Incorporating High-Speed, Optimizing Scheduling into NASA's EUROPA Planning Architecture

Richard Stottler[1] and David Breeden[2]
Stottler Henke Associates, Inc. San Mateo, CA, 94404

This paper describes an effort investigating improvements possible to NASA's EUROPA planning system development toolset through the addition of high-speed, high-quality scheduling algorithms. We determined that such additions were beneficial, feasible, and could be readily taken advantage of by planning system developers. We designed two different integration mechanisms (re-implementation of scheduling algorithms within EUROPA and interfacing an existing c++ library of scheduling algorithms to EUROPA), determined that there were advantages to each approach. We prototyped the reimplementation integration option and showed its benefits with two prototypes – one directed toward International Space Station (ISS) Extravehicular Activity (EVA) planning (that showed significantly better results in less time than EUROPA operating alone) and one directed toward automatic Score ISS crew scheduling that was able to quickly, automatically schedule with realistic data while obeying a large number of hard and soft constraints. (No automatic Score ISS scheduling capability currently exists.) Finally, we added the scheduling-enhanced EUROPA prototype to an independently-developed EUROPA application, which allowed it to finish planning and find an optimal schedule in under 4 minutes instead of not returning at all after 80 minutes.

Nomenclature

ADAPT	=	Advanced Diagnostics and Prognostics Testbed at NASA Ames
AI	=	Artificial Intelligence
ANML	=	Action Notation Modeling Language
AURORA	=	High-Speed, High-Quality, Intelligent Scheduling System Architecture and algorithms
DS1	=	Deep Space 1
EOS	=	Earth-Observing Satellite
EUROPA	=	Extensible Universal Remote Operations Planning Architecture, developed by NASA Ames
EVA	=	Extra Vehicular Activity (Space Walks)
ISS	=	International Space Station
LEE	=	Latching End Effector
MER	=	Mars Exploration Rover
NDDL	=	New Domain Description Language
PVTCS	=	Photovoltaic Thermal Control
SARJ	=	Solar Array Rotating Joint
Score	=	Developed by NASA Ames to help ISS operational planners to generate the daily ISS Schedule
SPDM	=	Special Purpose Dexterous Manipulator

I. Problem Overview

Traditionally, advanced, robust, autonomous planning systems have not focused on the scheduling decisions made by the planner. And high-quality, optimizing schedulers have rarely been integrated with such planning systems and when they have, it has been at a very coarse level, where the planner produces a completed tentative plan that is passed to the scheduler, which attempts to find feasible resources and time windows for the plan's tasks. If the scheduler cannot satisfy all of the plan's constraints, it replies with its failure, potentially including in its

[1] President, 951 Mariners Island Blvd., Suite 360, San Mateo, CA 94404, AIAA Member
[2] Software Engineer, 951 Mariners Island Blvd., Suite 360, San Mateo, CA 94404, AIAA Contributor

response the reasons for that failure (not enough of a specific resource available, for example). Little research regarding interleaving scheduling and planning has been performed. Meanwhile the Aurora scheduling architecture executes very rapidly and has solved scheduling problems in dozens of diverse, critical domains. We prototyped the integration of high-speed, high-quality scheduling algorithms from Aurora with NASA's existing EUROPA 2 architecture to provide real-time scheduling within EUROPA's planning process. This would create an easy to use development tool based both on EUROPA and Aurora for the creation of planning/scheduling systems.

As described by Smith et al., there tends to be a gulf between Artificial Intelligence (AI) planning research (which is typically directed toward problems with cascading levels of action choice) and research in scheduling (which tends to be applied toward optimizing the choice of assigning resources and time windows to specific activities that usually involve larger number of activities and/or resources but with significantly less choice as to what those activities are).[4] Yet all planning ultimately must result in specific resources carrying out specific actions (activities) at specific times, so all planning that actually results in execution ultimately results in a schedule. There are planning problems where the particular resource choice or timing is trivial or unimportant, especially compared to the issue of generating a valid sequence of actions to accomplish one or more goals. For example, there may only be one resource (e.g., 1 rover) to accomplish the activities in question and the order that activities are performed for that one resource will not matter if they do not compete for shared resources with other activities or change the resource's state significantly (e.g., changing a filter on the International Space Station (ISS) and cleaning a backup pump can be done in either order if they do not have any relationship with each other and therefore don't change the ISS's state significantly (other than the direct effects of having a fresh filter and cleaned backup pump)).

