Towards Probabilistic Plan Management

Laura M. Hiatt
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213
lahiatt@cs.cmu.edu

Reid Simmons
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
reids@cs.cmu.edu

Abstract

In temporally uncertain domains, taking uncertainty into account while planning leads to problems with scalability. One alternative to this is to plan deterministically and replan when execution deviates from schedule. In large, complex problems, however, replanning during execution can be prohibitively expensive. To address this, we have developed a general plan management framework called Probabilistic Plan Management (PPM). PPM probabilistically limits how far in the future it is necessary to consider tasks while repairing and replanning during execution. PPM also decides whether to replan based on the *probability* that a violated constraint will occur in execution, not on the presence of a conflict in the plan. These features decrease replanning during execution while ensuring that the quality of execution does not unduely suffer. In this paper, we describe our approach and discuss results in simulation that show large savings in the total time spent replanning during execution.

Introduction

Temporal uncertainty is characteristic of many complex, real-world domains. Ideally, one would take this uncertainty into account while planning. Because of the complexity of the problem, however, this as of yet cannot be done in a scalable, expressive way. Prottle (Little, Aberdeen, & Thiébaux 2005), ΔDUR (Mausam & Weld 2006) and Generalized Semi MDPs (Younes & Simmons 2004) all have yet to demonstrate sufficient scalability to handle realistically large problems.

A common alternative to planning under uncertainty is to plan with deterministic task durations and replan during execution in the face of contingencies. Mausam & Weld (2006), in fact, show that the ΔDUR planner that does so outperforms the ΔDUR planner that explicitly models temporal uncertainty. FF-rePlan is another such planner, which had large success in The Fifth International Planning Competition (Bonet & Givan 2006). During the schedule management process of executing, replanning and continuing with execution, however, we still want to take advantage of what we know about the duration distributions as much as possible.

As an example of this, think about what a person might do while executing a schedule with temporal uncertainty. Consider a person planning to run errands during the day and finish in time to meet friends for dinner at 7:00 pm. After they complete their first errand, they notice it took longer than anticipated. Still, the person continues running their errands as planned: they realize that other errands are likely to take less time than scheduled, and that they will probably still be on time for dinner. They will only rearrange their schedule if it becomes clear that they are going to be late for dinner otherwise. Furthermore, during this whole process, they think only about tasks scheduled in the near future. It is not necessary, for example, to think about the impact the late errand will have on their plans to go bowling next week or to change those plans if they have to rearrange their schedule.

Drawing on these ideas, we have developed a framework called Probabilistic Plan Management (PPM) that manages the execution of a plan using two novel components. Before the execution of each task, it calculates the *uncertainty horizon*, which limits the scope of tasks to be considered while managing the plan. This horizon represents a probabilistic boundary between tasks that are near in the future and likely to be executed as scheduled, and tasks that are far enough in the future that we do not care about conflicts with them.

After each task is executed, PPM uses *probabilistic flexibility* to determine whether a violated constraint within the uncertainty horizon is likely enough to occur to necessitate a replan. It calculates this likelihood using the actual duration distributions of the tasks, not the discrete scheduled duration. If the probability is low, it performs a fast repair method to eliminate conflicts within the horizon, if any, and continues with execution. Otherwise it replans.

We have implemented a preliminary version of PPM for total-order plans. When run in simulation, it shows large improvements in the total amount of time spent managing the execution of a plan.

Related Work

There are a variety of approaches that support decreasing the time spent replanning during execution. They fall into two main categories: decreasing the number of replans during execution by increasing the flexibility of a plan, and making each replan episode as fast as possible.

Increasing Plan Flexibility

Whereas fixed-times schedules specify only one valid start time for each task, flexible-times schedules allow execution intervals for tasks to drift as long as they satisfy the schedule's constraints, for example (Smith *et al.* 2006). This allows the system to adjust to unforeseen temporal events as far as the constraints will allow, triggering a replan only if all constraints in the current schedule cannot be satisfied.

IxTeT (Laborie & Ghallab 1995) is a temporal planner that generates partial-order plans with unbound variables. This allows IxTeT to repair plans at run time by searching through the partial plan space. If a valid plan does not exist in the partial plan space, however, the necessitated full replan has the potential to be even more expensive than it is for total-order planners.

