ICAPS05

DEM

System Demonstrations

Ioannis Refanidis
University of Macedonia, GREECE

ICAPS 2005
Monterey, California, USA
June 6-10, 2005

CONFERENCE CO-CHAIRS:

Susanne Biundo
University of Ulm, GERMANY

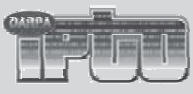
Karen Myers
SRI International, USA

Kanna Rajan
NASA Ames Research Center, USA

Cover design: L.Castillo@decsai.ugr.es

System Demonstrations

Ioannis Refanidis
University of Macedonia, GREECE

System Demonstrations

Table of contents

System Demonstrations

Preface

Following a well established tradition, ICAPS 2005 involves a System Demonstrations (SD) session being rich of interesting presentations. This year's SD program comprises eight demonstrations, mainly focusing on space and satellite applications, but also covering training, crisis management, mixed-initiative planning, web service composition and new planning algorithms.

In particular, Daniel Tran and his colleagues present the Autonomous Sciencecraft Experiment (ASE), currently flying onboard the Earth Observing-1 (EO-1) spacecraft. They demonstrate the potential for future space missions to use onboard decision making to respond autonomously to capture short-lived science phenomena. Their demonstration will include a live ASE contact from EO-1, as well as a full autonomous science scenario, during which ASE will command (a simulated) EO-1 to image science targets, process and analyze data and re-plan operations based on science results.

John Jaap and Patrick Meyer present Nexus, a P&S system supporting comprehensive modeling of tasks and resources, multi-user environment, remote/distributed access and incremental scheduling. Nexus aims at being a new P&S paradigm used directly by the end-users to produce their own timelines, e.g. by astronauts in space to schedule their own activities.

Ari Jonsson and his colleagues present MAPGEN, a mixed initiative activity plan generation system that is used in the Mars Exploration Rover mission surface operations to build activity plans for each of the rovers, each Martian day. The system supports activity plan editing and resource modeling together with an advanced constraint-based reasoning and planning module that supports building plans in an interactive fashion. The demo will show how the system has been used for actual Mars rover operations.

Tania Bedrax-Weiss and her colleagues present EUROPA2, an expressive and reusable platform that provides plan database services for building P&S systems. EUROPA2 treats planning as a dynamic constraint satisfaction problem, by incrementally adding constraints and variables as actions are selected to be in the plan. EUROPA2 has been used in LORAX, an ASTEP project concerning microbial sampling in an Antarctic Glacier, whereas it is currently used in another project concerning advanced robotic capabilities in the field.

Froduald Kabanza and his colleagues present Roman Tutor, a system aiming at teaching astronauts how to operate a robot manipulator deployed on the International Space Station. Operators must rely on cameras mounted on the manipulator and at strategic places of the environment where it operates. Roman Tutor uses a robot path-planner to automatically check errors of a student learning to operate the manipulator and to automatically produce illustrations of good and bad motions in training.

Marco Pistore and his colleagues present ASTRO, a set of tools that extend existing platforms for web service design and execution with automated composition and execution monitoring functionalities. ASTRO extends the Active WebFlow platform, a commercial tool for designing and developing BPEL4WS processes. The demo consists of a set of steps corresponding to the execution of service compositions and a monitor synthesis task.

Luis Castillio and his colleagues present SIADEX, an integrated framework to support decision making during crisis episodes, by providing realistic temporally annotated plans. SIADEX implements a forward state-based HTN temporal planner and a knowledge base in Protégé format. It also makes intensive use of web services to implement most of its capabilities, thus allowing accessing them from any device with internet connectivity.

Finally, Yixin Chen and his colleagues demonstrate SGPlan, a PDDL2.2 planner working by partitioning problem constraints by their subgoals into multiple subsets, solving each subproblem individually and resolving inconsistent global constraints across subproblems based on a penalty formulation. SGPlan won the first prize in the suboptimal metric track and a second prize in the suboptimal propositional track in the 4th International Planning Competition, 2004.

Before closing this quick introduction to this year's demonstrations, I have to thank all presenters for the time they have invested to make this event successful. I hope that this quick introduction has excited your interest in having a more thorough look at the short papers that follow in this booklet, and, furthermore, in attending the live demonstrations that promise to be fascinating. The presenters and I will wait you there!

Organizer

- *Ioannis Refanidis, University of Macedonia, GREECE*

DEMO: The Autonomous Sciencecraft Experiment Onboard the EO-1 Spacecraft

Daniel Tran, Steve Chien, Rob Sherwood, Rebecca Castano, Benjamin Cichy, Ashley Davies, Gregg Rabideau

Jet Propulsion Laboratory, California Institute of Technology
4800 Oak Grove Dr.
Pasadena, CA 91109
Firstname.Lastname@jpl.nasa.gov

Abstract

The Autonomous Sciencecraft Experiment (ASE), currently flying onboard the Earth Observing-1 (EO-1) spacecraft, integrates several autonomy software technologies enabling autonomous science analysis and mission planning. The experiment demonstrates the potential for future space missions to use onboard decision-making to respond autonomously to capture short-lived science phenomena. The software demonstration will consist of two sections: a real-time display of an ASE-commanded ground contact from the EO-1 spacecraft, and a simulation of the full ASE autonomous science-response scenario.

Introduction

The Autonomous Sciencecraft Experiment (ASE) [Chien et al., 2004] is currently flying onboard NASA's Earth Observing-1 (EO-1) spacecraft. Uploaded in the Fall of 2003, ASE has successfully commanded EO-1 operations and enabled autonomous retargeting to capture dynamic science events. ASE has currently collected over 1500 scenes and executed over 150 ambitious science-response scenarios. The ASE onboard flight software includes several autonomy components:

1. *Onboard science processing algorithms.* Science analysis algorithms process onboard image data to detect science events and suggest reactions to maximize science return.
2. *Onboard planning and scheduling software.* The Continuous Activity Scheduling Planning Execution and Replanning (CASPER) [Chien et al., 2000] system generates mission operations plans from goals uplinked by the EO-1 Science Team or inserted by the onboard science analysis module. The model-based planning algorithms enable rapid response to a wide range of operations scenarios based on models of spacecraft constraints.
3. *Robust execution software.* The Spacecraft Command Language (SCL) expands the CASPER mission plans to low-level spacecraft commands. SCL monitors the execution of the plan and has the flexibility and knowledge to perform improvements in execution as well as local responses to anomalies.

Building autonomy software for space missions presents a number of key challenges:

1. *Limited, intermittent communications*. EO-1 has eight ten-minute communications opportunities per day. This means that the spacecraft must be able to operate for long periods of time without supervision.
2. *Complex subsystems and controls*. A typical spacecraft has thousands of components carefully engineered to survive the rigors of space.
3. *Limited observability*. Bandwidth and onboard processing constraints limit the availability of engineering telemetry. Onboard software and ground operations teams must be able to operate the spacecraft on limited information.
4. *Limited computing resources*. An average spacecraft CPU offer 25 MIPS and 128 MB RAM – far less than a typical personal computer. The EO-1 team allocated 4 MIPS for all of the ASE software.
5. *High stakes*. A typical space mission costs hundreds of millions of dollars, any failure has significant economic impact.

ASE on EO-1 demonstrates an integrated autonomous mission using onboard science analysis, replanning, and robust execution. ASE performs intelligent science data processing, editing, and spacecraft retargeting, leading to a reduction in data downlinked and an increase in science return. These capabilities enable radically different missions with significant onboard decision-making allowing novel science opportunities. The paradigm shift toward highly autonomous spacecraft will enable future

NASA missions to achieve significantly greater science returns with reduced risk and reduced operations cost.

ASE has also teamed with NASA Ames Research Center to fly the Livingstone 2 Mode Identification and Diagnosis software in Fall 2004, which is used to monitor the health of EO-1 and detect fault conditions as they occur. By reconfiguring EO-1, ASE may recover functionality and continue on to meet its mission goals.

Live ASE Contact from EO-1

The first part of the demonstration will showcase a live contact from the EO-1 spacecraft *commanded onboard by ASE*. Telemetry from the spacecraft will be displayed remotely from the EO-1 Mission Operations Center at Goddard Spaceflight Center. The real-time telemetry stream will display the commands currently executing onboard EO-1 and the status of each ASE software component. During the demonstration ASE will downlink engineering and science data by issuing spacecraft commands to achieve the following objectives:

1. Point the X-Band phased-array antenna at the groundstation one minute prior to acquisition of signal (AOS).
2. At AOS, power on both transceivers and configure S Band for a 2 Mbit downlink rate.
3. Command the onboard solid state recorder to playback mode and begin streaming science data over the X-Band link.
4. Initiate the dumping of engineering data over S-Band.
5. Initiate the playback of the spacecraft event log through S-Band.
6. At loss of signal (LOS), turn off the S-Band transceiver, stop streaming science data, and begin streaming fill data.
7. Ten seconds after LOS, power down the X-Band transceiver and return the solid state recorder to standard operations mode.

A typical EO-1 contact lasts between 10 and 15 minutes.

Full Autonomous Science Scenario

The second part of the demonstration will simulate an ASE mission scenario. During this demonstration ASE will command a software simulation of the EO-1 spacecraft to image science targets, process and analyze onboard image data, and re-plan operations based on science results.

The demonstration will highlight the following capabilities of ASE:

1. *Autonomous Execution*. CASPER will generate a mission plan using an uplinked set of high-level goals requesting science observations and data downlinks. SCL will convert these plans to sequences of spacecraft commands and issue these commands to EO-1.
2. *Dynamic Event Tracking*. The onboard science analysis algorithms will detect an erupting volcano in one of the requested observations, and as a result recommend a follow-on observation.
3. *Data Editing*. During the analysis of another science observation, the analysis algorithms will determine an unacceptable percentage of cloud-obscured data. Based on this analysis the ASE software will recommend deleting this image and imaging an alternate target.
4. *Onboard Replanning*. CASPER will modify the onboard schedule to respond to the science analysis recommendations to insert new observations and delete low-value data from onboard storage.

Acknowledgement

Portions of this work were performed at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

We would like to acknowledge the important contributions of Nghia Tang and Michael Burl of JPL, Dan Mandl, Stuart Frye, and Stephen Ungar of GSFC, Jerry Hengemihle and Bruce Trout of Microtel LLC, Jeff D'Agostino of the Hammers Corp., Seth Shulman and Robert Bote of Honeywell Corp., Jim Van Gaasbeck and Darrell Boyer of ICS, Sandra Hayden, Adam Sweet, and Scott Christa of Ames Research Center, Michael Griffin and Hsiao-hua Burke of MIT Lincoln Labs, Ronald Greeley, Thomas Doggett, and Kevin Williams of ASU, and Victor Baker and James Dohm of University of Arizona.

References

S. Chien, B. Cichy, S. Schaffer, D. Tran, G. Rabideau, R. Bote, Dan Mandl, S. Frye, S. Shulman, J. Van Gaasbeck, D. Boyer, "Validating the EO-1 Autonomous Science Agent", *Working notes of the Workshop on Safe Agents*, AAMAS-2003.

S. Chien, et al. "The EO-1 Autonomous Science Agent," *International Conference on Autonomous Agents and Multi-agent Systems (AAMAS 2004).* New York City, NY, July 2004.

S. Chien, R. Knight, A. Stechert, R. Sherwood, and G. Rabideau, "Using Iterative Repair to Improve Responsiveness of Planning and Scheduling," *Proceedings of the 5th Intl. Conference on AI Planning and Scheduling*, Breckenridge, CO, April 2000.

A.G. Davies, R. Greeley, K. Williams, V. Baker, J. Dohm, M. Burl, E. Mjolsness, R. Castano, T. Stough, J. Roden, S. Chien, R. Sherwood, "ASC Science Report," August 2001. (available at ase.jpl.nasa.gov).

J. Kurien and P. Nayak. "Back to the future for consistency-based trajectory tracking." In Proceedings of the 7th National Conference on Artificial Intelligence (AAAI'2000), 2000.

Nexus: Planning Tomorrow, Today

John Jaap & Patrick Meyer

Missions Operations Laboratory
Marshall Space Flight Center
National Aeronautics and Space Administration
John.Jaap@nasa.gov

Abstract

To prepare for future human space flight programs, the Mission Operations Laboratory (MOL) at the Marshall Space Flight Center (MSFC) has been investigating new planning and scheduling paradigms. To support and prove this investigation, MOL technologists have developed a working prototype of a scheduling system to support the new paradigms. The new planning and scheduling system is called Nexus and has a web site at http://nexus.nasa.gov/. Nexus is based on a comprehensive modeling schema to capture all scheduling requirements typical to human space missions, an incremental scheduling engine tailored to the modeling schema, and remote access (including Personal Data Assistant (PDA) access) to the scheduling system. This paper describes the proposed paradigm shift and the enabling software. It also describes a typical Nexus demonstration which emphasizes how it works, how it enables the paradigm shift, and possible applications. Demonstrations include access to the full functionality of Nexus from a personal computer and access to limited functionally via a PDA.

Demonstrations

For demonstrations at the International Conference on Automatic Planning and Scheduling, the remote access configuration of Nexus is used; see Figure 1. A laptop PC and Pocket PCs (PDAs) use the internet to access the scheduling engine at MSFC.

Several key features of Nexus are shown in the demonstrations:

- Modeling Modal Resources, Tasks, & Sequences
- Request Submittal and Automatic Scheduling
- Multi-User Support
- Remote/Distributed Access
- Astronaut Autonomy Extrapolation

The technical details of the features are discussed later in this paper.

Comprehensive Modeling

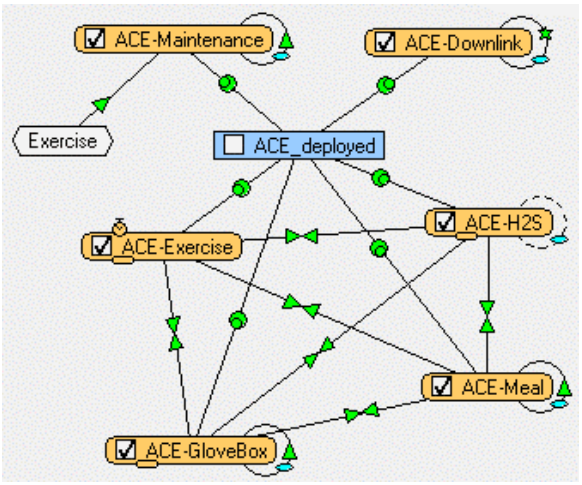


Figure 2 - A Complex Sequence/Task Model

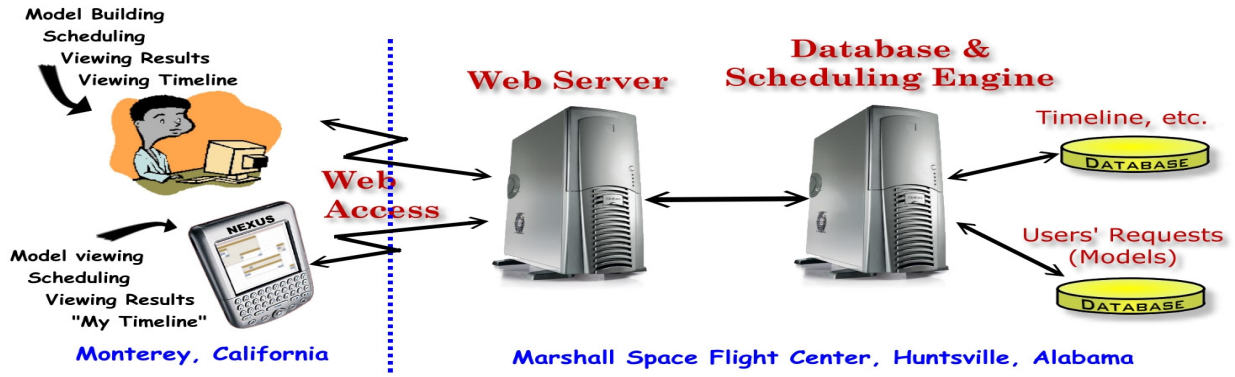


Figure 1 - Configuration of Nexus for the Demonstrations

Figure 2 is a screen shot of the canvas for building and editing a sequence. The rounded rectangles are embedded sequences, the rectangle is a task, and the hexagon is a public service. The lines connecting the nodes represent the temporal relationships. Editing is done graphically and via dialog boxes. The demonstration will also include modeling of tasks (which use resources) and implicit resource usage (modal resources).