But few problems in the space domain actually fall into this category as evidenced by the difficulty in coming up with the examples above. For example, even if there is only 1 rover (or vehicle or spacecraft), typically there will be multiple instruments onboard that will compete for limited resources such as power, bandwidth, thermal cooling, water, etc. Consider the following very simple example, where there are 4 separate instruments that have no overlap in capabilities so there is no choice as to which resource to use. There would be a desire to operate each in parallel to gather science in their four respective areas simultaneously but each, of course, uses power limited by how much is being generated by the solar panels at the moment (discounting battery capacity for this example). The activity networks for the four instruments are shown in Fig. 1with the height of each indicating its power requirement, which is also indicated by the number located at the top left corner of the activity. The number at the lower-right corner indicates the time duration. Activity D, for example, uses two units of power and its duration is 3 units of time. Of course there is the implicit desire to accomplish all of these activities in the shortest possible time. Most typical algorithms would produce the answer shown below in Fig. 2, which requires 9 units of time.

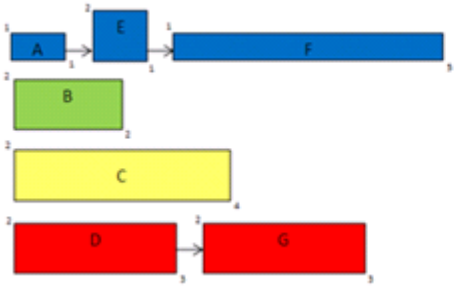

Figure 1.

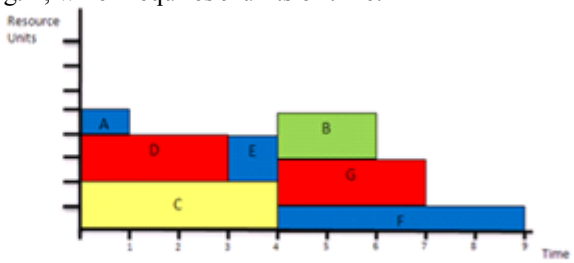

Figure 2.

Good scheduling algorithms would often produce the schedule below in Fig. 3, requiring 8 units of time. Very good schedulers should give the optimum result, shown in Fig. 4, requiring 7 units of time. High-quality scheduling decisions can get more science done in less time with the same resources.

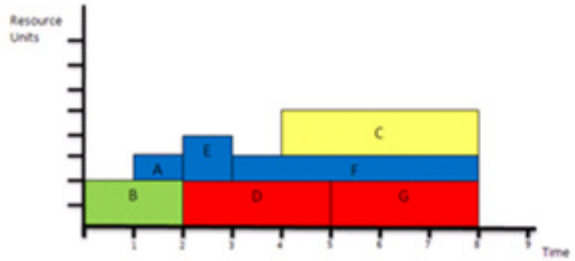

Figure 3.

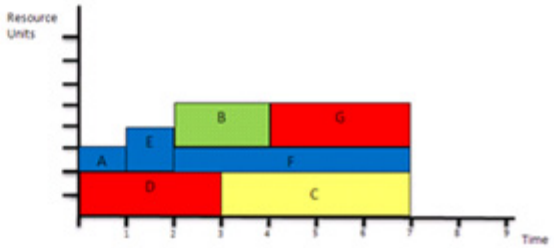

Figure 4.

II. Background

Meanwhile, EUROPA is a powerful software platform developed at NASA Ames Research Center for building easily configurable and extensible planners for a wide variety of domains. The intuitive and expressive New Domain Description Language (NDDL) enables the formulation of complex problems in a straightforward manner, and EUROPA's high degree of modularity and plug-in architecture make customization extremely low effort. EUROPA has been used for an impressive array of missions and research, including the Mars Exploration Rover (MER), Deep Space 1 (DS1), Earth-Observing Satellite (EOS) scheduling, and much more. NASA Ames is currently in the process of extending EUROPA for the Action Notation Modeling Language (ANML). ANML is a relatively new language for specifying planning domain models. ANML is an improvement over NDDL in many ways, especially in terms of being a higher level language with which to express planning domain concepts more naturally. Thus a combined EUROPA-Aurora would benefit ANML developers as well.

