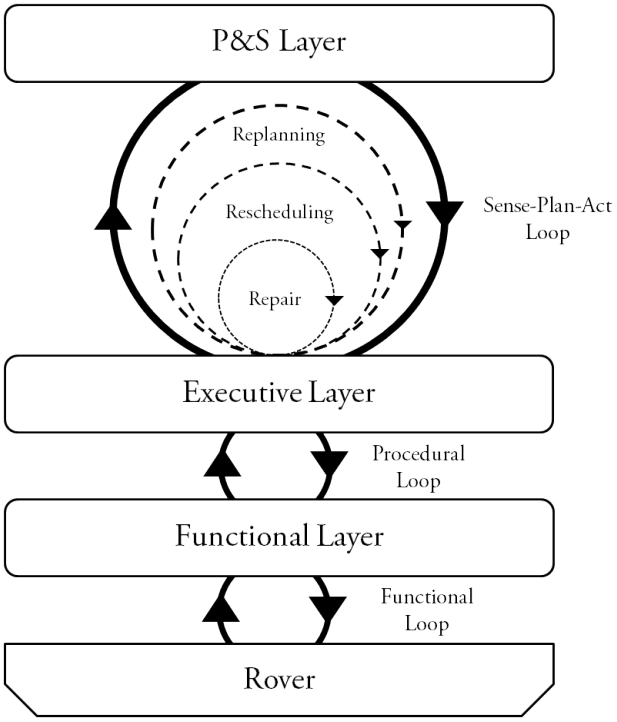

Figure 6.1: The typical multi-layered SPA-based architecture.

In this part of the thesis, we present a model-based control architecture targeted at generating and executing planetary rover mission plans inspired on the MSR mission concept.

In our architecture, we have placed ourselves at the top SPA-based P&S layer of the organization depicted in Figure 6.1, proposing a constrained-based flexible model that:

– allows to represent both time and resources, with special attention to the rover on-board battery power;

– integrates plan *re-scheduling* and plan *execution* on top of the same representation, allowing to hedge against domain uncertainty: (i) reacting against possibly disruptive events, and (ii) dynamically modifying the plan (e.g., to respond to unexpected science opportunities);

– allows the selection of the "re-planning depth" (repair Vs. re-scheduling), depending on the current execution conditions;

– allows to perform plan optimization and execution concurrently.

One of the most significant contributions of this work is the introduction of an energy management model explicitly represented and integrated within the scheduling process [Díaz *et al.*, 2013; 2011b], to the aim of maximizing rover's performances (i.e., in terms of routing optimization to maximize science return) in the context of long-term rover autonomy, where reliable projections of energy consumption are of great importance.

The rescheduling capabilities of our architecture have been tested by closing the execution loop with the 3DROV planetary rover system simulator [Poulakis *et al.*, 2008], an ESA asset providing a realistic behavioral reproduction (i.e., from the temporal, dynamic and energy-related standpoint) of all the rover subsystems (e.g., locomotion, drilling, etc,) necessary to perform the tasks covered in our experimental model.

The remainder of the chapter is organized as follows: we start providing a detailed description of the power-aware autonomous control architecture and its basic building blocks. Next, we present the integrated testbed platform built on the top of the ESA's 3DROV planetary rover system simulator [Poulakis *et al.*, 2008]. Following that, we conduct a formal analysis on the performance of the basic target capabilities of the controller through the dynamic simulation of two representative and comprehensive cases of study with 3DROV. Finally, a conclusions and future work section closes the chapter.

In this section we propose an integrated model-based execution control architecture $CoRe^p$ which enables the rover to: (a) deploy advanced decision-making

capabilities involving constraints concerned to both *global path-planning* and pure scheduling like *sequence-dependent setup-times, power production/consumption or multi-capacity resource allocation*; and (b) keep pace with the environmental uncertainty so that possible incoming disturbances threatening the plan under execution can be suitably managed.

6.2 Revisiting the Constraint-based PARC-MRS Problem from a Dynamic Perspective

Previously we provided a solution to the static dimension of the problem referred as PARC-MSR, where an integrated power-aware decision-making strategy was introduced to synthesize robust plans from a set of of high level mission requirements with special attention to the energy constraints. In this direction, current section aims at broadening previous problem scope by presenting an enhanced mission control architecture targeted at providing a reliable and efficient mission execution management based on the advanced decision-making capabilities previously mentioned, while supporting on-line plan optimization and dynamic management of new incoming activities.

Mission execution process starts with the synthesis of a feasible initial solution schedule. Next, the rover proceeds to its execution by timely dispatching the set of commands related to each rover activity[3]. The whole schedule execution relies on a reliable SPA closed-loop control model (see Figure 6.2), which continuously supervises/monitors the execution process so that as soon as a misalignment between the expected and the rover behaviour is detected, a contingency solving strategy is deployed with the aim of recovering the execution to a consistent state. For instance, suppose that the rover is navigating between two different waypoints i, k, and suffers from an unexpected delay as a consequence of a slippage while crossing a soft terrain segment on the path. The rover will detect (through its sensing and monitoring capabilities) that both the amount of time and energy[4] required to cover such a path

[3]The execution of each rover activity entails a downwards translation to predefined sequences of low level commands which directly operates the rover actuators, e.g., the navigation activity entails the operation of the locomotion s/s under a specific setting

[4]We assume that the rover is equipped with advanced motion control with modification of accelera-

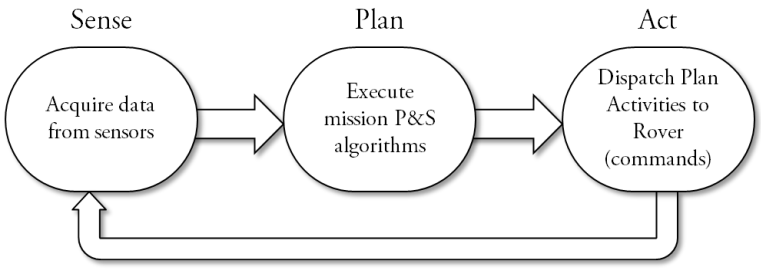

Figure 6.2: Sense-Plan-Act (SPA) closed-loop control model

segment is being higher than predicted, since the conditions of the terrain changed with respect to the expectations. Then, the rover will acknowledge the real situation by updating its internal model and triggering reactive mechanisms, if needed, to readjust the running schedule so that the contingency effects (i.e., the temporal delay and power overconsumption) are overcome.

Moreover, the plan execution can be disrupted by the arrival of new experiments (i.e., new locations to visit) that must be integrated "on-the-fly" in the currently executing schedule and that must be planned for, in order to guarantee both the feasibility and the quality of the rover's activities.

The rover will be also in charge of deploying a continuous plan optimization process running in the background in order to search better quality solution plans (in terms of makespan length), as the current plan (i.e., the most recent feasible solution found) is being executed.

6.3 The Power-aware Autonomous Control Architecture $CoRe^p$

In this section we propose an integrated model-based execution control architecture $CoRe^p$. It enables the rover to: (a) deploy advanced decision-making capabilities involving constraints concerned to both *global path-planning* and pure scheduling like *sequence-dependent setup-times, power production/consumption or multi-capacity resource allocation*; and (b) keep pace with the environmental uncertainty so that possible incoming disturbances threatening the plan under execution can be suitably

tion

managed. Figure 6.3 illustrates the main functional building blocks of the $CoRe^p$ autonomous controller. $CoRe^p$ implements a SPA closed-loop execution scheme which continuously guarantees the consistency of the entire plan execution process by exploiting advanced reasoning capabilities at its core, monitoring and analysing the execution status and deploying reactive strategies in case of misalignments on the internal model updating, i.e., on the face of possible deviations between the expected and the real execution process evolution. In addition, $CoRe^p$ allows deploying both continuous (on-line) plan optimization and dynamic management of new incoming mission activities.

More concretely, $CoRe^p$ is responsible for generating complete and consistent long-term mission plans covering kilometre-scale distances, and guaranteeing a safe and efficient plan execution through a flexible mission execution management process consisting on the following day-to-day readiness operations:

1. Synthesize feasible and robust baseline schedule solutions, exploiting a constraint-based solver called $ESTA^p$ (see [Díaz *et al.*, 2013] for more details).

2. Timely dispatch the nominal schedule and monitoring the execution evolution (Scheduler Dispatcher & Execution Monitoring module) on the basis of the rover telemetry such as position, orientation, command execution status and battery state of charge (SoC). The execution monitor is also aware of the arrival of new experiments (i.e., strings or sequences of rover activities defining a complete scientific experiment execution) externally submitted by the scientific team, or maybe as a result of an unexpectedly interesting science observation (*opportunistic science* support).

3. Update the internal mission execution model with the new information provided by the monitor, and detect possible misalignments between the planned and the real rover behaviour, in terms of timing and resource usage. This process is known as "execution monitoring & consistency checking".

4. Provide (on-line) alternative schedule solutions to face with the possible contingent situations (like command execution delays or battery overconsump-

tion), through the execution of reactive strategies to correct possible constraint violations after they have arisen as a result of the internal model updating.

5. Continuously search for better plans (in terms of makespan length) through the execution of an iterative sampling optimization loop which uses $ESTA^p$ in a similar fashion to the Iterative Sampling Earliest Solutions (ISES) [Cesta, Oddi, & Smith, 2002] strategy for makespan minimization: an efficient multi-pass approach which performs quite well in the face of scheduling problems that involve very large search spaces.

It is worth to mention that the controller reasons at a high abstract level in regards to path planning decisions, and therefore the kind of uncertainties considered must also defined at the same level. The "execution monitoring & consistency checking process" is involved with detecting and assessing temporal, resource usage and energy misalignments, but does not address local navigation issues like route deviations, as it is designed to work on top of a robotic system endowed with local navigation capabilities providing obstacle detection & avoidance, as well as smart, terrain-aware mobility features that mitigate the typical wheel slippage/sinking effects.

In the remaining of the section, we provide a detailed description of the all previous activities which compose the backbone of the plan execution management process: *constraint-based reasoning, execution monitoring & consistency checking* and *contingency solving.*

6.3.1 Constraint-based Reasoning

The attainment of the mission goals requires computing and executing a feasible schedule as a temporal arrangement of the experiment activities, while synchronizing the use of a set of different mission assets or resources. According to this problem formulation, 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 $a_k \in \{Drill_i, Rel_i : i = 1, 2, \ldots, n\} \subset A$, while the whole set of the following temporal, resource and energy constraints are satisfied.

– *Temporal Constraints* - Figure 6.4 shows in detail the temporal constraints involved within the execution of a single rover experiment: as mentioned before,

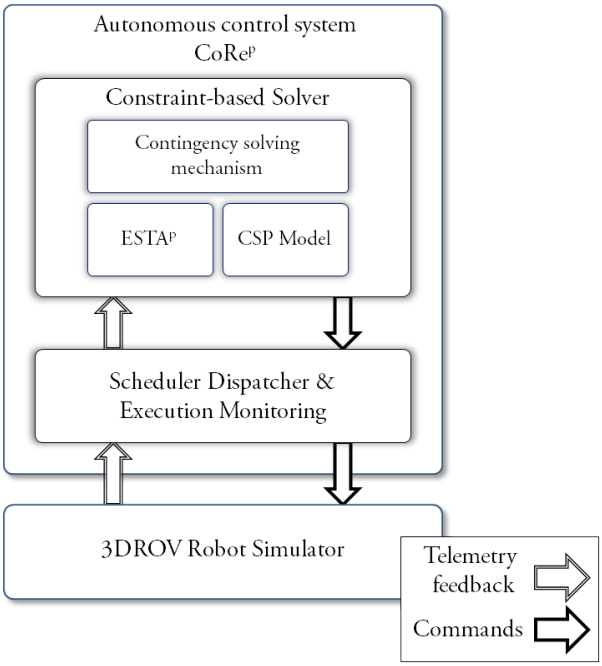

Figure 6.3: Conceptual schema of the autonomous control system $CoRe^p$

durations of the activities are flexible *but limited with maximum time windows in the case of the sample collection $Drill_i$ and sample release Rel_i activities*, so as to accommodate for some temporal flexibility during their execution.

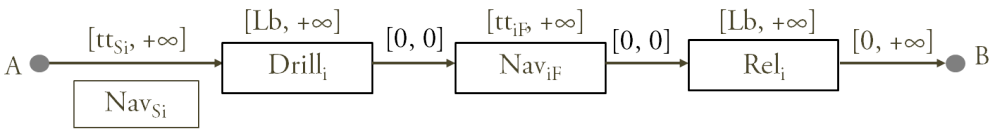

Figure 6.4: Temporal constraints involved within the execution of a single rover experiment

Pairs of consecutive activities in each experiment sequence Exp_i are stick to each other so that the end time of the preceding activity coincides with the start time of the succeeding activity (see the imposed interval constraints $[0, 0]$ in Figure 6.4). 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 oper-

ations, 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. The completion time of some of the Exp_i sequences might be constrained by an absolute deadline d_i. Finally, the whole plan is constrained by a global, maximum temporal horizon H, which might span a long term mission operation (up to several days).

– *Renewable resource constraints*. 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 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 ($Drill_i$) require a *science acquisition asset* and a *sample cache (SC_r)*, consisting of a drilling subsystem and a container with a finite capacity to store and transport up to *C* standard-sized samples, respectively. Navigation tasks demand the *Locomotion, Guidance, Navigation and Control (Loco & GNC) subsystem*, which provides all the functionalities that allow the rover to reliably reach a desired target. Figure 6.5 presents a Gantt representation diagram where three experiments and the resource profiles (i.e., resource usage over time) are represented: the drilling subsystem and the robotic arm are characterized as *binary resources*, since they can only be demanded by one activity at a time; while the sample cache is represented as a *cumulative (multi-capacity resource)* with a maximum capacity $C = 2$, so as a maximum of two different soil samples might be simultaneously transported. In general, the concurrent execution of activities using the same resource might cause a resource conflict if the total demand overpass the maximum capacity of the resource. For instance, we can spot in the figure a sample cache overconsumption during the time period on which the execution of three different navigation activities coincide.

