Proceedings of the Twenty-Third International Conference on Automated Planning and Scheduling

Scheduling with Contingent Resources and Tasks

Jussi Rintanen*
Department of Information and Computer Science
Aalto University, Finland

Abstract

Finding optimal schedules for the most commonly considered classes of scheduling problems is NP-complete. Best algorithms scale up to very large scheduling problems when optimality is not required and good solution quality suffices. These problems have *perfect information* in the sense that the resource availability, set of tasks, task duration, and other important facts, are fully known at the time of constructing a schedule. However, the assumption of perfect information is rarely satisfied, and real-world scheduling faces several forms of uncertainty, most notably with respect to durations and availability of resources. The effective handling of uncertainty is a major issue in applying scheduling in new areas.

In this work, we investigate the properties of a number of classes of problems of *contingent scheduling*, in which assignments of resources to tasks depend on resource availability and other facts that are only known fully during execution, and hence the off-line construction of one fixed schedule is insufficient. We show that contingent scheduling in most general cases is most likely outside the complexity class NP, and resides, depending on the assumptions, in PSPACE, Σ_2^p or Π_2^p. The results prove that standard constraint-satisfaction and SAT frameworks are in general not straightforwardly applicable to contingent scheduling.

Introduction

There are several approaches to handle uncertainty in scheduling (Beck and Wilson 2007), including the *proactive* approach in which the uncertain future events are explicitly and fully considered during scheduling time, and the *reactive* approach in which first a simplified full-information variant of the problem is solved as if uncertainty did not exist, and unanticipated events are dealt with as they arise. Our work will exclusively focus on proactive approaches, mainly because reactive approaches are in general inherently incomplete. Proactive approaches have further been split to redundancy-based, probabilistic and contingent (Beck and Wilson 2007; Herroelen and Leus 2005). Redundancy-based approaches offer a trade-off between the quality of a schedule and its robustness under delays or failures, by maintaining a reserve of potentially superfluous additional resources. The probabilistic approach observes failure probabilities and delays explicitly, but provides only one conventional schedule that maximizes a performance metric like the makespan. Finally, the contingent approach (which is the focus of this paper) recognizes that different schedules are needed under different contingencies, and computes them either off-line before the execution phase or on-line as information about the contingencies becomes available. This is the most general approach, eliminating the limitations (incompleteness, suboptimality) of the other approaches at the cost of increased complexity.

The fundamental decision problem in scheduling is whether a schedule of a given makespan exists. The most basic forms of scheduling are NP-complete (Garey and Johnson 1979). Different types of uncertainty in scheduling include the following (Davenport and Beck 2000).

1. Uncertainty about task durations
2. Introduction of new tasks during execution
3. Cancellation of existing tasks during execution
4. Uncertainty about resource availability
5. Uncontrollable alternative resources
6. Changes in the due dates
7. Changes in the ready dates

The complexity of scheduling with durational uncertainty follows from known results (Beck and Wilson 2007; Hagstrom 1988): determining the expected duration of a schedule with fixed task and resource choices is #P-hard, where #P is the class of counting problems where the output value corresponds to the number of executions of a nondeterministic Turing machine with a polynomial time bound. This is presumably harder than NP, and "almost" PSPACE-hard since PH$\subseteq$P$^{\#P}$ $\subseteq$PSPACE. We therefore address contingencies other than durational uncertainty, which have been not been investigated before.

Our work explicitly addresses uncertainty of forms 4 and 5. Uncertainty 6 and 7 are reducible to our basic framework, through restrictions on resource availability that force tasks to certain time intervals, but we do not address them explicitly. The overall impact of uncontrollability of tasks (2, 3) is

*Also affiliated with Griffith University, Brisbane, Australia, and the Helsinki Institute of Information Technology, Finland. This work was funded by the Academy of Finland (Finnish Centre of Excellence in Computational Inference Research COIN, 251170).

analogous to uncontrollability of resource availability, but is left out of this paper because of space limitations.

We consider bounded contingencies and worst-case cost measures. When there is no bound on the number of contingencies occurring simultaneously, worst-case cost measures consider each schedule to be as good as one in which all possible failures occur. Therefore, similarly to some applications of contingent scheduling (Drummond, Bresina, and Swanson 1994), we have a guarantee or an assumption that only certain combinations of contingencies can take place. The decision problem we address is testing whether there is a schedule that, in the worst case, doesn't exceed given resource and time constraints. Worst-case costs are preferable to expected costs in many situations, including risk-averseness for example when a schedule is executed only once and cost or time overruns are not an acceptable option. Expected costs are most relevant over long sequences of scheduling scenarios, when the outcomes of an individual scenario is interesting only in terms of its contribution to the overall cost over a long time frame. Also, a schedule that minimizes worst-case resource consumption tends to reduce the variability in different executions of the schedule, which by itself may be desirable.

Our results place contingent scheduling in complexity classes PSPACE, Π_2^p and Σ_2^p. This is the first time a detailed and rigorous complexity analysis of contingent scheduling problems has been carried out. Earlier works, which are algorithm-oriented, have indicated the apparent need for an exponential number of schedules (Drummond, Bresina, and Swanson 1994), suggesting – but not proving – a complexity beyond the class NP. As uncertainty in the kind of formalizations of scheduling we consider can be viewed as nondeterminism, results from probabilistic planning (Littman 1997), which subsumes general forms of scheduling, suggest an EXP upper bound. Our results tighten these obvious upper and lower bounds substantially, demonstrating the feasibility of the simplest quantified variants of CSP and SAT problems for contingent scheduling.