Policella *et al.* (2004) explore different ways of generating robust partial-order schedules. By taking a fixed-time schedule and expanding it to be partial-order, they see improvements with respect to fewer precedence orderings, temporal slack and disruptability over a schedule generated using a typical least-commitment approach.

Replanning Efficiency

O-Plan (Tate, Drabble, & Kirby 1994), the Open Planning Architecture, is an HTN planning system that combines both iterative repair and least commitment in its approach to planning. It represents plans as sets of constraints, and uses incremental constraint algorithms to repair the plans when constraints alter due to deviations from the plan during execution.

Like PPM, CLEaR (Estlin *et al.* 2001) distinguishes between short-term and long-term tasks. Its horizon, however, provides a distinction between tasks under the control of the executive versus the planner. This is in contrast to our system which distinguishes between tasks which are considered by the planner and tasks which are ignored.

ASPEN (Chien *et al.* 2000) is a planner that efficiently repairs fixed-time, total-order plans by iteratively making local, heuristic and sometimes randomized changes to the current, conflicted plan. While this is often very effective, because it does not perform a methodological search, there are no guarantees on how long it will take before a solution is found. We use this planner in our implementation of PPM.

Approach

During the execution of uncertain, temporal plans, it is likely that task durations will deviate from schedule, causing inconsistencies in the plan. While it is sometimes necessary to replan at such times to avoid missed deadlines and irreparable conflicts later during execution, due to the cost of planning complex problems it is also advantageous to replan infrequently, considering as few tasks as possible.

To explore this trade-off, we have developed a general framework, Probabilistic Plan Management (PPM), which manages plans during execution. PPM uses two novel components to manage plans during execution. It first calculates an *uncertainty horizon*. The uncertainty horizon limits the scope of tasks considered during plan management to tasks that are near enough to the present to be likely to be executed as they are planned, and thus should be conflict free. PPM then uses *probabilistic flexibility* to determine the likelihood of a successful, conflict-free execution of the plan up to the uncertainty horizon. If a violated constraint is too likely to ignore, then it uses a planner to *replan* the tasks within the horizon. Otherwise, a conflict probably will not occur and it uses a fast, incomplete *repair* method to eliminate any existing conflicts within the horizon and maintain a conflict-free plan. Note that we perform the analyses for the execution of every task; we want to consider cases both when a conflict is unlikely but the plan is inconsistent, and when a conflict is likely but the plan is consistent. Note also that by constraints we mean temporal constraints such as serial, min and max constraints. In the rest of this section, we describe the uncertainty horizon and probabilistic flexibility in more detail.

Uncertainty Horizon

In general, we would like to spend time managing only tasks that are near enough to the present to warrant the effort. In the short-term, we want to make sure our plan has a high chance of being successfully executed and so want it to be conflict free. Thus we consider short-term tasks while managing the plan. In the long-term, however, we expect things to go wrong and so do not care too much about conflicts farther in the future; those tasks would just be replanned again and again as other conflicts arise. So, we ignore longer-term tasks when managing the plan. The problem, then, is how to determine which tasks constitute the "short-term."

To formalize this, we introduce the *uncertainty horizon*. The uncertainty horizon is based on the probability that a plan will change between the current time and any given future time due to execution deviating from schedule. We define the uncertainty horizon as the probabilistic boundary between the short-term plan, where the likelihood of change is low, and the rest of the plan, where the likelihood of change is high. We assume here that all tasks t have a normally distributed duration with mean μ_t and standard deviation σ_t; however, other distribution representations could be used.

We begin by finding the range of non-conflicting end times for each task t, $[eet_t, let_t]$, assuming it starts at its scheduled time and using constraint propagation of the plan and its given constraints. We can then find the probability that each task will end in its valid range, $r_t = \text{normcdf}\left(let_t, \mu_t, \sigma_t\right) - \text{normcdf}\left(eet_t, \mu_t, \sigma_t\right)$[1]. The likelihood of some conflict arising during execution up to task T thus is $\left(1 - \prod_t^T r_t\right)$. The horizon is set at the task where the probability of a conflict rises above a user-defined horizon threshold. Intuitively, because the probability of a conflict grows as more tasks are considered, a smaller horizon threshold corresponds to a shorter horizon, and a larger horizon threshold corresponds to a longer horizon. The calculation for a schedule with T tasks is shown in Algorithm 1, where the variable $max\_horizon$ indicates that the horizon includes all scheduled tasks.