– *Energy constraints*. The rover energy supply is provided by a *powering sub-*

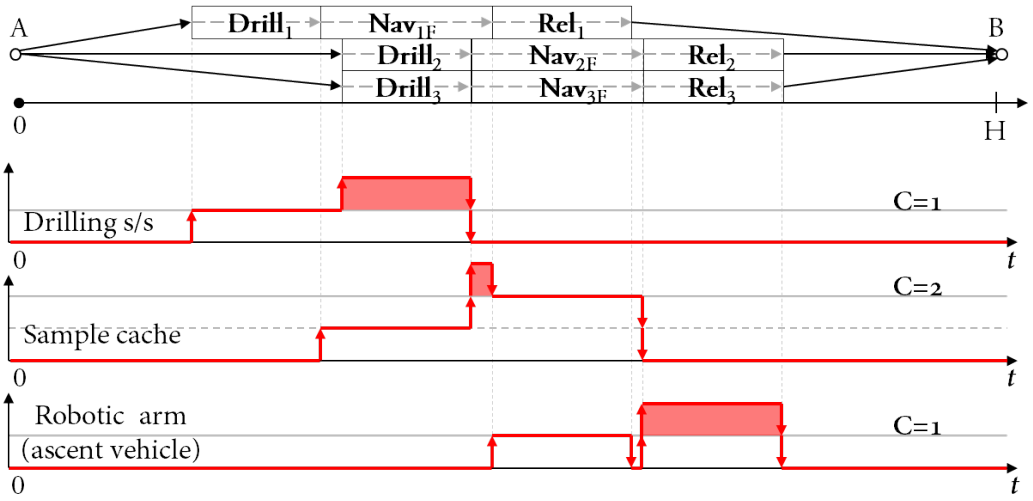

Figure 6.5: MSR problem example with three rover experiments and their corresponding resource profiles

system consisting of a combination of Solar Array (SA_r) as a primary power source, and a *Battery* (B_r). The battery is characterized by its maximum capacity or *saturation level* B_{max} and by a *minimum usage threshold* B_{thres}, representing the minimum battery energy level that can be reached, for safety reasons. Our hypothesis is 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. 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. The execution of the power *consuming activities* (i.e., soil extraction and navigation activities) require a certain amount of energy that *has to be completely available at the starting time of the activity*. Figure 6.6 depicts the battery state of charge profile relatively to the same problem instance presented earlier in Figure 6.5. As the figure shows, the $Nav_{i,j}$ activities demand a variable amount of energy $c(Nav_{ij})$, which depends on the traveling distance between the two different locations i and j, while the $Drill_i$ activities demand a constant amount of

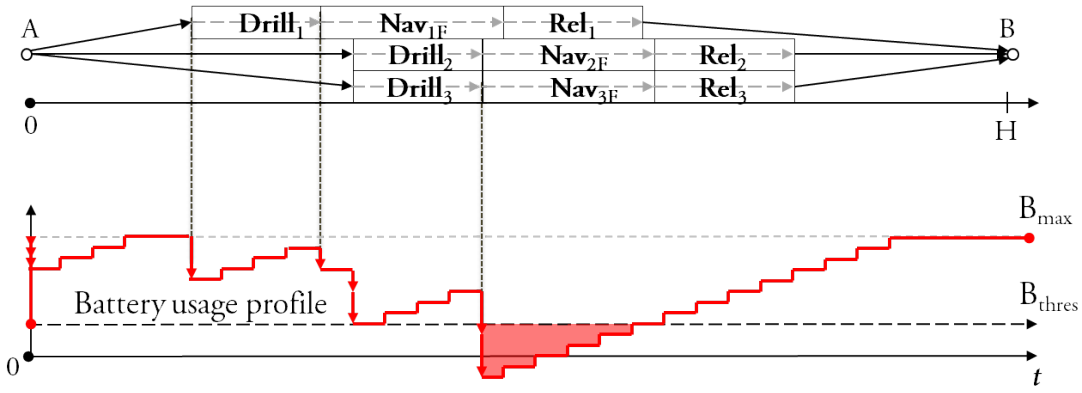

Figure 6.6: Energy production/consumption profile representation

power $c(Drill_i)$ necessary to operate the drill subsystem. In general, given a rover activity a, the energy consumptions $c(a)$ are modeled as *instantaneous activities*, whose start time is constrained to coincide with the start time of a (see Figure 6.6), whereas the continuous charging process during the daylight period is modeled as a sequence of small, discrete chunks of energy *productions* (see Figure 6.6) distributed along the complete horizon at a specific temporal rate. Each chunk of energy generates an amount of energy equal to the quantity of power collected at each piecewise segment; the result is a piecewise constant representation of the energy production profile (the interested reader may refer to [Díaz *et al.*, 2013] for further details about the energy model). More concretely, Figure 6.6 illustrates how the power demands of the energy consuming rover activities (i.e., $Drill_i$ and Nav_{ij} activities) combine with the energy production profile. It is worth noting if the battery is fully charged (i.e., saturated at its maximum capacity B_{max}), all the consecutive energy gains chunks are discarded until a new demand arrives.

A battery overconsumption occurs when the cumulative demand of the battery, pulls the curve representing the energy profile $B(t)$ below the minimum usage threshold B_{thres} (energy conflict). An energy feasible solution must satisfy the following constraint $B(t) \geq B_{thres}$ for each $t \in [0, H]$, see our work [Díaz *et al.*, 2013] for full description of the procedure for calculating the values $B(t)$).

The schedule execution process starts with the computation of a feasible and robust baseline schedule which synthesizes the rover behaviour on the success of the mission goals. More concretely, according to our MSR-based mission scenario, the rover receives a high-level description of a specific scientific experiments set, and exploits the profile-based, power-aware reasoning algorithm $ESTA^p$ [Díaz *et al.*, 2013] to provide an initial solution schedule so that all the considered mission constraints are satisfied. Furthermore, the same reasoning capabilities are deployed at the core of the execution control loop with the twofold aim of: (a) updating the internal model with the feedback provided by the execution monitor, and (b) repairing the possible misalignments arising on the face of incoming disturbances.

Very roughly, $ESTA^p$ represents a resource-driven, heuristic-biased solving algorithm able of reasoning upon a wide variety of temporal and resource constraints such as sequence-dependent setup-times (time lags between the execution of the rover activity pairs), maximum time-windows (maximum durations), multi-capacity resource demands and power production/consumptions. More in detail, the core characteristics of $ESTA^p$ are summarized as follows:

– **Multi-capacity resource demands and setup-time precedence relations.** The underlying *problem-solving strategy* basically combines the *precedence constraint posting* heuristic guideline used by the reference solving algorithm ESTA [Cesta, Oddi, & Smith, 2002], with an adaptation of the *dominance conditions* of the SP-PCP (Shortest Path-based Precedence Constraint Posting) algorithm proposed in [Oddi *et al.*, 2009] and Oddi2011. The *hybridization* of both previous techniques within a unique solving scheme provides (multi-capacity) cumulative resource allocation reasoning and setup-time (temporally bounded) precedence relations. For instance, in our reference mission scenario, the algorithm allows to allocate the rover activities by resolving resource over-consumptions, taking into account both the sample cache maximum capacity (i.e., a multi-capacity cumulative resource) and the travelling distances (i.e., setup-times) involved within the execution of two consecutive soil extraction activities $Drill_i$.

– **Power-awareness.** We adopted a well-know reference model proposed by

Simonis [Simonis & Cornelissens, 1995] to represent the battery resource (a consumable resource by definition) by means of a typical cumulative-based scheme as explained earlier in this section. Then, we merged this model with the general constraint-based representation framework, and introduced some enhancements within the underlying heuristic guideline in order to provide efficient power-based reasoning capabilities.

The knowledge encapsulated within the $ESTA^p$ heuristic guidelines allows the algorithm to iteratively balance the *resource contention peaks* (i.e., resource overcommitments) by posting new precedence constraints between rover activities until either a conflict-free schedule is found or a temporal inconsistency arises. Figure 6.7 illustrates one possible solution for a problem example like the one shown in figure 6.5 with three scientific experiments: it consists of a consistent temporal reallocation of the mission activities so that all the resource demands are contained within the permitted resource usage boundaries; the algorithm posted a new set of precedence constraints between the activities belonging to different experiments by separating the activities involved in resource overcommitments with the aim of *flattening the resource usage profiles* between the maximum and minimum usage limits.

6.3.2 Command dispatching and execution monitoring

Once the controller succeeds in computing a scheduling solution, the next step is about extracting the low-level command sequences related to each rover activity from the current schedule, and implementing an *execution strategy* by timely dispatching the commands to the rover for proper enactment. Given a feasible schedule, a command sequence is extracted in the shape of a *timeline* whose intervals (or tokens) represent the singular tasks to be executed in order to fulfill the plan's goals.

Figure 6.8 shows the command sequence timeline extracted from the same scheduling solution presented in Figure 6.7. As can be seen from the figure, the translation in this particular example is unambiguous. Every *Drill* and/or *Release* command (see the dashed down-arrows) is dispatched correspondingly to the start time of the related scheduling solution activity; all distance intervals between any *Drill* and/or any *Release* operation is interpreted as a *Navigation* activity, and the related Nav com-

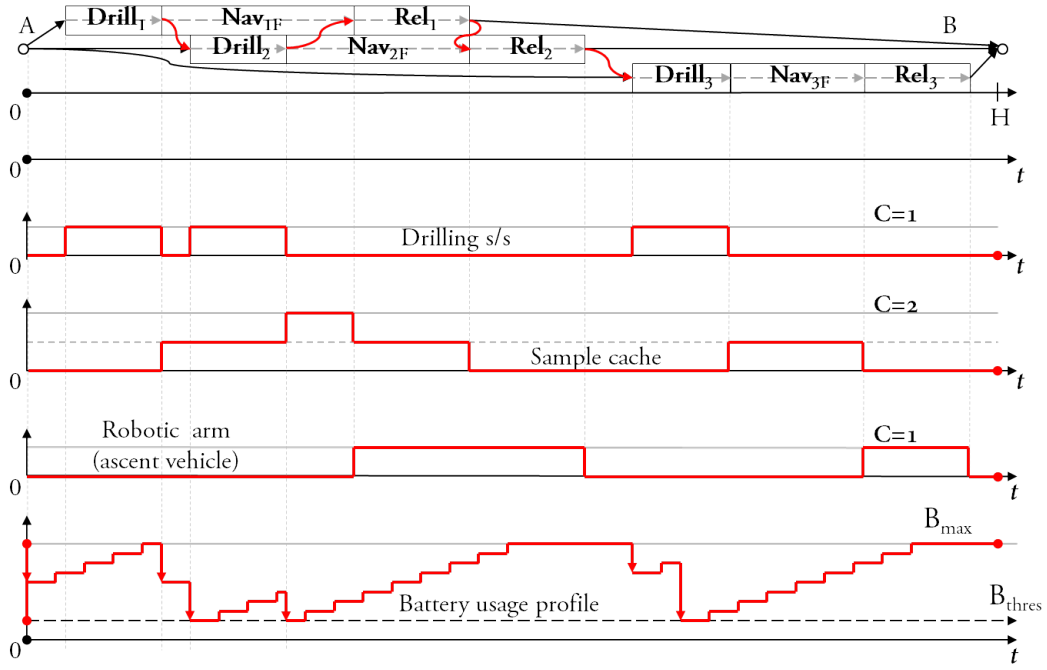

Figure 6.7: Feasible solution example: a set of solving constraints are posted so that all resource conflicts are solved

mand is issued accordingly. The executability of the resulting command sequence is a direct consequence of the temporal and resource feasibility of the scheduling solution the timeline is extracted from. As we will see in the next examples, the onset of possible ambiguities in the command timeline extraction will be solved by choosing one of the available execution strategies.

During the whole schedule execution process, the control system is at the same time continuously requesting telemetry data to the execution monitor so that the real execution status is reliably known at every moment. More concretely, the execution monitor collects the following specific execution feedback information:

– **Rover position and orientation.** This information is sampled to determine if the rover was delayed on the execution of locomotion activities. The monitor performs a temporal analysis on the evolution of the distance (both linear and angular) covered by the rover when moving between two different way-

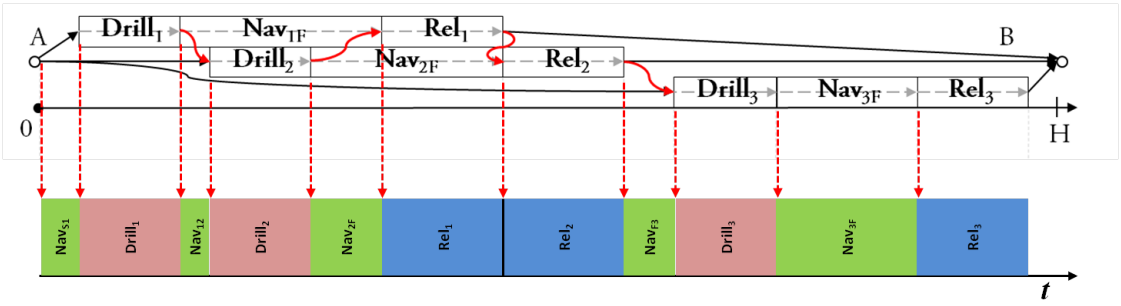

Figure 6.8: An example of Command Sequence extraction.

points by checking that the real and the estimated distances correctly match each other.

– **Command completion acknowledgement.** The execution monitor is also able of determining if the rover was delayed on the execution of every command by querying for its completion signal.

– **Battery State of Charge (SoC).** The detection of an anomalous battery usage (i.e., the battery SoC exhibits an unforeseen loss of energy along a specific time interval) is performed through the interpretation and comparison of the evolution of the real battery SoC with respect to the expected power consumption/production profile. For instance, if the rover is executing a sample collection activity, the monitor checks that the SoC values evolve according to the estimations contained within its model, i.e., by expecting that the SoC values suitably match with the estimated power and production rates related to such rover activity in nominal conditions (with a specific error margin).

According to the previous baseline schedule example of figures 6.7 and 6.8, the execution would proceed as follows: first, the rover moves to the experiment location regarding to the first experiment (as explained in section 6.3.1, the first navigation activity is simply modelled as a setup-time separation constraint), and performs the soil extraction and storage operations; then, the rover navigates towards the second experiment location and collects the related sample (at this point, it is worth to note that the sample cache is used at its maximum capacity $C = 2$); afterwards, the rover

goes to the final location and releases the two previous collected samples; finally, the rover travels to the last experiment location and proceeds as before to complete the science, navigation and sample release activities.

6.3.3 Contingency solving

The presence of environmental uncertainty requires the autonomous controller to be provided with flexible reactive strategies that allow the rover to adapt current solution schedule and generate alternative plans on the face of contingent situations. Concretely, contingency solving process continuously aims at restoring mission execution feasibility, by performing the following set of actions when a non-nominal situation is detected by the execution monitor:

– **Contingency effects projection and propagation (internal model updating).** Effects of the detected contingencies (in terms of temporal displacements and resource overconsumptions) are projected within its internal model with the aim of re-aligning it with respect to the real execution status. The internal model updating entails *injecting and propagating additional (temporal or resource) constraints within the nominal schedule*, so that the internal model becomes synchronized with the new situation.

  In case of temporal misalignment (e.g., an unexpected lag on the rover navigation between two different locations) the controller posts a new temporal constraint within its internal model so that the execution termination of the running activity is postponed. The effects stemming from the introduction of a punctual delay are propagated through the whole schedule by suitably stretching the underlying temporal network. Similarly, if a power overconsumption is detected, the controller reflects the amount of excess demand within the internal model and re-estimates the overall battery usage profile.

– **Global consistency checking and contingency solving.** Right after the internal model is updated, a global consistency checking process is performed with the aim of finding new temporal or resource conflicts. Figure 6.9 illustrates the three possible situations which might arise on this process: (a) if neither

temporal nor resource conflicts are detected (i.e., the nominal schedule succeed at absorbing the contingency effects so that no conflicts were created), the execution continues as expected; (b) if a *resource conflict* is introduced, the controller *triggers a re-scheduling process* on the attempt of solving the new resource overconsumption through the execution of a resolution strategy which exploits the $ESTA^p$ algorithm at its core; (c) if a *temporal inconsistency* arises (e.g., the maximum temporal horizon boundary is overpassed), then no solution can be provided and the mission execution is momentarily suspended (a human intervention is required).

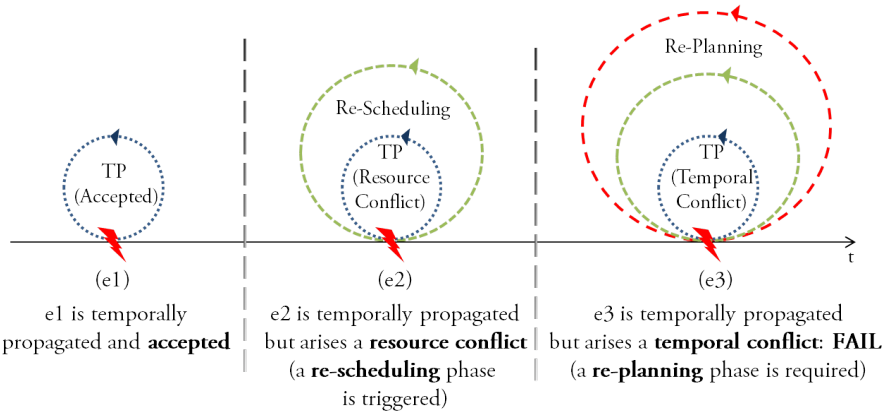

Figure 6.9: The three possible situations which might occur when the effects of an exogenous event e_i are injected and propagated.