The structure of the paper is as follows. We first define the classes of scheduling problems we analyze in this work. This is followed by the new contributions of the work, in the form of a detailed complexity analysis. Finally, we discuss related work, and we conclude the paper pointing out remaining open problems.

Problem Definition

We focus on unpredictable resource availability and unavailability in connection with worst-case performance measures. Since, in the worst case, any and every resource could be unavailable, we want to consider *bounded* formalizations of resource unavailability, so that the unavailability of a resource typically *guarantees* the availability of another resource. For example, we could have a guarantee that of three resources, at a given time point at least one of them is available. Our formalization can model temporary or permanent *failure* of resources, as well as other dependencies between resources for example caused by two or more high-level resources being dependent on the same exclusive lower level resource.

Since a given resource may fail or otherwise not be available, we also must address problems where the overall success of a schedule does not depend on the availability of any single resource. At the level of individual tasks this means that the completion of the task does not exclusively rely on a single resource, or that completing the task is not a necessary condition for the success of the schedule overall. In the first case there may be several alternative resources that may be used for completing the task, and in the second case there may be alternative tasks of which some need to be completed, but not necessary all of them.

Our formalization of alternative resources is quite general. For example, forms of uncertainty in task duration can be represented in the framework by having, for a given task, alternative resources with different durations to complete the task, and which resource is available is outside the scheduler's control. Task failure (at the point when it is ready to be started) can be represented as all of the relevant resources not being available. There are several other representational possibilities in our definitions which we will not discuss in more detail here.

Apart from resource availability and alternative resources for tasks, our formalization of scheduling is conventional. A scheduling problem consists of a partially ordered set T of *tasks* (with a strict (irreflexive) partial order), a set R of *resources* (machines, for example), and a relation $C \subseteq T \times R \times \mathbb{N}$ which expresses which resources can be used for completing which tasks and what is the duration. Each resource can be utilized by at most one task at a time. To cover the standard job-shop scheduling problem, we also include sets of tasks the execution of which cannot overlap, which can be used for representing the notion of jobs as it arises in job-shop scheduling. Overall, our definitions explicitly or implicitly covers most types of formalizations of scheduling, and then generalizes them to cover commonly occurring types of contingencies.

Although our definitions include a sufficient generality to represent several features of important real-world problems, our model ignores ones that don't interfere with complexity lower or upper bounds. For example, we only model *unary* resources and tasks that don't need more than one resource, because our hardness results don't need more, and it is obvious that more complex models of resources could be easily accommodated in the complexity upper bound proofs (membership in a complexity class.)

A scheduling problem is solved by assigning each task one of the alternative resources, and imposing a total ordering on the tasks that use the same resource. If the transitive closure of the union of the partial order on all tasks with the total orderings for the resources is irreflexive, then the assignment of resources to tasks qualifies as a solution. Now the resulting partial ordering can be topologically sorted and each task can be assigned a starting time satisfying the obvious constraints. This assignment of starting times is a *schedule*. The difference between the earliest starting time and the latest completion time of any task in a schedule is its *makespan*.

In a conventional schedule the tasks-resource assignment at a given time point is independent of past events, and hence

the behavior of the schedule is always the same. A contingent schedule, on the other hand, allows different task-resource assignments depending on how the execution of the schedule has proceeded so far. A contingent schedule can be viewed as a tree that assigns a possibly different task-resource assignment for every possible execution.

As we consider *bounds* on the contingencies with respect to resource availability, we have to express what is guaranteed about resources. A common constraint on resource availability is that *at least one* resource is available, or from two or more resources, *exactly one* resource is available. Similarly, a given resource could be available alternatively at different time points. A general language for expressing such dependencies is Boolean logic. We adopt general Boolean formulae, with connectives $\vee$, $\wedge$ and $\neg$, as the language for expressing dependencies in resource availability at different time points.

Since we consider alternative tasks, we also have to be able to specify which combinations of completed tasks are acceptable. Similarly to resources, we express the optionality of some tasks by Boolean formulae. We can restrict to formulas with $\vee$ and $\wedge$, because an objective in scheduling is minimizing costs and makespans, and the minimal number of tasks will be chosen in optimal schedules even without explicitly forbidding them by $\neg$. For example, under minimization, the constraint $t_1 \vee t_2$ is as good as the constraint $t_1 \leftrightarrow \neg t_2$, as no solution that unnecessarily includes both tasks can be preferable to one that includes only one of them.

To model resource *failures* and other forms of uncontrollable and unpredictable resource unavailability, we have a set U of uncontrollable resources.

Finally, the ability of the scheduler to detect the availability of resources is formalized by the parameter $H \in \mathbb{N} \cup \{\infty\}$ for the *observation horizon*. With $H = 1$, the scheduler knows whether $r \in U$ is available only right before the resource could be assigned to a task. With $H = \infty$, all resource availability is known when the execution of a schedule starts. As we will see later, the observation horizon is critical in determining the complexity of scheduling.