We calculate this boundary before the execution of each task and use it to limit the scope of our probabilistic flexibility analysis, ignoring tasks and constraints outside of the

[1]normcdf (X, μ, σ) is the normal cumulative distribution function and calculates the probability that a random variable drawn from $\sim N\left(\mu, \sigma^2\right)$ has a value less than or equal to X.

Algorithm 1: Uncertainty horizon calculation

```
$r = 1$
**foreach** $task$ $t$=1:$T$ **do**
    $r = r \cdot$ (normcdf($let_t, \mu_t, \sigma_t$) $-$
            normcdf($eet_t, \mu_t, \sigma_t$))
    **if** $(1 - r) >$ horizon_threshold **return** $t$
**end**
**return** $max\_horizon$
```

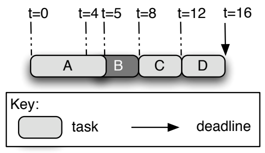

Figure 1: A schedule with four tasks and one deadline where the first task ran late, creating a conflict with the next task.

horizon. As execution progresses, the horizon correspondingly advances, including more and more tasks and constraints that were previously after the horizon. This potentially introduces conflicts, previously ignored, between tasks inside and outside of the horizon, due to overlapping execution intervals, mismatches in task decompositions, and other types of conflicts. Such conflicts, and conflicts arising from execution, are handled by the following probabilistic flexibility analysis.

Probabilistic Flexibility

Probabilistic flexibility distinguishes between a conflict that exists in the plan and a violated constraint that is likely to happen in execution. Consider the simple schedule shown in Figure 1. In this example, the schedule starts at time 0 and has a deadline at time 16. Each task has an assigned duration of 4, with a duration distribution $\sim N(3, 1)$. Note that we initially assign the duration to be $\mu + \sigma$ because we want a reasonable prior likelihood of making the deadline. When executed, task A ran late by 1, creating a conflict in the schedule.

Replanning to fix the conflict, however, is not really necessary: as there are no constraints on the tasks besides the final deadline, it does not matter when the individual tasks start and end as long as task D ends by time 16. Therefore, if we start task C right after task B and task D right after task C, we can effectively treat the three tasks as a single compound task with a combined duration distribution $\sim N(9, 3)$, which starts at 5, when task A ended, and has to end by time 16. The probability it does so is normcdf $\left(11, 9, \sqrt{3}\right) = 87.6\%$. This high probability of success indicates that replanning is not really necessary, as almost 88% of the time the schedule, as is, will execute without missing the deadline. So, we merely need to start task B at time 5, and continue with execution.

Probabilistic flexibility extends this simple example to analyze rich, complex plans. We assume, as with the above example, that tasks are started at the earliest possible time, allowing us to combine tasks' duration distributions and 'discover' extra flexibility in the plan. We first propagate the plan's constraints to find the conflict-free range of valid start and end times for each task (not necessarily including their current scheduled time), $[est_t, lst_t]$ and $[eet_t, let_t]$, given the range of durations each task could have during execution. Then, starting with the first unexecuted task and moving linearly through the plan, we combine the means and standard deviations of the tasks to find the distribution representing the cumulative end time of all tasks up to a given point.

The probability that this distribution has a value in the range $[eet_t, let_t]$, calculated as (normcdf $(let_t, \mu_t, \sigma_t) -$ normcdf (eet_t, μ_t, σ_t)), equals the probability of successful execution up to this time, given no prior conflict, $p(s_t|s_{1:t-1})$ (s_t represents the successful execution of task t). A simple application of the Bayes rule tells us that iteratively multiplying these probabilities together for all tasks gives us the desired probability of no conflict overall: $\prod_t^T p(s_t|s_{1:t-1}) = p(s_{1:T})$. The overall calculation for a plan with T tasks is shown in Algorithm 2, and gives us an approximation of what the probability of no conflict in the plan is.