III. Prototype Efforts

A. Score Domain and Prototyping Efforts

One of the prototyping efforts related to an automatic scheduling capability for Score, software used by ISS operational planners to generate the daily ISS schedule. This involves scheduling the astronauts' time and other in-space resources to the nearest minute. All of the astronauts' time is scheduled including sleep, pre and post sleep activities and meals. There are several hard and soft constraints that must be met by the schedule. Score is a graphical editing tool that utilizes EUROPA to check that no resource or temporal constraints are being violated and flags any that are. Besides the astronauts, other resources or conditions to schedule include communication bandwidth (which requires antenna line of site), power, data bandwidth, 3 lab camcorders, station orientation, PCs, High Rate Digital Link bandwidth, High Rate FM Digital Port, different types of Video Ports, Total System Video Bandwidth, Rack components, and many others. The number of resources and their complexity is surprisingly high. Many activities required 5 or 6 separate resources each. During the course of this effort we determined that it was both feasible and beneficial to provide an automatic scheduling capability for the Score tool.

To compile an automatic scheduling scenario, we pulled out from the data we received from NASA Ames all the activities from June 14th, 2011 that either required an astronaut resource or had a temporal constraint chain connection with an activity that required an astronaut resource. On that date there were 6 astronauts onboard to be scheduled (ISS_CDR, FE_1, and FE_3 through FE_6). We used the conditions and resources from the files and added additional constraints of different types as described below. To show some flexibility we also added a couple of pools of resources. The two pools we added were 1) FE_3_and_FE_4 and 2) FE_5_and_FE_6. Basically, instead of specifying a specific astronaut, a pool can be specified and the system will pick the one that leads to the best schedule.

1. Temporal constraints

- XFS = eXact Finish to Start. The first activity must end exactly when the second activity starts.
- FS = Finish to Start. The first activity must end before the second activity can start and there may or may not be a gap of time in-between.
- FS (0 – x minute Gap) = Finish to Start with an x minute gap at most. The first activity must end before the second activity can start and the gap in between cannot be longer than x minutes and a smaller gap is preferred.
- FS (Minimum x minute Gap) = Finish to Start but the gap between the two activities should be at least x minutes long (or longer).
- Late-End t = The activity must end by t or sooner.
- Early-Start t = The activity cannot start sooner than t.
- Concurrent = The two activities must occur at the same time.
- Not Concurrent = Two activities may not overlap in time.

2. Same Resource Constraint

In the case of a chain of activities where a specific resource is not specified but, instead, one of a resource set is specified, it often makes sense that whichever resource is chosen for one activity continues to be the one used for all activities in the chain. (E.g., if FE_3 performs the first activity in an experiment then he or she should continue performing the rest of the activities in the experiment.)

The prototype was able to generate a correct schedule that obeyed all of the hard constraints and optimized the

soft ones in a few seconds for the 130 activities and 20 different types of resources (including the 6 astronauts). In order to illustrate that the prototype was generally functional, we made several editing changes to the activities, rescheduling the entire schedule frequently to show the changes having an effect. This resulted in a 20-minute demonstration with 14 different parts. The demonstration sequence illustrated obeying the FS and FSX hard constraints, realizing the soft constraint associated with FS (0 – x minute Gap) when possible but always realizing the hard constraints associated with it, functioning correctly after deletions (thus on a (slightly) different set of tasks), obeying the FS (Minimum x minute Gap) hard constraints, obeying the hard constraints associated with chains of different types of constraints (one chain of constraints was Early-Start, XFS, Concurrent, XFS), obeying the same resource constraint, balancing assignments to different astronauts as required when durations were increased, and obeying the S-Band Condition constraint when it was added to an activity.

