Risk-Aware Scheduling throughout Planning and Execution

Andrew J. Wang
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
wangaj@mit.edu

Motivation

Scheduling is integral to many real-world logistics problems. It can be as simple as catching the bus in the morning, or as complex as assembling a commercial airliner. While simple applications render scheduling tools trivial, these tools have not been widely adopted for complex scenarios either. The larger the scenario, the greater the temporal uncertainty throughout the system, and many schedulers do not consider the probabilistic uncertainty in actions' durations. This makes them brittle to temporal disturbances or poor modeling, which incurs high replanning costs and unacceptable risks for the mission-critical nature of large applications.

Mitigating such risk is challenging because temporal reasoning is present at every level of planning and execution. Figure 1 diagrams the layers of reasoning for a planning executive to map logistical goals into real-world actions. First, the planner generates an grounded plan. The the scheduler produces a scheduling policy, which the dispatcher then follows to execute the plan's actions at appropriate times.

Although the scheduler is responsible for creating the scheduling policy, the planner and dispatcher still have to be aware of time: The dispatcher is responsible for keeping the plan on schedule. Hence, it needs to monitor actions' durations to make sure they match what the policy predicts. Likewise, the planner needs to know when the scheduler cannot produce a valid policy for a given plan, and use that information to avoid unschedulable plans.

When temporal uncertainty is factored into the scheduling algorithm, the planner and dispatcher must adapt to reason about the consequences of uncertainty as well. Thus, quantifying and managing scheduling risk requires one to apply probabilistic temporal reasoning across the entire planning and execution architecture. If this could be done efficiently, it would remove a significant obstacle to deploying large-scale logistics scenarios with confidence.

Problem statement

The goal of this thesis is to provide probabilistic guarantees on a plan's scheduling requirements. The approach is to augment an existing non-probabilistic planning and execution architecture to reason about risk-aware scheduling. In this architecture, the planner and scheduler first generate a plan and scheduling policy offline, which the dispatcher then executes online. However, if real-world disturbances break the policy, then the scheduler and planner may be invoked online incrementally to repair the remaining plan and schedule.

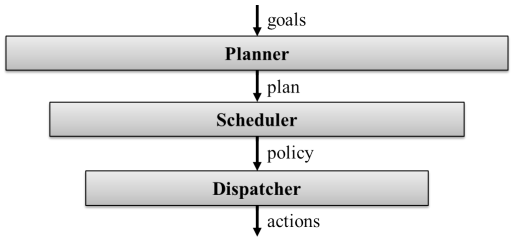

Figure 1: The role of scheduling in a plannning and execution architecture.

At its core, this work addresses the problem of chance-constrained scheduling, where the generated policy is guaranteed to satisfy a set of constraints that bound the risk of violating certain temporal requirements. A chance-constrained formulation was chosen over risk minimization for two reasons: First, risk minimization is an infinite goal that leads to overly conservative solutions, whereas a chance constraint specifies a clear bound to aim for. Second, multiple chance constraints may be specified over different subplans, which together indicate the user's risk priorities.

Given this problem framing, the proposed modifications to the architecture are as follows: The planner accepts a set of chance constraints accompanying the planning goals. When it generates a candidate plan, it also identifies what portions of the chance constraints would be affected by scheduling decisions, and sends those down to the scheduler. Now the scheduler does not just verify that the plan is temporally consistent, but that a scheduling policy exists which succeeds with sufficient likelihood in the context of each chance constraint. If such a policy cannot be found, then the planner tries again until the scheduler is successful.

When the dispatcher receives the chance-constrained scheduling policy, it must continuously monitor the risk to

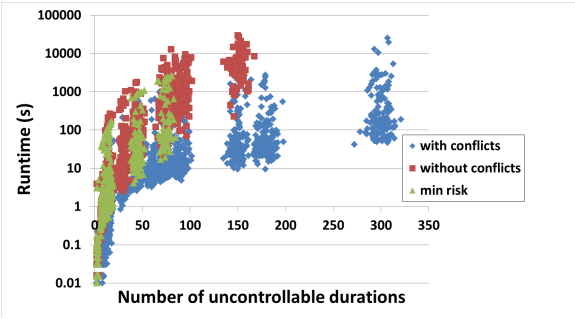

Figure 2: Conflict-directed risk allocation speeds up chance-constrained scheduling by an order of magnitude.

the temporal requirements during execution. If the risk exceeds what the chance constraints allow, then the scheduler is invoked to incrementally repair the scheduling policy. Similarly, if that cannot be done, then the condition is kicked up another level to the planner for incremental plan repair.