There are three complications in this algorithm to point out. First, simply summing the means of each of the tasks is not enough. We also need to account for minimal or exact time lags between tasks; e.g. if two tasks each had a mean of 3 but there was a required lag of 4 between them, their joint mean would be 10, not 6. At each iteration, therefore, we update the mean end time to be $\mu = \max(\mu + \mu_t, est_t + \mu_t)$.

Second, at each iteration we also use constraint propagation to update $[est_t, lst_t]$ for each $t < T$, given μ_{t-1}. This allows us to correctly handle situations like the above, where the later starting time of one task may push back the start time of another.

Third, after calculating $p(s_t|s_{1:t-1})$ for some task t, we adjust the new distribution so it approximates the end time distribution given *successful* execution up to that point, so we can more accurately calculate $p(s_{t+1}|s_{1:t})$ in the next iteration of the algorithm. For example, if there were a deadline at time 4 for a task and we assume it executes successfully, then the range of its end time is $[0, 4]$, not $[0, +\inf]$; this truncated distribution is a more accurate one to propagate forward to be combined with the distribution of the following task. We approximate this by finding the mean and standard deviation of the duration distribution truncated at the valid end times for the current task and treating it as the mean and standard deviation of a normal distribution: $\mu, \sigma =$ truncate $(eet_t, let_t, \mu, \sigma)$. To do so, we use the calculation outlined in (Barr & Sherrill 1999).

After each task is executed, probabilistic flexibility uses Algorithm 2 to calculate the probability of a successful execution of the plan up to the uncertainty horizon. This probability is compared with a user-defined flexibility threshold, which specifies how much confidence is required in the successful execution of the plan. A large flexibility threshold corresponds to a more conservative approach, replanning when there is high confidence of successful execution; note that at higher thresholds, it may replan even if there are no conflicts actually in the plan. A small threshold equates to a

Algorithm 2: Probabilistic flexibility calculation

```
$\mu = 0,\ \ \sigma = 0$
**foreach** $task\ i$=1:$T$ **do**
    $\mu = \max\left(\mu + \mu_t, est_t + \mu_t\right)$
    $\sigma = \sqrt{\sigma^2 + \sigma_t^2}$
    $p\left(s_1, ..., s_t\right) = p\left(s_1, ..., s_{t-1}\right) \cdot$
        $\left(\text{normcdf}\left(let_t, \mu, \sigma\right) - \text{normcdf}\left(eet_t, \mu, \sigma\right)\right)$
    **if** $t == T$ **break**
    $\mu, \sigma = \text{truncate}\left(eet_t, let_t, \mu, \sigma\right)$
**end**
**return** $p\left(s_1, ..., s_T\right)$
```

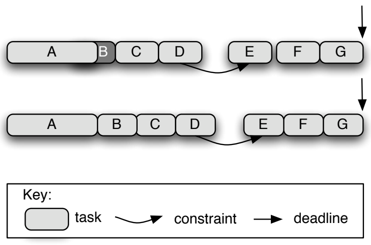

Figure 2: Adjusting a schedule to eliminate conflicts without replanning.

more permissive approach, allowing plans with less certain outcomes to continue being executed without replanning.

If the probability of successful execution does not meet this threshold, the tasks within the uncertainty horizon are replanned. Otherwise, we perform fast plan repair in order to superficially rid the plan of any conflicts that may exist in the horizon. To do so, we use the same valid ranges of start and end times for tasks as above that was generated by propagating plan constraints given the range of durations each task could have. Then we find a new valid plan while minimizing the change in duration, start and end times for each task. An example is shown in Figure 2. After this is done, we once again have a conflict-free plan up to the uncertainty horizon.

Experiments and Results

We implemented a preliminary version of PPM and ran an experiment to quantify its effect on plan execution. We used as our problem domain instances PSP94, PSP100 and PSP107 of the mm-j20 benchmark suite of the resource-relaxed Multi-Mode Resource Constrainted Project Scheduling Problem with Minimal and Maximal Time Lags (MRCPSP/max) (Schwindt 1998), an NP-complete problem. Each problem instance has 20 tasks. Tasks have 2, 3 or 5 modes, or decompositions, and both minimal and maximal time lags. Tasks do not have utility, but all tasks must be executed within their constraints for execution to be considered successful. We used the tasks' benchmark durations as their scheduled durations, but assigned each task t a normal distribution for its duration with a standard deviation that was randomly sampled from the range $[1, 2 \cdot \text{duration}_t/5]$, and a mean of