The availability of resources in $R \backslash U$ is determined solely by the use of other resources which are interdependent through the resource availability formulae α. Obviously, if the scheduler has already committed to using the resource r_1 at time 1, and the formula α entails $\neg(r_1@1 \wedge r_2@1)$, then the resource r_2 is not available at 1.

Definition 1 *A* scheduling problem *consists of the following components.*

1. *a finite set T of tasks*
2. *an irreflexive relation $<\subseteq T \times T$*
3. *a symmetric relation $\nsim \subseteq T \times T$*

The two relations respectively express ordering constraints for tasks and whether tasks are not allowed to overlap.

4. *a finite set R of resources*
5. *a set $U \subseteq R$ of* uncontrollable *resources*
6. *a relation $C \subseteq T \times R \times \mathbb{N}$*

This relation tells which resources can be used for the task and what is the duration of the task with this resource.

7. *a formula α on T (*the task specification*)*
8. *a formula β on $R \times \{0, \ldots, last\}$ (*the resource specification*)*
9. *an integer $last \in \mathbb{N}$ (*the makespan*)*
10. *$H \in \mathbb{N} \cup \{\infty\}$ (*the observation horizon*)*

The task specification α characterizes which combinations of tasks are sufficient. The resource specification expresses dependencies between resources in terms of their availability.

The execution of a schedule is characterized by sequences $R_0, \ldots, R_{last}$ and $S_0, \ldots, S_{last}$ respectively for the resources available and the task-to-resource assignments for the tasks starting at each time point. The sets R_i, which satisfy $(R \backslash U) \subseteq R_i \subseteq R$, are determined by the environment, and the sets $S_i \subseteq T \times R$ are determined by the scheduler. We often write the pairs $(r, t) \in S$ as $r@t$ for better readability.

Define $\mathit{available}_n(R_n) = (R_n \times \{n\}) \cup \{\neg(r, n) | r \in R \backslash R_n\}$. The sets R_i have to satisfy the following constraint.

1. The resource specification β is satisfied: $\beta \cup \bigcup_{i=0}^{n} \mathit{available}_i(R_i)$ is consistent.

For sets $S \subseteq T \times R$ we define $\mathit{tasks}(S) = \{t | (t, r) \in S \text{ for some } r \in R\}$, and $\mathit{resources}_n(S) = \{(r, n + i) | (t, r) \in S \text{ for some } t \in T, (t, r, d) \in C \text{ for some } d \in \mathbb{N}, 0 \leq i < d\}$. The latter is for the resources reserved by tasks in S for their duration.

Now the scheduler has to decide about the task-to-resource assignment $S_n \subseteq T \times R_n$ for the current time point based on $R_0, \ldots, R_{n-1+H}$ and subject to the following constraints.

1. Resources dependencies are obeyed: $\beta \cup \bigcup_{i=0}^{n} \mathit{resources}_i(S_i) \cup \bigcup_{i=0}^{n-1+H} \mathit{available}_i(R_i)$ is consistent.
2. Any resource is used by at most one task at a time: For all $(t, r) \in S_n$, there is no $(t', r) \in S_i$ for any $i \in \{0, \ldots, n\}$ such that $(t', r, d) \in C$ for $d > n - i$.
3. Exclusive tasks may not overlap: For all $(t, r) \in S_n$ there is no $(t', r') \in S_i$ for any $i \in \{0, \ldots, n\}$ such that $t \nsim t'$ and $(t', r', d) \in C$ for $d > n - i$.
4. The ordering $<$ is followed: $j \geq i + d$ whenever $(t, r) \in S_i$ and $(t', r') \in S_j$ and $t < t'$ and $(t, r, d) \in C$.
5. Tasks are started at most once: $\mathit{tasks}(S_i) \cap \mathit{tasks}(S_j) = \emptyset$ whenever $0 \leq i < j \leq \mathit{last}$.

The task specification is satisfied: $\alpha \cup \bigcup_{i=0}^{\mathit{last}} \mathit{tasks}(S_i) \cup \{\neg t | t \in T \backslash \bigcup_{i=0}^{\mathit{last}} \mathit{tasks}(S_i)\}$ is consistent.

Analysis of Complexity

Our results characterize the complexity of contingent scheduling in terms of the following features.

alternative tasks	Some combination of tasks has to be completed, not necessarily every task.
alternative assignments	For a task, there may be more than one resource that can be used for completing it.
alternative resources	Availability of a resource may be conditional on the availability or unavailability of other resources.
uncontrollable resources	Resources may be unavailable due to external uncontrollable reasons.

We formalize these features as follows.

Definition 2 *An instance of contingent scheduling has*

1. alternative tasks *if and only if* α *is not logically equivalent to* $\bigwedge T$,
2. alternative assignments *if and only if there are* $(t, r, d) \in C$ *and* $(t, r', d') \in C$ *such that* $r \neq r'$,
3. alternative resources *if and only if* β *is not logically equivalent to* $\bigwedge R$, *and*
4. uncontrollable resources *if and only if* $U \neq \emptyset$.

As pointed out earlier, uncontrollable resources without dependencies between them would mean unavailability of all resources in the worst case. With worst-case performance measures this would be uninteresting, as it would usually mean impossibility to schedule. This is why we always assume alternative resources whenever uncontrollability is present. Similarly, if resources are uncontrollable, we require either alternative tasks or for fixed tasks alternative ways of assigning resources to the tasks. Otherwise we would either be back to the inevitable failure in the worst case, or the uncontrollable resources would be irrelevant to the tasks, both cases being uninteresting.