During the contingency solving phase, a number of mutually conflicting issues generally arise. On the one hand, the need to rapidly provide an alternative solution in order to keep the execution going; on the other hand, the need to keep the "*plan stability*" under control. We identify the plan stability as an evaluation metric to inspect the plan execution process quality in terms of *minimal perturbation* ([Simmons & G., 1988], [Kambhampati & Hendler, 1992]); in particular, the plan stability is assessed by counting the number of activity pairs whose ordering has been reversed to refer to the differences that the reactive strategy induces before and after re-scheduling. In this context, we differentiate two plan recovering approaches, namely *local re-scheduling* and *from-scratch*

re-scheduling. By local re-scheduling we mean the repair process of adapting an invalidated plan to the new execution context whilst perturbing the plan as little as possible, i.e., by conserving its initial structure to the maximum extent possible. By contrast, from-scratch re-scheduling consists of the generation of a new plan without considering stability, so that the current plan is partially destroyed before re-scheduling by removing all the posted constraints resulting from previous scheduling steps. In general, plan stability is claimed as a valuable property to be considered on the design of reactive or adaptive strategies as demonstrated on [Fox *et al.*, 2006]: local repairing approaches typically lead to good quality plans in a shorter time and by performing a minimum computational effort (time and resources are precious and scarce), by assuming that the original plan can be a very useful guide to the synthesis of the new plan.

6.3.4 Re-scheduling Examples

The $CoRe^p$ execution control architecture executes rover sequence commands and acknowledges their effects from ESA's 3DROV planetary rover system simulator at a coarse level of granularity. For instance, the system handles information such as rover position and/or orientation changes, battery state of charge measurements over straight-line traversals and drill operations, and acknowledges high-level task completion signals, leaving the management of the issues such as obstacle avoidance or lower-level rover kinematics to the 3DROV simulator. Yet, as we shall see, the chosen share of responsibilities is sufficient to provide the rover with autonomous reasoning capabilities for long-range and time-extended missions that involve mission-level decisions against possible deviations from the baseline plan.

In the following, we intend to provide a couple of examples of typical situations that require high-level decision-making capabilities which might be easily handled without time-consuming human control. In the two examples that follow, the rover cache capacity is supposed to be equal to 3.

Example 1 Figure 6.10 (right side) presents a very simple plan synthesized as a solution of the rover exploration problem illustrated on the left side, composed of three locations to be visited to perform experiments, and carrying the soil samples

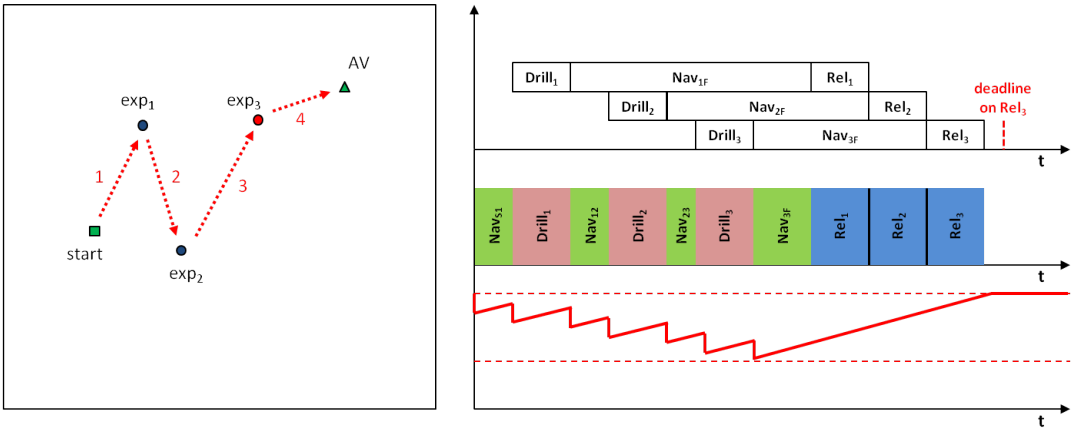

Figure 6.10: Example 1: Sketch representing a problem instance with 3 experiments (left); feasible plan solution (right), where the activity sequences are spotted at the top, the equivalent execution timeline in the middle, and the battery usage profile at the bottom.

to the Ascent Vehicle (AV) location. In the figure, the plan is presented in terms of the baseline schedule (top-right), the related command *timeline* (middle-right), and finally the power consumption (state of charge - SoC) profile (bottom-right). As can be observed, the solution entails the rover to move from the initial location to the exp_1 location (Nav_{S1}), perform the first experiment ($Drill_1$), move to the exp_2 location to perform the second experiment, and so forth. When all soil samples are collected, the plan entails that the rover finally moves to the AV location in order to release all samples in sequence. Note also that there exists a deadline constraint on the release time of sample #3, which is satisfied by the current solution. Relatively to the power consumption profile, please note that: (i) all necessary power is consumed at the beginning of each task (*Drill* and *Nav*), and (ii) the Release (*Rel*) tasks do not consume energy because the samples are collected by the AV's arm.

Let us now assume that the rover suffers an unpredicted overconsumption problem during the first traversal (see Figure 6.11, left side). Immediately after acknowledging the overconsumption exogenous event from the 3DROV Simulator, the event is injected and propagated in the current plan representation (see the bold down-arrow in the power profile of Figure 6.11, right side). As the figure shows, the projection of

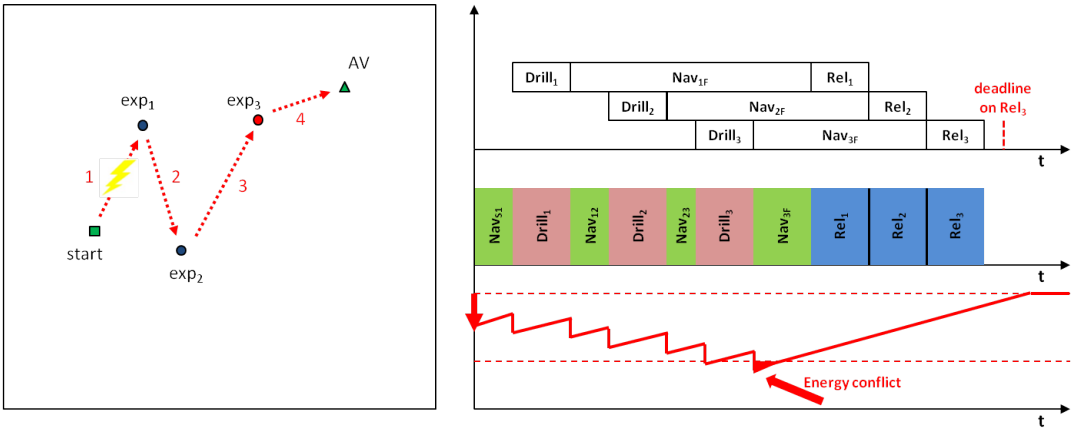

Figure 6.11: The rover suffers an unpredicted battery overconsumption during the first traversal, and an energy conflict is flagged.

the overconsumption's effects along the whole plan immediately generates an *energy conflict*, as the planned battery charging intervals are no longer sufficient to guarantee all the necessary power to the rover's future activities (i.e., the battery SoC profile falls below the minimum safety threshold).

Given this situation, a plan repair is necessary; it should be highlighted at this point that our model is *predictive*, in that it propagates the consequences of exogenous events in the future, therefore allowing the solver to re-adjust the solution in time while continuing the plan's execution. In the example at hand, the $CoRe^p$ system can keep executing the plan while at the same time re-scheduling the plan's future activities in order to resolve the energy conflict.

Figure 6.12 depicts a possible alternative solution which would be valid if the deadline constraint on the Rel_3 is not considered. Such solution would entail adding a battery charging activity after arriving to exp_3's locations, in order to store enough power to execute both the next $Drill_3$ and Nav_{3F} tasks, thus safely completing the plan to the AV location. Unfortunately though, this solution is invalid because of the existence of the deadline constraint in Rel_3.

Figure 6.13 presents a valid solution to the previous problem. As it can be observed, this solution is characterized by a worse makespan with respect to the previous one, but is constructed so as to respect the deadline constraint on Rel_3.

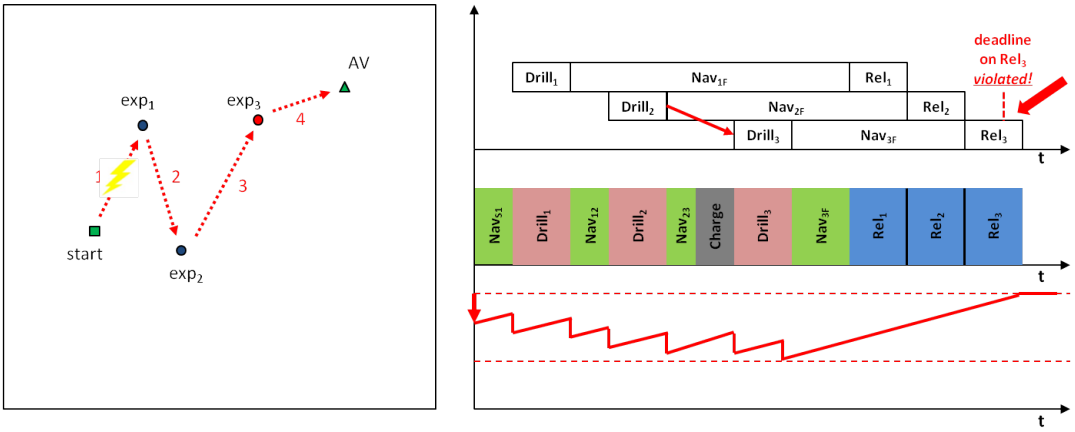

Figure 6.12: Possible solution to the energy conflict if the deadline constraint on Rel_3 was not considered: a new battery charging activity is added right before the execution of the $Drill_3$ activity.

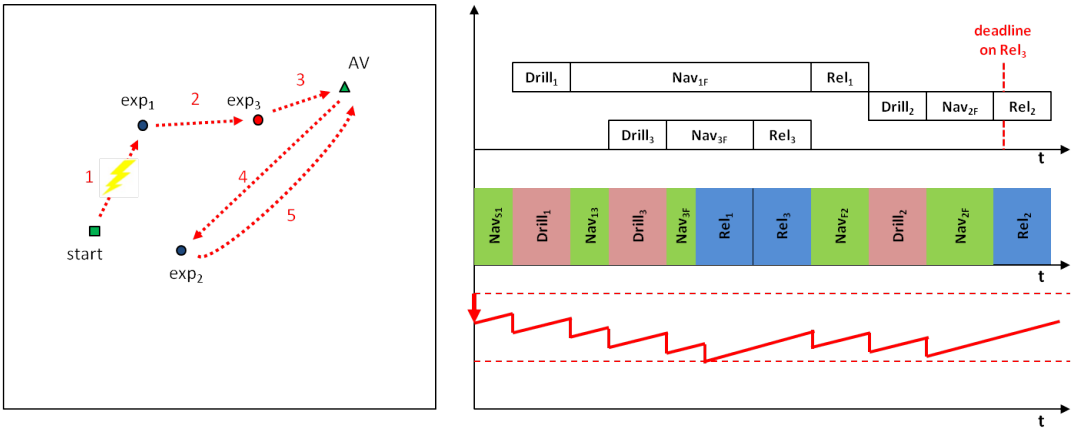

Figure 6.13: Feasible solution where the deadline constraint on Rel_3 is satisfied: the rover activities were completely reorganized.

This solution represents a clear example of how the rover is able to autonomously perform a high-level reasoning, significantly re-organizing the remaining experiments to perform, to the twofold aim of (i) maximizing the scientific return, and (ii) keeping all mission constraints satisfied. Observing the new solution, the rover not only decides to exchange the execution order of exp_2 and exp_3 tasks, but it also

chooses to deliver to the AV the samples #1 and #3 before returning to the exp_2 location in order to complete the plan (the reader can easily convince himself that simply exchanging the exp_2 and exp_3 order would still have violated the deadline).

Example 2 Figure 6.14 depicts a situation where the rover is called to execute two experiments exp_1 and exp_2 at two different locations, starting from the initial location $start$, and deliver both the acquired samples to the AV. In the depicted situation, the rover's activities are close to the night *hibernation period* (i.e., the black interval in the solution's timeline), which has a fixed duration and cannot be moved. In our model, the night periods are represented as "fixed activities" that demand an amount of energy proportional to their respective durations, and during which no power charging occurs (see the Battery SoC profile in Figure 6.14). In other words, the on-board reasoner is deputed to find a solution that takes into account the stringent power constraint imposed by the need to keep all instrument warm for the whole duration of the night inactivity period.

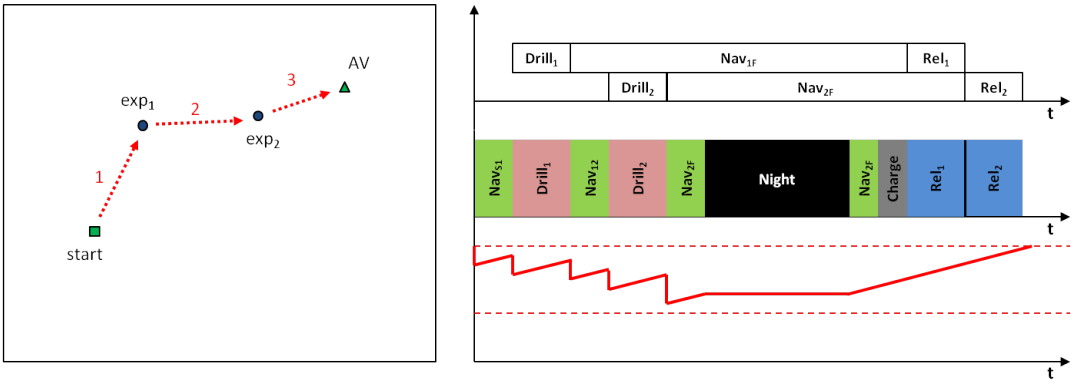

Figure 6.14: Example 2: Sketch representing a problem instance with 2 experiments and an hibernation period (left); a feasible plan solution is depicted (right), where the activity sequences are spotted at the top, the equivalent execution timeline in the middle, and the battery usage profile at the bottom.

Let us now suppose that an overconsumption issue occurs during the rover's first traversal Nav_{S1}; like in the previous example, this unpredicted event is immediately acknowledged and propagated in the plan's model. The results of such propagation

are depicted in Figure 6.15, which shows the onset of an energy conflict to occur right after the hibernation period. In this particular example, the rover autonomously acknowledges the fact that the current battery charge does not allow to safely reprise operations after the night time. Again, it is necessary to re-adjust the current solution to restore feasibility.

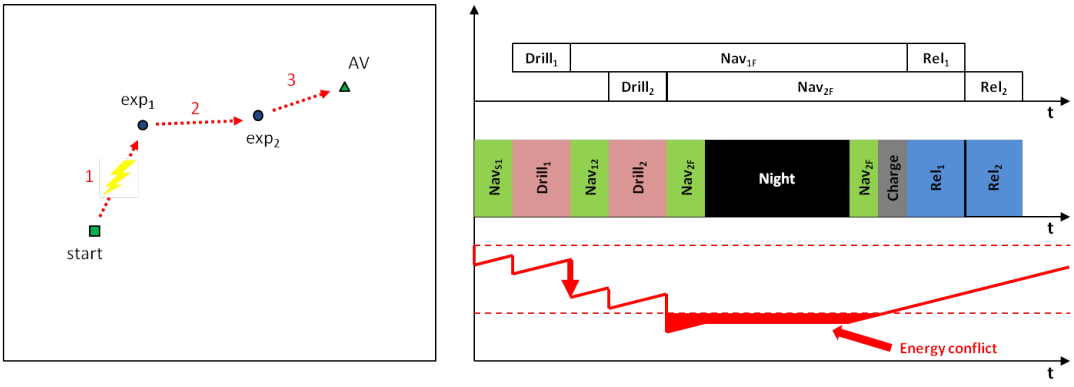

Figure 6.15: The rover suffers an unpredicted battery overconsumption during the first traversal, and an energy conflict is flagged.