Request Submittal and Automatic Scheduling

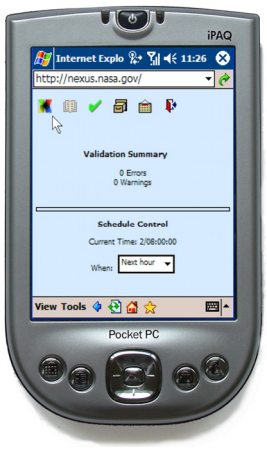


Figure 3 - Submitting from a PDA

Figure 3 shows the screen for submitting a request from the PDA. On a previous screen, the user chose a model to submit. Figure 4 shows the report that appears after the request is scheduled. The depicted request had a hierarchy of embedded sequences and activities. The figure also shows the mouse-over popup which gives details of an activity (shown) or a sequence. A tree control, shown on the left, is used to expand and collapse the results and a pan/zoom control is used to control the time axis.

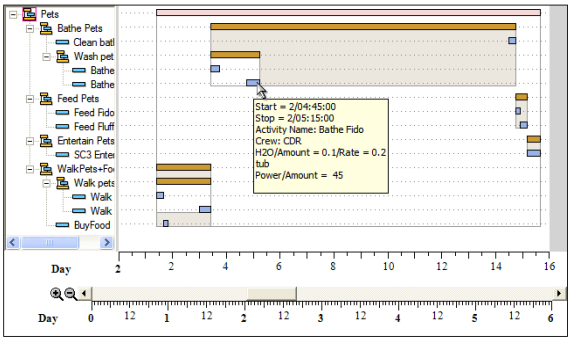


Figure 4 - Scheduling Report on PC

Multi-User Support

Nexus uses an incremental scheduling engine so that multiple users can contribute concurrently to a timeline. During the demonstrations, two users will simultaneously build one timeline. Each will see the results of the other's contributions.

Remote/Distributed Access

The demonstration is accomplished by accessing the database and scheduling engine in Huntsville, Alabama, at NASA's Marshall Space Flight Center. The backend software runs on a Windows OS running on Pentium III processors. Nexus can also be configured to run locally on a single PC.

Astronaut Autonomy Extrapolation

As described later in this paper, Nexus can be extrapolated to allow the crew (on orbit, on the moon or on Mars) to schedule their own activities and the activities of their companion systems. This usage of Nexus is demonstrated by assuming the Nexus is installed in situ. Astronauts have local access and PDA access when they are nearby. The PDA functionality is designed around crew usage, particularly the "My Timeline" screen.

A Paradigm Shift for Planning and Scheduling

The current state-of-the-art in modeling methodologies and scheduling engines results in a linear paradigm with knowledge contributed by task experts, vehicle experts, and scheduling engine experts. This paradigm requires significant effort and flow time. The task experts often struggle to enter their requirements using a language that is limited – often resorting to notes to fully describe their requirements. The vehicle and hardware experts then convert and augment this knowledge to further prepare the models for scheduling. The scheduling team then feeds the models to the scheduling engine. Since the models are incomplete, the team often has to "steer" the scheduler to produce an acceptable schedule.

Nexus enables a streamlined paradigm. The vehicle experts enter the system and hardware constraints independently of the task knowledge. The task experts enter the task requirements. The comprehensive modeling schema would allow them to specify all of the task requirements without resorting to notes for the scheduling team; consequently, these models are ready for automatic scheduling. Having models that express all the constraints allows the scheduling engine to operate automatically without human intervention.

The Nexus Project

Comprehensive Task Modeling

The modeling schema of Nexus captures all the requirements so that its automatic scheduler can function, and modeling can easily be done by various users including ground controllers and astronauts. Some of the key features of the task modeling schema are given below (Jaap, Davis & Richardson, 2004).

- Decomposition into salient components— Modeling is based on tasks and sequences.
- Task models allow hierarchies of constraints to be represented using statements like "one of" and "all of" in an outline format.
- Intuitive and rich expression of the relationships between components— Modeling employs common-sense modeling of temporal relationships using everyday concepts like sequential, during, and overlap. Innovative modeling of the continuance of resource usage between tasks, the fragmentation of tasks, and minimal percent coverage are included.
- Public Services— Modeling also includes the concept of public services, models that are scheduled at the request of another model.
- Experiment flexibility— Variable timing can be modeled at all levels. Additionally, alternate resources and alternate sequences can be modeled.
- Representation of nuances of the tasks— Even nuances, such as locking in alternate resources and one-to-one relationships are available.

Implicit Resource Modeling

Modal resources are the basic building blocks of a task model. Tasks are normally accomplished by using multiple pieces of equipment which, in turn, use other resources. Most types of equipment have various operating modes: e.g., a microwave oven has modes such as defrost, reheat, and cook. The power requirements of each mode are predefined. On a lunar or Martian base, the characteristics of each piece of equipment will be known to those building and integrating the equipment into the habitat. The equipment and the equipment modes can be modeled independently with respect to the tasks that will use the equipment. Mode models use an outline paradigm like that used by task models. Using mode models as building blocks of task models means that the person doing the task modeling does not need to know how the equipment is integrated into the habitat/vehicle system and does not need to know the details of the resource requirements; for example, the task modeler only needs to know that the camera will downlink high-quality video, without knowing the bandwidth, data rate or power requirements. This concept was proposed in a paper by Hagopian & Maxwell (1996).

Scheduling Engine

Nexus uses the Scheduling Algorithm for Temporal Relation Networks (SATRN) as an "incremental" scheduler. SATRN processes a single request (adding one or more tasks to the timeline) and then waits for another request.

SATRN converts the temporal relations of a sequence to time bounds on the sequence entities (embedded sequences or activities). As each entity is scheduled, the bounds on not-scheduled entities are shrunk. An activity's requirements are checked by a depth-first search. Variable activity durations are utilized to stay within the time bounds. When an entity cannot be scheduled, smart backtracking and reordering is used to unshrink the bounds on hard-to-schedule entities. Backtracking is also used to explore alternate requirements ("one-of" groups, scenarios, optional entities, and relationships to existing entities with multiple instances).

Configurations

- Multi-user— Multiple users can simultaneously build a single timeline through a centralized scheduling server. Users only need a computer and an Internet connection – even astronauts on orbit. This can enable those closest to the tasks being performed to enter their requirements and build the best schedule for themselves.
- Personal Data Assistant (PDA)— Crew self-planning is another configuration of Nexus. Astronauts could plan their own day rather than be tied to the traditional remote planning done today.
- Standalone— One user, on a single computer, working on one schedule is the classical configuration for planning and scheduling systems. This configuration would be the simplest setup of Nexus.

Applications of Nexus

Cost-savings applications

Nexus can allow the customers to produce their own timelines. The customers have the best knowledge of their scheduling requirements and know when a timeline meets their needs. Nexus enables them to perform their own scheduling because –

- The use of an incremental scheduler prevents the action of one user from impacting another user.
- The Nexus modeling schema captures all the requirements without resorting to artificial resources or difficult rule expressions.
- By submitting incrementally and getting immediate feedback from the scheduling engine, the users become virtual experts on modeling and scheduling.

Cost reduction is realized by reducing the size of the scheduling cadre and the flow time required to produce a schedule. A paper by Jaap and Muery (2000) describes in detail how a Nexus-like system could reduce the cost of *ISS* operations.

Crew-autonomy application

Nexus can allow the astronauts in space to schedule their own activities. Crew autonomy is essential as we explore further from home. The astronauts are aware of their situation and understand their needs and desires better than anyone. The light-time delay (up to 20 minutes from Mars) makes normal conversations with the earth-based support team impossible. The ground support team would build the baseline models and baseline schedule and upload it to the in-situ installation of Nexus. The astronauts would add to or delete from the timeline as desired. Ground support would also be able to modify the timeline using the remote access capabilities of Nexus.

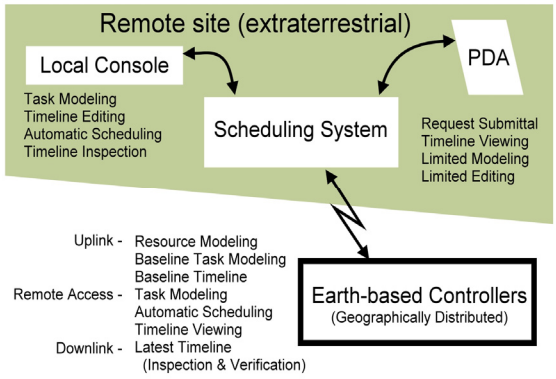


Figure 5 - In-Situ Configuration

For this application, Nexus is configured with the scheduling system, most current timeline, and other associated data located at the "extraterrestrial" site (e.g., exploration vehicle, lunar or Mars base) where the timeline is being executed; see Figure 5. This configuration ensures that the astronauts will always have access to a complete set of up-to-date planning information, and that any changes they make are applied to the currently executing timeline.

However, this operations concept does not imply that the astronauts will be tasked with performing the entire mission planning job. Since crew time is extremely valuable, the prime task of developing and maintaining the baseline timeline will still fall on ground-based support personnel. The level of astronaut participation in the planning process will be dictated by necessity (e.g., responding to real-time events) as well as by their personal preferences. In effect, the configuration provides an infrastructure which allows multiple parties (crew, ground support, and even autonomous systems) to simultaneously contribute to the development/maintenance of a single timeline.

In this application, Earth-based support will be responsible for collecting and entering into the scheduling system (i.e., "modeling") the underlying information needed to build the timeline. This information includes resource availability predictions, flight trajectory information, equipment/system status, and other planning constraints. The earth-based planners will also work with other flight controllers, vehicle/systems experts, and payload providers to model the many "tasks" that must be scheduled on the timeline. If appropriate, they may also create an initial timeline for a future period of time. All this preliminary work can be performed on the ground and the results then uplinked to the extraterrestrial remote site, where the information becomes part of the data set associated with the currently active timeline.

Once the planning information is within the scheduling system at the remote site, it will be available for use by the crew. From a local console, the crew will be able to view/inspect their timelines, make timeline edits (e.g., move a task, delete a task), schedule additional tasks via an interface to the automatic scheduling engine, and even edit the modeled tasks (e.g., change a specified task duration). A PDA-type interface to the scheduling system might also be available to the crew; however, capabilities at the PDA would necessarily be limited. Such an interface is depicted in Figure 6.

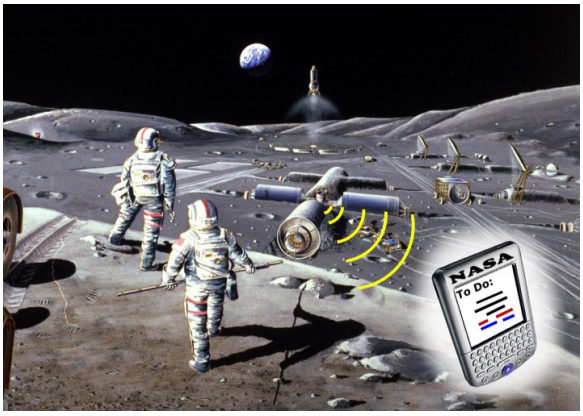


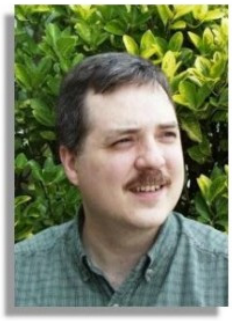
Figure 6 - Lunar Astronauts

Earth-based personnel can also remotely access the onboard scheduling system to inspect/verify the most current timeline information or to contribute timeline changes. To preserve precious crew time, it is envisioned that most extensive re-planning efforts will be performed by the earth-based planners, except in those cases where communications outages or delays preclude a timely ground response to a real-time event. The earth-based planners may also perform simpler timeline edits at the crew's request.

Providing the astronauts with the ability to manage the schedule will enable more autonomous crew/vehicle operations. Unlike today, the crew will be able to make a real-time schedule change and get immediate feedback that the change is feasible, thus supporting safe, reliable, and efficient crew operations.

Presenters

Patrick Meyer – Mr. Meyer began working for MSFC in 1990 developing graphical user interfaces for mission planning systems. His tasks have included making complex human-computer interactions more functional, modeling activities, and orbit analysis visualization tools. He graduated Magna Cum Laude with a B.S. in Computer Science from the University of Alabama in Huntsville. He has received numerous awards including a Space Act Award, a Silver Snoopy, a Director's Commendation, an MSFC Appreciation and several sustained superior performance awards.



John Jaap -- Mr. Jaap has been employed at the MSFC since 1980. He has developed mission planning software since 1972, and space activity scheduling software since 1978. Mr. Jaap received a B.A. in Mathematics from Mississippi State University. He is also the recipient of a Silver Snoopy, an NASA Exceptional Achievement Medal, a Space Act Award, and a Space Flight Awareness Honoree Award.

References

Hagopian, J., and Maxwell, T., "Explicit and Implicit Resources: A Simplified Approach to User Requirements Modeling," in proceedings of Space Ops 96, Fourth International Symposium on Space Mission Operations and Ground Data Systems, September 1996.

Jaap, J., Davis, E., and Richardson, L., "Maximally Expressive Modeling," in proceedings of the Fourth International Workshop on Planning and Scheduling for Space (23-25 June, 2004), Darmstadt, Germany.

Jaap, J., & Muery, K., "Putting ROSE to Work: A Proposed Application of a Request-Oriented Scheduling Engine for Space Station Operations," in proceedings of the Sixth International Conference on Space Operations (SpaceOps 2000), Toulouse, France, June 2000.

MAPGEN: Mixed-initiative activity planning for the Mars Exploration Rover mission

Mitchell Ai-Chang, John Bresina, Kimberly Farrell, Jennifer Hsu, Ari Jónsson, Bob Kanefsky, Michael McCurdy, Paul Morris, Kanna Rajan, Alonso Vera, Jeffrey Yglesias
Leonard Charest, Pierre Maldague
NASA Ames Research Center, MS 269-2, Moffett Field, CA 94035
Jet Propulsion Laboratory, 4800 Oak Grove Drive, Pasadena, CA 91109
jonsson@email.arc.nasa.gov

Abstract

The Mixed Initiative Activity Plan Generation system, MAPGEN, is one of the critical tools in the Mars Exploration Rover mission surface operations. The system is used to build activity plans for each of the rovers, each Martian day.

The MAPGEN system combines an existing tool for activity plan editing and resource modeling, with an advanced constraint-based reasoning and planning framework. The constraint-based planning component provides active constraint and rule enforcement, automated planning capabilities, and a variety of tools and functions that are useful for building activity plans in an interactive fashion.

In this demonstration, we will show the capabilities of the system and demonstrate how the system has been used for actual Mars rover operations. Since the demonstration of an earlier version at ICAPS 03, significant improvements have been made to the system. These include various additional capabilities that are based on automated reasoning and planning techniques, as well as a new Constraint Editor (CE) support tool.

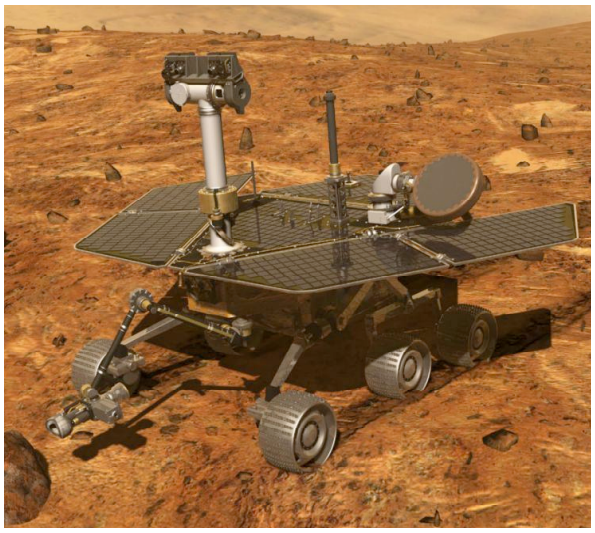


Figure 1: MER Rover

Overview

In January 2004, two NASA rovers, named Spirit and Opportunity, successfully landed on Mars, starting an unprecedented exploration of the Martian surface. Power and thermal concerns constrained the expected duration of this mission, leading to an aggressive plan for commanding both rovers every day.