So for the two cases with uncontrollable resources, we also consider alternative tasks or alternative resources for (a fixed set of) tasks, as listed in Table 1.

Next we proceed with the presentation of the main technical results of the work. The first result is a reduction from SAT to our formalization of scheduling. showing it NP-hard. The NP-hardness of several important scheduling problems such as job shop scheduling is already known (Garey and Johnson 1979), and we give this result because the main construction in the proof, which is new, will be used in subsequent proofs for establishing PSPACE-hardness of scheduling with uncertainty.

Theorem 3 *Scheduling without uncontrollable resources is NP-complete.*

Proof: Membership in NP is trivial: guess a schedule (tasks and an assignment of resources for all tasks in all time points) and verify in polynomial time that all constraints required for valid schedules are satisfied.

For the hardness proof, we give a reduction from 3-SAT with at most 3 occurrences of each propositional variable in the clause set. This restriction of 3-SAT is NP-complete (Garey and Johnson 1979).

feature	no uncertainty	problem 1	problem 2	not considered
uncontrollable resources		✓	✓	✓
alternative tasks	✓	✓		
alternative assignments	✓		✓	
	NP, Theorem 3	PSPACE, Theorem 5	PSPACE, Theorem 6	uninteresting

Table 1: Contingent scheduling problems

The scheduling problem has the following properties. The task set is unordered. All tasks have unit duration. Each task belongs to a set of three alternative tasks. Tasks correspond to literals and the sets of three alternative tasks correspond to clauses. Two alternative resources correspond to the two truth-values of a propositional variable.

The timeline is partitioned to segments of length 3, and in each segment, either the resource r_i or the resource $\widehat{r_i}$ is available. Hence either the tasks requiring resource r_i or the tasks requiring resource $\widehat{r_i}$ can be completed. By assumption, there are at most 3 of each type of tasks and hence all of them can be completed in the 3-step segment if needed.

Next we formalize this in detail. Let the 3-SAT instance consist of n propositional variables $p_1, \ldots, p_n$ and m clauses $I = \{c_1, \ldots, c_m\}$. The ith literal of clause c is denoted by $\text{lit}_i(c)$. The scheduling problem is as follows.

1. The tasks are $T = \{l_i^j | 1 \leq i \leq m, 1 \leq j \leq 3\}$.
2. The task ordering $<$ is the empty relation.
3. The task overlap relation $\nsim$ is the empty relation.
4. The resources are $R = \{r_1, \widehat{r_1}, \ldots, r_n, \widehat{r_n}\}$.
5. $U = \emptyset$
6. $C = \{(l_i^j, r_k, 1) | \text{lit}_j(c_i) = p_k, 1 \leq i \leq m, 1 \leq j \leq 3\} \cup \{(l_i^j, \widehat{r_k}, 1) | \text{lit}_j(c_i) = \neg p_k, 1 \leq i \leq m, 1 \leq j \leq 3\}$
7. $\alpha = \bigwedge_{i=1}^{m} (l_i^1 \vee l_i^2 \vee l_i^3)$
8. $\mathit{last} = 3n - 1$

9. Define

$$\theta_{i,k} = \bigwedge_{j=0}^{2} r_i@(3k-3+j)$$
$$\widehat{\theta}_{i,k} = \bigwedge_{j=0}^{2} \widehat{r_i}@(3k-3+j)$$
$$\neg\theta_{i,k} = \bigwedge_{j=0}^{2} \neg r_i@(3k-3+j)$$
$$\neg\widehat{\theta}_{i,k} = \bigwedge_{j=0}^{2} \neg\widehat{r_i}@(3k-3+j)$$
$$\beta = \bigwedge_{i=1}^{n} \Big((\theta_{i,i} \wedge \neg\widehat{\theta}_{i,i}) \vee (\neg\theta_{i,i} \wedge \widehat{\theta}_{i,i})\Big) \wedge \bigwedge\{\neg\theta_{i,j} \wedge \neg\widehat{\theta}_{i,j} | 1 \leq i \leq n, 1 \leq j \leq n, i \neq j\}$$

10. $H = 1$ (The value is irrelevant since everything is predictable.)

Clearly, given the 3-SAT instance, the scheduling problem can be constructed in polynomial time.

Assume I is satisfied by some assignment $v : \{p_1, \ldots, p_n\} \to \{0, 1\}$. We show that the scheduling problem has a solution.

Let $R_i = R$ for all $i \in \{0, \ldots, \mathit{last}\}$. Define, for every $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, 3\}$, $S_{3i-3+(j-1)} = \{(l_k^h, r_i)\}$ if the hth literal in c_k is p_i and c_k is the jth clause with an occurrence of p_i, and $S_{3i-3+(j-1)} = \{(l_k^h, \widehat{r_i})\}$ if the hth literal in c_k is $\neg p_i$ and c_k is the jth clause with an occurrence of $\neg p_i$.

For the scheduling problem to have a solution, the makespan bound $\mathit{last} = 3n - 1$ must be satisfied, some subset of the tasks have to completed to satisfy the task specification α, and the resource usage must comply with β.

