Contract Scheduling With Predictions

Spyros Angelopoulos, Shahin Kamali

▶ **To cite this version:**

Spyros Angelopoulos, Shahin Kamali. Contract Scheduling With Predictions. Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI), Feb 2021, Online Conference, United States. pp.11726-11733. hal-03439686

HAL Id: hal-03439686
https://hal.science/hal-03439686v1

Submitted on 22 Nov 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Contract Scheduling With Predictions

Spyros Angelopoulos,[1,2] **Shahin Kamali**[3]
[1] Centre National de la Recherche Scientifique (CNRS)
[2] Sorbonne Université, Laboratoire d'informatique de Paris 6, LIP6, Paris, France
[3] University of Manitoba, Winnipeg, Manitoba, Canada
spyros.angelopoulos@lip6.fr, shahin.kamali@umanitoba.ca

Abstract

Contract scheduling is a general technique that allows to design a system with interruptible capabilities, given an algorithm that is not necessarily interruptible. Previous work on this topic has largely assumed that the interruption is a worst-case deadline that is unknown to the scheduler. In this work, we study the setting in which there is a potentially erroneous *prediction* concerning the interruption. Specifically, we consider the setting in which the prediction describes the time that the interruption occurs, as well as the setting in which the prediction is obtained as a response to a single or multiple binary queries. For both settings, we investigate tradeoffs between the robustness (i.e., the worst-case performance assuming adversarial prediction) and the consistency (i.e, the performance assuming that the prediction is error-free), both from the side of positive and negative results.

Introduction

One of the central objectives in the design of intelligent systems is the provision of *anytime* capabilities. In particular, several applications such as medical diagnostic systems and motion planning algorithms require that the system outputs a reasonably efficient solution given the unavoidable constraints on computation time. *Anytime algorithms* offer such a tradeoff between computation time and quality of the output. Namely, in an anytime algorithm the quality of output improves gradually as the computation time increases. This class of algorithms was introduced first in (Boddy and Dean 1994) in the context of time-depending planning, as well as in (Horvitz 1988) in the context of flexible computation.

(Russell and Zilberstein 1991; Zilberstein and Russell 1996) introduced a useful distinction between two different types of anytime algorithms. On the one hand, there is the class of *contract* algorithms, which describes algorithms that are given the amount of allowable computation time (i.e, the intended query time) as part of the input. However, if the algorithm is interrupted at any point before this "contract time" expires, the algorithm may output a result that is meaningless. On the other hand, the class of *interruptible* algorithms consists of algorithms whose allowable running time is not known in advance, and thus can be interrupted (queried) at any given point throughout their execution.

Copyright © 2021, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

Although less flexible than interruptible algorithms, contract algorithms typically use simpler data structures, and are thus often easier to implement and maintain (Bernstein, Finkelstein, and Zilberstein 2003). Hence a natural question arises: how can one convert a contract algorithm to an interruptible equivalent, and at which cost? This question can be addressed in an ad-hoc manner, depending on the algorithm at hand; however, there is a simple technique that applies to any possible contract algorithm, and consists of repeated executions of the contract algorithm with increasing runtimes (also called *lengths*). For example, consider a *schedule* of executions of the contract algorithm in which the i-th execution has length 2^i. Assuming that an interruption occurs at time t, then the above schedule guarantees the completion of a contract algorithm of length at least $t/4$, for any t, as shown in (Russell and Zilberstein 1991). The factor 4 measures the performance of the schedule, and quantifies the penalty due to the repeated executions.

More formally, given a contract algorithm A, a *schedule* X is defined by an increasing sequence (x_i) in which x_i is the length of the i-th execution of A. For simplicity, we call the i-th execution of A in X the i-th *contract*, and we call x_i its *length*. The *acceleration ratio* of X, denoted by $\mathtt{acc}(X)$, relates an interruption T to the length of the largest contract that has completed by time T in X, which we denote by $\ell(X,T)$, and is defined as

$$\mathtt{acc}(X) = \sup_T \frac{T}{\ell(X,T)} \tag{1}$$

Intuitively, the acceleration ratio describes a trade-off between processor speed and resilience to interruptions. Namely, by executing the schedule X to a processor of speed equal to $\mathtt{acc}(X)$, one obtains a system that is as efficient as a single execution of a contract algorithm that knows when the interruption will occur, but runs in a unit-speed processor. Note that the schedule, and also its acceleration ratio, are determined by the contract lengths.

