Merging the Semantics of State Constraints for Collections of Activities

Russell Knight, Gregg Rabideau, and Steve Chien

Jet Propulsion Lab, M/S 126-347
California Institute of Technology
4800 Oak Grove Drive, Pasadena, CA 91109-8099
firstname.lastname@jpl.nasa.gov

Abstract

When scheduling a collection of activities, it is often useful to calculate the valid intervals for the collection with respect to shared resources and states. The constraints of these activities on shared resources and states need to be analyzed to avoid miscalculations due to interactions between the constraints. For shared resources, a combined profile is typically generated and used to compute the valid intervals. We introduce a technique for generating a combined profile for shared state constraints, which is subsequently used to compute valid intervals. The worst-case complexity of our technique is quadratic in the total number of resource and state constraints in the problem. We present empirical evidence indicating that our technique improves performance of our planner on real problems when compared to the performance of the same planner using more naïve techniques.

Introduction

This paper describes an approach to solving the problem of scheduling interdependent sets of activities in a combined scheduling/planning problem. In this approach, we compute a combined profile that encapsulates the requirements and effects of the collection of activities on shared states and resources. This combined profile is then used to determine the least-conflicting placements for the collection allowable in the current plan.

This problem is an important aspect of solving combined planning and scheduling problems. In many approaches to combined planning and scheduling, one alternates between finding activities to satisfy pre- and post-conditions (planning) and finding temporal assignments and resources for those activities (scheduling). Complex activity placement is also an important component of many scheduling problems, as finding temporal assignments for complex activities can be computationally challenging.

This work advances the approach of moving collections of activities whose temporal relationships among themselves are fixed. We observe similar techniques in job-centered scheduling approaches and tabu-search approaches. We proceed by describing our motivation, defining the problem, and describing the solution. Finally, we present empirical evidence in favor of our technique.

Motivation

We believe that reasoning about collections of activities (as opposed to individual activities) is helpful in finding a solution to a combined planning/scheduling problem. We believe this is due to the increased complexity introduced by the backtracking required for reasoning about individual activities. In a sense, we assume that decisions concerning the members of the collection will hold even when shifted “lock-step” in the schedule either forward or backward in time. Although research concerning flexible intervals between members of the collection might be fruitful, it is left as future work. Even the simplest of approaches (e.g. gradient descent) require that we know the least conflicting intervals for a given collection, therefore we wish to compute these intervals.

Problem

We assume some method for partitioning the complete set of activities into separate collections. This partitioning is a research area in its own right and outside the scope of this paper (although we do describe our technique for partitioning activities.)

We discover that naïve intersection of valid intervals for single activities in the collection render both false positives (intervals returned as being valid in fact cause constraint violations) and false negatives (intervals returned as causing constraint violations are in fact valid) with respect to the whole. This can only be due to interactions between the activities; or, more specifically, interactions between the constraints on shared states and resources.

For example, consider a pair of activities that affect a battery (see Figure 1). The first activity a_1 uses 10 amp-minutes, while the second activity a_2 restores 10 amp-minutes. If we schedule a_1 individually, we find no intervals that will not cause an over-subscription of the battery, because activity a_3 has already fully depleted the battery by the end of the current schedule. But, if we schedule these together, we find placements that are valid. The positive effect a_2 has on the schedule makes up for a_1’s usage. We wish to handle this sort of constraint-interaction for shared states and resources. We continue with some definitions.