By construction of our sets S_i, we have used only one of the two resources available in any of the 3-step schedule segments. Hence β is satisfied.

Let $l_k^1 \vee l_k^2 \vee l_k^3$ be any of the conjuncts of α. We will show that at least one of these three tasks is completed. By construction, there is a corresponding clause c_k in I. At least one of the literals in c_k is true under v. Let it be the first one $\neg p_i$ for some $i \in \{1, \ldots, n\}$ (proof for the unnegated case and for other positions is analogous.) By construction of the sets $S_0, \ldots, S_{\mathit{last}}$, we have $S_{3i-3+(j-1)} = \{(l_k^1, \widehat{r_i})\}$, where c_k is the jth clause with an occurrence of $\neg p_i$. Hence one of the three tasks in $l_k^1 \vee l_k^2 \vee l_k^3$ is completed. As this holds for any conjunct of α, the schedule is a valid solution.

For the proof in the other direction, consider a solution $R_0, \ldots, R_{\mathit{last}}, S_0, \ldots, S_{\mathit{last}}$ of the scheduling problem. Construct an assignment $v : \{p_1, \ldots, p_n\} \to \{0, 1\}$ as $v(p_i) = 1$ if $S_{3i-3+(j-1)} = \{(l, r_i)\}$ for some l and j, and $v(p_i) = 0$ otherwise. As for every conjunct $l_k^1 \vee l_k^2 \vee l_k^3$ of α one of the tasks is completed at some time point, one of the literals in the corresponding clause is satisfied by v by construction. Hence v satisfies I. □

The proof of the next theorem generalizes that of Theorem 3. Instead of the scheduler selecting the resources for every time point, at some points the selection of the available resources is outside the control of the scheduler. This complicates the task of the scheduler because it is no longer known in advance which resources will be available. This lifts the complexity from NP to PSPACE.

Lemma 4 *Deciding whether a QBF in 3-CNF and with at most 3 occurrences of any variable in the body is true, is PSPACE-complete.*

Proof: The construction is straightforward and essentially the same as for 3-SAT, introducing new variables to represent the fourth and further occurrences of a variable. The only difference is the quantification: the new variables are existential and are inserted in the prefix in front of the most closely following existential variables. □

Theorem 5 *The existence of schedules of any makespan, with alternative tasks, without alternative assignments, with uncontrollable resources, and with an observation horizon 1 is PSPACE-hard.*

Proof: Sketch: We extend the reduction from the proof of Theorem 3 to cover quantified Boolean formulae (QBF), an extension of the SAT problem. Similarly to the proof of Theorem 3, the resources are considered one at a time, in segments of three consecutive time points. The ordering of the resources is the same as the ordering of the corresponding propositional variables in the prefix of the QBF. By Lemma 4 we can restrict to QBF with at most three occurrences of any variable in the body and still have PSPACE-hardness.

The controllability of each resource depends on whether it is universally or existentially quantified. The $\forall$ variables of the QBF become the uncontrollable resources, and the $\exists$ variables are controllable ones. Hence $U = \{r_i \in R | p_i$ is universal in the QBF$\}$.

For this reduction it is essential that the observation horizon H equals 1, so that any decision by the scheduler depends only on the resource availability so far and is independent of the resource availability in the future. Otherwise the reduction is as in Theorem 3.

Testing the truth of the QBF, a PSPACE-complete problem (Garey and Johnson 1979), coincides with the solvability of the scheduling problem, which is consequently PSPACE-hard. □

In the above proof we used alternative tasks. Next we show that PSPACE-hardness continues to hold if we use alternative assignments instead.

Theorem 6 *The existence of schedules of a given makespan, without alternative tasks, with uncontrollable resources, with alternative assignments, and with an observation horizon 1 is PSPACE-hard.*

Proof: We adapt the ideas of the previous proofs to this problem. We consider two ways of performing each task, one with a resource with which the duration is short (1) and another with a long duration ($4n$). The resource with the long duration is exclusive to the task in question. The short resources come in pairs, similarly to the resources in the previous proofs. A clause corresponds to three tasks. The long resources for the three tasks may not temporally

overlap. The satisfaction of the clause corresponds to completing one of the tasks with a short resource. If the short resource is not available, the task must be completed with a long resource. If the tasks of one of the clauses are completed with long resources only, the makespan of the schedule is necessarily at least $12n$. If all of the clauses are completed with one or more short resources, the makespan is at most $3n + 4n + 4n = 11n$, using 3 instances of each of the n short resources in an ordering determined by the prefix of the QBF, followed by completing all the remaining tasks with long resources, with the up to two tasks for each clause sequentially with duration $4n+4n$ but the tasks for different clauses in parallel, yielding a makespan $3n + 4n + 4n$.

The QBF is true if and only if there is a schedule with a makespan of at most $11n$.

Similarly to the previous proof, we introduce uncontrollable resources, which allows encoding the quantifier alternation in QBF, leading to PSPACE-hardness.

Let the QBF consist of n propositional variables $p_1, \ldots, p_n$, ordered in the prefix in this order, and m 3-literal clauses $I = \{c_1, \ldots, c_m\}$. The predicate $\text{univ}(p_i)$ indicates whether the variable is universally or existentially quantified. The ith literal of clause c is denoted by $\text{lit}_i(c)$. By Lemma 4 we restrict to QBF with at most three occurrences of any variable in the body. The scheduling problem is as follows.