Contract scheduling has been studied in a variety of settings related to AI. It has long been known that the schedule $X = (2^i)$ has optimal acceleration ratio equal to 4 (Russell and Zilberstein 1991). Optimal schedules in multi-processor systems were obtained in (Bernstein et al. 2002). The generalization in which there are more than one problem instances associated with the contract algorithm was first

studied in (Zilberstein, Charpillet, and Chassaing 2003), in which optimal schedules were obtained for a single processor. The more general setting of multiple instances and multiple processors was first studied in (Bernstein, Finkelstein, and Zilberstein 2003) and later in (López-Ortiz, Angelopoulos, and Hamel 2014). (Angelopoulos, López-Ortiz, and Hamel 2008) considered the problem in which the interruption is not a fixed deadline, but there is a "grace period" within which the system is allowed to complete the execution of the contract. Measures alternative to the acceleration ratio were proposed and studied in (Angelopoulos and López-Ortiz 2009). More recently, (Angelopoulos and Jin 2019) studied contract scheduling in the setting in which the schedule is deemed "complete" once a contract reaches some prespecified end guarantees.

Contract scheduling is an abstraction of resource allocation under uncertainty, in a worst-case setting. As such it has connections to other problems of a similar nature, such as online searching under the competitive ratio (Bernstein, Finkelstein, and Zilberstein 2003; Angelopoulos 2015; Kupavskii and Welzl 2018). Note, however, that previous work follows a worst-case model of uncertainty, in which the performance must be guaranteed under all circumstances, and thus treats any prediction as adversarial and untrustworthy.

Our Setting: Contract Scheduling with Predictions

Previous work on contract scheduling has mostly assumed that the interruption is unknown to the scheduler, and thus can be chosen adversarially, in particular right before a contract terminates. In practice, however, the scheduler may have a certain *prediction* concerning the interruption. Consider the example of a medical diagnostic system. Here, the expert may know that the system will be likely queried around a *specific time*, (i.e., prior to a scheduled surgery). Another possible prediction may describe a partition of time in *intervals* in which the system will likely be queried. In the example of the medical diagnostic system, it is more likely that the consultation will be required over a weekday, than over a weekend.

We study two settings that capture the above scenarios. In the first setting, there is a prediction τ concerning the interruption T. That is, we have information about the exact time the interruption will occur. In the second setting, the prediction is in the form of answers to n *binary queries*, where n is a specified parameter. For example, a binary query can be of the form "Will the interruption occur within a certain subset of the timeline?". For both settings, the prediction is not necessarily trustworthy, and comes with an unknown *error* η.

The performance of the schedule is determined by two parameters: the first is the *robustness*, which is the worst-case acceleration ratio of the schedule assuming adversarial error (i.e., an adversary manipulating the prediction). The second is the *consistency* of the schedule, which is the acceleration ratio assuming that the prediction is error-free. In between these extremes, the acceleration ratio will be, in general, a function of the prediction error. This follows the recent framework in machine learning of robust *online computation* with predictions, as introduced in (Lykouris and Vassilvitskii 2018) for the caching problem, and later applied in (Purohit, Svitkina, and Kumar 2018) for other online problems such as ski rental and non-clairvoyant scheduling.

Our paper differs from the above works in two important aspects. The first is related to the nature of the results. More precisely, our aim is to complement the positive results obtained by specific schedules, with negative, i.e., impossibility results. This is in the spirit of recent work (Rohatgi 2020) which showed lower bounds on the competitive ratio of any caching algorithm, as a function of the prediction error, the cache size and the optimal cost. We are also interested in finding schedules that are *Pareto-efficient* with respect to the tradeoff between robustness and consistency; i.e., by fixing one of the two parameters, the other must be optimized. This is inspired by (Angelopoulos et al. 2020) which studied Pareto-efficient online algorithms with untrusted advice (Angelopoulos et al. 2020).