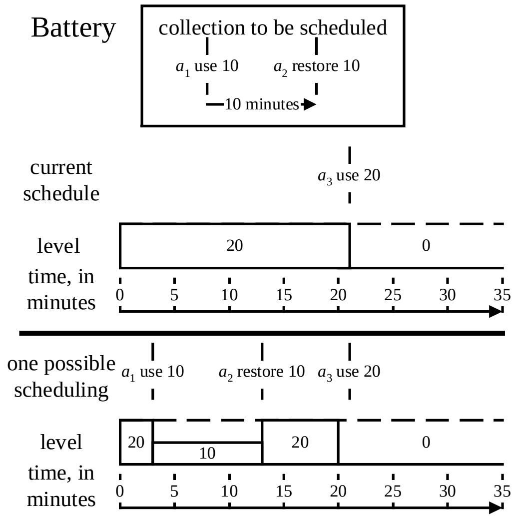

Figure 1 Battery interaction example

Definitions

Activity: consists of a start-time (s), an end-time (e), and a set of timeline constraints or reservations that represent the constraints of the activity with respect to shared states and resources (R). R is conjunctive, therefore we do not deal with multiple resources that could satisfy a single constraint. For simplicity, we assume that times are integers. We also assume that $s < e$.

Timeline: the representation of a shared state or resource over time. All timelines consist of a set of value units that represent the value of the timeline for all time-points in the horizon. All timelines also consist of a set of global constraints. Global constraints and values differ according to the type of timeline.

Depletable Resource Timeline: a timeline representing a resource that is added to or removed from over time. An example is a battery, where usage depletes the battery and recharging restores the battery. Global constraints include the minimum level, the maximum level, and the default or starting value.

Non-Depletable Resource Timeline: a timeline representing a resource that is used for a period of time and then relinquished back at the end of the usage. An example is a power bus, where usage takes up some of the capacity of the bus, and the cessation of usage restores the capacity. Obviously, a non-depletable timeline can be represented by a depletable timeline with a depleting reservation followed by a renewing reservation (reservations are described later), but we include these because the semantics of their reservations aids in describing the semantics of transformed depletable reservations. As with depletable timelines, global constraints include the minimum level, the maximum level, and the default value.

State Timeline: a timeline representing a state that can be either changed or required by reservations. Values are symbols; we use strings. Global constraints consist of a transition graph G whose nodes represent allowed values and whose edges represent allowed consecutive transitions from one value to the next, and a default value.

Reservation: a constraint on a shared state or resource for a specific interval to a specific value. Since reservations are part and parcel with activities, reservations inherit their start- and end-times (s, e) from the activity containing them. A reservation also consists of a reference to a timeline. The type of timeline that the reservation constrains determines the type of the reservation.

Depletable Reservation: consists of a start-time (the end-time is ignored) and a value (v). v is an integer representing the amount of capacity to be used by the reservation. The value of any point on such a timeline is the sum of all reservations at the time-point and previous, as well as the default value. See Figure 2. Note that the timeline models a value over time. The reservations are labeled with their values.

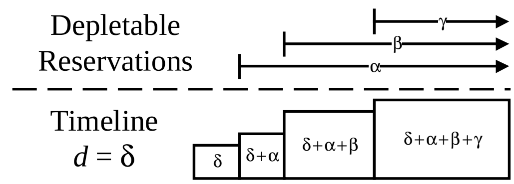

Figure 2 Effect of depletable reservations

Non-Depeletable Reservation: consists of a start- and end-time (s, e, as above) and a value (v). v is an integer representing the amount of capacity to be used by the reservation. The value of any point on such a timeline is the sum of all reservations that include the time-point in their temporal extents, as well as the default value. See Figure 3.

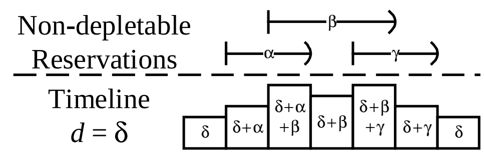

Figure 3 Effect of non-depletable reservations

State Reservation: we address two types of state reservations: changers and users.

A changer reservation consists of a start-time (s, the end-time is ignored) and a value (v). v is a symbol as described for state timelines. The value of any point on such a timeline is the value of the most recent changer reservation. If two or more changers are simultaneous to different values, the resultant value is invalid. See Figure 4.