As part of the process for generating these command loads, the MAPGEN tool provides engineers and scientists with an intelligent activity planning tool that allows them to more effectively generate complex plans that maximize the science return each day. The key to the effectiveness of the MAPGEN tool is an underlying constraint-based planning and reasoning engine.

Constraint-based Planning

The automated reasoning component of MAPGEN is based on an advanced constraint-based planning system called EUROPA (Jónsson, et al., 1999; Frank and Jónsson, 2003). In constraint-based planning, activities and states are described by predicate statements that hold over temporal intervals. The interval time-points and the predicate parameters are represented by variables connected by constraints. This approach supports a variety of complex planning constructs, including: activities with extended temporal durations, states that expire, exogenous events, complex constraints on parameters, temporal constraints linking activities and states, and subgoaling rules with conditions and disjunctions.

A constraint-based planning domain model defines a set of predicates, each of which has a set of parameters with possible values. The model also defines configuration constraints on predicates appearing in a plan. The notion of these configuration constraints is quite general and includes temporal and parametric constraints, as well as requirements for other activities and states in the plan. For example, a configuration rule may specify that any

Rover_Drive activity must be temporally contained by a CPU_On state.

In constraint-based planning, a partial plan consists of a set of activities and states, connected by constraints. The partial plan may be incomplete, in that rules are not satisfied and pending choices have not been made. The planning process then involves modifying a partial plan until it has been turned into a complete and valid plan.

Traditional search-based methods accomplish this by trying different options for completing partial plans, and backtracking when constraints or rules are found to be violated. Constraint reasoning methods, such as propagation and consistency checks can be used to eliminate options and identify dead-ends early. In constraint-based planning, arbitrary changes can be made to a candidate plan, supporting user changes, random exploration and a variety of other methods for building plans.

Automated Reasoning in MAPGEN

The MAPGEN system combines the core capabilities of APGEN (Maldague, et.al, 1998), an existing plan editing tool, with the automated reasoning functionality of EUROPA. The automated reasoning component adds three key capabilities to the activity plan generation process.

The first is that constraints and rules are actively enforced. Without active enforcement, constraint violations are only identified after the violation has been created. As an example, consider a constraint specifying that a picture must be taken between 10:10 and 10:30. Without active constraint enforcement, the user can schedule the activity at any time. If the chosen time is outside the allowed time frame, the system notifies the user that the constraint is violated. With active constraint enforcement, the system can continuously maintain that the picture is scheduled within the given timeframe. When the user attempts to move the activity outside the interval, the system prevents it from moving further than the end of the interval.

The second is a variety of automated search techniques, such as completing partial plans, and fixing plans that violate resources or constraints. To complete a plan, or part of a plan, a variation of a backtracking search engine is used. The key difference is that when it appears that backtracking is thrashing, the search mechanism can choose to eliminate a low priority activity from the plan. This avoids the computational expense of exploring all options before rejecting an activity that cannot fit into the plan, but at the cost of completeness.

Finally, the automated reasoning capabilities are used to provide a variety of tools that assist the users in building activity plans. For example, users can move activities interactively, immediately seeing the impact of temporal

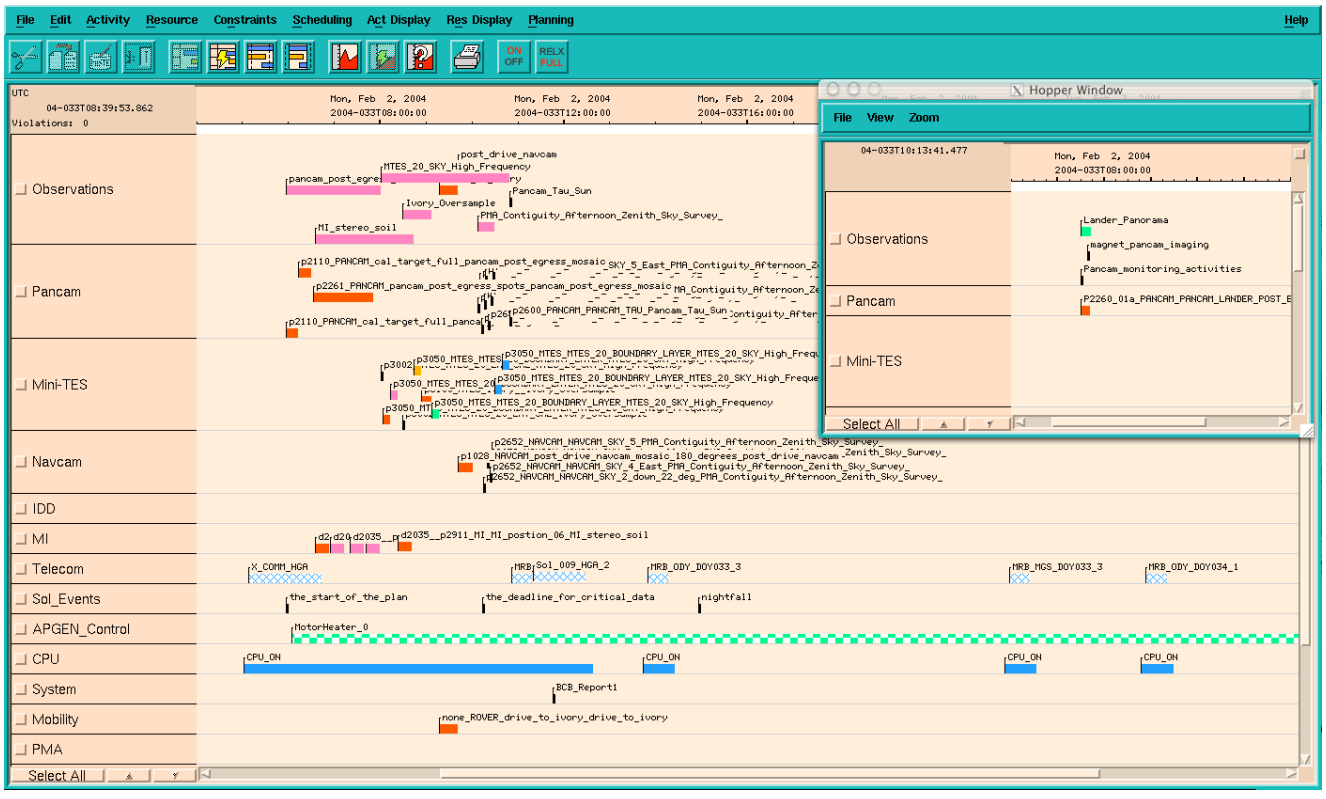


Figure 2. MAPGEN interface showing a simple plan and the hopper

constraints, while getting tactile feedback on the limits posed by the constraints.

Constraint Editor

The model of rover activities, states and rules results in constraints automatically being instantiated for partial plans. Such constraints specify general rules, such as any occurrence of a Rover_Drive activity requiring that the on-board CPU be turned on. But each day there are other constraints that apply only to specific instances. For example, there may be a constraint saying that the activity instance Drive_To_Big_Rock must be completed before 13:30 local Mars time. In order for the automated reasoning system in MAPGEN to be sufficiently informed, this information must be made available to it each day.

The MAPGEN interface, which uses an existing mission activity plan editing tool, is only aimed at editing activity instances, and does not provide a good interface for adding and editing constraint instances. To make up for this, a Constraint Editor tool was developed. This tool can read in an activity plan, including activities and constraints, and allows the user to add, change and edit constraints. The updated set of constraints is then passed on to MAPGEN, which takes it into account in its automated reasoning.

MAPGEN in Operations

The MAPGEN tool is used in a complex and challenging commanding process for the MER rovers. Towards the end of each Martian day, a rover will downlink data to Earth. The essence of the commanding process is then to use this and other data to determine what the rover will do the following day. The process starts with analysis of the data, both in terms of scientific information and information about the state and health of the rover. This feeds into a process where scientists and engineers construct sets of desired activities for the following day. The challenge of the activity planning process is to select an optimal set of requests, and then schedule them in such a fashion that all flight rules are satisfied, all required support activities are in place, and all resource limits are respected. This is where MAPGEN is used. The resulting activity plan is then used to construct the detailed set of commands to be sent up to the rover.

MAPGEN and the activity planning process

To give an indication of how the system is used, we will now provide an overview of how the activity planning might proceed on a typical day. The inputs into the activity planning process are:

- An engineering skeleton plan, which specifies communications activities and other engineering-related activities.
- A set of prioritized science observation requests, each of which may consist of multiple activities, some of which are linked by temporal constraints.

The first step is to codify the constraints scientists have specified informally for their observation requests. This is done in the Constraint Editor, where temporal constraints are specified using a mixture of graphical and fill-out

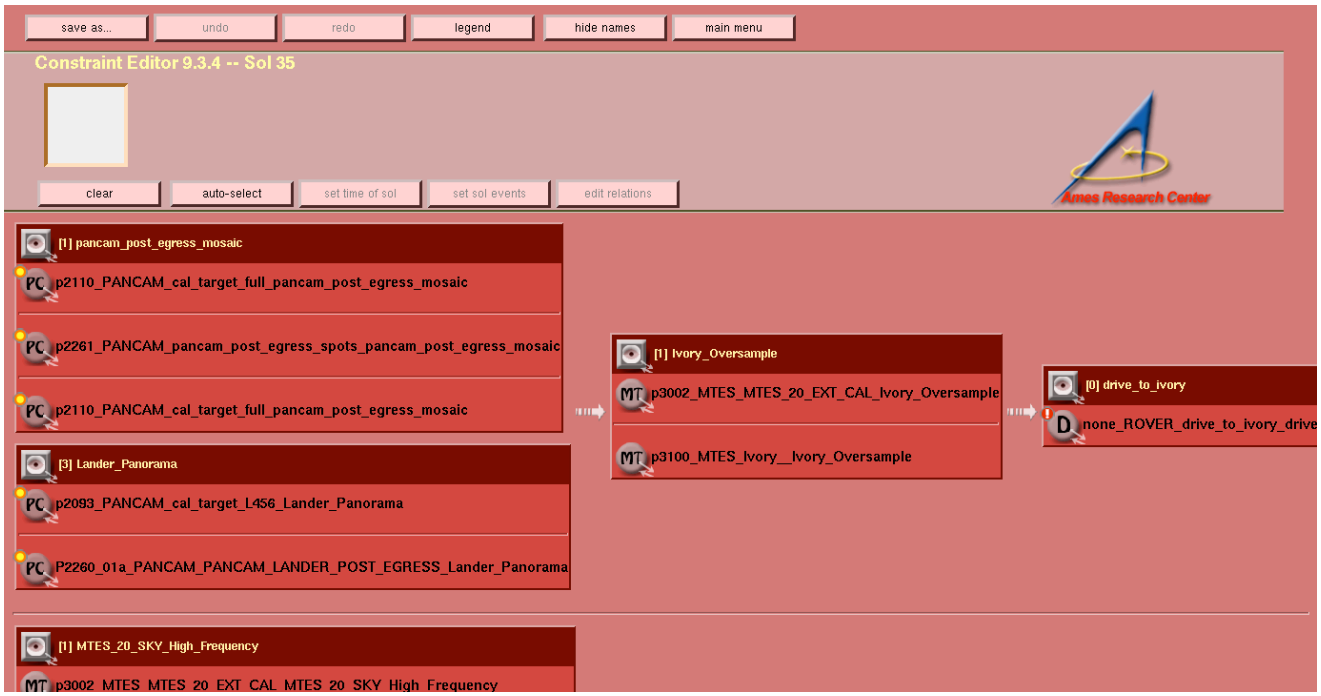


Figure 3. Constraint Editor Interface

interfaces. Common constraints include:

- Ordering relations between observations and activities. For example, an activity involving the rover arm's microscopic imager must be done prior to the rover driving to a new location, whereas the imaging activity to localize the rover at the new location is to be done after the drive completes.
- Temporal distance relations between activities. For example, an imaging activity should be at most twenty minutes from the associated calibration image. Another example is that periodic thermal spectroscopy activities of the sky should be spaced at least half an hour apart but no more than an hour apart.
- Time-of-day constraints for activities. The most common examples are activities requiring daylight, as is the case for most imaging activities. Other examples include more specific time bounds, so as to achieve a certain level of illumination, or to avoid shadows.

Next, the engineering skeleton plan, the science observation requests and the now-codified constraints are read into MAPGEN, and the core activity planning process starts. The goal of this process is to have a valid and safe plan that satisfies all applicable rules and limitations, but at the same time, achieves as many science requests as possible, weighted by priorities. This challenging problem is further complicated by some of the flight rules not being codified for the EUROPA planning system, and by there being complex preferences that the mission operators may have regarding the desired plan.

At the start of the planning process, the science observations are kept in a hopper, which is a holding area for activities that are currently not in the plan. The MAPGEN operator will then work on building the plan, using a somewhat iterative process inter-leaving plan extension operations with plan modification operations.

To add to the plan, the operator will add activities from the hopper to the plan. This can be done in a number of different ways:

- Fully automated planning where as many activities as possible weighted by priorities, are fit into the plan, such that codified flight rules and constraints are satisfied.
- Selective planning, where selected activities are fit into the plan, if possible, such that flight rules and constraints are satisfied.
- Placement planning, where selected activities are fit into the plan, as close as possible to a time specified by the user.

In order to ensure the plans also satisfy non-codified rules and preferences, the MAPGEN operators often need to make adjustments to the plans. The most common ones are changes in start times, and removal of activities, but sometimes the operator may also add new activities or modify the specifics of given activities.

The process of making manual modifications is made easier in MAPGEN by the continuous automatic enforcement of constraint and flight rules. This means the operator can concentrate on the desired changes and not worry about introducing violations by accident. Consider, for example, moving an activity to a different time. The user does this by dragging the activity in the interface, but the MAPGEN system will limit the range of this move to within where the activity placement will satisfy applicable rules and constraints. For another example, consider the addition, modification or removal of activities in the plan. The MAPGEN system will automatically ensure that new and modified activities do not overlap incompatible activities, and will also automatically update necessary support activities, such as making sure the on-board computer is turned on when needed.

The final activity plan is presented at a plan approval meeting where scientists, mission managers, and rover system experts go over the plan and approve it.

Conclusion

The MAPGEN tool is one of only a handful of AI-based planning and scheduling tools to be used to build plans for operating spacecraft, and is the first to be used for operating a rover on another planet. The tool has performed well in the MER mission, and has been used for every nominal command uplink since surface operations started in January 2004. More importantly, MAPGEN has made a notable positive impact on the mission operations. By actively enforcing flight rules and providing significant assistance with plan generation, the tool has allowed engineers to build more complex plans that achieve more science in less time than they otherwise could have.

References

Bresina, J, Jónsson, A., Morris, P., and Rajan, K. 2005. Activity Planning for Mars Exploration Rovers. In Proceedings of 15th International Conference on Automated Planning and Scheduling (ICAPS 2005).

Frank, J., and Jónsson, A., 2003. "Constraint-based Attribute and Interval Planning". In *Constraints*, 8(4), p 339-364.

Jónsson, A., Morris, P., Muscettola, N., and Rajan, K. 1999. Next generation Remote Agent planner. In Proceedings of the Fifth International Symposium on Artificial Intelligence, Robotics and Automation in Space (iSAIRAS99).

Maldague, P., Ko, A., Page, D., and Starbird, T., 1998. APGEN: A multi-mission semi-automated planning tool. In First International NASA Workshop on Planning and Scheduling, Oxnard, CA.