The second difference is related to the nature of the problem we study. Unlike "natural" online optimization problems in (Lykouris and Vassilvitskii 2018) and (Purohit, Svitkina, and Kumar 2018), contract scheduling under the acceleration ratio poses certain novel challenges. Most notably, it is not the case that the performance improves monotonically as the error decreases. To see this, consider an interruption T, a prediction τ for T, and a schedule X for prediction τ. Suppose that a contract finishes right before τ in X: this is intuitively bad, because even with very small error, it is possible that X barely misses to complete its largest contract by time T. But it is also possible that if the error is very large, T happens to occur right after another contract terminates in the schedule. This is a "best-case" scenario for the schedule: it completes a contract right on time. This observation exemplifies the type of difficulties we face. Another difficulty is that there may exist schedules that are Pareto optimal for the pair of consistency and robustness, but whose performance falls back to the worst-case acceleration ratio for *any* non-zero error. Such schedules are clearly undesirable, which is another challenge we must overcome.

Results

We first consider the setting in which the prediction τ is the interruption time T. The prediction τ comes with an *error* $\eta \in [0, 1]$ such that $T \in [\tau(1-\eta), \tau(1+\eta)]$. We show how to obtain a Pareto-optimal schedule by showing a reduction from an online problem known as *online bidding* (Chrobak and Kenyon-Mathieu 2006). This allows us to use, as black-box, a Pareto-optimal algorithm of (Angelopoulos et al. 2020), and obtain a schedule with the same ideal performance. But there are two complications: this schedule cannot tolerate *any* errors (see the discussion above), and is also fairly complex. We give another simple schedule with the same robustness and consistency, and thus also Pareto-optimal. We then show how to extend this schedule to the realistic setting in which $\eta \neq 0$, and we complement the positive results with lower bounds on the performance of any schedule.

In the second part we study the setting in which the prediction is in the form of answers to n *binary queries*, for some given parameter n, i.e., we would like to combine the

advice of n binary experts. Thus, the prediction is an n-bit string, and the prediction error $\eta \in [0, 1]$ is defined as the fraction of the erroneous bits in the string. First, we show an information-theoretic lower-bound on the best-possible consistency one can hope to achieve in this setting, assuming optimal robustness equal to 4. We then present and analyze a family of schedules, parameterized by the range of error that each schedule can tolerate. There are several challenges here: the analysis must incorporate several parameters such as the error η, the number of queries n and the desired robustness r. Moreover, we need to define queries that are realistic and have a practical implementation. To this end, each query is a *partition* query of the form "Does interruption T belong to $\mathcal{T}$?", where $\mathcal{T}$ is a subset of the timeline.

Due to space limitations, we omit several technical proofs. All omitted details can be found in (Angelopoulos and Kamali 2020).

Other related work There are several recent works that study algorithms with ML predictions in a status of uncertainty. Examples include online rent-or-buy problems with multiple expert predictions (Gollapudi and Panigrahi 2019), queuing systems with job service times predicted by an oracle (Mitzenmacher 2020), online algorithms for metrical task systems (Antoniadis et al. 2020), and online makespan scheduling (Lattanzi et al. 2020). Clustering with noisy queries was studied in (Mazumdar and Saha 2017). Our setting is also related to the field of robust optimization (Kouvelis and Yu 2013), in which the input is uncertain and the objective is to devise efficient algorithms for any input realization. In our setting, however, the concept of the "input" is much broader, and entails potentially partial information that could be helpful to the decision maker.