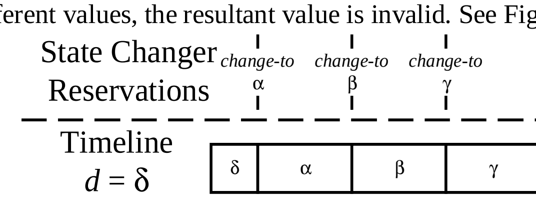

Figure 4 Effect of state changer reservations

A user reservation consists of a start- and end-time (s, e) and a value (v). v is a symbol as described for state timelines. A user reservation constrains the timeline such that for all points during its temporal extent, the value of the timeline is the same as *v*. Note that user reservations have no direct effect on the value of the timeline.

Conflict: a conflict is a violation of any constraint. These include: 1) over or under subscriptions, where a resource timeline value lies outside of its allowed range, 2) state transition violations, where a state timeline value is immediately followed by a value for which no corresponding transition arc exists in its transition graph, 3) state usage violations, where a state timeline value differs from a user reservation constraining the timeline during the temporal extent of the reservation.

Solution Description

Our approach to computing the valid intervals (*I*) is as follows: 1) gather all reservations of all activities, 2) partition these according to timeline (i.e. for each timeline, create a set that contains all of the reservations for the timeline), 3) compute the valid intervals for each partition (*P*), 4) translate and intersect the valid intervals. We focus on step 3. We know that if the reservations in *P* do not interact, then we can compute valid intervals for *P* by computing those for each individual reservation in *P*. This is accomplished by first computing each set of intervals I_r for each individual reservation *r* in *P*. We then translate I_r by the difference between the start-time of the reservation and the earliest start-time in *P*, and intersecting all of the resultant intervals.

The key point of our approach: transforming *P* into a set of non-interacting reservations *P'* is one way of making the computation of valid intervals tractable.

Even if reservations in *P* interact, the reservations in *P'* do not, thus we can compute valid intervals using simple translation and intersection of the valid intervals for each individual reservation in *P'*.

To obtain *P'*, we merge the semantics of the existing reservations. This is straight-forward for resource reservations, but state reservations are problematic. For example, if *P* consists only of a changer reservation to "on" preceding a user reservation of "on", the *P'* requires a reservation between them that ensures that either no changes occur, or that the last change that does occur accommodates the user. Our solution technique is described more fully in (Knight, R., Rabideau, G., and Chien, S. 2000.) The complexity of our technique is quadratic in the total number of reservations, and scales at a constant factor cost over more naive techniques.

Empirical Evaluation

We now describe an empirical comparison of an aggregate search technique using our informed approach for determining valid placements for collections of activities to the same search technique using a naïve approach. We evaluate two aspects of our algorithm: 1) quality in terms of speed and accuracy, and 2) efficacy in terms of conflict reduction, both in scheduling and combined planning and scheduling.

In our empirical analysis we use four models (and corresponding problem set generators): 1) the EO1 spacecraft operations domain, 2) the Rocky-7 Mars rover operations domain, 3) the DATA-CHASER shuttle payload operations domain, and 4) the New Millennium Space Technology Four landed operations domain.

Within each model and corresponding problem set, we generate random problems that include a background set of fixed activities and a number of movable activity groups. The activity groups are placed randomly. The goal is to minimize the number of conflicts in the schedule by performing planning and scheduling operations.

To solve each problem, we use the ASPEN (Automated Scheduling and Planning Environment) system using an "iterative repair" algorithm, which classifies conflicts and attacks them each individually (Rabideau, G., Knight R., Chien S., Fukunaga A., Govindjee A. 1999). Conflicts occur when a plan constraint has been violated; this constraint could be temporal or involve a resource or state timeline. Conflicts are resolved by performing one or more schedule modifications such as moving, adding, or deleting activities. The iterative repair algorithm continues until no conflicts remain in the schedule, or a timeout has expired.