1. The tasks are $T = \{l_i^j | 1 \leq i \leq m, 1 \leq j \leq 3\}$,
2. The task ordering $<$ is the empty relation.
3. The task overlap relation $\nsim$ is the empty relation.
4. The resources are $R = \{r_1, \widehat{r_1}, \ldots, r_n, \widehat{r_n}\} \cup \{r_i^j | 1 \leq i \leq m, 1 \leq j \leq 3\}$. Here r_i^j are the long resources specific to each task.
5. $U = \{r_i | 1 \leq i \leq n, \text{univ}(r_i)\}$
6. $C = \{(l_i^j, r_k, 1) | \text{lit}_j(c_i) = p_k\} \cup \{(l_i^j, \widehat{r_k}, 1) | \text{lit}_j(c_i) = \neg p_k\} \cup \{(l_i^j, r_i^j, 3n) | 1 \leq i \leq m, 1 \leq j \leq 3\}$.
7. $\alpha = \bigwedge_{i=1}^{m} (l_i^1 \wedge l_i^2 \wedge l_i^3)$
8. $\mathit{last} = 12n - 1$
9. $$\begin{aligned} \epsilon = \bigwedge_{i=0}^{\mathit{last}} \bigwedge_{j=1}^{m} (& \neg(r_j^1 @ i \wedge r_j^2 @ i) \wedge \\ & \neg(r_j^1 @ i \wedge r_j^3 @ i) \wedge \\ & \neg(r_j^2 @ i \wedge r_j^3 @ i)) \end{aligned}$$
$$\theta_{i,k} = \bigwedge_{j=0}^{2} r_i @ (3k - 3 + j)$$
$$\widehat{\theta}_{i,k} = \bigwedge_{j=0}^{2} \widehat{r_i} @ (3k - 3 + j)$$
$$\neg\theta_{i,k} = \bigwedge_{j=0}^{2} \neg r_i @ (3k - 3 + j)$$
$$\neg\widehat{\theta}_{i,k} = \bigwedge_{j=0}^{2} \neg\widehat{r_i} @ (3k - 3 + j)$$
$$\begin{aligned} \beta = \epsilon \wedge \bigwedge_{i=1}^{n} & \left((\theta_{i,i} \wedge \neg\widehat{\theta}_{i,i}) \vee (\neg\theta_{i,i} \wedge \widehat{\theta}_{i,i}) \right) \\ & \wedge \bigwedge \{ \neg\theta_{i,j} \wedge \neg\widehat{\theta}_{i,j} | 1 \leq i \leq n, 1 \leq j \leq n, i \neq j \} \end{aligned}$$
10. $H = 1$

The conjunct ϵ in β prevents the overlap of tasks with long duration. This way an unsatisfied clause forces the makespan to be at least $12n$.

We only sketch the argument for the equivalence between the existence of a schedule and the truth of the QBF. As in the previous proof, there is a close correspondence between the AND-OR tree for the QBF and the alternating decisions of the scheduler and the environment/adversary. If the QBF evaluates to *true*, the scheduler has a strategy to always choose resources so that there is at least one short task for every 3-task set, and the makespan is at most $11n$. If the QBF evaluates to *false*, then for some 3-task set all of them are long, entailing makespan $12n$. □

We have shown different variants of the scheduling problem PSPACE-hard. To establish PSPACE-completeness, we additionally need to show membership in PSPACE.

Theorem 7 *Determining whether an instance of contingent scheduling can be solved with given resource bounds is in PSPACE.*

Proof: We only sketch the idea of the proof. The problem can be solved by a depth-first AND-OR tree search algorithm with a depth that is proportional to the number of tasks. The AND-nodes correspond to the environment decisions about resource availability, and the OR-nodes correspond to the scheduler's assignments of resources to tasks. Only a polynomial amount of memory is needed in each node of the search tree for checking that the alternative decisions by the environment and the scheduler satisfy the constraints stated in the definition of executions. □

In the previous PSPACE-hardness results it is essential that the observation horizon is short, so that the scheduler decisions must be interleaved with observations. If the observation horizon is unbounded so that all relevant future contingencies can be observed in the beginning of the execution, then the problem is easier.[1]

Theorem 8 *Scheduling with observation horizon ∞ is Π_2^p-complete.*

Proof: We sketch a proof. Membership in Π_2^p is by an algorithm that tests the existence of a schedule by a nondeterministic polynomial-time Turing machine with an oracle for NP. The Turing machine guesses all the observations (resource availability), and then tests with an NP oracle the possibility of assigning resources to tasks.

Π_2^p-hardness is by a reduction from QBF with the prefix $\forall\exists$. The evaluation problem of this class of QBF is Π_2^p-complete. The reduction follows the previous hardness proofs. The long observation horizon corresponds to the adversary first determining which uncontrollable resources are available, corresponding to outer $\forall$ quantifiers, followed by the scheduler choosing which of the remaining resources to use, corresponding to the inner $\exists$ quantifiers. □

[1] Notice that this is not the same as having no uncertainty, because the question of whether a schedule exists must be answered *before* any of the actual contingent information becomes available.