EUROPA 2: Plan Database Services for Planning and Scheduling Applications

Tania Bedrax-Weiss and **Conor McGann** and **Michael Iatauro**
QSS Group, Inc.
Intelligent Systems Division
NASA Ames Research Center
Mailstop 269-4
Moffett Field, CA 94035-1000
{tania,cmcgann,miatauro}@email.arc.nasa.gov

Abstract

Introduction NASA missions require solving a wide variety of planning and scheduling problems with concurrent operations and temporal dependencies; simple resources such as robotic arms, communications antennae and cameras; complex replenishable resources such as memory, power and fuel; and complex constraints on geometry, heat and lighting angles. Planners and schedulers that solve these problems are used in ground tools as well as onboard systems. The planners are usually embedded in larger applications such as multi-agent systems, execution systems, or mixed-initiative systems. The diversity of planning problems and applications of planners and schedulers precludes a "one-size fits all" solution. However, as different as these applications may seem, the underlying technologies are common across planning domains and applications. The difference lies in the specific configuration of technologies. For example, one application may require intensive resource calculations while another may not deal with resources at all. One application may require temporal reasoning while another may not. Providing a common platform for building planners and schedulers is essential for the success of these applications because it encourages reusability, reduces cost, and leverages state-of-the-art technologies across applications. EUROPA$_2$ is a more expressive and powerful system than EUROPA (Frank & Jónsson 2003) that provides plan database services for building planning and scheduling systems.

Planning Capabilities In order to solve the wide range of NASA problems we've built a planning and scheduling system that can: 1) reason with flexible temporal intervals; 2) reason about finitely quantified propositions; 3) do limited reasoning with infinite and dynamic domains; 4) reason with procedural constraints in addition to declarative constraints; 5) do conditional subgoaling; 6) reason with objects that have structure which can be inherited; 7) reason about the existence of propositions with properties; 8) reason about properties that must be true of sets of propositions; and 9) reason about resource production and consumption.

Flexible temporal intervals are necessary for execution systems where large part of the uncertainty lies in the duration of states. Allowing domains to be dynamic is necessary when dealing with objects that are "discovered" during the planning process, such as paths and acquisition targets. Being able to reason about the existence of propositions is important when requesting for instance that in order to move between acquisition targets there must be a path between them. Furthermore, reasoning about sets of propositions or objects is important when requesting that they have properties in common. All of these features add tremendous power to the reasoning and allows users to build complex NASA applications.

Planning with EUROPA 2 EUROPA$_2$ takes the approach to solve planning as a dynamic constraint satisfaction problem, by incrementally adding constraints and variables as actions are selected to be in the plan. A planner uses a problem description to initialize the plan database via transactions. The plan database consults the domain model to infer constraints and variables that must be added as a consequence of the transactions. The planner consults the plan database for conditions that need to be satisfied and commits to either adding a new state or specifying a variable. The plan database again consults the domain model and propagates if a new state has been added or just propagates if a variable is specified. The cycle continues until a plan is found.

Planning domain descriptions for EUROPA$_2$ are written in the New Domain Description Language (NDDL). NDDL provides an object-oriented syntax and semantics that makes it convenient to express sophisticated relationships among elements of a partial plan. NDDL expressivity includes: static objects (useful for describing composite entities that don't change over time), temporally scoped predicates, resources, object composition, object inheritance, conditional subgoaling, inifinite domains (limited capability), existential quantification, universal quantification (over finite domains), reference own constraints, define own base types.

A plan in EUROPA$_2$ is represented by a network of tokens (representing states) linked by constraints. A token implements a predicate and is a flexible time interval that has flexible start, end, and duration, as well as parameter variables. Temporal variables from a token are related to each other via temporal constrains. EUROPA$_2$ framework provides implementations for all of the Allen relations (Allen 1984) which are handled by a temporal reasoning compo-

nent. Internally the component is implemented as a simple temporal network (Dechter, Meiri, & Pearl 1991). Variables represent event times with domains that are temporal intervals and constraints specify distance bounds on variable pairs. The advantage of simple temporal networks is that consistency can be determined in polynomial time. The EUROPA$_2$ framework allows a user to exchange this component with another one that provides the services of temporal propagation.

Parameter variables of tokens are related to each other via constrataints that can be declarative or procedural in nature. The EUROPA$_2$ framework also exploits constraint reasoning inefficiencies present in many computations such as those using comparison, arithmetic and differential operators by providing procedural constraint propagation services. Procedures replace some of the declarative constraints. These also allow reasoning over variables with open domains and support real-valued reasoning. EUROPA$_2$ provides a framework for implementing and registering domain-dependent constraints.

Resource reasoning is another special case of propagation services. It works by bounding resource usage as actions are placed in the plan. Calculating tight bounds helps identify resource conflicts early and provide guidance to the planner. In EUROPA$_2$ resource calculations are made by a simulation providing earliest start time and latest end time resource profile. It creates a critical path and applies mutual exclusion reasoning to propagate integrated resource bounds and detect conflicts. We plan to provide an alternative implementation that uses maximal flow analysis to compute tight bounds as in (Muscettola 2002).

Constraints are managed by Propagators which aggregate constraints of specific types and processes them internally. A propagator is registered with a Constraint Engine that is responsible for managing variable changes and invoking propagators in reaction to these changes. Propagators can contain scheduling policies but we have not experimented with that yet. We also provide the ability to disable and enable constraints which improves efficiency in networks with many redundant constraints. The constraint engine triggers propagation in each of the propagators to quiescence.

A Rules Engine manages the model subgoaling rules and is tasked with invoking the rules that apply as a consequence of planning decisions. EUROPA$_2$ supports rules with multiple subgoals, arbitrary relationships among subgoals via constraints, existential and universal quantification over sets of subgoals or objects, and conditional rules. Users can also create their own rules to suit their specific applications.

Architecture The design of EUROPA$_2$ prioritized 1) Extendibility - making it easy to integrate new functionality and straightforward to customize existing capabilities; 2) Configurability - making it easy to selectively combine system components to meet the particular application needs; 3) Efficiency - providing fast access to key operations, and ensuring information can easily flow between collaborating elements. This was achieved by providing a kernel API that identifies the fundamental primitives of our underlying constraint-based planning paradigm: Objects, Tokens, Constraints and Variables. A large number of component implementations may be placed as specializations of this framework. We allow flexible connections to be made between components through standardized interfaces with accessible attachment points. For example, if temporal constraints are not important in a problem, the temporal propagator may be removed and/or replaced with a default propagator. Since subgoaling has been encapsulated in a rules engine the behavior can be replaced, customized or omitted if necessary. Selection and composition is useful as it allows systems to avoid incurring costs for components that are not required.

EUROPA$_2$ provides several points of extension for application builders: 1) constraints; 2) propagators; 3) object model; 4) subgoaling rules; 5) specialized domains.

Applications LORAX (Life Observing Remote Antarctic Explorer) is a recent ASTEP funded project to conduct microbial sampling around a 100 km perimeter of an Antarctic Glacier. The mission requires long duration autonomous operation using renewable energy sources (i.e. solar and wind). The autonomy architecture includes an on-board planner and executive implemented on EUROPA 2. The application requires managing a thermal reservoir with complicated constraints that relate temperature to energy levels which are best implemented as a custom resource type. In the early phase of the project, the planner and executive are being used as a simulator to evaluate robot design trade-offs.

EUROPA$_2$ is currently being used by the Collaborative Decision Systems Program to demonstrate advanced robotic capabilities in the field. The field test is using EUROPA$_2$ for planning target selection, instrument placement, navigation, and execution. The executive is encapsulated by an IDEA agent (Muscettola *et al.* 2002) which communicates with the hardware.[1]

References

Allen, J. F. 1984. Towards a general theory of action and time. *Arti cial Intelligence* 23:123–154.

Dechter, R.; Meiri, I.; and Pearl, J. 1991. Temporal constraint networks. *Arti cial Intelligence* 49:61–94.

Frank, J., and Jónsson, A. 2003. Constraint based attribute and interval planning. *Journal of Constraints* 8(4).

Muscettola, N.; Dorais, G.; Fry, C.; Levinson, R.; and Plaunt, C. 2002. Idea: Planning at the core of autonomous reactive agents. In *Proceedings of the 3d International NASA Workshop Planning and Scheduling for Space*.

Muscettola, N. 2002. Computing the envelope for stepwise constant resource allocations. In *Proceedings of the* 8^{th} *International Conference on the Principles and Practices of Constraint Programming*.

[1]This research was supported by NASA Ames Research Center and the NASA Collaborative Decision Systems program.

Roman Tutor: A Robot Manipulation Tutoring Simulator

Khaled Beghith and **Froduald Kabanza**
Université de Sherbrooke
{khaled.belghith, froduald.kabanza}@usherbrooke.ca

Roger Nkambou and **Mahie Khan**
Université du Québec a Montréal
{nkambou.roger, khan.mahie}@uqam.ca

Leo Hartman
Canadian Space Agency
leo.hartman@space.gc.ca

Abstract

This demonstration will be about *Roman Tutor*, a system that we are developing to teach astronauts how to operate a robot manipulator deployed on the International Space Station (ISS). Operators do not have a direct view of the scene of operation on the ISS and must rely on cameras mounted on the manipulator and at strategic places of the environment where it operates. *Roman Tutor* uses a robot path-planner, not to control the manipulator, but to automatically check errors of a student learning to operate the manipulator, and to automatically produce illustrations of good and bad motions in training.

Introduction

Designing software that teaches requires, in advanced cases, the implementation of "intelligence" capabilities. After all, best human teachers are those mastering the subject they teach, having communication skills and understanding the student's solving process in order to help him. However, as we still do not understand well how to model human cognition, efforts in the design of intelligent software-based education systems remain experimental. In this line of inquiry, we have been developing an intelligent tutoring software, *Roman Tutor*, to support astronauts in learning how to operate the Space Station Remote Manipulator System (SSRMS), an articulated robot arm mounted on the international space station (ISS). Figure 1 illustrates a snapshot of the ISS with the SSRMS.

The SSRMS is operated from a robotic workstation located inside one of the ISS modules, and equipped with three video monitors, each displaying a view from one of the 14 cameras mounted on the ISS exterior and the SSRMS. Crewmembers operating the SSRMS have no direct view of the ISS exterior other than the three monitors. In fact, choosing the right camera views to display is part of the tasks for operating the manipulator.

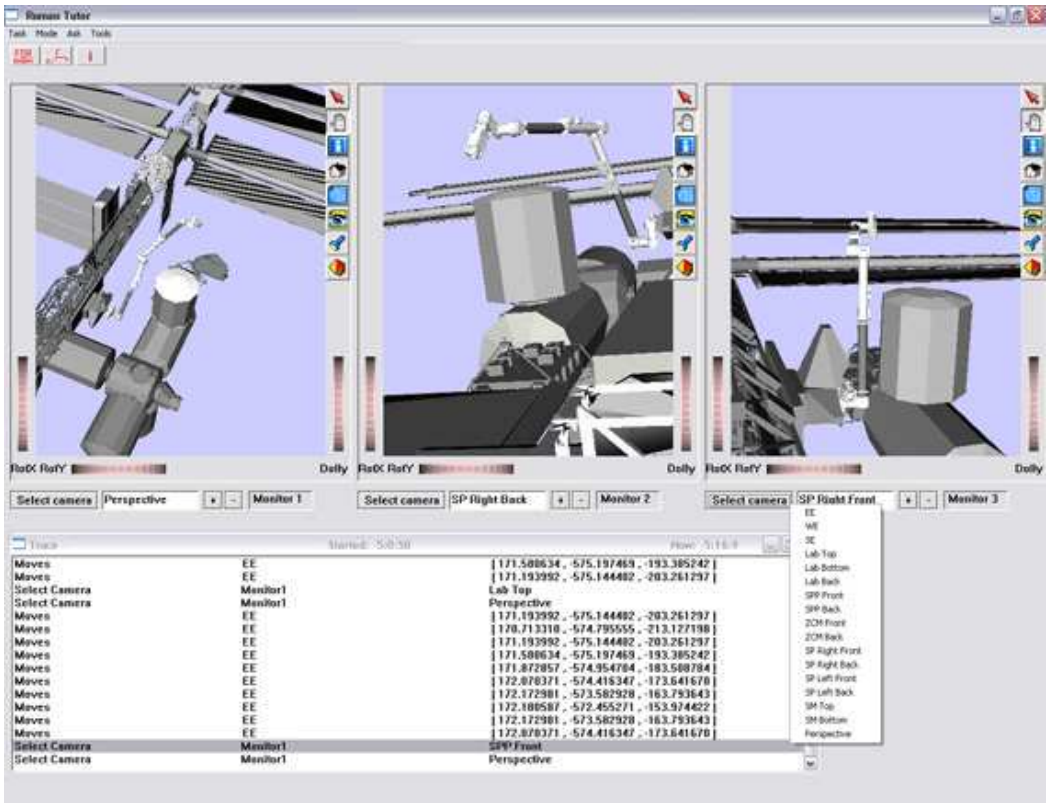
Even though they are supported by on-ground operators and have precise safety protocols to follow during their operations, training provides crewmembers with (1) manipulator teleoperation skills, (2) the ability to carry out operations independently (i.e., without ground support) and (3) detailed knowledge of SSRMS design that is required for the tasks scheduled for their mission and SSRMS operation. Consequently the associated training is challenging due to the limited direct view of the ISS exterior, unpredictable lighting conditions, the complexity of the SSRMS (seven degrees of freedom), high masses, costly payloads, and vulnerable robot mechanics requiring very slow speeds to avoid overruns and to reduce risks of oscillations, collisions or singularities.

Figure 1 ISS with the SSRMS

Roman Tutor is still under development; at ICAPS, we would like to demonstrate its current features that include a path-planner called FADPRM and used to provide tutoring feedback to the student. To illustrate, when an astronaut is learning to move a payload, *Roman Tutor* invokes the FADPRM path-planner periodically to check whether there is a path from the current configuration to the target, and provides feedback accordingly.

FADPRM path-planner not only computes collision free paths among obstacles, as is normal for a classic path-planner, but is also capable of taking into account the lim-

Figure 2 Roman Tutor Student Interface



ited direct view of the ISS, the lighting conditions and other safety constraints about operating the SSRMS. We developed such a robot-path planner by extending the probabilistic roadmap method (Sanchez & Latombe 2001) with constraints on safe and unsafe corridors of operation. In addition this planner proves useful in the automatic generation of movies that illustrate good and bad operations.

In the next section we describe *Roman Tutor*'s architecture, outline its basic functionalities and discuss details about its main components, mainly the FADPRM path-planner. We, then, conclude with a discussion on related work.

Architecture and Basic Functionalities

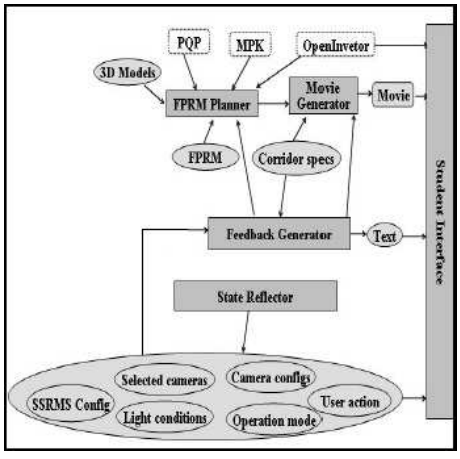
Main Components

One challenge in developing a good training simulator is of course to have the simulator in the first place. Then we need to integrate useful training strategies, in particular a model for tracing the student to assess his progress on a task and a process to provide really useful, intelligent feedback.

Roman Tutor works with any robot manipulator provided a 3D model of the robot and its workspace are specified. The system includes the following components among others (Figure 3): a graphic user interface, a feedback generator, a path planner, a movie generator, and third-party libraries: Proximity Query Package (PQP) (Larsen *et al.* 2000), Open Inventor from Silicon Graphics, and Motion Planning Kit (MPK) (Sanchez & Latombe 2001).