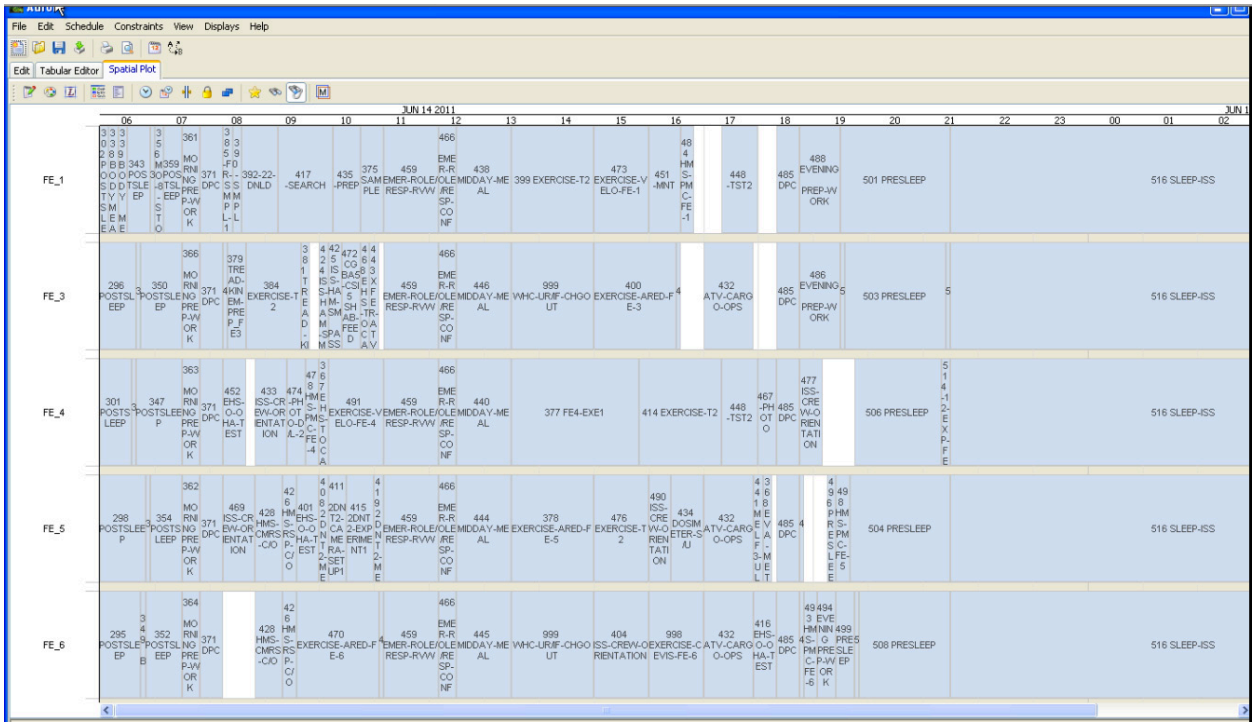

Figure 5. Sample Schedule Generated by the Score Scheduler Prototype.

B. EVA Prototype Implementation

We prototyped plug-ins to the default EUROPA solver to intimately integrate some of Aurora's scheduling techniques. Scheduling decisions in the EUROPA solver are made by resolving object flaws (also called "threats"). The EUROPA solver can be configured to use custom logic to resolve threats via the solver's plug-in architecture.

1. The Extrahicular Activity (EVA) test domain

The EVA domain was hand-translated to NDDL from the PDDL-e description found in TRACLabs' public subversion repository at http://svn.traclabs.com/svn/3t/trunk/planner/domains/nasa/nasa2.pddl.[3] The principal goals modeled for the EVA are:

- Grapple bar stow beam installation
- Special Purpose Dexterous Manipulator (SPDM) Latching End Effector (LEE) lubrication
- Port side Solar Array Rotating Joint (SARJ) lubrication
- Photovoltaic Thermal Control System (PVTCS) ammonia reservoir refill

The model includes the following auxiliary tasks/concepts to support the above goal tasks:

- Egress and ingress
- Equipment stowage, possession and containment
- Traveling and location

We translated a subset of this domain to NDDL. This subset modeled the stow beam installation and SPDM LEE lube as monolithic tokens, while the SARJ lube and ammonia fill were decomposed one level into chains of tasks.

All tasks were constrained to occur between egress and ingress. The only resources in the translated model were the two crew members. No additional auxiliary tasks or resources were modeled. Egress, ingress and the first two tasks in the PVTCS ammonia fill chain required both crew members, whereas all other tasks only required a single crew member.