The above results establish the complexity of making the next-step scheduling decisions (starting a task with given available resources) as definite information about resource availability becomes available. They do not say how and when the reasoning for this is ultimately performed, whether before or during the execution. This is a key question in practical implementations of contingent scheduling,

There are two main approaches to implement the decision making process. The first, often impractical one, involves constructing a tree of scheduling decisions that covers all possible contingencies that may arise. The size of the trees is typically exponential. The advantage of the approach is that during the execution time, no expensive decision making remains. The second approach is to do the computation for the next-step decisions starting from scratch after the relevant information becomes available. The memory consumption in this scenario does not have to be prohibitive, even in the worst case, as shown by our results which placed all the problem variants in PSPACE. However, the computation for solving any NP-hard problem (including ones that are also Π_2^p-complete or PSPACE-complete) takes an exponential time in the worst case and is therefore not necessarily scalable to large problems. The choice between these two approaches is essentially a trade-off between memory consumption (in terms of the size of data structures computed off-line to reduce the amount of computation needed on-line) and the amount of computation needed on-line when facing the next decisions.

A particular trade-off, embodied for example in the work by Drummond et al. (1994), is to allow some incompleteness at the cost of getting both fast on-line decision making during execution and reasonably small contingent schedules. To test whether such small schedules will solve the problem instance at hand is substantially easier than testing the existence of solutions in general.

Theorem 9 *Determining whether an instance of contingent scheduling with a restriction to polynomial size schedules has a solution with a given makespan is Σ_2^p-complete.*

Proof: Sketch: The decision problem can be solved in polynomial time by a nondeterministic Turing machine with an NP-oracle. The Turing machine guesses a contingent schedule in nondeterministic polynomial time, and then uses the oracle to verify that the schedule is valid. The oracle guesses a combination of contingencies and then verifies the validity of the schedule under these contingencies, which takes polynomial time.

Σ_2^p-hardness of contingent scheduling can be established by the proof idea already used in proofs of Theorems 3, 5 and 6, by reducing a $\exists\forall$ QBF to contingent scheduling with polynomial-size schedules. □

Related Work

Drummond et al. (1994) constructed contingent schedules with a bound on the number of failure points for scheduling a telescope. When the number of possible failure points in a schedule is N, and there is a bound B on the number of those failures actually occurring, there are $C = \bigcup_{n=0}^{B} \left| \begin{matrix} N \\ B \end{matrix} \right|$ situations that need to be covered. If B is independent of N as N grows, the number of contingencies is linear in the size of the scheduling problem. Hence, the uncertainty is handled by computing those C schedules, starting the execution of the schedule that is most likely (typically the one that does not involve any failures), and after detecting a failure switching to a schedule for the new failure combination.

Beck and Wilson (2007) consider a a job-shop scheduling problem with stochastic durations. They point out that computing the expected makespan of a given schedule is #P-complete, as indicated by a result by Hagstrom (1988).

Frank and Dearden (2003) analyze the complexity of a number of scheduling problems with uncertainty about resource consumption and job utility, and show them to be NP-complete. In their model, if the resources required for a job are not available, the job is not executed. Otherwise the execution of the schedule proceeds as originally planned. Since the approach does not consider proactive scheduling, it is unsurprising that the complexity is not above NP.

Benedetti et al. (2008) investigate adversarial scheduling problems which, similarly to our problems, allow only a limited amount of uncertainty. Their focus is in the representation and implementation of the problem as QCSP. QCSP is of equal representational power as QBF, and hence PSPACE-complete. A straightforward adaptation of our results justifies the use of QCSP (instead of CSP) in these applications. Also Nightingale (2009) proposes quantified CSPs, in the context of scheduling with machine failures with given probabilities, with the goal of producing a schedule with a success probability exceeding some constant.

The planning problem in AI is related to various scheduling problems. While planning with full information and polynomially sized plans is in NP (Kautz and Selman 1996), contingent planning (worst-case cost measures, partial observability, nondeterminism) under the same restrictions is Σ_2^p-complete (Rintanen 1999; Baral, Kreinovich, and Trejo 2000), and can be solved by reduction to quantified Boolean formulae with two quantifier alternations (Rintanen 2007). With unrestricted (exponential) plan sizes and a polynomial decision horizon the planning problem is PSPACE-complete (Turner 2002), similarly to the contingent scheduling problem with partial observability, whereas with unlimited decision horizons the planning problem is 2-EXP-complete (Rintanen 2004).

Corresponding probabilistic planning problems, formalized as partially observable Markov decision processes (POMDP), with compactly-represented exponentially large underlying search spaces, are similarly very difficult, for example EXPSPACE-complete when restricted to exponentially long decision horizons (Mundhenk et al. 2000) and unsolvable for many infinite horizon problems (Madani, Hanks, and Condon 2003). The complexity of partially observable Markov decision processes coincides with that of our most general contingent scheduling problem when the state set and the transition relation of the POMDP have been represented enumeratively and the length of the decision horizon is proportional to the number of states (Papadim-

itriou and Tsitsiklis 1987). To reduce contingent scheduling to POMDPs, more complex compact representations (Mundhenk et al. 2000) must be used, because the underlying state-space is exponential.