A snapshot of the user interface is shown on Figure 2. It emulates the ISS robotic workstation using three screens (for the three monitors). The keyboard is used to operate the robot (the SSRMS in our case). In *command mode*, one controls the joints directly; in *automatic mode*, one moves the end-effector, small increments at a time, relying on inverse kinematics to calculate the joint rotations.

Figure 3 Roman Tutor Architecture



In Figure 2, different cameras are selected, displaying the same robot configuration with different views. The perspective camera (on the left) can inspect the entire ISS 3D model. It is used in training tasks aimed at helping a student to develop a mental 3D model of the ISS, but it's not physically available on the ISS. Normal training uses replica models of the ISS for the same purpose.

The robot free workspace is segmented into zones with each zone having an associated degree of desirability, that is, a real number in the interval [0 1], depending on the task, visual cue positions, camera positions, and lighting conditions. The closer the *dd* is to 1, the more the zone is desired. Safe corridors are zones with *dd* near to 1, whereas unsafe corridors are those with *dd* in the neighborhood of 0.

The *state reflector* periodically updates the student's actions (i.e, keyboard inputs) and their effects on the ISS environment (robot configuration, cameras mapped to the monitors, their view angles, and the operation mode). It also monitors lighting conditions.

The *feedback generator* periodically checks the current state to trigger feedback to the student, using rules that are preconditioned on the current state information and the current goal. These are "teaching" expert rules and can be as efficient as the available teaching expertise allows. The *feedback generator* also changes the lighting conditions based upon specification rules in the current state.

Training Tasks

Students could carry out on *Roman Tutor* several kinds of training tasks, which can be classified as open, recognition, localization, or robot manipulation. Open tasks are those in which the learner interacts with the simulator, without any formally set goal, with minimal assistance configured in the system's preferences (e.g., collision warning and avoidance).

Recognition tasks train to recognize the different elements in the workspace. An example is to show a picture of an element of the ISS and ask the student to name it and describe its function.

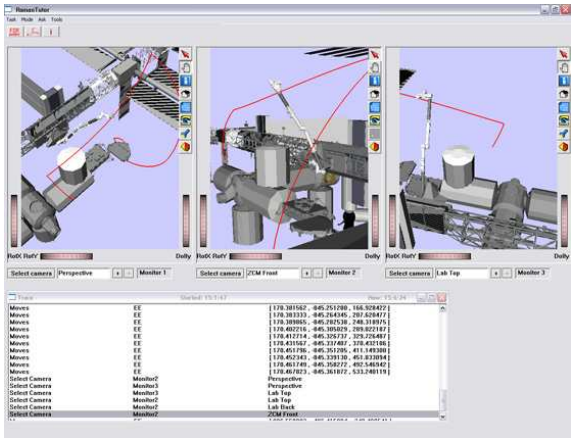
Localization tasks train to locate visual cue or ISS elements and to relate them spatially to each others. An example is to show a picture of a visual cue and ask the student to make it visible on the screen using an appropriate selection of cameras; or we can ask to name elements that are above another element shown on the screen.

Robot operation tasks deal with moving the manipulator (avoiding collision and singularities, using the appropriate speed, switching cameras as appropriate, and using the right operation mode at different stages), berthing, or mating. An illustration is to move the arm from one position to another, with or without a payload. Another example is to inspect an indicated component of the ISS using a camera on the end-effector. These tasks require the student to be able to define a corridor in a free workspace for a safe operation of the robot and follow it. The student must do this based on the task, the location of cameras and visual cues, and the current lighting conditions. Therefore localization and navigation are important in robot operations.

Tasks are made more or less unpredictable by dynamically changing the lighting conditions, thus requiring the revalidation of safe corridors. Feedback rules can take into account how long the student has been trying on a subtask and how good or bad he is progressing on it.

As most complex tasks deal in one way or another with moving the SSRMS, for the simulator to be able to understand students' operations in order to provide feedback, it must be aware of the space constraints and be able to move the arm by itself. A path-planner inside *Roman Tutor* that calculates arm's moves without collision and consistent with best available cameras views is then the key training resource on which other resources and abstract tutoring processes hinge. Figure 4 illustrates a solution path computed by the FADPRM path-planner given to the student on the *Roman Tutor* user interface to help him carry on his task.

Figure 4 A Path from FADPRM



The FADPRM path-planner

For efficient path planning, we pre-process the robot workspace into a roadmap of collision-free robot motions in regions with highest desirability degree. More precisely, the roadmap is a graph such that every node n in the graph is labeled with its corresponding robot configuration n.q and its degree of desirability *n.dd*, which is the average of *dd* of zones overlapping with *n.q.* An edge *(n,n')* connecting two nodes is also assigned a *dd* equal to the average of *dd* of configurations in the path-segment *(n.q,n'.q)*. The *dd* of a path (i.e., a sequence of nodes) is an average of *dd* of its edges.

The calculation of a configuration *dd* and a path *dd* is a straightforward extension of collision checking for configurations and path segments. For this, we customized the Proximity Query Package (PQP) (Larsen *et al.* 2000). The 3D models for the ISS, the SSRMS and zones are implemented using a customization of Silicon Graphics' Open Inventor. The robot is modeled using Motion Planning Kit (MPK), that is, an implementation of Sanchez and Latombe's PRM planner (Sanchez & Latombe 2001).

Following probabilistic roadmap methods (PRM) (Sanchez & Latombe 2001), we build the roadmap by

picking robot configurations probabilistically, with a probability that is biased by the density of obstacles. A path is then a sequence of collision free edges in the roadmap, connecting the initial and goal configuration. Following the Anytime Dynamic A* (AD*) approach (Likhachev *et al.* 2005), to get new paths when the conditions defining safe zones have dynamically changed, we can quickly re-plan by exploiting the previous roadmap. On the other hand, paths are computed through incremental improvements so the planner can be stopped at anytime to provide a collision-free path and the more time it is given, the better the path optimizes moves through desirable zones.

Therefore, our planner is a combination of the traditional PRM approach (Sanchez & Latombe 2001) and AD* (Likhachev *et al.* 2005) and it is flexible in that it can into account zones with degrees of desirability. We call it Flexible Anytime Dynamic PRM (FADPRM). More precisely, FADPRM works as follows. The input is: an initial configuration, a goal configuration, a 3D model of obstacles in the workspace, a 3D specification of zones with corresponding *dd* , and a 3D model of the robot. Given this input:

- To find a path connecting the input and goal configuration, we search backward from the goal towards the initial (current) robot configuration. Backward instead of forward search is done because the robot moves, hence its current configuration, is not necessarily the initial configuration; we want to re-compute a path to the same goal when the environment changes before the goal is reached.
- A probabilistic queue *OPEN* contains nodes of the frontier of the current roadmap (i.e., nodes are expanded because they are new or because they have previously been expanded but are no longer up to date w.r.t. to the desired path) and a list *CLOSED* contains non frontier nodes (i.e., nodes already expanded).
- Search consists of repeatedly picking a node from *OPEN*,generating its predecessors and putting the new ones or out of date ones in OPEN.
- The density of a node is the number of nodes in the roadmap with configurations that are a short distance away (proximity being an empirically set parameter, taking into account the obstacles in an application domain). The distance estimate to the goal takes into account the node's *dd* and the Euclidean distance to the goal. A node n in *OPEN* is selected for expansion with probability proportional to :

  $$(1-\beta)/density(n)+\beta*goal-distance-estimate(n)$$
  $$with\ 0 \leq \beta \leq 1.$$

  This equation implements a balance between fast-solution search and best-solution search by choosing different values for β. With β near to 0, the choice of a node to be expanded from *OPEN* depends only on the density around it. That is, nodes with lower density will be chosen first, which is the heuristic used in traditional PRM approaches to guaranty the diffusion of nodes and to accelerate the search for a path (Sanchez & Latombe 2001). As β approaches 1, the choice of a node to be expanded from *OPEN* will rather depend on its estimated distance to the goal. In this case, we are seeking optimality rather than speed.
- To increase the resolution of the roadmap, a new predecessor is randomly generated within a small neighborhood radius (that is, the radius is fixed empirically based on the density of obstacles in the workspace) and added to the list of successors in the roadmap generated so far. The entire list predecessors is returned.
- Collision is delayed: detection of collisions on the edges between the current node and its predecessors is delayed until a candidate solution is found; if there is a collision, we backtrack. Collisions that have already been detected are stored in the roadmap to avoid doing them again.
- The robot may start executing the first path found.
- Concurrently, the path continues being improved by replanning with an increased value of β.
- Changes in the environment (moving obstacles or changes in *dd* for zones) cause updates of the roadmap and replanning.

Conclusion

The current prototype of *Roman Tutor* includes a new path-planner called FADPRM to handle workspaces with desirable and undesirable zones, and basic tutoring feedbacks. When completed, potential benefits to future organizational training strategies are (1) the simulation of complex tasks at a low cost (e.g., using inexpensive simulation equipment and with no risk of injuries or equipment damage) and (2) the installation anywhere and anytime to provide "just in time" training. Crewmembers will be able to use it onboard of the ISS to comprehend a workaround for a repair. For very long missions, they will use it to train regularly in order to maintain their skills. We also believe that ideas behind *Roman Tutor* could inspire training systems for other applications of robot manipulators.

Acknowledgment

We are grateful to the Canadian Space Agency, The Infotel Group and the Natural Sciences and Engineering Research Council of Canada (NSERC) for their logistic and financial support. We also thank the Robotics Laboratory at the University of Stanford for making MPK available to us, and Mitul Saha for help on MPK.

References

Larsen, E.; Gottschalk, S.; Lin, M.; and Manocha, D. 2000. Fast distance queries using rectangular swept sphere volumes. In *IEEE International Conference on Robotics and Automation*, 4:24–28.

Likhachev, M.; Ferguson, D.; Gordon, G.; Stentz, A.; and Thrun, S. 2005. Anytime dynamic a*: An anytime, replanning algorithm. In *International Conference on Automated Planning and Scheduling*.

Sanchez, G., and Latombe, J.-C. 2001. A single-query bi-directional probabilistic roadmap planner with lazy collision checking. In *Ninth International Symposium on Robotics Research*.

ASTRO: Supporting Composition and Execution of Web Services*

M.Trainotti M.Pistore

University of Trento
Via Sommarive 14
38050 Povo (Trento) - Italy

G.Calabrese G.Zacco G.Lucchese
F.Barbon P.Bertoli P.Traverso

ITC-Irst
Via Sommarive 18
38050 Povo (Trento) - Italy

Abstract

Web services are rapidly emerging as the reference paradigm for the interaction and coordination of distributed business processes. In several research papers we have shown how advanced automated planning techniques can be exploited to automatically compose web services, and to synthesize monitoring components that control their execution. In this demo we show how these techniques have been implemented in the ASTRO toolset (http://www.astroproject.org), a set of tools that extend existing platforms for web service design and execution with automated composition and execution monitoring functionalities.

Introduction

Web services are rapidly emerging as the reference paradigm for the interaction and coordination of distributed business processes. The ability to automatically plan the composition of web services, and to monitor their execution, is therefore an essential step toward the real usage of web services.

In previous works (1; 2; 3), we have shown how automated planning techniques based on the "Planning via Model Checking" paradigm can effectively support these functionalities. More precisely, the algorithms proposed in (1; 2; 3) are based on web service specifications described in BPEL4WS, a standard language that can be used both for describing existing web services in terms of their interfaces (i.e., of the operations that are needed to interact with them) and for defining the executable code that implements composite services.

Automated web service composition starts from the description of a number of protocols defining available external services (expressed as BPEL4WS specifications), and a "business requirement" for a new composed process (i.e., the goal that should be satisfied by the new service, expressed in a proper goal language). Given this, the planner must synthesize automatically the code that implements the internal process that, exploiting the services of the external partners, achieves the business requirement. This code is then emitted as executable BPEL4WS code.

The automated synthesis techniques provided by the "Planning via Model Checking" framework can be also exploited to generate process monitors, i.e., pieces of code that detect and signal whether the external partners behave consistently with the specified protocols. This is vital to detect unpredictable run-time misbehaviors (such as those that may originate by dynamic modifications of the partners' protocols), or other events in the executions of the web services that need to be reported and analyzed.

Notice that these problems require to deal with nondeterminism (since the behavior of external services cannot be foreseen a priori), partial observability (since their status is opaque to the composed service), and extended goals (since realistic business requirements specify complex expected behaviors rather than just final states). By tackling the problem of composing and monitoring web services, we have shown the capabilities of the "Planning via Model Checking" approach in realizing such a complex planning task.

In this demo we show how these techniques can extend existing commercial platforms for web service design and execution. More precisely, we describe the ASTRO toolset (http://www.astroproject.org), which implements automated composition and monitor generation functionalities as extensions of the Active WebFlow platform. Active WebFlow (http://www.activebpel.org/) is a commercial tool for designing and developing BPEL4WS processes which is based on the Eclipse platform. It also provides an open-source BPEL4WS execution engine, called Active BPEL. By implementing automated composition and monitoring within Active WebFlow, these advanced functionalities can be combined with the other "standard" functionalities provided by the platform (such as inspecting BPEL4WS code, writing or modifying business processes, deploying these processes and executing them) and become integral part of the life cycle of business process design and execution.

The rest of the paper is structured as follows. We start with the description of a service composition scenario which is used to illustrate the proposed approach.

*This work is partially funded by the MIUR-FIRB project RBNE0195K5, "Knowledge Level Automated Software Engineering", and by the MIUR-PRIN 2004 project "Advanced Artificial Intelligence Systems for Web Services".

Then we describe the architecture and the functionalities of the ASTRO toolset. Finally, we present a demonstration of the application of this toolset to the reference composition scenario.

A service composition scenario

The demo is based on a classical web service composition problem, namely that of the Virtual Travel Agency (VTA). It consists in providing a combined flight and hotel booking service by composing two separate, independent existing services: a Flight booking service, and a Hotel booking service.

The Hotel booking service becomes active upon a request for a room in a given location (e.g., Paris) for a given period of time. In the case the booking is not possible (i.e., there are no available rooms), this is signaled to the request applicant, and the protocol terminates with failure. Otherwise, the applicant is notified with information about the hotel (e.g., Hilton), cost of the room, etc. and the protocol stops waiting for either a positive or negative acknowledgment. In the first case, an agreement has been reached and the room is booked. In the latter case, the interaction terminates with failure.

The protocol provided by the Flight booking service is similar. It starts upon a request for flights that guarantee to stay in a given location (e.g., Paris) for a given period of time. This might not be possible, in which case the applicant is notified, and the protocol terminates failing. Otherwise, information on the flights (carrier, cost, schedule...) are computed and returned to the applicant. The protocol suspends for either a positive or negative acknowledgment, terminating (with success or failure resp.) upon its reception.

The expected protocol that the user will execute when interacting with the VTA goes as follows. The user sends a request to stay in a given location during a given period of time, and expects either a negative answer if this is not possible (in which case the protocol terminates, failing), or an offer indicating hotel, flights and cost of the trip. At this time, the user may either accept or refuse the offer, terminating its interaction in both cases.

Of course several different interaction sequences are possible with these services; e.g., in a *nominal* scenario, none of the services answers negatively to a request; in non-nominal scenarios, unavailability of suitable flights or rooms, as well as user refusals, may make it impossible to reach an agreement for the trip. Taking this into account, the business requirement for the composed service is composed of two subgoals. The "nominal" subgoal consists in reaching the agreement on flights and room. This includes enforcing that the data communicated to the various processes are mutually consistent; e.g., the number of nights booked in the hotel depends on the schedule of the selected flights. The "recovery" subgoal consists in ensuring that every partner has rolled back from previous pending requests, and must be only pursued when the nominal subgoal cannot be achieved anymore.

By automated composition of the VTA process, we mean the automated generation of the code that has to be executed on the VTA server, so that requests from the user are answered combining the Flight and Hotel services in a suitable way. This composition has to implement the two sub-goals described above.