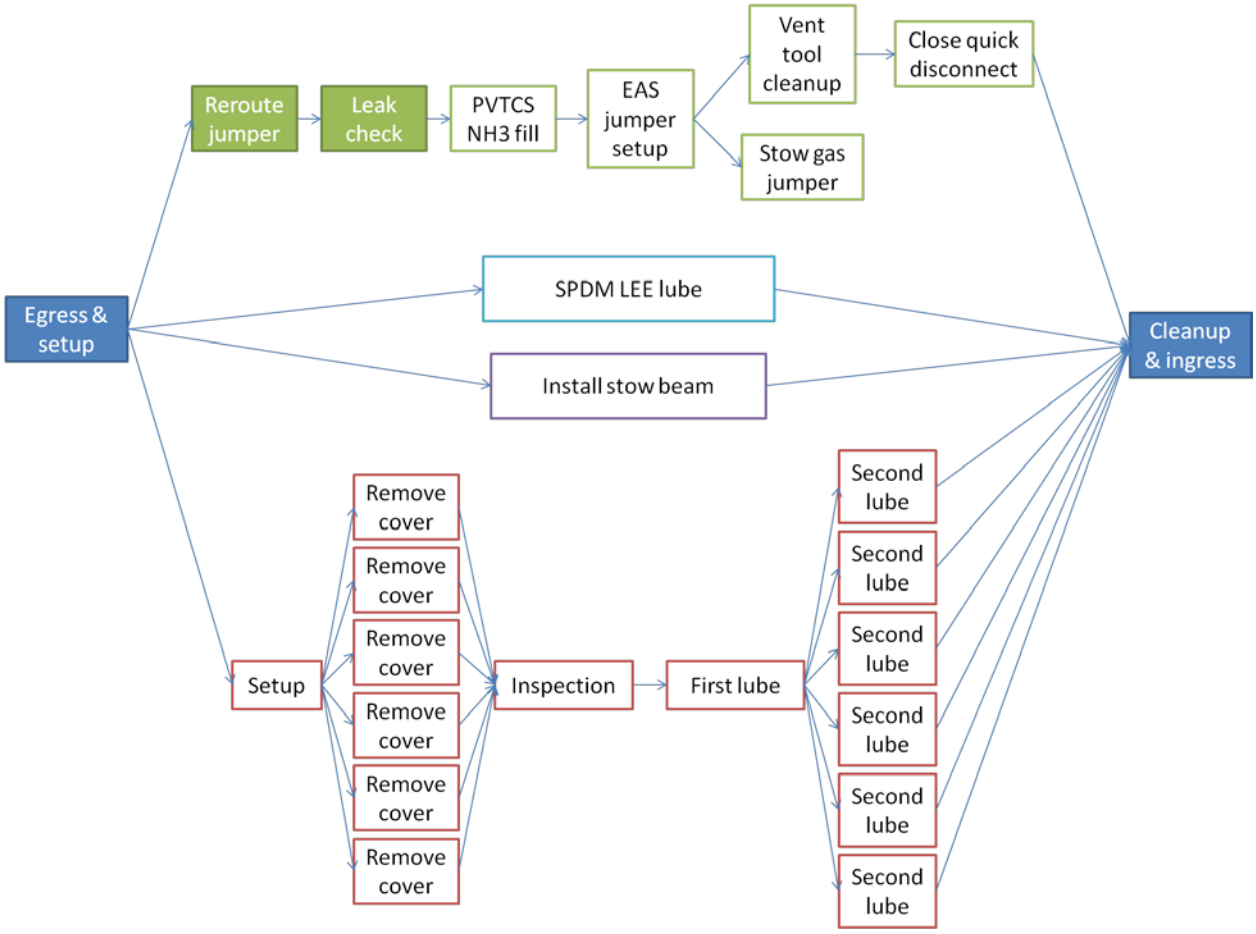

Figure 6. EVA temporal network. Child tasks must occur after parent tasks. Shaded tasks require both crew members, and all others require 1 crew member.

The minimal makespan for these tasks is 5 hours and 31 minutes. This is obtained by assigning the SPDM LEE lube, PVTCS ammonia fill tasks, and one lubrication of two of the SARJ covers to one crew member, and all other tasks to the other (the rest of the SARJ lube tasks and the stow beam installation).

2. Prototype evaluation

Below are the results of scheduling the tasks with the default solver and with the improved solver using our plug-ins. Due to compiler compatibility issues, the debug libraries were used and so the run times are longer than they would be operationally for both the default and prototype solvers. We initially used a planning horizon of 6 1/2 hours as this was the planning horizon for the real EVA procedures.