Conclusions

Our results motivate and justify the study of contingent scheduling with new solution methods: Π_2^p-hard and PSPACE-hard problems are generally not practically solvable in standard constraint satisfaction frameworks such as CSP, SAT or integer linear programming (IP). Quantified variants of these problems, including quantified Boolean formulae (QBF) (Stockmeyer 1976) and quantified CSPs (Bordeaux and Monfroy 2002) do have sufficient power to represent Π_2^p-complete problems and everything until PSPACE. The question is, how do these quantified problems – as representational frameworks – help developing more effective methods for contingent scheduling.

Open questions remain. We already referred to the #P-hardness of the most basic questions about fixed schedules with uncertainty about task durations. Conditioning task selection according to already observed durations would seem to lift the complexity of scheduling further, as we would have a combination of expected costs and alternation between the environment decisions and the scheduler decisions. This raises the question whether the problems are complete for PSPACE or some class above it, with implications to possible reductive approaches to scheduling. The #P-complete scheduling problem with durational uncertainty can be reduced to (weighted) model-counting problems (Bacchus, Dalmao, and Pitassi 2003) in polynomial time. Which reductive methods have sufficient expressivity when other forms of contingencies are introduced?

References

Bacchus, F.; Dalmao, S.; and Pitassi, T. 2003. Algorithms and complexity results for #SAT and Bayesian inference. In *Foundations of Computer Science, 2003. Proceedings. 44th Annual IEEE Symposium on*, 340–351. IEEE.

Baral, C.; Kreinovich, V.; and Trejo, R. 2000. Computational complexity of planning and approximate planning in the presence of incompleteness. *Artificial Intelligence* 122(1):241–267.

Beck, J. C., and Wilson, N. 2007. Proactive algorithms for job shop scheduling with probabilistic durations. *Journal of Artificial Intelligence Research* 28:183–232.

Benedetti, M.; Lallouet, A.; and Vautard, J. 2008. Modeling adversary scheduling with QCSP+. In *Proceedings of the 2008 ACM Symposium on Applied Computing*, 151–155. ACM.

Bordeaux, L., and Monfroy, E. 2002. Beyond NP: Arc-consistency for quantified constraints. In *Proceedings of the 8th International Conference on Principles and Practice of Constraint Programming*, 17–32. Springer-Verlag.

Davenport, A. J., and Beck, J. C. 2000. A survey of techniques for scheduling with uncertainty. Unpublished manuscript.

Drummond, M.; Bresina, J. L.; and Swanson, K. 1994. Just-in-case scheduling. In *Proceedings of the 12th National Conference on Artificial Intelligence*, 1098–1104. AAAI Press.

Frank, J., and Dearden, R. 2003. Scheduling in the face of uncertain resource consumption and utility. In Rossi, F., ed., *Principles and Practice of Constraint Programming – CP 2003: 9th International Conference*, 832–836. Springer-Verlag.

Garey, M. R., and Johnson, D. S. 1979. *Computers and Intractability*. San Francisco: W. H. Freeman and Company.

Hagstrom, J. 1988. Computational complexity of PERT problems. *Networks* 18(2):139–147.

Herroelen, W., and Leus, R. 2005. Project scheduling under uncertainty: Survey and research potentials. *European Journal of Operations Research* 165:289–306.

Kautz, H., and Selman, B. 1996. Pushing the envelope: planning, propositional logic, and stochastic search. In *Proceedings of the 13th National Conference on Artificial Intelligence and the 8th Innovative Applications of Artificial Intelligence Conference*, 1194–1201. AAAI Press.

Littman, M. L. 1997. Probabilistic propositional planning: Representations and complexity. In *Proceedings of the 14th National Conference on Artificial Intelligence (AAAI-97) and 9th Innovative Applications of Artificial Intelligence Conference (IAAI-97)*, 748–754. AAAI Press.

Madani, O.; Hanks, S.; and Condon, A. 2003. On the undecidability of probabilistic planning and related stochastic optimization problems. *Artificial Intelligence* 147:5–34.

Mundhenk, M.; Goldsmith, J.; Lusena, C.; and Allender, E. 2000. Complexity of finite-horizon Markov decision process problems. *Journal of the ACM* 47(4):681–720.

Nightingale, P. 2009. Non-binary quantified CSP: algorithms and modelling. *Constraints* 14(4):539–581.

Papadimitriou, C. H., and Tsitsiklis, J. N. 1987. The complexity of Markov decision processes. *Mathematics of Operations Research* 12(3):441–450.

Rintanen, J. 1999. Constructing conditional plans by a theorem-prover. *Journal of Artificial Intelligence Research* 10:323–352.

Rintanen, J. 2004. Complexity of planning with partial observability. In *ICAPS 2004. Proceedings of the Fourteenth International Conference on Automated Planning and Scheduling*, 345–354. AAAI Press.

Rintanen, J. 2007. Asymptotically optimal encodings of conformant planning in QBF. In *Proceedings of the 22nd AAAI Conference on Artificial Intelligence (AAAI-07)*, 1045–1050. AAAI Press.

Stockmeyer, L. J. 1976. The polynomial-time hierarchy. *Theoretical Computer Science* 3(1):1–22.

Turner, H. 2002. Polynomial-length planning spans the polynomial hierarchy. In *Logics in Artificial Intelligence, European Conference, JELIA 2002*, 111–124. Springer-Verlag.