The scheduler entertains non-conflicting placements when moving activity groups. In the *control* trials the scheduler does so using the naïve algorithm for computing valid placements. In the *experiment* trials the scheduler is using the informed method to compute valid placements. In all cases for each domain, both trials are using the same set of heuristics at all other choice-points (e.g., selection of a conflict or activity group to attempt to repair, where to place within computed valid intervals, etc.). Note that simple (non-aggregate) operations are available in both real domains, although they are of limited comparative utility. Using only non-aggregate operations, the problems are intractable within reasonable time bounds. This is because the distance in terms of sub-optimal moves from one local optima to the next is $O(n)$ and the space to be searched is $O(m^n)$ where *n* is the number of activities in a movable collection and *m* is the number of possible locations given by a naïve calculation of legal intervals for an individual activity. For example, in the EO1 domain, *n* ranges from 23 to 56; in the Rover domain, *n* ranges from 8 to 17. Note that naïve calculation for valid intervals for an individual activity are adequate for that activity.

We forego further description of the domains due to space constraints.

Quality Assessment

To assess the quality of the algorithm, we directly compare the accuracy and speed of the informed search mechanism to the naïve intersection approach and a random placement approach. We assess the accuracy of the competing approaches by comparing the intervals for legal placement that each algorithm returns to the correct intervals. We assess the speed of the three approaches by measuring the

CPU time taken by each algorithm to compute its intervals.

Accuracy is a desirable property in any algorithm to determine "good" placements for aggregated activities. Ideally, a legal interval generator would return exactly those times that are legal placements for the member activity. This would mean that the algorithm would be sound (e.g., all times in the interval returned would be legal) and complete (e.g., all legal times would be returned by the algorithm). Table 1 shows the results of this evaluation on the ST4, EO1, Rover, and DCAPS domains. Because the informed method is complete and sound, it returns all of the correct interval(s) and no incorrect intervals. However, the naïve intersection method has both false positive (soundness) and false negative (completeness errors). As the data shows, it tends to miss the majority of the legal interval (by failing to recognize positive interactions between activities in the collection).

	EO1	Rover	DCAPS	ST4
informed	$0/_{14870}$	$0/_{10030}$	$0/_{5100}$	$0/_{96890}$
naïve	$14880/_{10}$	$4250/_{5780}$	$5700/_{3600}$	$51690/_{45220}$
random	2742250/ 2757120	$16101/_{17104}$	$2100/_{7200}$	360150/ 457040

Table 1 Errors in Calculated Intervals ($^{\text{avg. errors}}/_{\text{avg. interval size}}$)

Speed is another desirable property of a legal times algorithm. Ideally, a legal interval generator would take very little CPU resources to compute. Table 2 shows the average CPU time taken by each algorithm to compute its estimation of the legal intervals. Because the random algorithm simply returns the whole interval it takes effectively zero time. We also observe that the informed algorithm takes more time than the naïve algorithm. This not surprising as it must first transform the basic state and resource reservations into non-interacting reservations, then compute legal intervals for each, and then intersect them. The naïve approach need only perform the latter two steps.

	EO1	Rover	DCAPS	ST4
informed	1.8925	.025	.3528	.0315
naïve	.1506	.025	.1090	.0300
random	0	0	0	0

Table 2 Average time to compute intervals, in seconds

Efficacy Assessment

We assess the efficacy of our algorithm in terms of conflict reduction. We compare the number of conflicts reduced for scheduling operations, and the effect this has on solving combined planning/scheduling problems.

In terms of scheduling, ideally, placing an aggregate at a time recommended by a legal times algorithm should result in an improved schedule (i.e. one with fewer conflicts). Table 3 shows the average reduction in the number of conflicts in the schedule after placement of the aggregate. Note that because having an unplaced activity is a conflict, by default each algorithm gets a score of *n* if there are *n* activities in the aggregate just for placing the activity (i.e. in the absence of any state or resource conflicts introduced or removed). We see that the informed algorithm strictly outperforms the others.