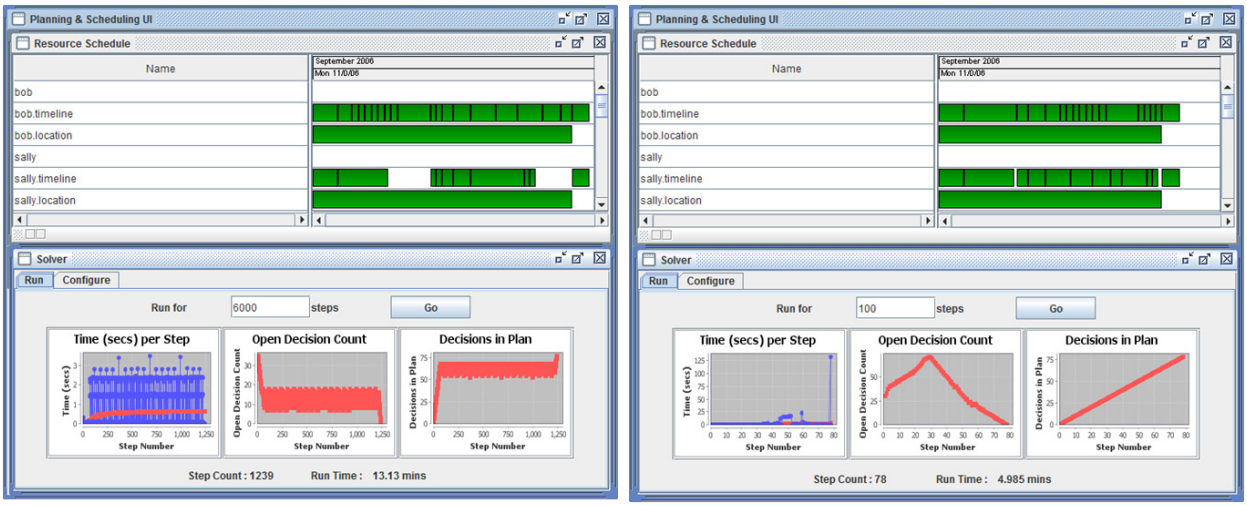

Figure 7. Results from scheduling the EVA tasks with the default solver (left) and the prototype solver (right). Important features to note: 1) the near-optimal makespan produced by the prototype solver. 2) The oscillations in the Decisions in Plan with the default solver indicate heavy backtracking, whereas the prototype solver does not backtrack at all. 3) The plot of the open decision count with the prototype solver, showing an initial planning stage where open decisions increase as tokens are activated, followed by a scheduling stage where open decisions decrease to zero as remaining flaws are resolved.

The default solver produced a plan with a makespan of 6 hours, 25 minutes and took 1239 steps with a run time of 13.13 minutes. By contrast, the prototype solver produced a plan with a makespan of 5 hours, 35 minutes and took 78 steps with a run time of 4.99 minutes.

Comparison based on makespan, however, is biased since our prototype is designed to optimize makespan, whereas the default solver is not explicitly intended to optimize any metric. Therefore, we examined the effect of shortening the plan horizon on the respective solvers. Below are these results.

Table 1. Results from scheduling with decreasing plan horizons.

	Default solver			Prototype solver		
Plan horizon	**Step count**	**Run time**	**Makespan**	**Step count**	**Run time**	**Makespan**
6 hr, 30 min	1239	13.1 min	6 hr, 25 min	78	4.99 min	5 hr, 35 min
5 hr, 47 min	77	0.06 min	5 hr, 43 min	93	3.82 min	5 hr, 31 min
5 hr, 46 min	77	0.06 min	5 hr, 43 min	1218	31.8 min	5 hr, 36 min
5 hr, 34 min	20000	118 min	Timeout	5196	95 mins	5 hr, 32 min
5 hr, 31 min	20000	123 min	Timeout	13459	318.8 mins	Timeout