Figure 6.16 finally depicts two different solutions to the previous problem, both feasible, but characterized by different makespan. In fact, the $CoRe^p$ not only attempts to restore feasibility providing a new solution as soon as possible to allow the plan's execution, but it also performs a continuous optimization process through which the restored solution is optimized. Figure 6.16 presents two different steps of this optimization process; the solution labelled (b) represents an example of non-optimal plan, which consists of executing the first two drills before the hibernation period, leaving the battery with just enough the power to safely pass the night, and spending the first part of the next day charging the battery before executing the last traversal.

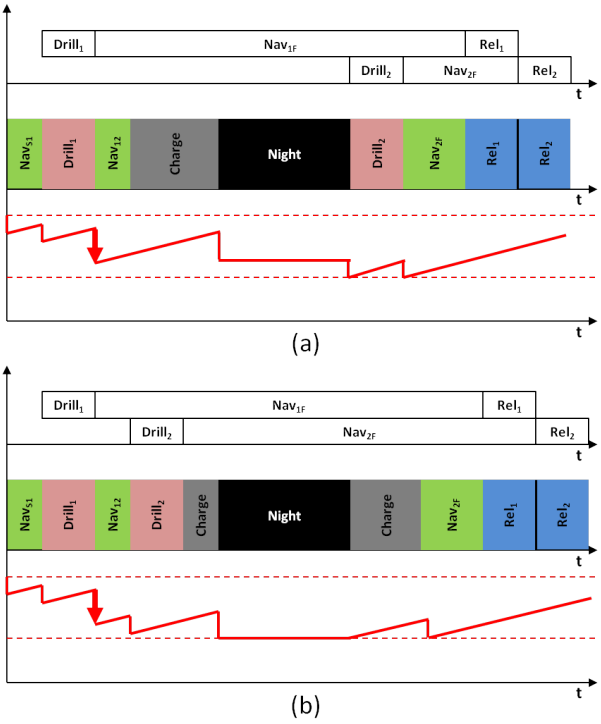

Figure 6.16: Two feasible solutions: the second experiment is postponed to the second day (a) by inserting a charging activity before the hibernation; both drills activities are executed before the hibernation, by increasing the charging activity duration after the hibernation.

The solution labelled (a) represents a better solution, where the rover decides to postpone the second drill to the following day after reaching the exp_2 location, thus allowing for more battery power for the night, thus allowing the execution of the second drill right after the night period.

6.4 Summary

In this chapter we have presented an integrated model-based execution control architecture ($CoRe^p$) targeted at generating and safely executing robust mission plans, by exploiting a constraint-based flexible model that allows to:(a) represent both time and resources, with special attention to the rover power requirements; (b) integrate

plan re-scheduling and plan execution on top of the same representation, allowing to hedge against domain uncertainty by reacting to possible incoming disturbances and dynamically modifying the plan (e.g., to respond to unexpected science opportunities); and (c) perform on-line plan optimization concurrently to plan execution.

With the aim of conducting a formal analysis on the performance of the core capabilities of the autonomous controller, we built an integrated testbed platform on top of the 3DROV planetary rover system simulator, an ESA asset that provides a realistic behavioural reproduction (i.e., from the temporal, dynamic and energy-related standpoint) of all the rover subsystems (e.g., locomotion, drilling, etc,) necessary to perform the tasks covered in our experimental model.

Chapter 7

An Analysis on the Performance of the Execution Control Process

In this part of the dissertation we start presenting an comprehensive testbed platform integrated with 3DROV (and advanced software simulation platform), created with the twofold aim of (i) validating the whole $CoRe^p$ autonomous control architecture; and (ii) performing an experimental analysis on the performance of the control architecture's capabilities, throughout two representative cases of study. Next, we conduct an analysis on the performance of the basic target capabilities of the controller running on the integrated testbed platform, through the dynamic simulation of two representative study cases.

7.1 The Integrated Testbed Platform

We developed a testing workbench platform which exploits the capabilities of the 3DROV simulation framework according to following the configuration set-up (see figure 7.1):

– Our **autonomous control system** $CoRe^p$ was developed on top of the APSI-TRF [Cesta & Fratini, 2008; Cesta *et al.*, 2009], an advanced constraint-based software development environment which provides the necessary assets to model and reason upon instances of our problem domain of reference according to

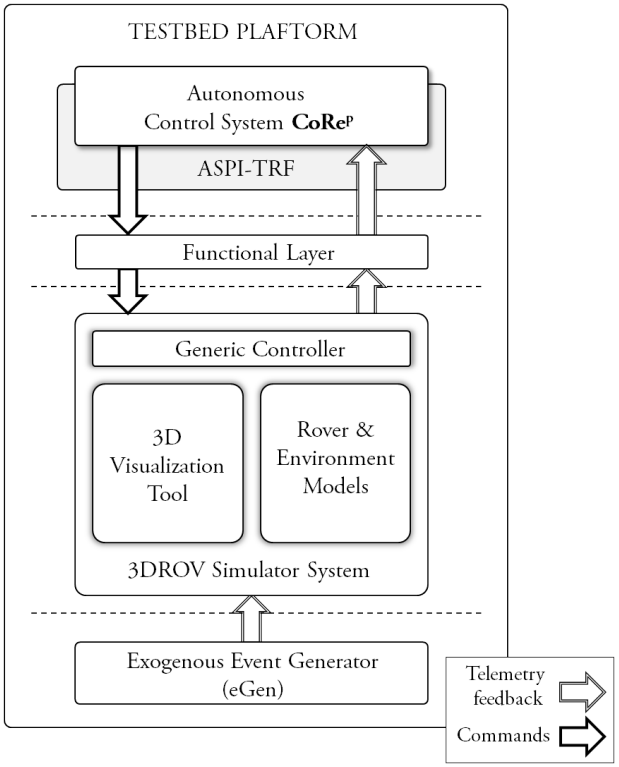

Figure 7.1: The integrated testbed platform with 3DROV simulator.

a CSP representational and reasoning scheme. $CoRe^p$ system was designed according to a multi-thread pattern which allows to execute the different execution processes involved in the Sense-Plan-Act loop control scheme in a concurrent fashion like the telemetry management & command dispatching, execution monitoring & consistency checking, contingency solving and continuous improvement.

– We used the 3DROV [Poulakis *et al.*, 2008] tool, an advanced software simulation platform created to the aim of providing ESA's A&R section with a supporting tool for the complete development process of planetary robot systems within *realistic mission scenarios*.

  One of the many interesting features of 3DROV is that it provides the necessary elements to accurately reproduce both the robotic systems and the envi-

ronmental surrounding aspects involved within a planetary mission operation context. Its modular and distributed scheme design (see figure 7.1) basically consists of the following integrated building blocks: the *Generic Controller*, the *Rover and Environment models* and the *3D visualization component*. The Generic Controller encapsulates all the required functionality [1] to operate with all the rover subsystems (i.e., locomotion, drill, sample cache, etc.). It relies on the ESA's SIMSAT simulation framework [1], and assumes the role of the on-board flight software by controlling all the rover operations at low level, and providing standard simulation services such as scheduling of events and time management. The Generic Controller represents the 3DROV's interface with our autonomous control system architecture via the *Functional Layer*, which will be described later. The Rover and Environment models are a compound of accurate and integrated models representing both: (i) rover physical s/ss like kinematics, power or thermal, sensors and scientific instruments; and (ii) environmental features like the ephemeris and timekeeping, terrain and atmospheric conditions. The 3D visualization component is the front-end of the 3DROV simulator and allows us to track (in real-time) the evolution of the simulation execution through a 3D representation of the complete mission scenario.

3DROV provides a basic controller which can be used as a rudimentary interface to external systems, and enables a safely operation of the rover in a *close-to-real-time* manner. Concretely, the nominal simulation speed provided by the rover is 60 times slower than real time, but allows to be (safely) accelerated in a factor of 10, so the simulation speed results only 6 times slower than real time. Therefore, if we consider an accelerated simulation framework, the clock frequency used by the $CoRe^p$ controller must be reduced in a factor of 6 in order to make both systems temporally consistent (synchronous). The connection between both systems was implemented through a simple *Remote Procedure Call (RPC) schema*: a socket-based communication protocol which defines the

[1] 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) [Putz & Elfving, 1992] framework.

semantics of the communication dialogue of two computer programs (client-server), and encapsulates the details for this remote interaction. According to this schema, the simulator's controller implements the server stub, and therefore our controller implements a set of clients to commit the rover commands and query for the telemetry. It is important to mention that, despite its simplicity, this communication scheme allows to manage several requests at a time, a fundamental issue given the multi-threaded nature of our controller: it might perform different telemetry requests (orientation, position, SoC and correct command termination checking) and initiates a new rover command execution in the same simulation cycle.

– Between the controller and the 3DROV simulation system we designed a middle module as a **functional layer** with the following purposes: (i) interfacing both systems (i.e., the controller and 3DROV) through a TCP/IP-based communication relay based on a simple Remote Procedure Call (RPC) schema[2], (ii) performing the downstream translation from the high-level rover activities defining the solution schedule to low-level command sequences to operate the different rover subsystems in 3DROV; and (iii) requesting and performing an upstream translation of the telemetry data provided by the simulator.

  Figure 7.2 depicts the data transfer flowing from the $CoRe^p$ system to the 3DROV simulator on each execution cycle: the execution of each rover activity entails a downwards translation to predefined sequences of low level commands which directly operates the rover actuators. In the other way around, the execution monitor continuously collects telemetry data which is interpreted back to reliably determine the execution status, like the rover position and orientation, command completion acknowledgement and the battery state of charge (SoC).

– Although the 3DROV simulator provides a realistic and meaningful source of uncertainty, we decided to couple it with an **exogenous event generator**

[2] A socket-based communication protocol which defines the semantics of the communication dialogue of both the controller (client) and 3DROV simulator (server), and encapsulates the details for this remote interaction

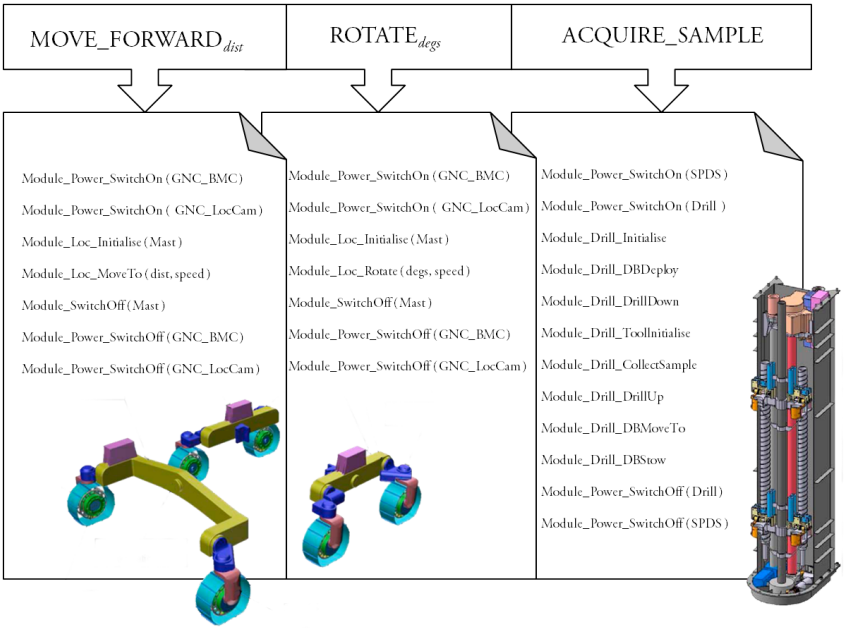

Figure 7.2: Top-down decomposition of the three main commands: move forward, rotate and acquire sample.

(eGen) which "increases the degree of uncertainty" in a controlled fashion, with the aim of augmenting the level of stress posed on the controller's reactive mechanisms. Concretely, eGen provides a second source of uncertainty and supports the definition and injection of a set of *exogenous events* to modify the nominal rover behaviour at a certain extent. An exogenous event is characterized through the following parameters: (a) by its class or type (i.e., linear or angular speed reduction/increment, and battery consumption increment/reduction); (b) the instant at which it starts affecting the rover behaviour; and (c) its magnitude or extent. The set of exogenous events are defined before the simulation execution starts and are injected *on-the-fly*. For instance, we might schedule a temporal exogenous event which provokes a reduction on the rover speed of $0.2m/s$. during the whole execution of the second navigation activity. In this way we expect a major number of contingencies detected and suitably addressed by the controller.

Next, we conduct an experimental analysis on the performance of the controller through the execution of two representative cases of study.

The experimental analysis is targeted at both validating the main concepts and capabilities of the autonomous controller $CoRe^p$, and provide a quantitative insight on the performance of the control architecture.

The following analysis is focused on the assessment of the key characteristics of the control architecture:

- **Domain model correctness and consistency:** checking that the constraint-based representation model actually encapsulates the most important characteristics of the MSR-like mission scenario of reference, and represent an accurate and meaningful depiction of the mission execution status along the whole simulation process.

- **Reasoning capabilities performance:** checking that the controller succeeds in computing feasible but also efficient (in terms of solution quality) solution schedules that represent realistic estimations in time, resource usage and power demand when applied to practical cases of study.

- **Uncertainty management:** checking the robustness of the solution schedules on the absorption of temporal and resource variations during their execution, as well as the controller's flexibility and efficacy on the deployment of reactive mechanism to face with environmental disturbances (including the dynamic insertion of new experiments).

7.2 Experimental Settings

The study cases here considered represent two different simulations of the same problem instance (Table 7.1 summarizes its parameter settings): the problem entails the synthesis and execution of a schedule solution consisting of five scientific experiments allocated in a 50 m^2 area of easy traversability. The rover contains a sample cache with a transportation capacity of up to three soil samples, hence the rover is forced to reach the ascent vehicle and release the first soil samples *at least once* be-

fore returning to the experiment locations and complete the plan. Finally, the battery is not allowed to be depleted below 95% of its total capacity.

Table 7.1: Configuration parameters of the baseline problem instance

Baseline problem features		
Attribute	*Value*	*Observations*
Num. Experiments	5 experiments	–
Terrain size	50 m^2	–
Sample Cache cap.	3 samples	Max number of soil samples transported concurrently
Working cycle duration	1440 mins	–
Hibernation (inactivity) period duration	400 sec.	–
Activity period duration	1040 mins	–
Soil Extraction & Storage activity dur.	170 mins	–
Release Sample activity dur.	5 mins	–
Linear & rot. speed	1 m./sec. (linear) 12 deg./sec. (rot)	Rover moves by performing straight traversals and/or simple rotations
Max. battery usage threshold	5%	Battery starts 100% charged
Power cons. locomotion	0.235% (1 meter or degree)	–
Power cons. soil extraction & storage	9%	In total
Power cons. heater	0.0001% (1 sec.)	–
Power production	0.024% (1 sec.)	–
Execution feedback sampling rate	1 Hz	The frequency at which the controller acquires execution data

In the simulations, the time scale was intentionally accelerated by a factor of 60 so that a full working cycle is collapsed into minutes. The aim was to allow to complete

a realistic mission simulation execution, which might span several martian sols, into a few hours. In the case under examination, the complete working cycle duration is 1440 s. long with an inactivity (hibernation) period lasting 400 s., meaning that the rover activity period duration takes about 1040 s. per working cycle. It is worth to mention that hibernations are represented by periods of darkness (i.e., no battery charging is possible) where the rover remains in a suspended mode with a low-power consumption.