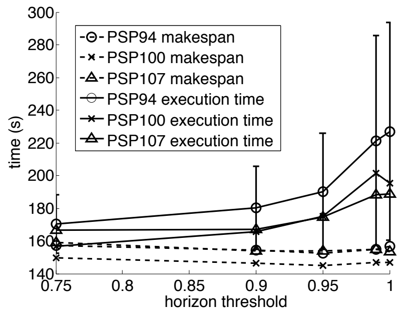

Figure 3: Average makespan and execution time for PPM$_{\text{uh}}$.

$(\text{duration}_t - \sigma_t/2)$; after experimenting with several values these numbers seemed to present the best trade-off of robustness to deviations in execution against efficient execution. Finally, when performing our probabilistic flexibility analysis we added a deadline for the last task in the original schedule, since the problem does not have explicit deadlines represented in it.

To do the constraint propagation required by our analysis, we used a version of the AC3 algorithm (Machworth 1977) that propagates constraints both forward and backward. To generate our initial plan and to perform replans, we used the ASPEN planner, as described in Related Work.

We ran our experiment on a 3.4 GHz Pentium D computer with 2 GB RAM. To simulate task execution, we iteratively picked a random execution duration for each task based on its duration distribution, updated the plan, and used PPM to manage and repair or replan tasks as necessary. The experiment compared four conditions: (1) replan all tasks whenever there is a conflict in the plan, without using PPM (PPM$_{\text{no}}$); (2) use PPM with only the uncertainty horizon, and when there is a conflict in the plan always replan only those tasks within the uncertainty horizon (PPM$_{\text{uh}}$); (3) use PPM with only probabilistic flexibility and perform the probabilistic flexibility analysis on all tasks and repair or replan all of them as necessary (PPM$_{\text{pf}}$); (4) use PPM with both the uncertainty horizon and probabilistic flexibility (PPM$_{\text{full}}$). We performed 250 simulation runs for each condition and problem instance combination.

We also varied the thresholds for the uncertainty horizon and probabilistic flexibility. Recall that a higher horizon threshold corresponds to a longer horizon, where more tasks are replanned each episode, and a lower threshold corresponds to a shorter horizon. Similarly, a larger flexibility threshold corresponds to replanning more and more confident plans.

The results for PPM$_{\text{uh}}$ are shown in Figures 3 and 4. Note that in these graphs a horizon threshold of 1 corresponds to no uncertainty horizon, or PPM$_{\text{no}}$. We stop at a threshold of 0.75 as this is the minimal horizon of looking at only 2-3 tasks in the future. Figure 3 shows both the average makespan of the executed plan as well as the average makespan plus average time spent managing, which we refer to as execution time. It also has error bars for PSP94 showing the standard deviation of the execution time. Figure 4

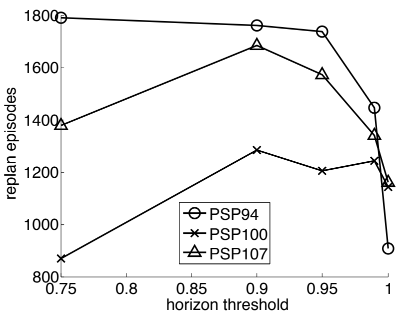

Figure 4: Total number of replan episodes for PPM_{uh}.

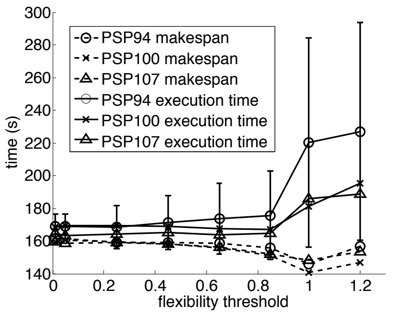

Figure 5: Average makespan and execution time for PPM_{pf}.

shows the total number of replan episodes.

The results for PPM_{pf} are shown in Figures 5 and 6. The flexibility threshold of 1.2 refers to no probabilistic flexibility, or PPM_{no}. Note that this is different than a flexibility threshold of 1, which can potentially still modify the plan such as starting tasks at the earliest possible time or replanning even if there are no conflicts in the plan. Figure 5 shows both the average makespan of the executed plan and the execution time, and Figure 6 shows the total number of replan episodes performed.