After the VTA process has been generated, its executions must be monitored, in order to detect problems in the interactions with the other partners participating to the scenario. Properties to be monitored include "correctness" checks (e.g., the partners obey the declared protocols; the flight schedules are compatible with the requests...). It is also possible to monitor "business" properties, such as the fact that, when an offer for a trip is sent to the user, this offer gets accepted or not.

The ASTRO toolset

This section presents a general overview of the ASTRO toolset. At the current stage, it consists of the following tools: WS-gen, WS-mon, WS-console and WS-animator.

WS-gen is responsible for generating the automated composition. It consists in a back-end layer and a front-end layer. The back-end layer takes as input the BPEL4WS specifications of the interaction protocols that the composite service has to implement, a "choreographic" file describing the connections between the composition's partners, and a goal file defining the composition requirement. It consists of two applications (see Fig.1): BPELTranslator converts the BPEL4WS specification files and the choreography file in an intermediate (.smv) file which is adequate for representing "Planning via Model Checking" problems; WSYNTH takes as input the problem domain, computes the plan which fulfills the requirements, and emits the plan in BPEL4WS format. The front-end (see Fig.2) is responsible for controlling the composition process and for managing the generated BPEL4WS specification; it has been implemented as an Eclipse plugin, and is hence integrated in the Active WebFlow environment.

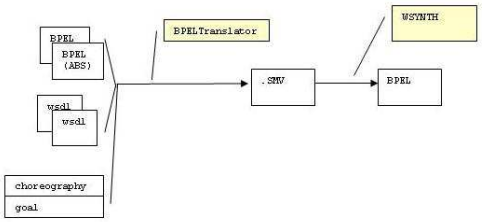


Fig. 1: WS-gen architecture

WS-mon is responsible for generats the Java code implements the monitors for the composed process and deploying them to the monitor framework. Similar to WS-gen, it consists in a back-end layer and a front-end

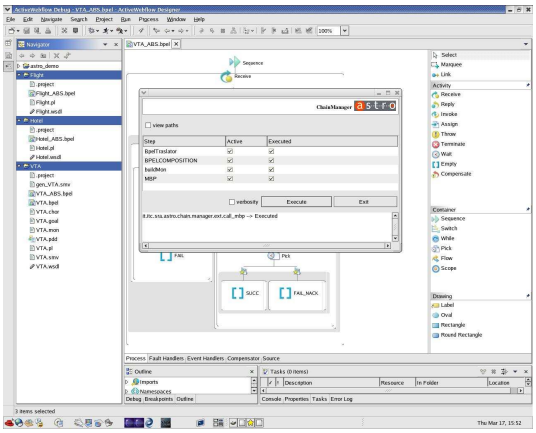


Fig. 2: WS-gen front end

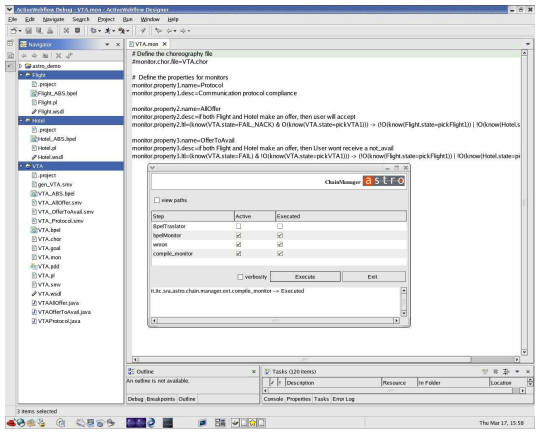


Fig. 4: WS-mon front end

layer. The back-end takes as input BPEL4WS specifications and a "choreographic" file, while the goal file is replaced by a file specifying the properties to by monitored. The back-end layer consists in three applications (see Fig.3): BPELTranslator, which is in common with WS-gen, converts the BPEL4WS specification files and the choreography file in a .smv file which describes the problem domain; WMON takes as input the problem domain, computes the plan which fulfills the monitoring requirements, and emits this plan in Java format; and the DEPLOYER compiles the Java class and deploy them to the monitor framework. The front-end (see

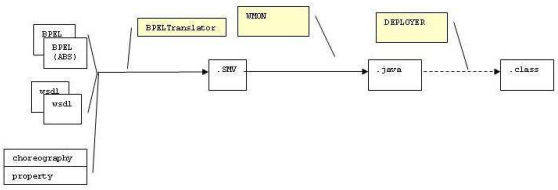


Fig. 3: WS-mon architecture

Fig.4), which is responsible for controlling the monitor generation process, has been implemented as an Eclipse plugin, and is hence integrated in the Active WebFlow environment.

The run-time monitor framework is responsible for executing the monitors associated to a given process every time an instance of that process is executed. It is also responsible for reporting the status of these monitors to the user in a convenient way. It consists of a back-end layer and a front-end layer (see Fig.5). The back-end layer has been implemented as an extension of the Active BPEL execution engine; the main goal is to sniff the input/output messages directed to the process that has to be monitored and to forward them to the Java monitors instances. The front-end implementation, WS-console, extends the Active BPEL administration console in order to present the status of the

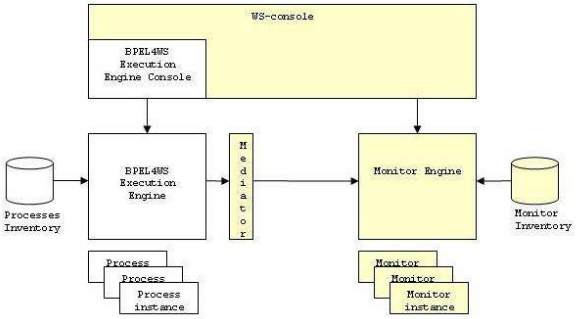


Fig. 5: Monitor framework architecture

monitors associated with each process instance. In this way, violations of the monitored properties are easy to be checked by the user (see Fig.6).

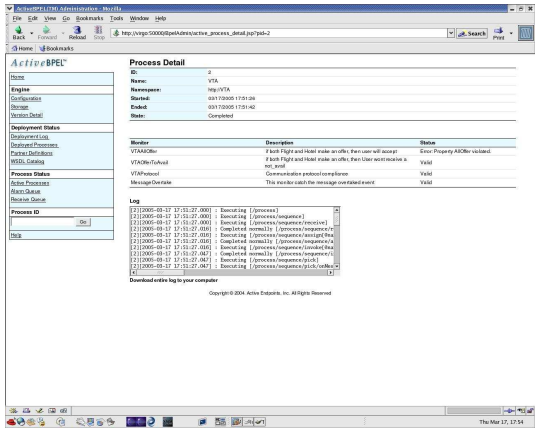


Fig. 6: WS-console

Finally, WS-animator (see Fig.7) is another Eclipse plugin, which gives the user the possibility to "execute" the composite process (in our case, the VTA). More precisely, it allows the user to play the roles of the actors

interacting with the composite process, while the Active WebFlow engine executes it.

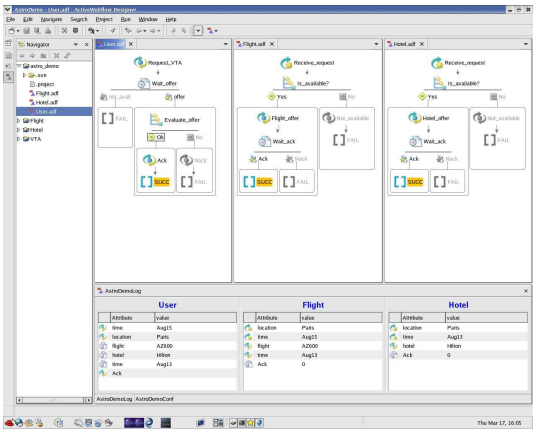


Fig. 7: WS-animator

The DEMO

In this section we describe a demonstration of the capabilities of the ASTRO toolset. The demo consists of a set of steps corresponding to the execution of service composition and a monitor synthesis task (see Fig.8).

Step 1. Within Active WebFlow, the user selects the projects (Flight, Hotel, VTA) which are part of the composition scenario. These projects contain the WSDL and the abstract BPEL4WS files describing the interfaces of the existing web services (and the protocol that the VTA has to expose to the end user). Moreover, the VTA project contains the .goal, .mon and .chor configuration files; those files define the requirements and choreography for the process and monitor composition.

Step 2. WS-gen is invoked. After the generation is terminated, the left panel gives a glimpse of the generated files; in particular, the composed process VTA.bpel is ready to be deployed to the BPEL4WS execution engine.

Step 3. The composed process is deployed into the Active BPEL execution engine via the Active WebFlow console; now the composed process is ready to receive the client requests.

Step 4. After the composition and deployment of the BPEL4WS process, WS-mon is used to generate the associated monitors. The left panel gives a glimpse of the generated files, and in particular the Java files implementing the monitor processes.

Step 5. To test the generated service, User, Flight and Hotel processes are executed in WS-animator, while the composed process is executed in Active BPEL execution engine. This configuration gives the possibility to test the composed process controlling the execution of the partner processes.

Step 6. After the execution of a nominal scenario within WS-animator, all the services end in a SUCCESS state. In this scenario, the user has request an offer to the composite service for a flight and a hotel specifying a date and a location. The Flight has received the flight request from the composite service, checked for its availability and sent back the flight number and date. The Hotel has received the request for an hotel reservation for a date (the one sent by the Flight) from the composite service, checked for its availability and sent back the hotel. The user has received the offer from the composite service and has accepted it.

Step 7. WS-console presents the states of the monitors for the instance of the VTA service corresponding to the nominal scenario presented above. All the monitor instances are valid.

Step 8. After the execution of a scenario where the User refuses a travel offer, WS-animator shows all the services terminated in a FAIL state. In this scenario, the user has requested an offer to the composite service for a flight and a hotel specifying a date and a location. The Flight has received the flight request from the composite service, checked for its availability and sent back the flight number and date. The Hotel has received the request for an hotel reservation for a date (the one sent by the Flight) from the composite service, checked for its availability and sent back the hotel. The user has received the offer from the composite service and has denied it. The denial has been forwarded to Flight and Hotel.

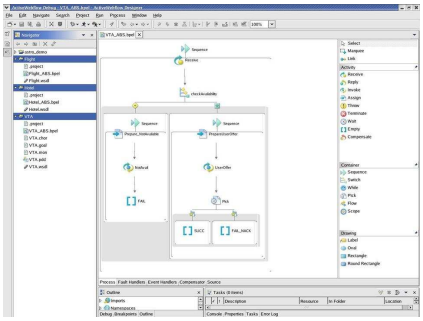
Step 9. WS-console presents the states of the monitors for the instance of the VTA service of the scenario where the user refuses the travel offer. This scenario violates one of the monitored properties, namely "if both Flight and Hotel make an offer, the user will accept it". This violation is reported in WS-console.

References

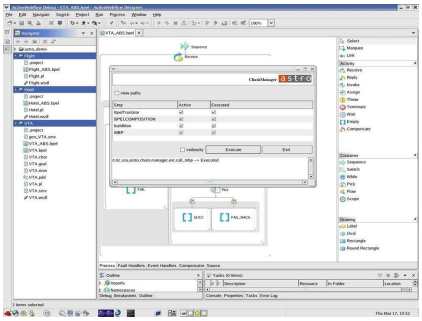
[1] Pistore, M.; Barbon, F.; Bertoli, P.; Shaparau. D.; and Traverso, P. 2004. Planning and Monitoring Web Service Composition. In *Proc. AIMSA'04*.

[2] Pistore, M.; Traverso, P.; and Bertoli, P. 2005. Automated Composition of Web Services by Planning in Asyncronous Domains. In *Proc. ICAPS'05*.

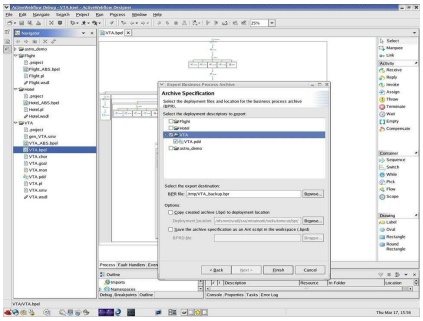
[3] Pistore, M.; Marconi, A.; Bertoli, P.; and Traverso, P. 2005. Automated Composition of Web Services by Planning at the Knowledge Level. In *Proc. IJCAI'05*.



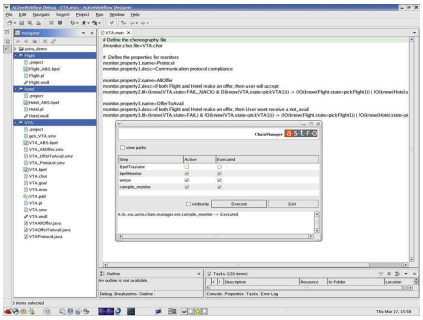
Step 1



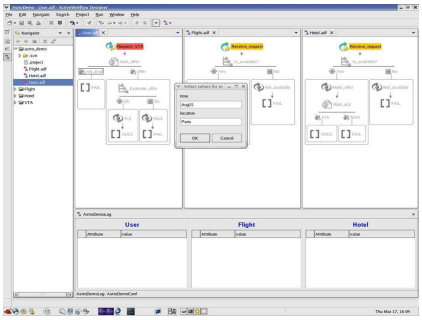
Step 2



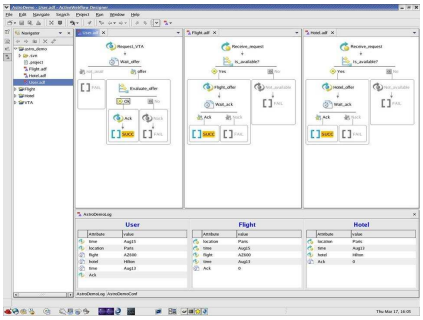
Step 3



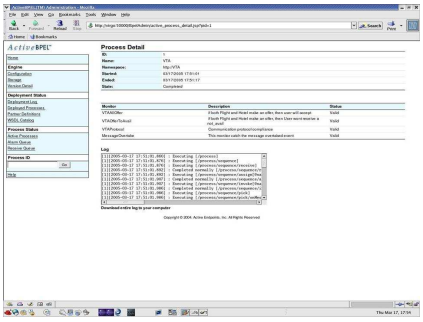
Step 4



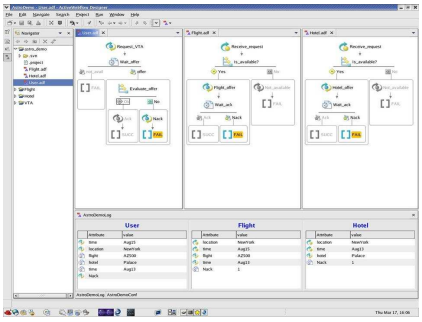
Step 5



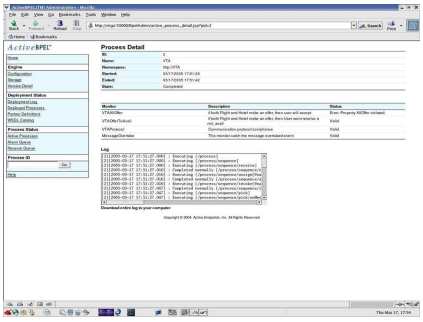
Step 6



Step 7



Step 8



Step 9

Fig. 8: The demo

SIADEX. An integrated planning framework for crisis action planning*

L. Castillo, J. Fdez-Olivares, O. García-Pérez and F. Palao
Dept. of Computers Science and Artificial Intelligence
ETSI Informática, University of Granada
18071, Granada, SPAIN
{L.Castillo,Faro,Oscar,Palao}@decsai.ugr.es
Phone:+34.958.240803, +34.958240805

Abstract

SIADEX is an integrated framework to support decision making during crisis episodes by providing realistic temporally annotated plans of action. The main component of SIADEX is a forward state-based HTN temporal planner.