This architecture leverages an existing planner (Wang and Williams 2015b), scheduler (Shu, Effinger, and Williams 2005), and dispatcher (Levine and Williams 2014), which have been developed by former and current labmates. The purpose of this work is to augment each module's algorithm and interfaces to support chance-constrained scheduling.

Current work

The offline scheduling component has been developed and published recently (Wang and Williams 2015a). This constitutes the core algorithm, because the form in which it expresses the scheduling policy determines how the planner and dispatcher will interface to it. The algorithm is based on the concept of risk allocation, which has been applied in the contexts of path planning (Ono and Williams 2008) and scheduling (Fang, Yu, and Williams 2014). In risk allocation, a chance constraint's risk bound gets allocated to relevant portions of the plan, allowing one to bound the risk locally and independently.

The novel contribution in this work is to make risk allocation iterative, by discovering temporal conflicts that can help improve the risk allocation. This makes the solution process efficient because the problem size starts small and grows gradually through subsequent iterations. In contrast, prior art casts the problem as a single large optimization. The runtime performance of our conflict-directed approach is shown in Figure 2 to rival prior art by about an order of magnitude.

In our approach, a master problem generates candidate risk allocations, while a subproblem checks them for temporal feasibility. The risk allocations map the original probabilistic plan into a deterministic form called the simple temporal network with uncertainty (STNU). Therefore, the subproblem may leverage efficient STNU controllability algorithms (Vidal and Fargier 1999) to compute scheduling policies that are chance-constrained by the master's risk allocation. These algorithms can discover temporal conflicts, too, which are sent to the master for further consideration.

Future timeline

The goal is to integrate the core scheduling algorithm into a comprehensive planning and execution architecture over the next three years. The first year will focus on interfacing above with the generative planner. When a plan is passed to the scheduler, the scheduler needs to know, for each chance constraint, the relevant actions in the plan, in order to allocate that chance constraint's risk bound. This step requires a semantic distinction between temporal requirements and actions' durations, which the scheduler ignores, so the planner must reason about it. When the scheduler deems a plan temporally infeasible with respect to the chance constraints, it must also inform the planner why, so such scenarios may be avoided in further candidate plans. The reason is precisely that the set of temporal conflicts learned by the subproblem are incompatible with the provided chance constraints.

The second and third years will then address incremental rescheduling with respect to the dispatcher and the planner. When the original plan is modified—either through chance constraints, temporal requirements, or action durations—these incremental changes may be captured so that the scheduler can make corresponding updates to the policy, rather than compute it from scratch. When the dispatcher observes actions' durations that have exceeded their risk allocation bounds, the scheduler should update its model of past durations and propagate them within the prevailing policy. If the policy cannot be successfully updated, then the scheduler invokes the planner's incremental replanning interface, which could modify the chance constraints or the temporal requirements. These requirement changes would need to be propagated through the scheduling policy too.

References

Fang, C.; Yu, P.; and Williams, B. C. 2014. Chance-constrained probabilistic simple temporal problems. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*.

Levine, S. J., and Williams, B. C. 2014. Concurrent plan recognition and execution for human-robot teams. In *Twenty-Fourth International Conference on Automated Planning and Scheduling*.

Ono, M., and Williams, B. C. 2008. Iterative risk allocation: A new approach to robust model predictive control with a joint chance constraint. In *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, 3427–3432. IEEE.

Shu, I.-h.; Effinger, R. T.; and Williams, B. C. 2005. Enabling fast flexible planning through incremental temporal reasoning with conflict extraction. In *ICAPS*, 252–261.

Vidal, T., and Fargier, H. 1999. Handling contingency in temporal constraint networks: from consistency to controllabilities. *Journal of Experimental & Theoretical Artificial Intelligence* 11(1):23–45.

Wang, A. J., and Williams, B. C. 2015a. Chance-constrained scheduling via conflict-directed risk allocation. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.

Wang, D., and Williams, B. C. 2015b. tburton: A divide and conquer temporal planner. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.