Figure 7 shows the percent of runs that were executed to completion for PPM_{pf}. Those that did not complete failed due to the planner timing out during a replan, either because a solution did not exist or the solution was not found in time. All other statistics reported are compiled from only those runs that completed successfully. For the sake of redundancy we omit the same graph for PPM_{uh}.

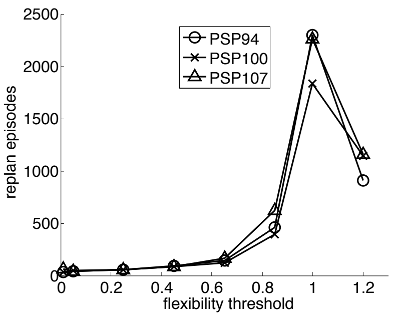

Figure 6: Total number of replan episodes for PPM_{pf}.

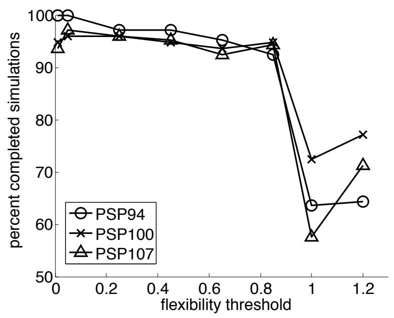

Figure 7: Percent completion for PPM_{pf}.

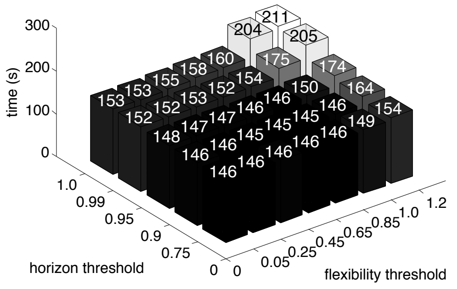

Figure 8: Average execution time of PSP94 for PPM_{full}.

We show the average execution time results for simulations of the PSP94 problem instance using PPM_{full} in Figure 8. Note again that a horizon threshold of 1 corresponds to PPM_{pf}, and a flexibility threshold of 1.2 corresponds to PPM_{uh}. The bar with a horizon threshold of 1 and a flexibility threshold of 1.2, then, is PPM_{no}.

Discussion

PPM_{uh} shows interesting results for the number of replan episodes in Figure 4. As the threshold decreases from 1, the number of replan episodes first tends to *increase* due to needing to replan extra conflicts caused by the uncertainty horizon. At lower thresholds, however, the number of replans levels off and decreases. We believe that here the number of tasks replanned is so few that at this low threshold there are fewer conflicts between tasks inside and outside the horizon, as there are fewer opportunities for conflicts such as mismatched decomposition conflicts.

By comparing Figures 3 and 4 we see that the plan makespan decreases as the number of replans increases, as frequent replanning helps maintain a plan's optimality. Despite the rise in replan episodes, however, the overall execution time still decreases as desired since we are replanning fewer tasks each time.

As Figures 5 and 6 show, PPM_{pf} also makes gains in execution time. At a threshold of 1, the number of replans is much higher than it is for PPM_{no}, due to probabilistic flexibility pessimistically calling for replans on schedules with even the slightest chance of failing during execution. That quickly falls to lower levels and shows large savings in execution time.

Figure 7 shows (as does the corresponding graph for PPM_{uh}) that as both the thresholds decrease we generally see higher, then lower, then often higher completion rates. We believe these fluctuations are due to fewer / more replan episodes timing out because there are fewer / more replan episodes overall. If we were to increase or decrease the amount of time allowed for replanning, we would expect both the completion rate and the time spent replanning both to increase or decrease, respectively. Similarly, for more methodical planners, as the thresholds increase we expect the completion rate and replanning time would both increase as well.

The high variances in Figures 3 and 5 we believe are due to the randomness of execution, as well as the planner.