Two simulations were executed to the aim of testing the main controller's capabilities, in particular its adaptability under two different scenarios characterized by different levels of uncertainty. In the first simulation (MSR_0) all the information about the scientific experiments to perform is known from the beginning of the execution, and the only source of uncertainty considered is the one naturally provided by the simulator itself (i.e., eGen is not used); in the second simulation (MSR_1), four experiments are scheduled from scratch to constitute the baseline schedule, while the fifth experiment is injected and scheduled *on-the-fly*. In addition, a set of exogenous events are injected during the simulation execution (via the *eGen* Exogenous Event generator), in order to further disturb the rover's nominal behaviour in a controllable fashion. Table 7.2 presents the set of exogenous events injected during the second simulation, describing the events' type, magnitude and affected plan's activity.

Table 7.2: Exogenous events description (second simulation MSR_1 execution)

Set of exogenous events injected during MSR_1		
Exogenous Event Type	*Value*	*Affected activity*
Linear speed reduction	-0.5 m/sec.	Plan's 1^{st} Traversal
Consumption rate increment	+0.01%	Plan's 1^{st} Traversal
Linear speed reduction	-0.3 m/sec.	Plan's 3^{rd} Traversal
Angular speed reduction	-6 deg/sec.	Plan's 2^{nd} Rotation
Consumption rate increment	+0.02%	Plan's 2^{nd} Rotation

During the simulations, plan executability is continuously maintained through re-schedulings each time the current solution is made infeasible due to the occurrences of disturbing events.

7.3 Experimental Results

In order to realize a trade-off between reactivity and solution quality (see Section 6.3.3), the re-scheduling strategy applied in case of resource conflict detection is as follows: on a first attempt, the controller iterates $ESTA^p$ *"n times"* (where $n \leq 10$) in a stochastic fashion. If the reasoner does not succeed at finding a feasible solution, on a second attempt the same reasoning process is repeated by first removing all the previously imposed solving constraints: all the temporal constraints injected during both the initial resolution process and all the previous re-scheduling steps are retracted, before re-scheduling; in this way, the flexibility of the schedule in execution (and thus the reasoner's freedom of decision) is significantly increased, i.e., it is more likely to succeed. In case of a success, the resulting schedule solution consists of a similar ordering where the activities which are pending to be executed are reshuffled and/or delayed with respect to the previous solution.

Figures 7.3 and 7.4 present the preliminary experimental results corresponding to the plan executions obtained in both simulations. In the figures, the x axis represents time (in seconds), and the ticks mark the instants (not in scale) at which a new plan has been produced during the execution due to the onset of a conflict making the current solution infeasible, while the y axis provides a measurement of the solution makespan (in minutes). All produced solutions are highlighted with labels characterizing the explanation behind the execution of the related re-scheduling process. The labels have the following meanings:

- *RD (Rotation Delay)* = execution delay due to slow rotation
- *CTD (Command Termination Delay)* = the commanded task has ended later than expected
- *TD (Traversal Delay)* = the traversal activity has ended later than expected
- *NE (New Experiment)* = a new experiment has been added to the plan
- *BOT (Battery Overconsumption on Traversal)* = battery overconsumption during a traversal

- *BOR (Battery Overconsumption on Rotation)* = battery overconsumption during a rotation.

Lastly, the $Mksp$ column represents the solution makespan length (expressed in minutes).

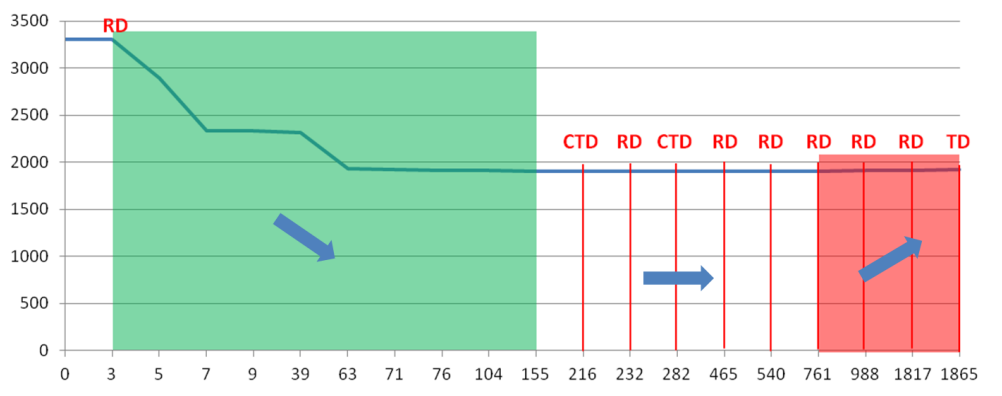

Figure 7.3: Set of feasible solutions generated during the first simulation, where the x axis represents the instants (in seconds) at which plans were synthesized (not scaled), and the y axis provides a measurement of the solution makespan (in minutes).

Each simulation described by both figures entails the *off-line* synthesis of a baseline low quality solution, immediately followed by the execution start as soon as the baseline solution is ready. The idea behind this choice is to demonstrate the *on-line* optimization and control capabilities of the system, by simulating from the start a typical situation where the scheduler is called to continuously attempt new improvements to the running solution, as the latter is being executed. Of course, only the part of the solution which has not yet commenced execution undergoes the optimization process.

As described in Figure 7.3, the MSR_0 simulation starts the execution of a high makespan ($\approx$ 3300) solution, when a rotational delay (RD) is detected and propagated, and a new solution is found at $t = 3$. The stability of the makespan indicates that the event was absorbed by the flexibility of the solution. At the same time, the optimization process performs a series of makespan improvements on the current solution (green area in Figure 7.3) by readjusting the execution order of the future experiments' tasks, ultimately allowing the makespan to decrease to the 1906 value

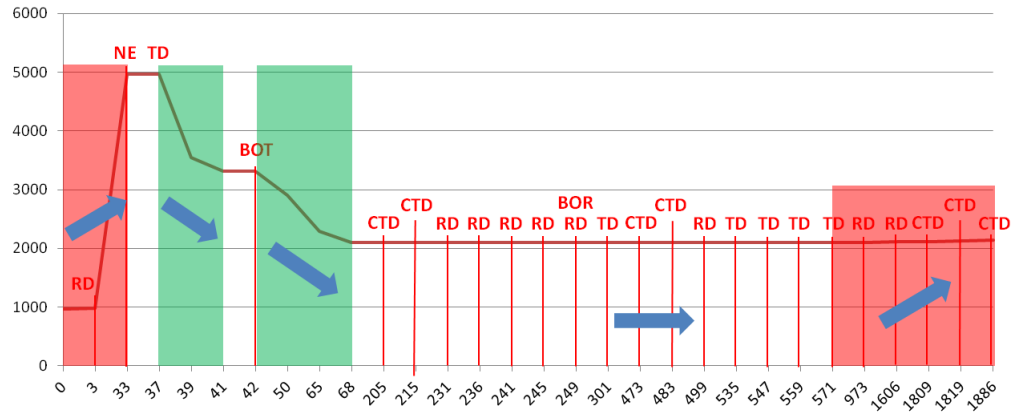

Figure 7.4: Set of feasible solutions generated during the second simulation, where the x axis represents the instants (in seconds) at which plans were synthesized (not scaled), and the y axis provides a measurement of the solution makespan (in minutes).

by $t = 155$. No need to perform readjustments due to non-nominal behaviour on behalf of 3DROV is detected during this time.

From the instant $t = 216$ onward, the telemetry data obtained from the rover determine a number of significant deviations from the rover's nominal performance, hence requiring the production of new solutions. As the figure show, five alternative solutions are synthesized between the instants $t = 216$ and $t = 540$. It should be observed that the solution's makespan remains unchanged despite the re-schedulings; this is due to two main reasons: (i) the temporal flexibility of the solutions, which allows to absorb temporal delays within a certain extent, in combination with (ii) the re-scheduling policy used, which attempts to maintain a certain degree of *continuity* between the various solutions, allowing a heavy re-shuffling of the activities only when no solution can be found otherwise.

During the last part of the execution described in Figure 7.3 (red area between $t = 761$ and $t = 1865$) four more solutions are computed due to misalignments between the scheduling model's predictions and the real 3DROV's performances. Incidentally, it can be observed that these new solutions do entail an slight increase of the solution's makespan (i.e., from 1906 to 1923), due to the fact that the solution's flexibility can no longer accommodate further deviations without affecting the plan's

quality. Lastly, it should be noted how, once the solution has reached a good quality level (i.e., $makespan = 1906$ at $t = 155$), the optimization does not find any further improvements for the rest of the execution.

Figure 7.4 describes the second execution simulation. As previously mentioned, in this case only four of the five experiments to be executed by the rover are known at creation time of the baseline solution; the fifth experiment location is provided at plan execution time. The idea is to demonstrate the capabilities of the reasoner to accept, schedule and optimize new activities in the current plan as the latter is being executed.

As the figure shows, the baseline plan is characterized by a makespan equal to 973, i.e., a rather good quality solution; after the initial solution is obtained, the execution can start. As in the previous MSR_0 simulation, a rotational delay (RD) is detected and a new solution is found at $t = 3$. As opposed with the previous case, in the current MSR_1 simulation there is a slight increase in the makespan (see the upward arrow in the picture), due to the fact that the running plan is already very tight, and therefore its inherent flexibility (i.e., the capability to absorb delays) is limited. Seconds later, the fifth experiment is added to the plan and a new solution representing the complete routing for all tasks to be executed is synthesized at $t = 33$. Note that this solution entails a significant increase in the makespan ($= 4960$), as the plan represents the first feasible solution found, which must still undergo optimization. The optimization phase continuously running in the background produces five new solutions in the interval between $t = 37$ and $t = 68$. As Figure 7.4 shows, the optimization steps are interleaved with two re-schedulings due to a traversal delay (TD at $t = 37$) and a battery overconsumption on traversal (BOT at $t = 42$), respectively. The best solution found is finally obtained at $t = 68$, characterized by a makespan equal to 2099.

Likewise in the MSR_0 simulation, due to detected deviations from the rover's nominal performance, the solver is called to produce a number of new solutions in the interval between $t = 205$ and $t = 571$, all characterized by the same quality (i.e., exploiting the solution's flexibility). As the execution proceeds and new re-schedulings are necessary, such flexibility is eventually exhausted, and the makespan starts to increase again (see the climbing arrow in the figure) reaching the value 2134,

as it can be observed in the interval between $t = 571$ and $t = 1886$, the instant at which the execution terminates. Lastly, it should be noted that the larger makespan obtained in the MSR_1 simulation is mainly due to the fact that in the second case, the 3DROV's non-nominal behavior (i.e., the disrupting factor) has been artificially augmented through the action of the *eGen* Event Generator, ultimately resulting in a larger number of re-schedulings during plan execution.

Figures 7.5 and 7.6 provide a graphic representation of: (a) the connection between the rover speed evolution –both angular (top) and linear (middle)– and the re-scheduling triggers occurring as a consequence of a reduction on the nominal speed; and (b) the connection between the battery usage evolution as the state of charge (SoC) and the re-scheduling triggers resulting from a battery overconsumption (bottom). A further objective of our analysis was to test the precision of out predictive model with respect to 3DROV's nominal performances. To this aim, we analyzed in both figures the SoC telemetry data captured during the whole execution, and focused on the deviations between the real "battery usage" and the estimations provided by our predictive model.

In particular, Figure 7.5 how the rover performance deviations in terms of linear and angular speed during the MSR_0 execution are reflected in the power consumption. It can be observed how the SoC regularly undergoes a steeper consumption trend in correspondence with the occurrence of single as well as "bursts" of exogenous events (e.g., at $t = 3, 232, 465, 540, 761, 988, 1817$ for the angular speed case, and $t = 1865$ for the linear speed case). Outside those instants, the consumption rate seems to re-gain the ordinary steepness. Looking at Figure 7.5, it should be observed as a charging operation is eventually necessary immediately after the consumption peak occurred at $t = 988$; in that case, the rover battery is recharged so as to allow the feasible execution of all the remaining tasks, until plan completion. Lastly, it should be observed how both the real and the expected SoC profiles follow the same trend and are characterized by an acceptable deviation, mainly due to the conservative nature of the predictive model utilized in this study (see [Díaz *et al.*, 2013]).

Figure 7.6 refers to the MSR_1 simulation, where the severity of the rover's non-nominal performances have been augmented.

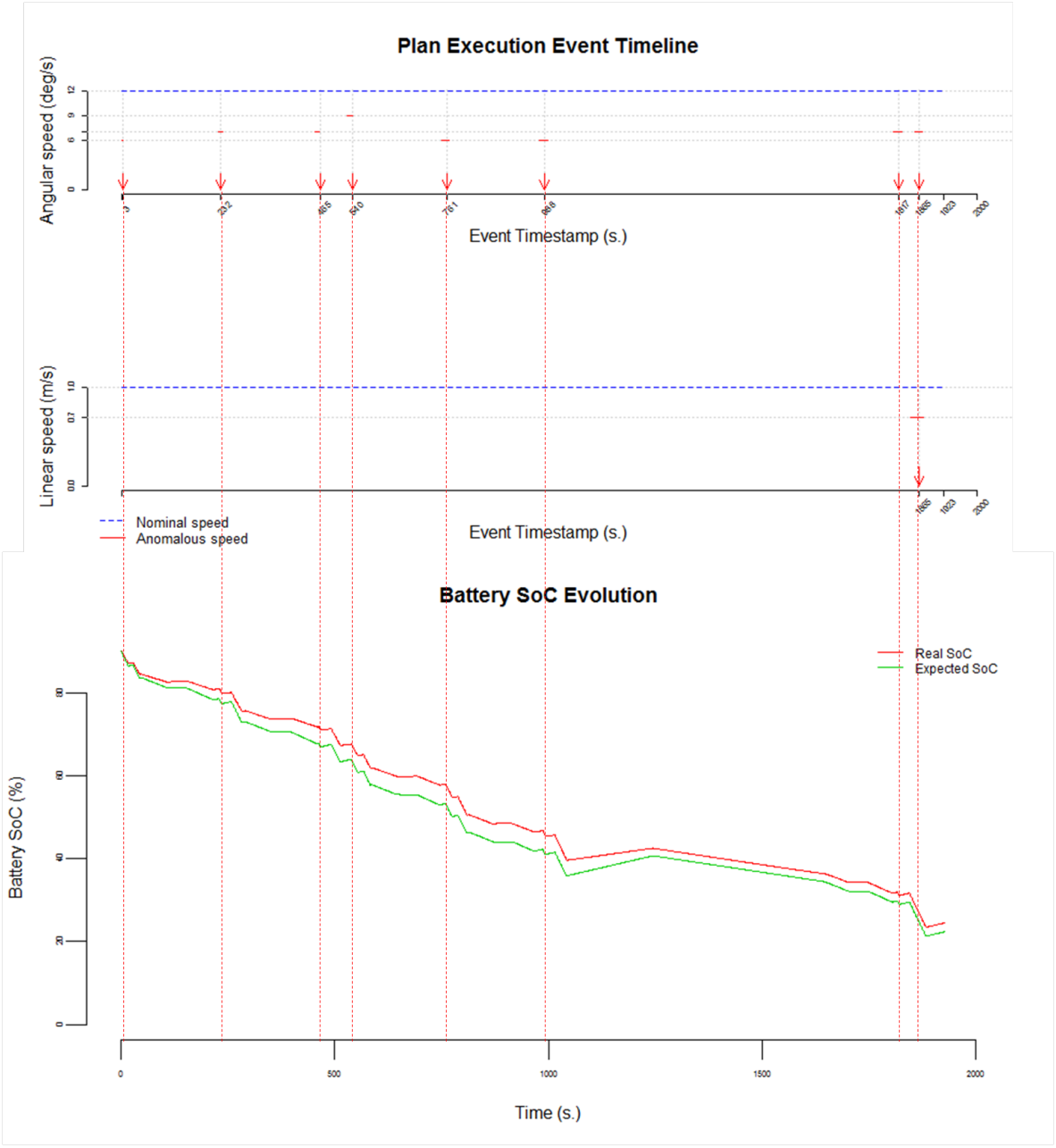

Figure 7.5: First simulation telemetry analysis. Relation between: (a) the rover speeds, angular (top) and linear (middle), and the re-scheduling triggers occurring as a consequence of a speed reduction; and (b) the battery state of charge (SoC) and the re-scheduling triggers resulting from a battery overconsumption (bottom)

From the qualitative standpoint, the information conveyed by the figure is similar to the previous case, e.g., the battery consumption rate significantly increases at the occurrences of exogenous events. The difference with respect to the previous case is mainly quantitative, as the execution is affected by a larger number of occurrences, relatively to both the linear and angular speed delays. As a consequence, the battery is depleted more rapidly and necessitates a re-charge at an earlier stage during plan execution (around $t = 550$). As another consequence, it can be observed how the predictive model suffers a slightly larger misalignment in terms of SoC projection.