Introduction

The design of plans of activity for crisis situations is a very sensitive field of application for mature AI planning and scheduling techniques (Allen *et al.* 1995; Myers 1999; Biundo & Schattenberg 2001; Avesani, Perini, & Ricci 2000). However, a successful approach requires a subtle integration of several research and development issues like

- Integration of several technologies. These systems are not usually a monolithic approach, but a composition of technologies that integrate with each other with different functionalities like planning (to determine the appropriate set of activities), scheduling (to handle time and resources), pathfinding (to find optimal movement plans in complex networks), etc.
- Enhancing the role of end-users. End users of these systems are not expected to have a background knowledge on AI, therefore the system must use user-friendly interfaces in order for end-users to establish goals, to understand what the system does and what the system is demanding without having to use a technical language like a planning domain description language or a constraint programming language.
- Flexible knowledge representation. The system must represent a large amount of data coming from heterogenous sources of information like GPS locations of resources, facilities, legal issues that constraint the activities (for example, contracting conditions or durations of shifts), exogenous events (i.e., meteo conditions, day and night periods), and many more. Even more, although this might seem simple it is a very important issue, the system must access to all this information on-line, that is, extracting it from legacy databases and translating them into known planning and scheduling domain description languages.
- Support of distributed and concurrent access of end-users. Usually, these systems are operated in hostile environments like a forest fire, natural disasters scenarios, etc, and most of the inputs come from (and most of the outputs are directed to) end-users located at these places. These systems are too complex as to be installed and run on small devices with limited computation capability like a laptop or a PDA, therefore providing a centralized high-capability computing facility with full connectivity and accessibility to end users is a valuable feature.
- Integration with legacy software. Not only the income but also the outcome must be redirected to legacy software so that end users may painlessly understand, process and deliver activity plans. In this case a user-friendly output to GIS or project management and monitoring is strongly required.
- Quick response. Last, but not least, the system must be very efficient so that it obtains a response in an acceptable time with respect to the own latency time of the crisis situation (that may range from minutes to hours).

The architecture of SIADEX

SIADEX is an open problem solving architecture based on the intensive use of web services to implement most of its capabilities (Figure 1). Its main components are the knowledge base (named BACAREX), that stores in Protege ((National Library of Medicine)) all the knowledge that would be useful for the planning engine, and the planning module (named SIADEX), the core of the architecture in charge of building fire fighting plans.

The planning algorithm and its knowledge representation are built as two independent modules, which are accessible from any device with internet connectivity (a desktop computer, a laptop or a PDA). This allow users to query or modify the state of the world by using almost any existing web browser or by using a well known GIS software (ESRI), recall that during a crisis episode most of the objects and resources are associated have geographical properties (see Figure 2). In the same way, temporal plans designed by SIADEX may be downloaded into some project management software in the form of Gantt charts (see Figure 3) or any other software commonly used by technical staff. The basic process is as follows.

*This work is being funded by the Andalusian Regional Ministry of the Environment, under research contract NET033957

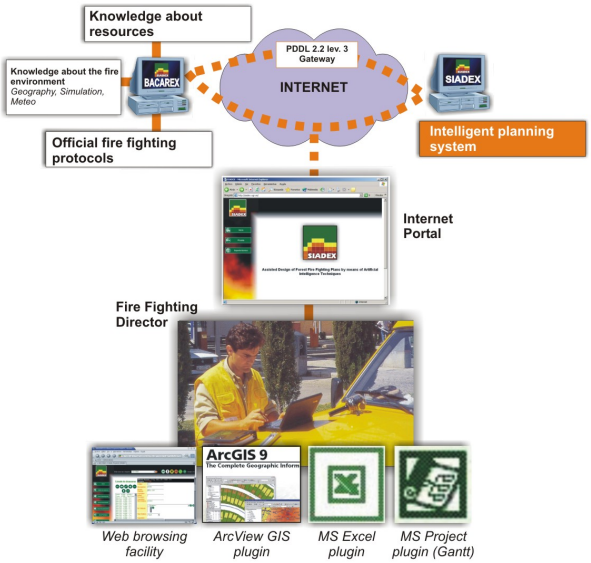


Figure 1: Architecture of SIADEX

Describing the problem The fire fighting scenario is introduced by the technical staff consisting of the targets areas, the general attack procedures and an estimation of the number of resources to be used. This may be done by the web browsing utility or, much more easily, by the ArcView GIS software plugin (ESRI).

Storing the scenario The fire fighting scenario is stored in Protege format in BACAREX. Therefore, knowledge about resources and fire scenario share the same representation. All this information is visible to other users by means of the web browsing facility, although only the knowledge about the problem may be accessed (the knowledge about the domain, tasks and actions, is only visibile for the development team).

Requesting a plan The planning engine is not able to read the domain and the problem stored in Protege, therefore a PDDL Gateway has been implemented that translate problem and domain into PDDL 2.2 level 3 (Edelkamp & Hoffmann 2004). After that, the planning engine is called and a plan is obtained (or not).

Displaying the plan The plan obtained may be displayed in a number of "user-friendly" alternatives like Microsoft Excel, in the form of a chronogram, or Microsoft Project in the form of a Gantt chart (see Figure 3).

Plan execution and monitoring (in development) The plan may be launched for execution, distributed amongst all the technical staff with some resposibility in the fire fighting episode, and concurrently monitored.

Domain knowledge

The knowledge about the planning objects (places, facilities, task forces, resources, etc) is stored in an ontology of the problem represented in Protégé, an ontology editor and

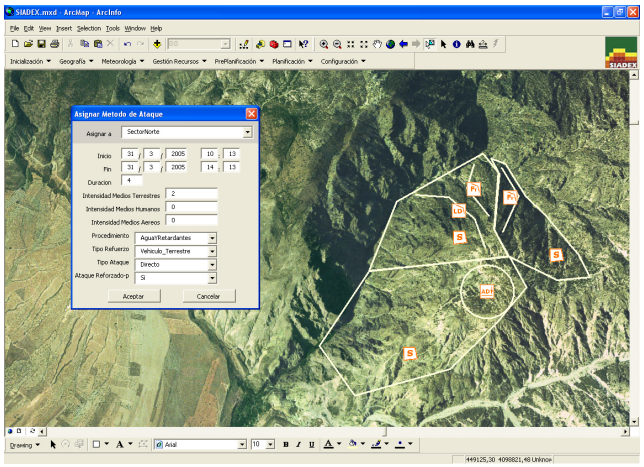


Figure 2: The ArcView plugin

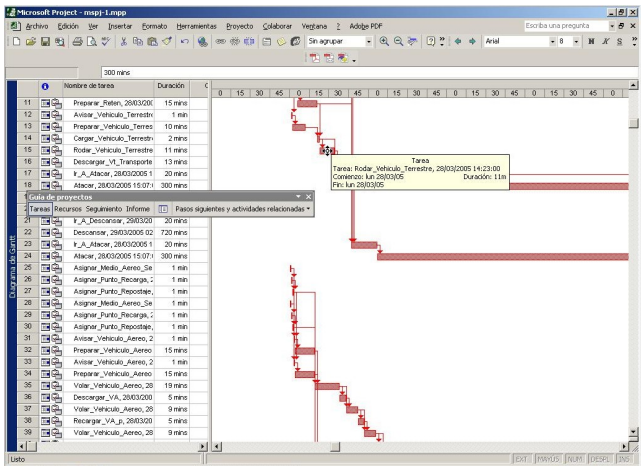


Figure 3: Gantt chart output

knowledge acquisition tool (National Library of Medicine). A web browsing tool has been designed so that end users may easily access to the hierarchy of objects, to query or modify their properties, without having to know anything related to knowledge representation[1]. This hierarchy of objects also supports the definition of the goal scenario (geographical targets, goal tasks, etc) either from a web browser or from a GIS software. Once a plan is requested by the user, the knowledge stored in this knowledge base is then translated into PDDL 2.2 level 3 (Edelkamp & Hoffmann 2004), with support for timed initial literals and derived predicates, following the next outline:

- Classes of the ontology are translated into a hierarchy of PDDL types.
- Instances are translated as typed planning objects (only the slots relevant for the planning process are translated).
- The domain is stored directly in the form of tasks, meth-

[1]The development team may also use the standard Protégé shell to run knowledge consistency checking and validation.

ods and actions compliant with PDDL 2.2 level 3, so it does not need to be translated.

- Other constraints of the problems are also translated accordingly like maximun legal duration of shifts (fluent), day/night events (timed initial literals), activity windows over the scenario (deadline goals), etc.

The planner

The planning module is a forward state-based HTN planning algorithm with the following features:

- Primitive actions are fully compliant with PDDL 2.2 with durative actions and numeric capabilities (Figure 4).
- It makes use of an extension of PDDL to represent timed HTN tasks and methods.
- SIADEX's domains also embed some functionalities to control and prune the search in order to make the planning process more efficient.
- SIADEX also supports the use of external functions calls by embedding Python scripts in the domain definition, to access external sources of information or perform complex computations during the planning process (Figure 5).

```
(:durative-action Refuel_Plane
 :parameters (?a - Aircraft ?p - Refueling_Point)
 :prettyprint "?start > Aircraft ?a starts refueling
               at ?p. Finishing at ?end"
 :duration (= ?duration (refueling_time ?a))
 :condition  (and
  (in_fire ?a ?fire)
  (over all (daylight ?fire))
  (GIS ?p ?gis)
  (current_position ?a ?gis))
 :effect (and
  (state ?a refuelling)
  (at end (assign (current_autonomy ?a)
          (max_autonomy ?a))))
)
```

Figure 4: PDDL 2.2 level 3 primitive actions

Temporal and resource reasoning

One of the most important features of SIADEX is that it allows a powerful handling of temporal knowledge. SIADEX's plans are built on top of a temporal constraint network (Dechter, Meiri, & Pearl 1991) that records temporal and causal dependencies between actions so that, although it is a state based process, plans may have a partial order structure with temporal references either qualitative or numeric. This allows SIADEX to obtain very flexible schedules (N. Policella 2004) that might be redesigned during the execution of the plan to adapt to unforeseen delays without the need to replan. In addition to this, SIADEX also supports the definition of constraints on the makespan of the plan and deadline goals over primitive and compound tasks

```
 (:functions
 (distance ?x1 ?y1 ?x2 ?y2)
{
 import math
 return math.sqrt ( (?x2 - ?x1) *
    (?x2 -?x1) +
    (?y2 - ?y1) *
    (?y2 - ?y1))
}
...
```

Figure 5: Embedded python in the domain

(in the case of compound tasks, deadline goals are inherited by its component tasks).

Since SIADEX handles numerical objects like PDDL fluents (that can also be dynamically linked to external Python calls) it achieves a basic handling of numeric resources.

References

Allen, J.; Schubert, L.; Ferguson, G.; Heeman, P.; Hwang, C.; Kato, T.; Light, M.; Martin, N.; Miller, B.; Poesio, M.; and Traum, B. 1995. The TRAINS project: a case study in building a conversational planning agent. *Experimental and Theoretical Artificial Intelligence* 7:7–48.

Avesani, P.; Perini, A.; and Ricci, F. 2000. Interactive case-based planning for forest fire management. *Applied Intelligence* 13(1):41–57.

Biundo, S., and Schattenberg, B. 2001. From abstract crisis to concrete relief - a preliminary report on combining state abstraction and htn planning. In *6th European Conference on Planning (ECP-01)*.

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

Edelkamp, S., and Hoffmann, J. 2004. The language for the 2004 international planning competition. http://ls5-www.cs.uni-dortmund.de/ edelkamp/ipc-4/pddl.html.

ESRI. http://www.esri.com.

Myers, K. L. 1999. CPEF: A continuous planning and execution framework. *AI Magazine* 20(4):63–69.

N. Policella, S. Smith, A. C. A. O. 2004. Generating robust schedules through temporal flexibility. In *14th International Conference on Automated Planning and Scheduling, ICAPS*.

National Library of Medicine. http://protege.stanford.edu/.

Subgoal Partitioning and Resolution in SGPlan*

Yixin Chen, Chih-Wei Hsu, Benjamin W. Wah
Department of Electrical and Computer Engineering
and the Coordinated Science Laboratory
University of Illinois at Urbana-Champaign
Urbana, IL 61801, USA
URL: http://manip.crhc.uiuc.edu/programs/SGPlan
Email: {chen, chsu, wah}@manip.crhc.uiuc.edu

Introduction

In this demo, we illustrate the idea and operations of SGPlan (Chen, Hsu, & Wah 2004; Wah & Chen 2005), a PDDL2.2 planner that won the first prize in the suboptimal temporal metric track and a second prize in the suboptimal propositional track in the Fourth International Planning Competition (IPC4), 2004. SGPlan is the only planner that won in two tracks of the competition. Since SGPlan is a suboptimal planner, it did not participate in the optimal track.

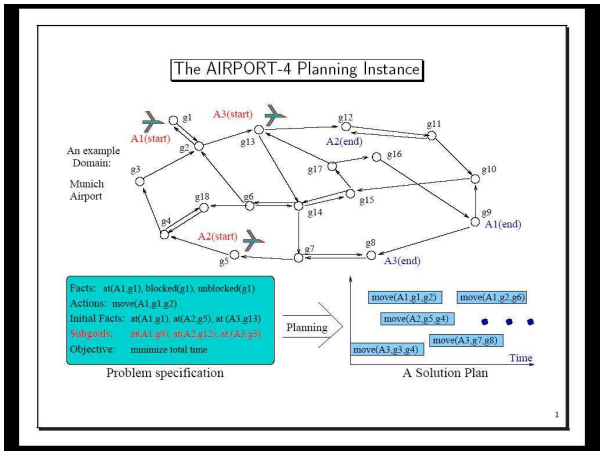
SGPlan was designed based on the key observation that many planning applications have a clustered structure of their constraints. Specifically, we have found that constraints are highly localized by their subgoals. Based on this observation, we have proposed to partition problem constraints by their subgoals into multiple subsets, solve each subproblem individually, and resolve inconsistent global constraints across subproblems based on a penalty formulation.

In this demo, we illustrate the key observation of constraint locality in some application domains, the constraint partitioning approach we have used in SGPlan, and the process of resolving inconsistent global constraints across partitioned subproblems.

The intended audience of this demonstration includes all researchers and practitioners in planning and scheduling, especially those who are interested in deterministic PDDL2.2 planning and scheduling. We will present the slides in the demonstration, discuss the technology with interested audience, and answer questions regarding the SGPlan system.

The executable of SGPlan is downloadable from our website (http://manip.crhc.uiuc.edu/programs/SGPlan). The system has been evaluated on all the test problems from IPC3 and IPC4. Complete evaluation results can be found in our website and a paper submitted to the Journal of Artificial Intelligence Research (Chen, Hsu, & Wah 2005).

*Research supported by National Science Foundation Grant IIS 03-12084 and National Aeronautics and Space Administration Grant NCC 2-1230.



Slide 1: The AIRPORT-4 Planning Instance.

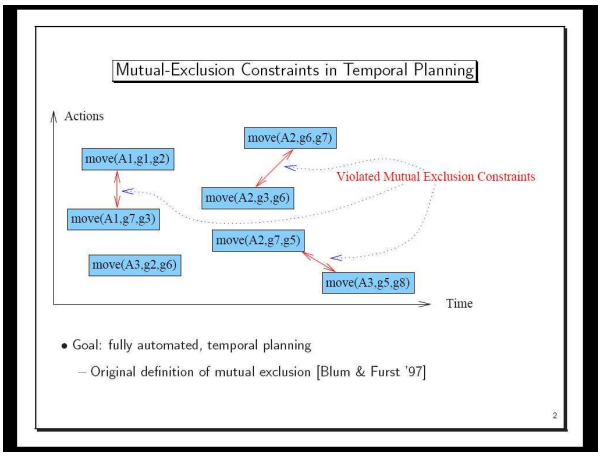
Observations on Constraint Locality

Our approach is based on the observations that the constraints in many planning applications are not purely random but highly structured, and that these constraints can be clustered by their subgoals.