The above analyses apply when we consider the simulations that incorporate both probabilistic flexibility and the uncertainty horizon shown in Figure 8. As the thresholds decrease, the average execution time in general decreases as it replans fewer tasks less often. Unlike when we used PPM_{uh}, however, when we use the uncertainty horizon and probabilistic flexibility together, as the horizon shortens, the repair overhead and number of replan episodes decrease. We believe this is because probabilistic flexibility helps to avoid the extra replanning in PPM_{uh} caused by conflicts lingering at the horizon. The graph shows also that after a certain point the analysis is not extremely parameter sensitive, and the execution time plateaus at a global minimum.

Overall, the benefit of PPM is clear. Using a horizon threshold of 0.9 and a flexibility threshold of 0.85, PPM_{full} decreases the schedule execution time from an average of 187s, 1071 replan episodes and a 71% completion rate to an average of 144s, 121 replan episodes and a 93% completion rate.

Conclusions and Future Work

The data suggest strong potential for Probabilistic Plan Management, a framework that manages plans during execution. We lessen the amount of time spent replanning during runtime by replanning only tasks in the near future when a conflict is likely. By doing so, we see an average reduction in execution time of 23% on our benchmark problems.

We are currently working on adapting our analysis to manage multi-agent plans with a flexible-times representation and discrete task duration distributions. This involves reasoning about the probabilities of successful execution across agents, as tasks on one agent may affect the deadline of another, and adapting the analysis to account for the increased computational demands of the problem. We also plan to add support for consumable resources, such as fuel or battery consumption. This would allow an agent to reason about the probability that tasks will be successfully executed given the resources available to the agent, as well as shared resources between agents.

Acknowledgements

The authors would like to thank the Jet Propulsion Laboratory for the use of ASPEN for this research.

This research was sponsored in part by NASA grant NNA04CK90A and by National Science Foundation Fellowship No. DGE-0234630. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity.

References

Barr, D. R., and Sherrill, E. T. 1999. Mean and variance of truncated normal distributions. *The American Statistician* 53(4):357–361.

Bonet, B., and Givan, B. 2006. Results of probabilistic track in the 5th international planning competition. In *Proceedings of ICAPS 2006*.

Chien, S.; Rabideau, G.; Knight, R.; Sherwood, R.; Engelhardt, B.; Mutz, D.; Estlin, T.; Smith, B.; Fisher, F.; Barrett, T.; Stebbins, G.; and Tran, D. 2000. Aspen - automated planning and scheduling for space mission operations. In *Space Ops*.

Estlin, T.; Volpe, R.; Nesnas, I.; Mutz, D.; Fisher, F.; Engelhardt, B.; and Chien, S. 2001. Decision-making in a robotic architecture for autonomy. In *Proceedings of iSAIRAS 2001*.

Laborie, P., and Ghallab, M. 1995. Planning with sharable resource constraints. In *Proceedings of IJCAI-05*.

Little, I.; Aberdeen, D.; and Thiébaux, S. 2005. Prottle: A probabilistic temporal planner. In *Proceedings of AAAI-05*. Pittsburgh, PA: AAAI Press.

Machworth, A. K. 1977. Consistency in networks of relations. *Artificial Intelligence* 8(1):99–118.

Mausam, and Weld, D. S. 2006. Probabilisitic temporal planning with uncertain durations. In *Proceedings of AAAI-06*.

Policella, N.; Smith, S. F.; Cesta, A.; and Oddi, A. 2004. Generating robust schedules through temporal flexibility. In *Proceedings of ICAPS 2004*.

Schwindt, C. 1998. Generation of resource-constrained project scheduling problems subject to temporal constraints. Technical Report WIOR-543, Institut fur Wirtschaftstheorie und Operations Research, Universit at Karlsruhe.

Smith, S. F.; Gallagher, A.; Zimmerman, T.; Barbulescu, L.; and Rubinstein, Z. 2006. Multi-agent management of joint schedules. In *Proceedings of the 2006 AAAI Spring Symposium on Distributed Plan and Schedule Management*.

Tate, A.; Drabble, B.; and Kirby, R. 1994. O-Plan 2: an open architecture for command, planning and control. In Fox., M., and Zweben, M., eds., *Knowledge Based Scheduling*. Morgan Kaufmann.

Younes, H. L. S., and Simmons, R. G. 2004. Solving generalized semi-markov decision processes using continuous phase-type distributions. In *Proceedings of AAAI-04*.