7.4 Summary

In this part of the thesis we presented as main contribution, an integrated testbed platform built on top of an existing ESA asset, namely the 3DROV planetary rover system simulator. This benchmark platform is developed with the twofold aim of (i) validating the whole $CoRe^p$ autonomous control architecture; and (ii) performing an experimental analysis on the performance of the control architecture's capabilities, throughout two representative cases of study.

The experimentation here described is based on a first integration achievement between a constraint-based scheduler and an realistic and independent simulator.

One aspect under analysis, and which is of particular significance in the Martian domain, is related to the modelling of the terrain.

The actual model used does not take into account features like terrain slope and/or the orographic shape of the surroundings; the terrain is considered as a flat, completely traversable area.

Another interesting aspect under analysis is related to the relation between the solar flux rate and the battery power charge. It is known that the charging rate of the solar arrays onboard the rover is strictly dependent on the time of day, inclination of the solar flux w.r.t. to the array panels, current illumination conditions (e.g., shady Vs. sunlit terrain), etc.

Lastly, the scalability aspect of the proposed solution is being studied. In [Díaz *et al.*, 2013] we have demonstrated how the proposed $Esta^p + ISES$ solvers are

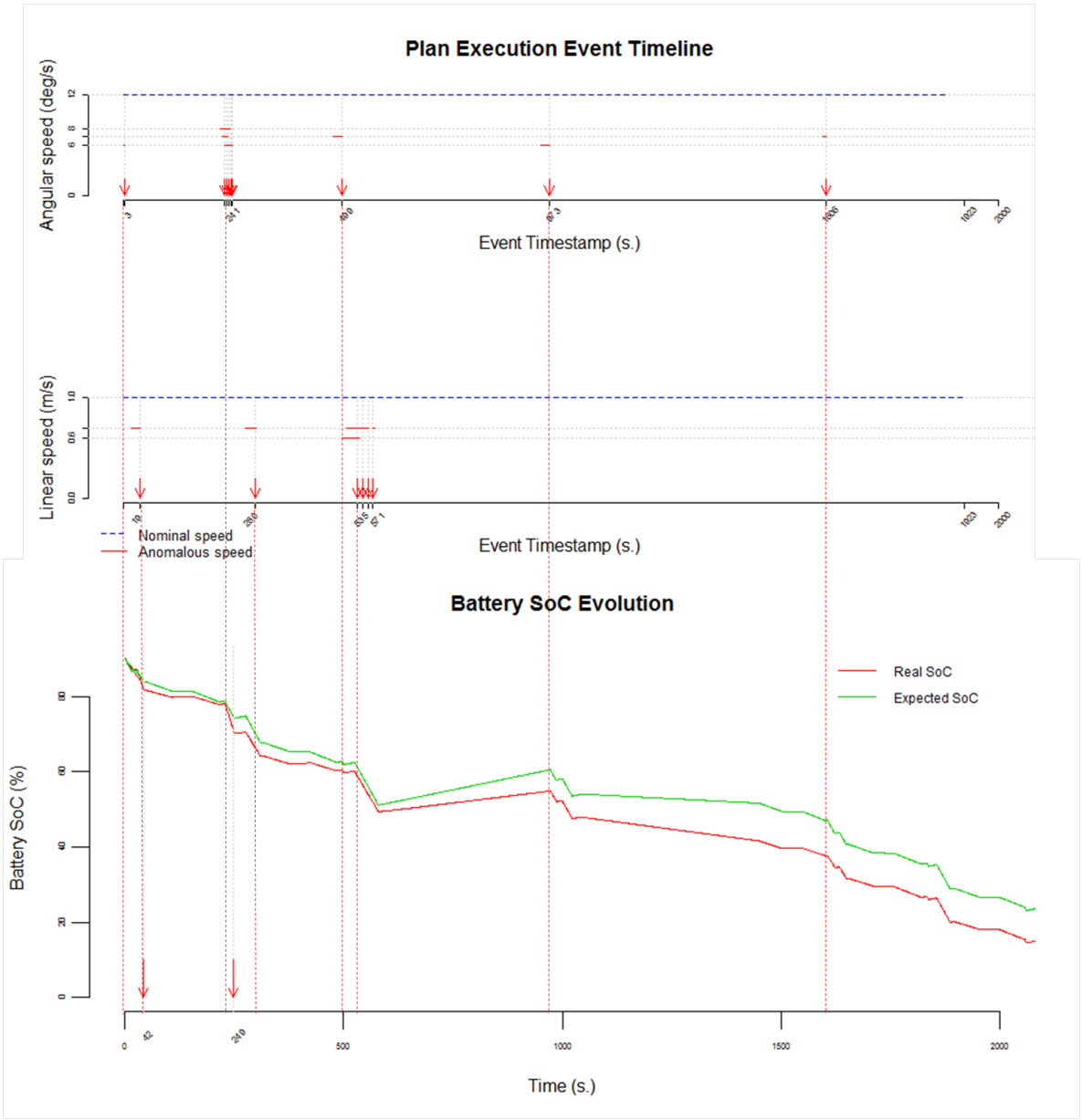

Figure 7.6: Second simulation telemetry analysis. Relation between: (a) the rover speeds, angular (top) and linear (middle), and the re-scheduling triggers occurring as a consequence of a speed reduction; and (b) the battery state of charge (SoC) and the re-scheduling triggers resulting from a battery overconsumption (bottom)

capable of producing plans composed of up to 30 experiments (corresponding to a period of uninterrupted and autonomous rover activity between 9 and 34 sols, given the involved traversal distances). Due to the stringent requirements imposed to the dynamic nature of the experimentation performed in the present work, the number of experiments composing the running plan has been limited to 5, but analyses are being planned to study the possibility to increase such number also for the dynamic case.

Part IV

Conclusions and future work

Chapter 8

Conclusions and Future Work

On this last chapter we provide a summary of the main contributions presented along this Ph.D. thesis, and outline some interesting research activities for the future as ongoing topics in the continuation of this work.

8.1 Conclusions

The work presented in this Ph.D. thesis was developed within the NPI ESA programme *Autonomy for Interplanetary Missions* (ESTEC-No. 2169/08/NI/PA), conceived as a response to the increasing interest of evolving current telerobotic capabilities towards a complete mixed-initiative strategy implementation that enables shifting control modes from purely manual to fully autonomous operations.

In this context, we studied and exploited existing AI scheduling and control execution techniques to propose a solution for a MSR, which combines both robust constraint-based robot action scheduling and flexible reactive schedule execution management in the face of high executional uncertainty. On the one hand, first of the two major challenges we faced was about providing high-level reasoning capabilities to enable the rover synthesizing complete command plans on the management of complex temporal and resource constraints. On the other hand, second major problem was about providing an efficient execution process of the command plans while preserving the safety of the mission.

As a result, we have presented as the core contribution of this work a power-

aware, model-based autonomous control architecture ($CoRe^p$) for managing the execution of robot actions in the context of a planetary mission exploration inspired in the MSR mission concept. More concretely, in this dissertation we have introduced the following results:

- A *scheduling problem domain inspired in the MSR mission concept*. We proposed a model of the world inspired in the MSR mission concept, a long-range planetary exploration scenario, and introduced a scheduling problem called PARC-MRS based on the baseline requirements of the MSR mission scenario. The proposed model encapsulates a wide range of interesting features which makes it particularly challenging, as it involves: (i) 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; (ii) resource management, by analyzing the *energy* production/consumption profiles of all the plan activities; (iii) 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. We proposed a study of the benchmarking problem tailored to the MSR domain, and produced a methodology to generate meaningful PARC-MRS problem instances.

- An *advanced constraint-based solving algorithm*. We presented a profile-based, power-aware reasoning algorithm ($ESTA^p$) for providing feasible solution plans to the so-called scheduling problem PARC-MRS, as well as a meta-heuristic for solution optimization. $ESTA^p$ provides the controller with advanced reasoning capabilities like the synthesis of complete command plans involving 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. More concretely, the reasoner we propose provides an extension of a well-known constraint-based, resource-driven procedure which exploits power-aware reasoning capa-

bilities 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, it is worth to mention the successful exploitation of a well known methodology to represent renewable resources by means of a classical cumulative scheme, as a core concept to model and solve the PARC-MRS problem.

- A *flexible model-based autonomous control architecture* for planetary rover mission operations. The proposed controller $CoRe^p$ implements a single (model-based) SPA closed-loop execution scheme to safely command the robot activities considered in the context of the MSR key mission scenario, through a seamless integration of advanced decision-making capabilities (provided by $ESTA^p$) within a flexible execution process targeted at generating and safely executing mission plans. $CoRe^p$ architecture also supports (on-line) continuous plan optimization concurrently with the plan execution process, as well as dynamic integration of new rover activities within the running plan *on-the-fly*.

- An *integrated testbed platform* built on top of an outstanding planetary rover simulation platform (3DROV) with the aim of validating the core capabilities of the $CoRe^p$ control architecture. Concretely, we designed an experimental model targeted at both: (a) validating the synthesized solution schedules with respect to practical cases of study; and (b) demonstrating the *degree of adaptability* of the autonomous controller when inducing executional uncertainty on the basis of the robustness and flexibility of both the solution schedules and execution management process respectively.

8.2 Future Work

The content of this Ph.D. thesis can be seen as an ongoing work. We can mention the following future research paths that can be carried out as a continuation of the results here presented:

- Refinement of the problem model. The internal model encapsulated within the controller can be improved by including additional aspects of the problem do-

main. For instance, the terrain model can be refined to take into account characteristics such as slope, compactness, roughness, etc. that can be exploited by the scheduling engine to provide more accurate solution plans. The battery model can also be improved by considering more realistic, non-monotonic battery charging/consumption profiles.

- Improve the heuristic rationale used by the scheduling engine. The decision-making strategy used for the resource conflict detection and resolution might be improved by considering multiple criteria, by synthesizing and ranking the MCS on the basis of weighted, multi-objective optimization heuristics over a three-dimensional Euclidean space.

- Provide a better command extraction method. Solution pans synthesized by $ESTA^p$ actually represents a set of possible low-level command (rover action) sequences: given a feasible schedule, a possible command sequence is extracted in the shape of a *timeline* whose intervals (or tokens) represent a particular task to be executed to fulfil the plan's goals. The extraction method here used is static (i.e., it is always the same along the whole mission execution). We recall that every Drill and/or Release command is dispatched correspondingly to the start time of the related scheduling solution activity; all distance intervals between any Drill and/or any Release operation is interpreted as a Navigation activity and battery charging intervals. Battery charging intervals where the rover stays still are extracted as *idle periods* right after a navigation (or hibernation) activity *by default*. A different method might chose to extract them in the other way around, or maybe slicing them by interleaving navigation activities and idle periods on the basis of a more complex executional criteria.

- Timeline-based reasoning schema. Our solving algorithm $ESTA^p$ exploits a model defined at an *activity abstraction level*, i.e., represented in terms of partially-ordered sequence activities and a set of temporal, precedence and resource usage constraints. In a timeline-based representation, the problem domain is decomposed in terms of subsystems or entities, modelled as state variables which can evolve over time (resources are modelled as a special type of state variable). A timeline is a logic structure used for reasoning about the

evolution of a state variable over a period of time. Different state variables are related each other by means of synchronization constraints, and the problem solving strategy is reduced to instantiate feasible timelines (one for each entity) given a initial and final state, by satisfying all the synchronization constraints. The timeline-based paradigm allows modelling in a more natural way planning problems, and represents an ideal framework for the integration of reasoning and control execution capabilities.

- Consider a *multi-reactor* control architecture design. A multi-reactor-based consists of a distributed and interacting collection of SPA loops, which are encapsulated on the shape of reactors. A reactor is an autonomous subsystem capable of reasoning about its internal state by considering (and impacting over) the internal state of other reactors (by using a timeline-based representation of the problem domain). Each reactor internally manages a set of timelines that define the evolution of its internal state. A reactor can observe and influence over other reactors through a well-defined communication scheme (synchronization process), that guarantees a consistent and unique shared view of the overall system state. Multi-reactor based control execution systems enables *scalability* in terms of P&S in a dynamic perspective, as the scope of deliberation and execution is interleaved and partitioned along different agents (reactors) both functionally and temporally. An outstanding example that implements this kind of architecture is T-REX [McGann *et al.*, 2007].

- Consider different space exploration scenarios. The solution here presented might be generalized to be used in other application domains where groups of autonomous rovers are in charge of collaborating on the fulfilment of the specific mission goals. This kind of problem domains that involve the synchronization of *multiple agents*, are naturally addressed through the application of Computational Intelligence (CI) techniques based on *decentralized approaches*, in contrast to centralized solutions like the one presented in this work. There are many outstanding examples in literature where distributed methods have been successfully applied to solve real-world problems. For instance, in [Li *et al.*, 2012], a multi-agent system is proposed to tackle the

service restoration problem in power distribution networks; while in [Abielmona *et al.*, 2011], a novel agent-based architecture is proposed for territorial or perimeter-security applications.

Bibliography

[Abdeddaïm *et al.*, 2007] Abdeddaïm, Y.; Asarin, E.; Gallien, M.; Ingrand, F.; Lesire, C.; and Sighireanu, M. 2007. Planning Robust Temporal Plans: A Comparison Between CBTP and TGA Approaches. In *International Conference on Automated Planning and Scheduling (ICAPS), Providence, Rhode Island, USA*, 2–9. AAAI.

[Abielmona *et al.*, 2011] Abielmona, R. S.; Petriu, E. M.; Harb, M.; and Wesolkowski, S. 2011. Mission-driven robotic intelligent sensor agents for territorial security. *IEEE Comp. Int. Mag.* 6(1):55–67.

[Albus *et al.*, 1987] Albus, J.; McCain, H.; Lumia, R.; and of Standards, U. S. N. B. 1987. *NASA/NBS Standard Reference Model for Telerobot Control System Architecture (NASREM)*. NBS technical note. U.S. Department of Commerce, National Bureau of Standards.

[Allen, 1983] Allen, J. F. 1983. Maintaining knowledge about temporal intervals. *Commun. ACM* 26(11):832–843.

[Anderson, Smith, & Weld, 1998] Anderson, C. R.; Smith, D. E.; and Weld, D. S. 1998. Conditional effects in graphplan. In *Artificial Intelligence Planning System (AIPS), Pittsburgh Pennsylvania, USA*, 44–53. AAAI.

[Aschwanden *et al.*, 2006] Aschwanden, P.; Baskaran, V.; Bernardini, S.; Fry, C.; Moreno, M.; Muscettola, N.; Plaunt, C.; Rijsman, D.; and Tompkins, P. 2006. Model-unified planning and execution for distributed autonomous system control. In *Fall Symposium on Spacecraft Autonomy, Association for the Advancement of Artificial Intelligence (AAAI), Boston, Massachusetts, USA*.

[Bacchus & Kabanza, 2000] Bacchus, F., and Kabanza, F. 2000. Using temporal logics to express search control knowledge for planning. *Artificial Intelligence* 116, Issues 12:123191.