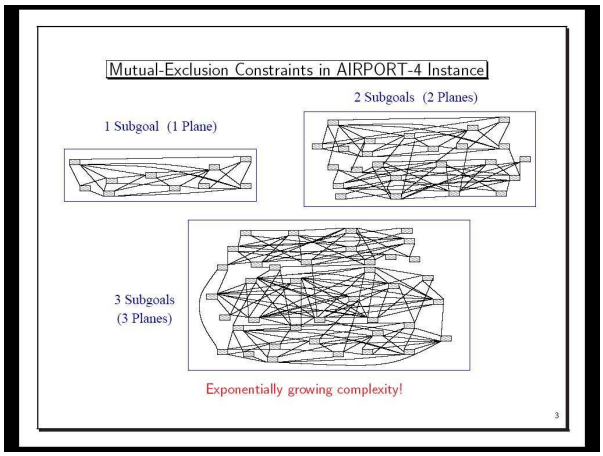
For example, in AIRPORT, an airport scheduling domain, each subgoal is a destination in an airport. Most mutual-exclusion constraints are localized within a subgoal relating two actions from the same subgoal, and only a small fraction of global constraints relate multiple subgoals. We illustrate this observation as follows.

Slide 1 illustrates the topology an airport in the AIRPORT-4 instance. The example involves a planning task for moving three airplanes from their starting positions to some destination gates. To apply a PDDL2.2 planning system to solve the problem, we first model this problem in the PDDL2.2 language. The PDDL2.2 model specifies the facts, actions, initial state, and goals of the planning problem. Note that in the goals include multiple subgoals, one for each airplane. A solution plan, shown in the right bottom part of Slide 1, is a temporal plan where actions have durations, and can have overlapping execution times.

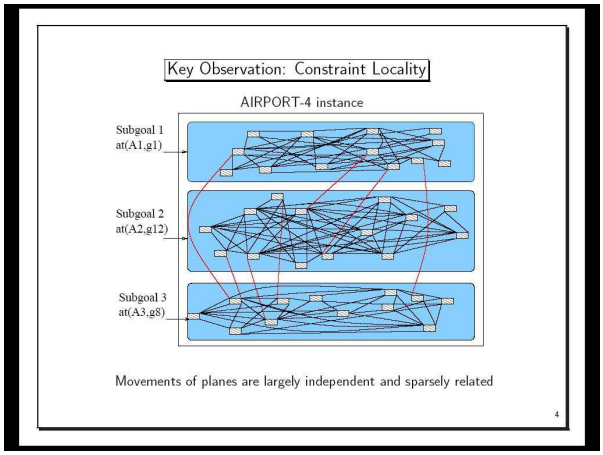
Slide 2 illustrates the constraints in a temporal planning problem whose actions have durations. Constraints



Slide 2: Mutual-Exclusion Constrains in Temporal Planning.



Slide 3: Mutual-Exclusion Constrains in AIRPORT-4 Instance.



Slide 4: Key Observation: Constraint Locality.

in PDDL2.2 temporal planning problems are mutual-exclusion constraints proposed in Graphplan (Blum & Furst 1997). We show examples of violated mutual-exclusion constraints in the AIRPORT-4 instance. For example, in the diagram in Slide 2, the two actions move(A1,g1,g2) and move(A1,g7,g3) are mutual exclusive and cannot be scheduled simultaneously in the plan as shown in the diagram. This is obviously true because that an airplane cannot be at different gates at the same time. A plan is a solution plan if there are no violated mutual-exclusion constraints.

Slide 3 plots the structure of the mutual-exclusion constraints in three solution plans generated by LPG on the AIRPORT-4 instance. These plans correspond to, respectively, one, two, and three airplanes (subgoals). We plot actions in boxes and a line between two actions if they are related by a mutual-exclusion constraint. We see that the number of actions and the number of constraints grow in proportion to the number of subgoals, which lead to an exponential growth in search complexity.

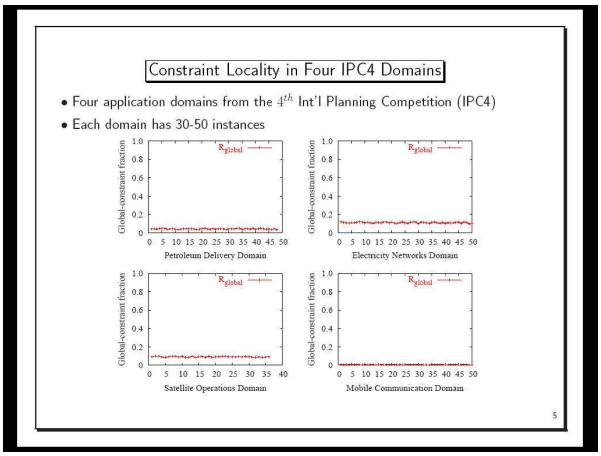
Slide 4 shows that the seemingly random constraints in the solution plans generated by LPG are in fact highly structured and can be clustered by their subgoals. To see this, we generate a plan for each of the three subgoals in the AIRPORT-4 instance, compose the plans together, and plot all the actions and constraints. We show that most constraints are local constraints relating two actions from the same subproblem, and that only a few global constraints (shown in red lines) relate two actions from different subproblems. This observation is intuitively sound because the movements of airplanes are largely independent. Two airplanes interact with each other only when they are at the same position and can be scheduled independently most of the time.

We also show in the demo that the constraint-locality property is observed in other IPC4 domains. Slide 5 plots the ratio of global constraints to all constraints under subgoal partitioning in four other IPC4 domains. It shows that the fraction of global constraints with respect to the total number of constraints is consistently low.

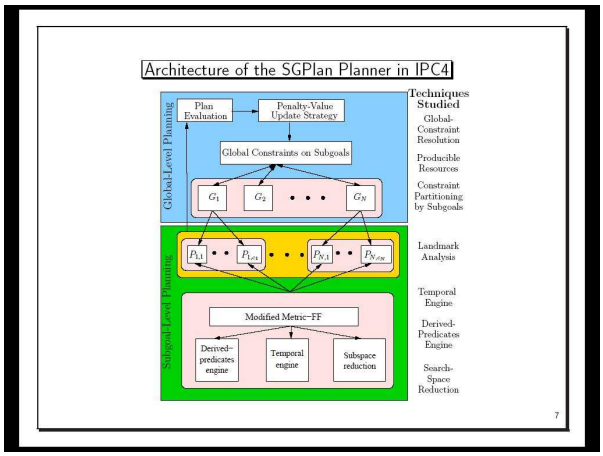
Although not shown in the slides, we have also observed similar constraint locality in some planning systems modelled by languages other than PDDL2.2, where there is no such notion as subgoals. For example, in the ASPEN system for space-rover planning for the Jet Propulsion Laboratory of NASA, the planning problem is modelled by a specific language that only defines actions and their temporal features, and the task is to find a conflict-free plan that optimizes a preference score while satisfying all the temporal constraints among actions. We have found that for ASPEN planning problems, constraints demonstrate temporal locality, i.e. the constraints are clustered by their active times. We therefore have proposed to partition the constraints by time.

Constraint Partitioning in SGPlan

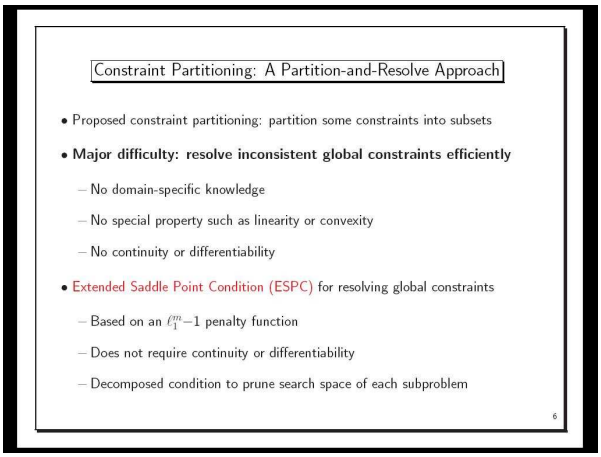
In SGPlan, we use a planning algorithm based on constraint partitioning. The approach partitions a planning problem into multiple subproblems by its subgoals, and



Slide 5: Constraint Locality in Four IPC4 Domains.



Slide 7: Architecture of the SGPlan Planner in IPC4.



Slide 6: Constraint Partitioning: A Partition-and-Resolve Approach.

solves each subproblem individually before composing the solutions. As there are global constraints relating multiple subproblems, the main technical problem addressed in the design of SGPlan is the resolution of inconsistent global constraints.

Resolving global constraints for temporal planning is particularly difficult than some previous partitioning approach for several reasons. First, we are interested in fully-automated planning and do not have any domain-specific knowledge about the application problems. Efficient partitioning algorithms, such as HTN planning, do exist when there are domain knowledge about how to partition a task into multiple subtasks. Second, existing efficient decomposition methods for linear and convex optimization cannot be used because the constraint functions are symbolic, nonlinear, and do not have linearity or convexity properties. Separable programming for constrained search usually requires the functions to have separable structures and be linear or convex. Third, mathematical conditions requiring continuity and differentiability cannot be derived because the constraints may be discrete and not in closed form. Due to these difficulties, previous constraint programming methods, such as penalty and Lagrangian methods, cannot be applied for resolving inconsistent global constraints.

To overcome these difficulties, Slide 6 presents our proposed Extended Saddle Point Condition (ESPC) based on a new ℓ_1^m-penalty function (Wah & Chen 2005). Our theory provides a necessary and sufficient condition for characterizing constrained local optimal solutions without continuity and differentiability assumptions on constraints. Moreover, the condition can be decomposed into a partitioned form, where each subproblem is associated with a necessary local condition. The partitioned condition greatly reduces the search space of each subproblem by pruning candidate subplans that do not satisfy the ESPC condition. We can then view the planning problem as a constrained nonlinear optimization problem, partition their constraints, and apply the ESPC condition to resolve the global constraints efficiently.

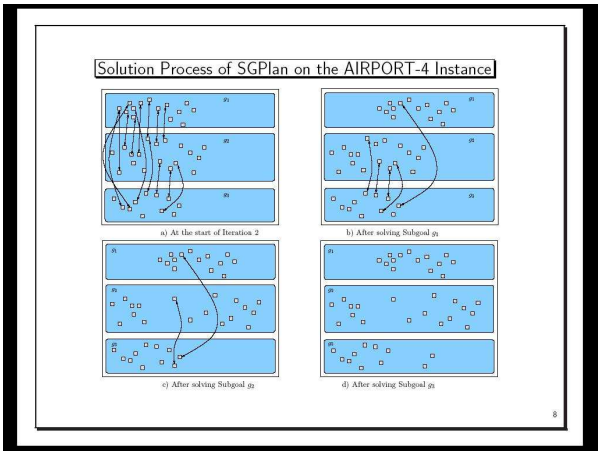
Slide 7 illustrates the architecture of our SGPlan planner based on constraint partitioning. In SGPlan, a planning problem is partitioned into multiple subproblems by its subgoals, and may be further partitioned using Landmark analysis (Porteous, Sebastia, & Hoffmann 2001). Landmark analysis detects some intermediate facts that have to be made true before achieving certain subgoals. Therefore, a subproblem may be further partitioned into a series of smaller problems, each trying to achieve a subset of landmark facts. SGPlan then uses a modified version of the FF planner (Hoffmann & Nebel 2001) to find a solution plan for each subproblem that satisfies all local constraints and that minimizes a modified objective function. In order to bias the search towards resolving global inconsistencies, our modified objective function includes the original objective function and the penalty terms for violated global constraints. In each iteration, the penalty values

are updated after solving all the subproblems and evaluating the global constraints.

Resolution of Inconsistent Global Constraints

Next, we illustrate the rapid reduction of violated global constraints in SGPlan. In addition, we show the effectiveness on using ESPC in SGPlan by comparing it with a greedy search without ESPC.

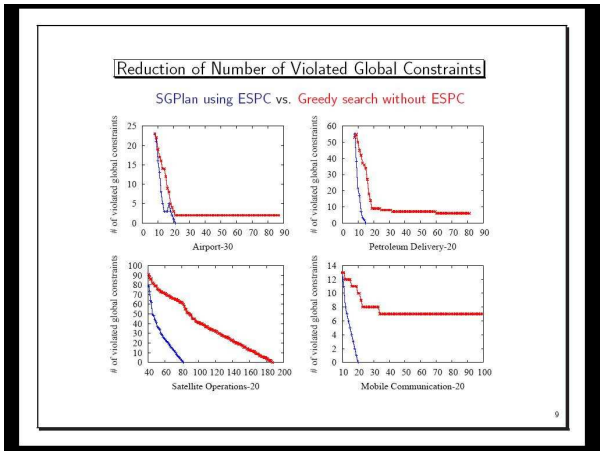
Slide 8 demonstrates the solution process of SGPlan on the AIRPORT-4 instance with three subgoals. In the first iteration, we generate a plan for each subgoal. We see that there are some violated global constraints (shown in arrows) at the start of the second iteration. In the second iteration, SGPlan solves each subproblem individually, using the ℓ_1^m-penalty function to bias the search towards resolving the violated global constraints. It is clear that the number of global constraints is reduced quickly, and a solution plan is found after the second iteration.



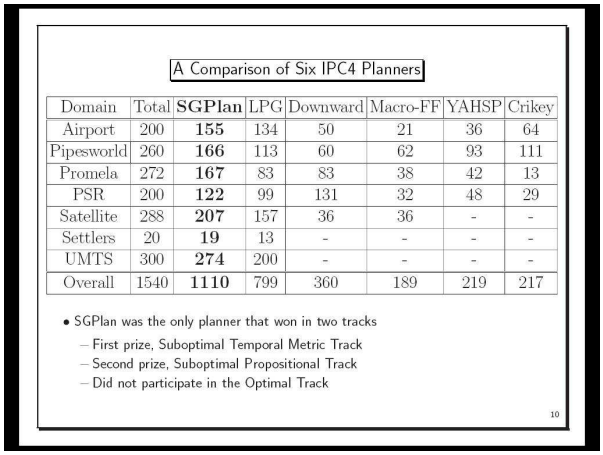
Slide 8: Solution Process of SGPlan on the AIRPORT-4 Instance.

Slide 9 demonstrates the effectiveness of using ESPC on four example instances in four application domains. In each case, we plot the number of violated global constraints with respect to the number of subproblems solved by SGPlan. As a comparison, we also tested a greedy search algorithm without using ESPC and a penalty function to bias the search. In each instance, we generated three alternative plans and accepted the one with the minimum number of violated global constraints. Clearly, SGPlan using ESPC can resolve inconsistent global constraints much more efficiently than the greedy algorithm.

Slide 10 summarizes the evaluation results of SGPlan on the seven IPC4 domains. For each domain, we show the total number of instances and the corresponding number of instances solved by SGPlan and five other leading IPC4 planners within the 30-minute CPU-time and 1-GB memory limits. The results show



Slide 9: Reduction of Number of Violated Global Constraints.



Slide 10: A Comparison of Six IPC4 Planners

that SGPlan is consistently better than other planners in most IPC4 domains.

References

Blum, A. L., and Furst, M. L. 1997. Fast planning through planning graph analysis. *Artificial Intelligence* 90:281–300.

Chen, Y. X.; Hsu, C.-W.; and Wah, B. W. 2004. SGPlan: Subgoal partitioning and resolution in planning. In *Proc. Fourth Int'l Planning Competition*. Int'l Conf. on Automated Planning and Scheduling.

Chen, Y. X.; Hsu, C.; and Wah, B. W. 2005. Temporal planning using subgoal partitioning and resolution in sgplan. *Journal Artificial Intelligence Research*.

Hoffmann, J., and Nebel, B. 2001. The FF planning system: Fast plan generation through heuristic search. *J. of Artificial Intelligence Research* 14:253–302.

Porteous, J.; Sebastia, L.; and Hoffmann, J. 2001. On the extraction, ordering, and usage of landmarks in planning. In *Proc. European Conf. on Planning*, 37–48.

Wah, B., and Chen, Y. X. 2005. Fast temporal planning using the theory of extended saddle points for mixed nonlinear optimization. *Artificial Intelligence*.