	EO1	Rover	DCAPS	ST4
informed	21.2878	1.9042	25.2549	5.800
naïve	19.9978	1.8339	13.8416	0.9997
random	19.2978	-0.7837	21.4041	3.0785

Table 3 Effectiveness through conflict reduction

To assess the effect that the algorithm has in solving planning/scheduling problems, we examine the number of conflicts over time (in terms of CPU usage) and the total number of problems solved for each domain. If superior, the informed search should result in faster reduction of conflicts and more problems being completely solved.

We generate twenty random problems for each domain and run ASPEN with twenty different random seeds for each combination of problem and technique. Note that we do not guarantee that the problems are solvable.

We evaluate the performance of our technique versus the performance of the naïve technique in terms of the number of iterations to solve conflicts, amount of time to solve conflicts, and the total number of problems solved for each domain.

For the EO1 operations domain, the naïve technique and informed technique perform similarly at first. This is because a number of the conflicts do not involve interacting reservations and hence the naïve technique can solve them. However, many of the conflicts involve interacting reservations. Because the informed technique correctly handles these interactions, it is able to solve these conflicts. Thus in the longer term the informed algorithm is able to solve more conflicts in the schedule.

For the Rocky-7 Mars rover operations domain, the informed technique appears to strictly dominate the naïve technique. Interestingly, conflict count rises before it falls for both algorithms. This is due to the added planning necessary to solve conflicts. Adding activities leads to more conflicts initially, but eventually leads to solutions.

For the New Millennium ST4 Landed Operations domain, the informed technique strictly outperforms the naïve technique. Conflict count rises before it falls as in the rover domain and for the same reason, except the algorithm employing the naïve technique never recovers. Many of the activities in a group interact, therefore the naïve technique often makes mistakes in recommending placements for activity groups. Because of this faulty advice, repair using the naïve approach actually increases the number of conflicts in the schedule.

For the DCAPS domain, the informed technique strictly outperforms the naïve technique. In this domain, almost all of the activities in a group interact, leading to similar consequences as the ST4 domain.

In terms of number of problems solved, we observe that ASPEN employing the informed scheduling technique is able to completely solve (i.e., remove all conflicts) more problems than ASPEN employing the naïve approach in all five domains (Table 4).

	EO1	Rover	DCAPS	ST4	total
informed	149/400	390/400	387/400	243/400	1169/1600
naïve	60/400	243/400	1/400	0/400	304/1600

Table 4 Problems Solved

These empirical results imply that aggregate reasoning is effective in real domains, both in terms of number of constraint violations repaired and in terms of overall time to reach a desired solution quality, as long as we use an informed scheduling function. We have computed the statistical confidence that the average final number of conflicts using the informed search method is less than the final number of conflicts using the naïve method. For the EO1, ST4, and DCAPS domains, this confidence is greater than 99.9%. For the Rover domain, this confidence is greater than 98%.

Related Work

There are a number of related systems that perform both planning and scheduling. IxTeT (Laborie, P., Ghallab, M. 1995) uses least-commitment approach to sharable resources that does not fix timepoints for its resource and state usages.

HSTS (Muscettola, N. 1993) enforces a total order on timepoints affecting common shared states and resources, allowing more temporal flexibility. We believe that our technique is applicable in this case at a greater computational expense (while still being polynomial), and future research should address this issue.

Both IxTeT and HSTS are less committed representations than our grounded time representation and this flexibility incurs a greater computational expense to detect and/or resolve conflicts.