[Baglioni *et al.*, 2006] Baglioni, P.; Fisackerly, R.; Gardini, B.; Giafiglio, G.; Pradier, A.; Santovincenzo, A.; Vago, J.; and Van Winnendael, M. 2006. 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.*

[Baskaran *et al.*, 2006] Baskaran, V.; Muscettola, N.; Rijsman, D.; Plaunt, C.; and Fry, C. 2006. Intelligent Rover Execution for Detecting Life in the Atacama Desert. In *Fall Symposium on Spacecraft Autonomy, Association for the Advancement of Artificial Intelligence (AAAI), Washington, DC*.

[Beck & Wilson, 2007] Beck, J. C., and Wilson, N. 2007. Proactive algorithms for job shop scheduling with probabilistic durations. *Artificial Intelligence Research* 28:183–232.

[Beetz, 2001] Beetz, M. 2001. Runtime plan adaptation in structured reactive controllers. In Andre, E., and Sen, S., eds., *International Conference on Autonomous Agents, Barcelona, Spain*.

[Blum & Furst, 1995] Blum, A. L., and Furst, M. L. 1995. Fast planning through planning graph analysis. *Artificial Intelligence* 90(1):1636–1642.

[Bonet & Geffner, 1999] Bonet, B., and Geffner, H. 1999. Planning as heuristic search: New results. In *Recent Advances in AI Planning, European Conference on Planning (ECP), Durham, UK*, 360–372.

[Bonet & Geffner, 2000] Bonet, B., and Geffner, H. 2000. Planning with incomplete information as heuristic search in belief space. In *International Conference on Artificial Intelligence Planning Systems (ICAPS), Breckenridge, CO, USA*, 52–61. AAAI.

[Bonet & Geffner, 2001] Bonet, B., and Geffner, H. 2001. Planning and control in artificial intelligence: A unifying perspective. *Applied Intelligence* 14(3):237–252.

[Bouguerra, Karlsson, & Saffiotti, 2007] Bouguerra, A.; Karlsson, L.; and Saffiotti, R. 2007. Active execution monitoring using planning and semantic knowledge. In *Workshop on Planning and Plan Execution for Real-World Systems (ICAPS), Providence, Rhode Island, USA*.

[Bresina *et al.*, 2005] Bresina, J. L.; Jonsson, A. K.; Morris, P. H.; and Rajan, K. 2005. Activity Planning for the Mars Exploration Rovers. In Biundo, S.; Myers, K. L.; and Rajan, K., eds., *International Conference on Automated Planning & Scheduling (ICAPS), Monterey, California, USA*, 40–49. AAAI.

[Brooks, 1986] Brooks, R. A. 1986. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation* 2(10).

[Brooks, 1990] Brooks, R. A. 1990. The Behavior Language; User's Guide. Technical Report AIM-1227, MIT Artificial Intelligence Lab.

[Ceballos *et al.*, 2011] Ceballos, A.; Bensalem, S.; Cesta, A.; De Silva, L.; Fratini, S.; Ingrand, F.; Ocon, J.; Orlandini, A.; Py, F.; Rajan, K.; Rasconi, R.; and Van Winnendael, M. 2011. A Goal-Oriented Autonomous Controller for Space Exploration. In *Advanced Space Technologies in Robotics and Automation (ASTRA), Noordwijk, The Netherlands*, volume 11. ESA.

[Cesta & Fratini, 2008] Cesta, A., and Fratini, S. 2008. The timeline representation framework as a planning and scheduling software development environment. In *Workshop of the UK Planning and Scheduling Special Interest Group, Edinburgh, United Kingdom*.

[Cesta *et al.*, 2007] 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 *et al.*, 2009] Cesta, A.; Cortellessa, G.; Fratini, S.; and Oddi, A. 2009. Developing an End-to-End Planning Application from a Timeline Representation Framework. In *Conference on Innovative Applications of Artificial Intelligence (IAAI), Pasadena, California, USA*.

[Cesta, Oddi, & Smith, 2002] 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 & Smith, 1994] Cheng, C., and Smith, S. 1994. Generating Feasible Schedules under Complex Metric Constraints. In *National Conference on AI (AAAI), Seattle, Washington, USA*.

[Chien *et al.*, 2004] Chien, S.; Sherwood, R.; Tran, D.; Cichy, B.; Rabideau, G.; no, R. C.; Davies, A.; Lee, R.; M, D.; Frye, S.; Trout, B.; Hengemihle, J.; Shulman, S.; Ungar, S.; and Brakke, T. 2004. The EO-1 Autonomous Science Agent. In *Autonomous Agents & Multi Agent Systems (AAMAS), New York City, NY, USA*.

[Chien *et al.*, 2005] Chien, S.; Sherwood, R.; Tran, D.; Cichy, B.; Rabideau, G.; Castaño, R.; Davies, A.; Mandl, D.; Frye, S.; Trout, B.; D'Agostino, J.; Shulman, S.; Boyer, D.; Hayden, S.; Sweet, A.; and Christa, S. 2005. Lessons learned from autonomous sciencecraft experiment. In *Autonomous Agents & Multi Agent Systems (AAMAS), Utrecht, Netherlands*, 11–18. ACM.

[Cimatti *et al.*, 2003] Cimatti, A.; Pistore, M.; Roveri, M.; and Traverso, P. 2003. Weak, strong, and strong cyclic planning via symbolic model checking. *Artificial Intelligence* 147(1-2):35–84.

[Crandall & Goodrich, 2001] Crandall, J., and Goodrich, M. 2001. Experiments in adjustable autonomy. In *IEEE Systems, Man, and Cybernetics*, volume 3, 1624–1629 vol.3.

[Culberson & Schaeffer, 1996] Culberson, J. C., and Schaeffer, J. 1996. Searching with pattern databases. In *Advances in Artificial Intelligence (Lecture Notes in Artificial Intelligence), Dresden, Germany*, 402–416. Springer-Verlag.

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

[Díaz *et al.*, 2011a] Díaz, D.; R-Moreno, M.; Cesta, A.; Oddi, A.; and Rasconi, R. 2011a. Scheduling a Single Robot in a Job-Shop Environment through Precedence Constraint Posting. In *International Conference on Industrial, Engineering & Other Applications of Applied Intelligent Systems (IEA/AIE), Syracuse, NY, USA*.

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

[Díaz *et al.*, 2013] Díaz, D.; Moreno, M.-D.; Cesta, A.; Oddi, A.; and Riccardo, R. 2013. Efficient Energy Management for Autonomous Control in Rover Missions. *IEEE Computational Intelligence Magazine, Special Issue on Computational Intelligence for Space Systems and Operations* 8:12–24.

[Drger, Finkbeiner, & Podelski, 2006] Drger, K.; Finkbeiner, B.; and Podelski, A. 2006. Directed model checking with distance-preserving abstractions. In Valmari, A., ed., *SPIN*, volume 3925 of *Lecture Notes in Computer Science*, 19–34. Springer.

[D.Wettergreen *et al.*, 2002] D.Wettergreen; Dias, B.; Shamah, B.; Teza, J.; Tompkins, P.; Urmson, C.; Wagner, M.; and Whittaker, W. 2002. Experiments in sun-synchronous navigation. In *IEEE International Conference on Robotics and Automation (ICRA), Washington D.C., USA*.

[E.-Martin, R-Moreno, & Castaño, 2010] E.-Martin, Y.; R-Moreno, M.; and Castaño, B. 2010. PIPSS*: A System based on Temporal Estimates. In *International Conference on Artificial Intelligence (SGAI), Cambrige, UK*, 123–136. Springer.

[E.-Martín, Rodríguez-Moreno, & Smith, 2014] E.-Martín, Y.; Rodríguez-Moreno, M. D.; and Smith, D. E. 2014. Progressive heuristic search for probabilistic planning based on interaction estimates. *Expert Systems* 31(5):421–436.

[Estlin *et al.*, 2005] Estlin, T.; Gaines, D.; Chouinard, C.; Fisher, F.; Castano, R.; Judd, M.; and Nesnas, I. 2005. Enabling autonomous rover science through dynamic planning and scheduling. In *IEEE Aerospace Conference, Big Sky, MT, USA*.

[Estlin *et al.*, 2007] 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, Roma, Italy*, 4911–4918.

[Fikes & Nilsson, 1971] Fikes, R., and Nilsson, N. J. 1971. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence* 2(3/4):189–208.

[Firby, 1987] Firby, R. J. 1987. An investigation into reactive planning in complex domains. In *Association for the Advancement of Artificial Intelligence (AAAI), Seattle, Washington, USA*, 202–206. Morgan Kaufmann.

[Fong *et al.*, 2008] Fong, T.; Bualat, M.; Deans, M.; Allan, M.; Bouyssounouse, X.; Broxton, M.; Edwards, L.; Elphic, R.; Flückiger, L.; Frank, J.; Keely, L.; Kobayashi, L.; Lee, P.; Lee, S. Y.; Lees, D.; Pacis, E.; Park, E.; Pedersen, L.; Schreckenghost, D.; Smith, T.; To, V.; and Utz, H. 2008. Field testing of utility robots for lunar surface operations. In *American Institute of Aeronautics and Astronautics (AIAA), San Diego, California, USA*.

[Fox & Long, 2001] Fox, M., and Long, D. 2001. Stan4: A hybrid planning strategy based on subproblem abstraction. *AI Magazine* 22(3):81–84.

[Fox *et al.*, 2006] Fox, M.; Gerevini, A.; Long, D.; and Serina, I. 2006. Plan stability: Replanning versus plan repair. In *International Conference on Automated Planning and Scheduling (ICAPS), Cumbria, United Kingdom*, 212–221. AAAI Press.

[Frank & Dearden, 2003] Frank, J., and Dearden, R. 2003. Scheduling in the face of uncertain resource consumption and utility. In Rossi, F., ed., *Principles and Practice of Constraint Programming CP 2003*, volume 2833 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg. 832–836.

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

[Frank & Morris, 2007] Frank, J., and Morris, P. H. 2007. Bounding the resource availability of activities with linear resource impact. In *International Conference on AI Planning and Scheduling (ICAPS), Providence, Rhode Island, USA*.

[Gat, 1991] Gat, E. 1991. Integrating reaction and planning in a heterogeneous asynchronous architecture for mobile robot navigation. *SIGART Bull.* 2(4):70–74.

[Gat, 1998] Gat, E. 1998. Three-layer architectures. In Kortenkamp, D.; Bonasso, R. P.; and Murphy, R., eds., *Artificial Intelligence and Mobile Robots: Case Studies of Successful Robot Systems*. Cambridge, MA: MIT Press. 195–210.

[Gerevini & Long, 2006] Gerevini, A., and Long, D. 2006. Plan constraints and preferences in PDDL3. In *Workshop on Soft Constraints and Preferences in Planning at the International Conference on Automated Planning and Scheduling (ICAPS), Cumbria, UK*.

[Ghallab & Laruelle, 1994] Ghallab, M., and Laruelle, H. 1994. Representation and control in ixtet, a temporal planner. In *Second International Conference on Artificial Intelligence Planning Systems (AIPS), Chicago, Illinois, USA*, 61–67. AAAI.

[Halsey, Long, & Fox, 2004] Halsey, K.; Long, D.; and Fox, M. 2004. Crikey - a temporal planner looking at the integration of scheduling and planning. In *Workshop on Integration Scheduling Into Planning at the International Conference on Automated Planning and Scheduling (ICAPS), Whistler, British Columbia, Canada*, 46–52.

[Helmert, 2006] Helmert, M. 2006. The fast downward planning system. *Journal of Artificial Intelligence Research* 26(1):191–246.

[Hoey *et al.*, 1999] Hoey, J.; St-aubin, R.; Hu, A.; and Boutilier, C. 1999. SPUDD: Stochastic planning using decision diagrams. In *Conference on Uncertainty in Artificial Intelligence, Stockholm, Sweden*, 279–288. Morgan Kaufmann.

[Hoffmann, Porteous, & Sebastia, 2004] Hoffmann, J.; Porteous, J.; and Sebastia, L. 2004. Ordered landmarks in planning. *Journal of Artificial Intelligence Research* 22(1):215–278.

[Hoffmann, 2001] Hoffmann, J. 2001. FF: The Fast-Forward Planning System. *AI magazine* 22:57–62.

[Hoffmann, 2003] Hoffmann, J. 2003. The Metric-FF Planning System: Translating "Ignoring Delete Lists" to Numeric State Variables. *Journal of Artificial Intelligence Research (JAIR)* 20:291–341.

[Hsu & Liu, 2007] Hsu, H.-H., and Liu, A. 2007. A flexible architecture for navigation control of a mobile robot. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on* 37(3):310–318.

[Inagaki & Itoh, 1996] Inagaki, T., and Itoh, M. 1996. Trust, autonomy, and authority in human-machine systems: Situation-adaptive coordination for systems safety. In *Cognitive Systems Engineering in Process Control International Symposium (CSEPC), Kyoto, Japan*, 176–183.

[Ingrand *et al.*, 1996] Ingrand, F. F.; Chatila, R.; Alami, R.; and Robert, F. 1996. PRS: a high level supervision and control language for autonomous mobile robots. In *IEEE International Conference on Robotics and Automation, Minneapolis, Minnesota, USA, April 22-28, 1996*, 43–49.

[Ingrand *et al.*, 2007] Ingrand, F.; Lacroix, S.; Lemai-Chenevier, S.; and Py, F. 2007. Decisional autonomy of planetary rovers. *Journal of Field Robotics* 24(7):559–580.

[Jensen & Veloso, 2000] Jensen, R. M., and Veloso, M. M. 2000. OBDD-based universal planning for synchronized agents in non-deterministic domains. *Journal of Artificial Intelligence Research* 13:189–226.

[Jónsson *et al.*, 2000] Jónsson, A. K.; Morris, P. H.; Muscettola, N.; Rajan, K.; and Smith, B. D. 2000. Planning in interplanetary space: Theory and practice. In *Artificial Intelligence Planning and Scheduling Systems (AIPS), Toronto, Canada*, 177–186. AAAI.

[Kambhampati & Hendler, 1992] Kambhampati, S., and Hendler, J. A. 1992. A Validation-structure-based Theory of Plan Modification and Reuse. *Artificial Intelligence* 55(2-3):193–258.

[Katz & Domshlak, 2008] Katz, M., and Domshlak, C. 2008. Structural patterns heuristics via fork decomposition. In *International Conference on Automated Planning and Scheduling (ICAPS), Sydney, Australia*, 182–189. AAAI.

[Kim & Chung, 2006] Kim, G., and Chung, W. 2006. Tripodal schematic control architecture for integration of multi-functional indoor service robots. *IEEE Transactions on Industrial Electronics* 53(5):1723–1736.

[Kim, Williams, & Abramson, 2001] Kim, P.; Williams, B. C.; and Abramson, M. 2001. Executing reactive, model-based programs through graph-based temporal planning. In Nebel, B., ed., *Joint Conference on Artificial Intelligence (IJCAI), Seattle, Washington, USA*, 487–493. Morgan Kaufmann.

[Koehler *et al.*, 1997] Koehler, J.; Nebel, B.; Hoffmann, J.; and Dimopoulos, Y. 1997. Extending planning graphs to an adl subset. In Steel, S., and Alami, R., eds., *ECP*, volume 1348 of *Lecture Notes in Computer Science*, 273–285. Springer.

[Kurniawati, Hsu, & Lee, 2008] Kurniawati, H.; Hsu, D.; and Lee, W. S. 2008. Sarsop: Efficient point-based pomdp planning by approximating optimally reachable belief spaces. In *Robotics: Science and Systems, Zurich, Switzerland.*

[Laborie, 2003] Laborie, P. 2003. Algorithms for propagating resource constraints in ai planning and scheduling: Existing approaches and new results. *Artificial Intelligence* 143(2):151–188.

[Li *et al.*, 2012] Li, H.; Sun, H.; Wen, J.; Cheng, S.; and He, H. 2012. A fully decentralized multi-agent system for intelligent restoration of power distribution network incorporating distributed generations [application notes]. *IEEE Computational Intelligence Magazine* 7(4):66–76.