It is evident from this data that the default solver is capable of improving remarkably with the right planning horizon, but it is not stable. Also, the fact that the prototype solver had a much tougher time with tighter planning horizons is interesting. The problem is that the only plan that will work requires that two of the SARJ lube tasks be moved after the chain is scheduled, and the solver is configured to prefer assigning all tokens in a chain to the same resource. With a longer planning horizon (a minimum of 5 hours and 47 minutes), the prototype solver is able to obtain the suboptimal plan with all SARJ lube tasks on one crew member in a single pass, then postprocesses the plan to reach the optimal makespan quickly. When the horizon is too tight for all tokens of each chain to be on the same resource, however, we are forced to backtrack. In this particular case, since the only solution involves changing the allocation of two tokens that are assigned relatively early on, the backtracking is very painful. This is one indication that a local search strategy that more closely mirrors Aurora's approach would be more robust for difficult problems. In this situation Aurora would schedule all the tokens, knowing there are conflicts, and then find the solution in postprocessing.

C. Out-of-the-box application to an unrelated domain

The prototype plug-ins were additionally tested with a completely unrelated domain, planning recovery activities on the ADAPT (Advanced Diagnostics and Prognostics Testbed) Electrical Power System.[2] Out of the box, it only took a few minutes for a user with no knowledge of our plug-in design or scheduling algorithms (or scheduling in general) to install our plug-ins and configure his EUROPA solver to use them. Without the plug-ins, his sample problem did not return after 80 minutes and 6600 steps. With the plug-ins, the optimal solution was found after 3.46 minutes and 200 steps. This strongly indicates both the benefits and the feasibility of these scheduling algorithm additions to EUROPA.

IV. Conclusion

The Prototyping effort proved the feasibility of adding high-quality scheduling algorithms to EUROPA. The above discussion also points to another finding: EUROPA and Aurora's algorithms are highly complementary. EUROPA's back-tracking search is very appropriate for the action generation planning stage but far less so for the scheduling step, which Aurora's near-linear algorithms are more suited for but which are totally inappropriate for the planning stage. Consequently Aurora has no support for back-tracking whereas EUROPA has very sophisticated support for it so that the combined EUROPA-Aurora EVA prototype could take advantage of it readily. (For Aurora this might mean back-tracking in order to reverse a fairly small number of large, relatively arbitrary decisions that have a major impact on the schedule as a whole.) Meanwhile Aurora has much better support for the concept of moving and shuffling tasks (as opposed to back-tracking) so those actions are accomplished much more easily within Aurora. EUROPA follows a least commitment strategy that can sometimes be useful, along with the associated sophisticated constraint propagation, which is to be used as a basis for easily computing data associated with some of Aurora's more complex heuristics (such as bottleneck scheduling). However, Aurora's maximal commitment strategy is more runtime efficient, when it is applicable. Aurora provides significant support for generating global knowledge about the initial and evolving scheduling problem to be used in sophisticated heuristics as does EUROPA (with ready access at all times to the evolving plan database). Finally Aurora provides for a very explicit post-processing step, which is often an important component to scheduling as was illustrated in the Phase I prototype; the type of task moving/shuffling often associated with this step is most easily performed in Aurora.

References

[1]Bonasso, P., Boddy, M., "Planning for Human Execution of Procedures Using ANML," Proceedings of the ICAPS 2010 Scheduling and Planning Applications woRKshop (SPARK 2010), Vol.1, AIAA, Reston, VA, 2010, pp.21-29.

[2]"Diagnostic Problem II – ADAPT," *Second International Diagnostic Competition (DXC 10)*, 2010. [http://sites.google.com/site/dxcompetition2010/home/tracks/diagnostic-problem-ii. Accessed 5/21/12]

[3]"Revision 2469/trunk/planner/domains/nasa," TRACLabs. [http://svn.traclabs.com/svn/3t/trunk/planner/domains/nasa/nasa2.pddl. Accessed 5/21/12]

[4]Smith, D., Frank, J., and Jónsson, A., "Bridging the Gap Between Planning and Scheduling," The Knowledge Engineering Review, Vol. 15, No. 1, 2000, pp. 47-83.

[5]Frank, J., Jónsson, A., "Constraint-based attribute and interval planning, " Constraints, Vol. 8, No. 4, 2003, pp. 339-64.

[6]Smith, D., Frank, J., and Cushing, W., "The ANML Language," David E. Smith – Publications [online database], URL: http://ti.arc.nasa.gov/profile/de2smith/publications/ [cited 25 May 2012].