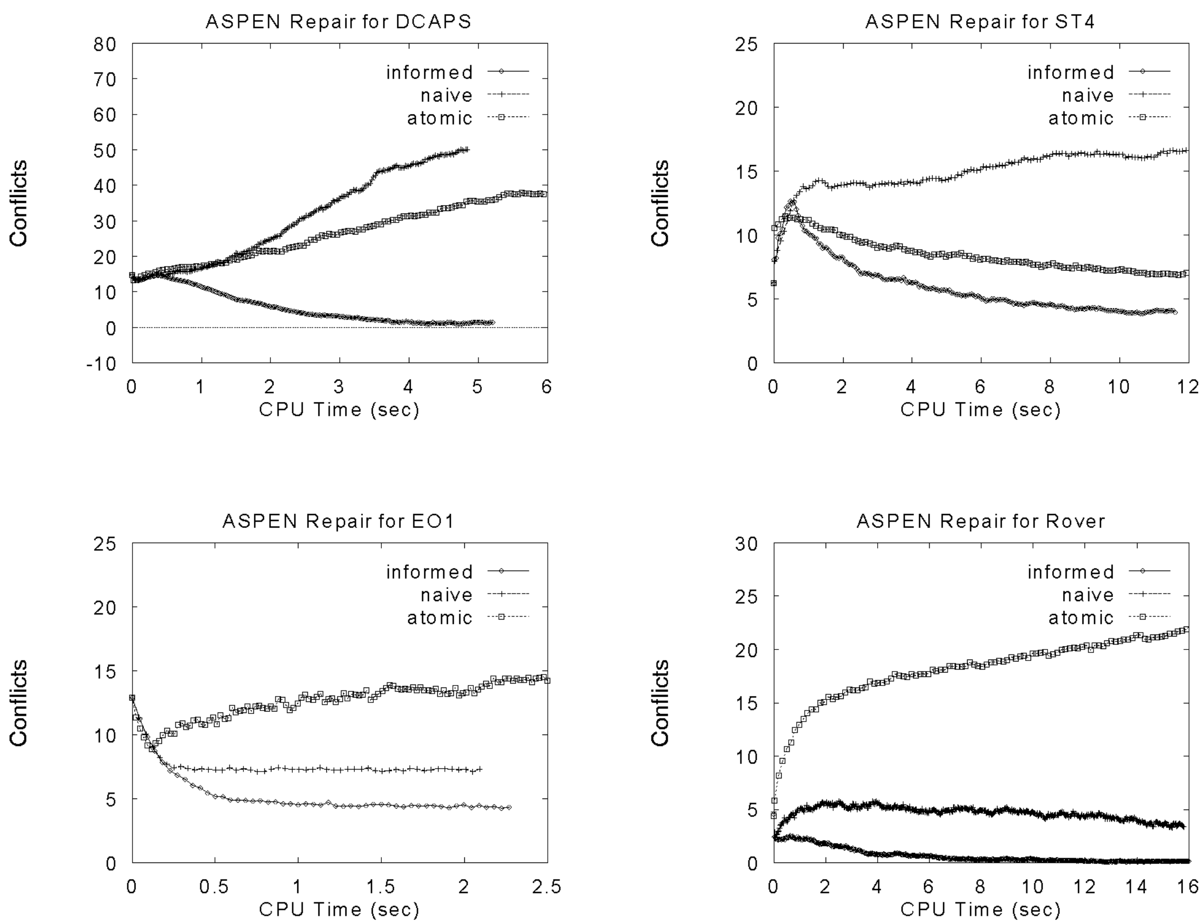

Figure 5 Conflicts over time for domains

O-PLAN (Drabble, B., and Tate A. 1984) also deals with state and resource constraints. O-PLAN's resource reasoning uses optimistic and pessimistic resource bounds to efficiently guide its resource analysis when times are not yet grounded. Like ASPEN, O-PLAN also allows multiple constraint managers which would enable it to perform general reasoning when times are unconstrained and more efficient reasoning in the case where all timepoints are grounded (also enabling aggregate informed search as described in this paper). SIPE-2 (Wilkins, D., 1998) handles depletable/non-depletable resource and state constraints as planning variables using constraint posting and reasons at the same level of commitment as IxTeT.

(Cesta, Oddi S., and Smith S. 1998) apply constraint-posting techniques to satisfy multi-capacitated resource problems at the same level of commitment. Depletable/non-depletable resource constraints are easily transformed to multi-capacitated resource constraints. None of these systems generally consider aggregate operations in their search space.