[Maes, 1991] Maes, P. 1991. The Agent Network Architecture (ANA). *SIGART Bull.* 2(4):115–120.

[Maimone, Leger, & Biesiadecki, 2007] Maimone, M. W.; Leger, P. C.; and Biesiadecki, J. J. 2007. Overview of the Mars Exploration Rovers Autonomous Mobility and Vision Capabilities. In *IEEE International Conference on Robotics and Automation (ICRA), Roma, Italy*.

[Majercik & Littman, 1998] Majercik, S. M., and Littman, M. L. 1998. MAXPLAN: A new approach to probabilistic planning. In *International Conference on Artificial Intelligence Planning Systems (AIPS), Pittsburgh, Pennsylvania, USA*, 86–93. AAAI.

[Mariela, 2008] Mariela, B. 2008. *A multi-agent architecture with distributed coordination for an autonomous robot*. Ph.D. Dissertation, Universitat de Girona.

[Mausam & Weld, 2008] Mausam, and Weld, D. S. 2008. Planning with durative actions in stochastic domains. *Journal of Artificial Intelligence Research* 31(1):33–82.

[McGann *et al.*, 2007] McGann, C.; Py, F.; Rajan, K.; Thomas, H.; Henthorn, R.; and McEwen, R. 2007. T-rex: A model-based architecture for auv control. *Workshop on Planning and Plan Execution for Real-World Systems, Providence, Rhode Island, USA*.

[McGann *et al.*, 2008a] McGann, C.; Py, F.; Rajan, K.; Ryan, J. P.; and Henthorn, R. 2008a. Adaptive control for autonomous underwater vehicles. In *AAAI Conference on Artificial Intelligence, Chicago, Illinois, USA*, 1319–1324. AAAI Press.

[McGann *et al.*, 2008b] McGann, C.; Py, F.; Rajan, K.; Ryan, J. P.; Thomas, H.; Henthorn, R.; and McEwen, R. S. 2008b. Preliminary Results for Model-Based Adaptive Control of an Autonomous Underwater Vehicle. In *Experimental Robotics, The International Symposium (ISER), Athens, Greece*, volume 54 of *Springer Tracts in Advanced Robotics*, 395–405. Springer.

[McGann *et al.*, 2009] McGann, C.; Berger, E.; Bohren, J.; Chitta, S.; Gerkey, B. P.; Glaser, S.; Marthi, B.; Meeussen, W.; Pratkanis, T.; Marder-Eppstein, E.; and Wise, M. 2009. Model-based, hierarchical control of a mobile manipulation platform. In *Workshop on Planning and Plan Execution for Real-World Systems, in International Conference on Automated Planning and Scheduling (ICAPS), Thessaloniki, Greece*.

[Mishkin *et al.*, 1998a] Mishkin, A.; Morrison, J.; Nguyen, T.; Stone, H.; and Cooper, B. 1998a. Operations and Autonomy of the Mars Pathfinder Microrover. In *IEEE Aerospace Conference, Snowmass at Aspen, CO, USA*.

[Mishkin *et al.*, 1998b] Mishkin, A.; Morrison, J.; Nguyen, T.; Stone, H.; Cooper, B.; and Wilcox, B. 1998b. Experiences with operations and autonomy of the Mars Pathfinder microrover. In *IEEE Aerospace Conference, Snowmass at Aspen, CO, USA*.

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

[Murphy, 2000] Murphy, R. R. 2000. *Introduction to AI Robotics*. MIT Press.

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

[Muscettola, 1993] Muscettola, N. 1993. HSTS: Integrating Planning and Scheduling. Technical Report CMU-RI-TR-93-05, Robotics Institute, Pittsburgh, PA.

[Muscettola, 2004] Muscettola, N. 2004. Incremental maximum flows for fast envelope computation. In *Automated Planning and Scheduling (ICAPS), Whistler, British Columbia, Canada*, 260–269.

[N. Muscettola & Plaunt, 2002] N. Muscettola, G. Dorais, C. F. R. L., and Plaunt, C. 2002. IDEA: Planning at the Core of Autonomous Reactive Agents. In *Workshops at the Artificial Intelligence Planning and Scheduling (AIPS) Conference*.

[Nau, Ghallab, & Traverso, 2004] Nau, D.; Ghallab, M.; and Traverso, P. 2004. *Automated Planning: Theory & Practice*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

[Nayak *et al.*, 1999] Nayak, P. P.; Kurien, J.; Dorais, G.; Millar, W.; Rajan, K.; Kanefsky, R.; Bernard, E. D.; Jr., B. E. G.; Rouquette, N.; Smith, D. B.; Tung, Y.-W.; Muscoletta, N.; and Taylor, W. 1999. Validating the DS1 Remote Agent Experiment. In *International Conference on Artificial Intelligence, Beijing, China*, volume 440.

[Nesnas *et al.*, 2003] Nesnas, I. A. D.; Wright, A.; Bajracharya, M.; Simmons, R.; Estlin, T.; and Kim, W. S. 2003. Claraty: An architecture for reusable robotic software. In *SPIE Aerosense Conference, Orlando, Florida, United States*.

[Nicola, Morris, & Muscettola, 2001] Nicola, P. M.; Morris, P.; and Muscettola, N. 2001. Dynamic control of plans with temporal uncertainty. In *International Joint Conference on Artificial Intelligence (IJCAI), Seattle, Washington, USA*, 494–502.

[Nigenda & Kambhampati, 2003] Nigenda, R. S., and Kambhampati, S. 2003. Altaltp: Online parallelization of plans with heuristic state search. *Journal of Artificial Intelligence Research (JAIR)* 19:631–657.

[Nilsson, 1984] Nilsson, N. J. 1984. Shakey the robot. Technical Report 323, AI Center, SRI International, 333 Ravenswood Ave., Menlo Park, CA 94025.

[Nilsson, 1994] Nilsson, N. J. 1994. Teleo-reactive programs for agent control. *Journal of Artificial Intelligence Research* 158.

[Nourbakhsh & Genesereth, 1996] Nourbakhsh, I., and Genesereth, M. 1996. Assumptive planning and execution: A simple, working robot architecture. *Autonomous Robots* 3(1):49–67.

[Nuijten & Pape, 1998] Nuijten, W., and Pape, C. L. 1998. Constraint-Based Job Shop Scheduling with Ilog-Scheduler. *J. Heuristics* 3(4):271–286.

[Oddi & Smith, 1997] Oddi, A., and Smith, S. 1997. Stochastic Procedures for Generating Feasible Schedules. In *National Conference on AI (AAAI), Providence, Rhode Island, USA*, 308–314.

[Oddi *et al.*, 2009] Oddi, A.; Rasconi, R.; Cesta, A.; and Smith, S. 2009. Iterative-Sampling Search for Job Shop Scheduling with Setup Times. In *Constraint Satisfaction Techniques (COPLAS), Thessaloniki, Greece*.

[Oddi *et al.*, 2011] 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.

[Papadimitriou & Steiglitz, 1998] Papadimitriou, C. H., and Steiglitz, K. 1998. *Combinatorial Optimization: Algorithms and Complexity*. Dover Publications.

[Peot & Smith, 1992] Peot, M. A., and Smith, D. E. 1992. Conditional nonlinear planning. In Hendler, J., ed., *International Conference on Artificial Intelligence Planning Systems (AIPS), College Park, Maryland, USA*. San Mateo, CA: Kaufmann. 189–197.

[Policella *et al.*, 2004] Policella, N.; Smith, S. F.; Cesta, A.; and Oddi, A. 2004. Generating robust schedules through temporal flexibility. In Zilberstein, S.; Koehler, J.; and Koenig, S., eds., *International Conference on Automated Planning and Scheduling (ICAPS), Whistler, British Columbia, Canada*, 209–218. AAAI.

[Policella *et al.*, 2007] Policella, N.; Cesta, A.; Oddi, A.; and Smith, S. F. 2007. From precedence constraint posting to partial order schedules: A CSP approach to robust scheduling. *AI Community* 20(3):163–180.

[Poulakis *et al.*, 2008] Poulakis, P.; Joudrier, L.; Wailliez, S.; and Kapellos, K. 2008. 3DROV: A Planetary Rover System Design, Simulation and Verification Tool. In *International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS), Hollywood, USA*.

[Puterman, 1994] Puterman, M. L. 1994. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. New York, NY, USA: John Wiley & Sons, Inc., 1st edition.

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

[Refanidis & Vlahavas, 1999] Refanidis, I., and Vlahavas, I. P. 1999. Grt: A domain independent heuristic for strips worlds based on greedy regression tables. In *European Conferences on Planning (ECP), Durham, United Kingdom*, volume 1809 of *Lecture Notes in Computer Science*, 347–359. Springer.

[Richards, Kuwata, & How, 2003] Richards, A.; Kuwata, Y.; and How, J. P. 2003. Experimental demonstrations of real-time MILP control. In *Guidance, Navigation, and Control Conference (GNC), Austin, Texas, USA*. AIAA Paper 2003-5635.

[Richter & Westphal, 2010] Richter, S., and Westphal, M. 2010. The LAMA Planner: Guiding Cost-based Anytime Planning with Landmarks. *Journal of Artificial Intelligence Research* 39(1):127–177.

[Rosenblatt, 1995] Rosenblatt, J. 1995. Damn: A distributed architecture for mobile navigation - thesis summary. In *Journal of Experimental and Theoretical Artificial Intelligence*, 339–360. AAAI Press.

[Rossi, Venable, & Yorke-Smith, 2006] Rossi, F.; Venable, K. B.; and Yorke-Smith, N. 2006. Uncertainty in Soft Temporal Constraint Problems: A General Framework and Controllability Algorithms for the Fuzzy Case. *Journal of Artificial Intelligence Research* 27:617–674.

[Russell & Norvig, 2003] Russell, S. J., and Norvig, P. 2003. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2 edition.

[Sadeh, 1991] Sadeh, N. M. 1991. *Look-ahead Techniques for Micro-opportunistic Job Shop Scheduling*. Ph.D. Dissertation, Pittsburgh, PA, USA.

[Sherwood *et al.*, 2001] Sherwood, R.; Mishkin, A.; Chien, S.; Estlin, T.; Backes, P.; Cooper, B.; Rabideau, G.; and Engelhardt, B. 2001. An Integrated Planning and Scheduling Prototype for Automated Mars Rover Command Generation. In *European Conference on Planning (ECP), Toledo, Spain*, 441–444.

[Simmons & G., 1988] Simmons, R., and G., R. 1988. A Theory of Debugging Plans and Interpretations. In *Association for the Advancement of Artificial Intelligence (AAAI), Madison, Wisconsin, USA*, 94–99. The MIT Press.

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

[Smith & Cheng, 1993] Smith, S., and Cheng, C. 1993. Slack-Based Heuristics for Constraint Satisfaction Scheduling. In *National Conference on AI (AAAI), Washington, D.C., USA*.

[Smith & Weld, 1998] Smith, D. E., and Weld, D. S. 1998. Conformant Graphplan. In *Association for the Advancement of Artificial Intelligence and Innovative Applications of Artificial Intelligence Conferences (AAAI/IAAI)*, 889–896. AAAI Press / The MIT Press.

[Smith *et al.*, 2007] Smith, T.; Thompson, D. R.; Wettergreen, D. S.; Cabrol, N. A.; Warren-Rhodes, K. A.; and Weinstein, S. J. 2007. Life in the atacama: Science autonomy for improving data quality. *Journal of Geophysical Research: Biogeosciences* 112(G4).

[Smith, Thompson, & Wettergreen, 2007] Smith, T.; Thompson, D. R.; and Wettergreen, D. S. 2007. Generating exponentially smaller POMDP models using conditionally irrelevant variable abstraction. In *International Conference on Applied Planning and Scheduling (ICAPS), Providence, Rhode Island, USA*.

[Srivastava, Kambhampati, & Do, 2001] Srivastava, B.; Kambhampati, S.; and Do, M. B. 2001. Planning the project management way: Efficient planning by effective integration of causal and resource reasoning in realplan. *Artificial Intelligence* 131(1-2):73–134.

[Steel *et al.*, 2012] Steel, R.; Hoffmann, A.; Cimatti, A.; Roveri, M.; Kapellos, K.; Donati, A.; Policella, N.; and Niezette, M. 2012. Innovative Rover Operations Concepts – Autonomous Planner (IRONCAP) – Supporting Rover Operations Planning on Ground. In *SpaceOps, Stockholm, Sweden*.

[Tompkins, Stentz, & Whittaker, 2004] Tompkins, P.; Stentz, A.; and Whittaker, W. 2004. Mission-Level Path Planning for Rover Exploration. In *International Conference on Intelligent Autonomous Systems (IAS), Amsterdam, The Netherlands*.

[Treiman *et al.*, 2009] 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, National Research Council, Washington, D.C., USA*.

[van Winnendael, Baglioni, & Vago, 2005] van Winnendael, M.; Baglioni, P.; and Vago, J. 2005. Development of the ESA ExoMars Rover. In *International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS), Munich, Germany*.

[Wander & Frstner, 2013] Wander, A., and Frstner, R. 2013. Innovative Fault Detection, Isolation and Recovery Strategies On-Board Spacecraft: State of the Art and Research Challenges. In *Proceedings of Deutscher Luft- und Raumfahrtkongress, Berlin, Germany*.

[Weld, Anderson, & Smith, 1998] Weld, D. S.; Anderson, C. R.; and Smith, D. E. 1998. Extending graphplan to handle uncertainty & sensing actions. In *Association for the Advancement of Artificial Intelligence and Innovative Applications of Artificial Intelligence Conferences (AAAI/IAAI), Madison, Wisconsin, USA*, 897–904. AAAI Press / The MIT Press.

[Wettergreen *et al.*, 2005a] Wettergreen, D.; Cabrol, N.; Baskaran, V.; Caldern, F.; Tompkins, P.; Villa, D.; Williams, C.; and Wagner, M. 2005a. Second experiments in the robotic investigation of life in the Atacama Desert of Chile. In *International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS), Munich, Germany*.

[Wettergreen *et al.*, 2005b] Wettergreen, D.; Cabrol, N.; Teza, J.; Tompkins, P.; Urmson, C.; Verma, V.; Wagner, M. D.; and Whittaker, W. 2005b. First experiments in the robotic investigation of life in the atacama desert of chile. In *IEEE International Conference on Robotics and Automation (ICRA), Barcelona, Spain*, 873–878. IEEE.

[Wettergreen *et al.*, 2008] Wettergreen, D.; Wagner, M.; Jonak, D.; Baskaran, V.; Deans, M.; Heys, S.; Pane, D.; Smith, T.; Teza, J.; Thompson, D.; Tompkins, P.; and Williams, C. 2008. Long-Distance Autonomous Survey and Mapping in the Robotic Investigation of Life in the Atacama Desert. In *International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS), Hollywood, USA*.

[Wettergreen *et al.*, 2009] Wettergreen, D.; Jonak, D.; Kohanbash, D.; Moreland, S.; Spiker, S.; and Teza, J. 2009. Field experiments in mobility and navigation with a lunar rover prototype. In *Field and Service Robotics (FSR), Cambridge, Massachusetts, USA*, 489–498.

[Williams *et al.*, 2003] Williams, B. C.; Ingham, M. D.; Chung, S. H.; and Elliott, P. H. 2003. Model-based programming of intelligent embedded systems and robotic space explorers. *IEEE* 91(1):212–237.

[Wood, 2002] Wood, E. G. 2002. Multi Mission Power Analysis Tool. In *Information Technology (IT) Symposium, Pasadena, CA, USA*.

[Younes & Simmons, 2002] Younes, H. L. S., and Simmons, R. G. 2002. On the role of ground actions in refinement planning. In Ghallab, M.; Hertzberg, J.; and Traverso, P., eds., *International Conference on Artificial Intelligence Planning Systems, Toulouse, France*, 54–62. AAAI.