Conclusion

This paper has described the use of informed transformation techniques to improve the efficiency of scheduling sets of interdependent activities. We describe our algorithm for processing interacting state and resource requirements of a cluster of interdependent activities into a set of independent requirements and use these to search for placements for the activity set. We show empirically that our informed search method outperforms the alternative approach of searching for legal placements on problems from space exploration domains.

Acknowledgements

We wish to thank Stephen Smith of Carnegie Mellon University and Richard Korf of the University of California, Los Angeles for their helpful suggestions concerning this paper. This work was performed at the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration.

References

Cesta, Oddi S., and Smith S. 1998. "Profile-Based Algorithms to Solve Multiple Capacitated Metric Scheduling Problems." *Proc. AIPS98*, pp. 214-223.

Dechter, R., Meiri I., and Pearl J. 1991. "Temporal Constraint Networks," *Artificial Intelligence*, 49, 1991, pp 61-95.

Drabble, B., and Tate A. 1984. "Use of Optimistic and Pessimistic Resource Profiles to Inform Search in an Activity Based Planner," *Proc. AIPS94.*

Estlin, T., Chien, S., and Wang, X. 1997. "An Argument for an Integrated Hierarchical Task Network and Operator-based Approach to Planning," in *Recent Advances in AI Planning*, S. Steel and R. Alami (eds.), Notes in Artificial Intelligence, Springer—Verlag, 1997, pp. 182-194.

Fukunaga, A., Rabideau, G., Chien, S., Yan, D. 1997. "Towards an Application Framework for Automated Planning and Scheduling," *Proc. of the 1997 International Symposium on Artificial Intelligence, Robotics and Automation for Space*, Tokyo, Japan, July 1997.

Kautz, H., and Selman B. 1996. "Pushing the Envelope: Planning, Propositional Logic, and Stochastic Search." *AAAI-96.*

Knight, R., Rabideau, G., and Chien, S. 2000. "Computing Valid Intervals for Collections of Activities with Shared States and Resources." *AIPS-2000.*

Laborie, P., Ghallab, M. 1995. "Planning with Sharable Resource Constraints," Proc. IJCAI-95, 1643-1649

Muscettola, N. 1993. "HSTS: Integrating Planning and Scheduling." *Intelligent Scheduling*. Morgan Kaufmann, March 1993.

Rabideau, G., Chien, S., Backes, P., Chalfant, G., and Tso, K. 1999. "A Step Towards an Autonomous Planetary Rover," *Space Technology and Applications International Forum*, Albuquerque, NM, February 1999.

Rabideau, G., Knight R., Chien S., Fukunaga A., Govindjee A. 1999, "Iterative Repair Planning for Spacecraft Operations Using the ASPEN System," *iSAIRAS '99*, Noordwijk, The Netherlands, June 1999.

Sherwood, R., Govindjee, A., Yan, D., Rabideau, G., Chien, S., Fukunaga, A. 1998. "Using ASPEN to Automate EO-1 Activity Planning," *Proc. of the 1998 IEEE Aerospace Conference*, Aspen, CO, March 1998.

Tate A., Drabble, B., and Dalton, J. 1996. "O-Plan: a Knowledge-based planner and its application to Logistics," *Advanced Planning Technology, Technological Achievements of the ARPA/RL Planning Initiative*, AAAI Press, 1996, pp. 259-266.

Vidal, V., and Regnier, P., 1999. "Total Order Planning is More Efficient than we Thought." *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, AAAI Press, 1999, pp. 591-596.

Wilkins, D., 1998. "Using the SIPE-2 Planning System: A Manual for Version 4.22." SRI International Artificial Intelligence Center, Menlo Park, CA, November 1998.

Zweben, M., Daun, B., Davis, E., and Deale, M., 1994. "Scheduling and Rescheduling with Iterative Repair," in *Intelligent Scheduling*, Morgan Kaufman, 1994.