Concerning contract scheduling, the work that is closest to ours is (Zilberstein, Charpillet, and Chassaing 2003), in which there is stochastic information about the interruption, and the objective is to optimize the expected quality of the output upon interruption. The optimal scheduling policy in (Zilberstein, Charpillet, and Chassaing 2003) is based on a Markov decision process, hence no closed-form solution is obtained. More importantly, their schedule does not provide worst-case guarantees (i.e., a bound on the robustness), but only average-case guarantees for the given distribution, which is also assumed to be known.

Preliminaries

A contract schedule is defined by a sequence $X = (x_i)_{i\geq 1}$ of contract lengths, or *contracts*, where x_i as the *i-th contract* in X. We will always denote by T the time at which an interruption occurs. We will make the standing assumption that an interruption can occur only after a unit time has elapsed. With no prediction on T, the worst-case acceleration ratio of X is given by (1); this is the *robustness* of X, which we denote by r_X, or simply r, if the schedule is implied. With a prediction, the acceleration ratio of X is simply defined as $T/\ell(X, T)$. Given a prediction, the *consistency* of X is its acceleration ratio assuming $\eta = 0$. We will say that a schedule has *performance* (r, s) if it has robustness r and consistency s. We will refer to any schedule with robustness at most r as r-robust. In a *Pareto-optimal* schedule, these are in a Pareto-optimal relation.

Given a schedule $X = (x_i)$, it is easy to see that the worst-case interruptions occur infinitesimally prior to the completion of a contract. Hence the following useful formula.

$$r_X = \sup_{i\geq 1} \frac{\sum_{j=1}^{i} x_j}{x_{i-1}}, \tag{2}$$

where x_0 is defined to be equal to -1.

The class of *exponential* schedules describes schedules in which the i-th contract has length a^i, for some fixed a, which we call the *base* of the schedule. For several variants of the problem, there are efficient schedules in this class. The robustness of an exponential schedule with base a is equal to $a^2/(a-1)$ (Zilberstein, Charpillet, and Chassaing 2003), and for $a = 2$ the corresponding schedule has optimal robustness 4 (Russell and Zilberstein 1991). Let

$$c_r = \frac{r - \sqrt{r^2 - 4r}}{2} \text{ and } b_r = \frac{r + \sqrt{r^2 - 4r}}{2},$$

then it is easy to verify that for any given $r \geq 4$, an exponential schedule with base $a \in [c_r, b_r]$ has robustness at most 4. This fact will be useful in our analyses.

In the *online bidding* problem (Chrobak and Kenyon-Mathieu 2006), we seek an increasing sequence $X = (x_i)$ of positive numbers (called *bids*) of minimum *competitive ratio*, defined formally as

$$\sup_{u\geq 1} \frac{\sum_{j=1}^{i} x_j}{u} \ : x_{i-1} < u \leq x_i, \tag{3}$$

where u is the *target* value. In words, we seek a strategy (x_i) for submitting bids, given some unknown target (or threshold) u, and we pay a cost equal to the sum of all bids up the first bid that is at least as large as u. The competitive ratio of the strategy is the maximum ratio of this cost divided by the target u.

Without predictions, online bidding is equivalent to contract scheduling: given an increasing sequence $X = (x_i)$, both its acceleration ratio and its competitive ratio can be described by (2). We say that a bidding sequence has performance (r, s) with a given prediction if it has robustness r and consistency s with respect to its competitive ratio.

Interruption Time as Prediction

We first consider the setting in which the prediction τ describes the interruption time T. The prediction comes with an *error* $\eta \in [0, 1]$, defined as follows. If $T \geq \tau$, then we define η to be such that $T/\tau = (1 + \eta)$, and if $T \leq \tau$, then we define η to be such that $T/\tau = (1 - \eta)$. In the former case, we will say that the error is *positive*, otherwise we will say that the error is *negative*. Regardless of the sign of error, we have that $T \in [\tau(1 - \eta), \tau(1 + \eta)]$.

We will also study settings in which the error η is bounded by a quantity $H \leq 1$ which may or may not be known to the schedule. We thus distinguish between *H-oblivious* and *H-aware* schedules. Note that if η is bounded by H then

$$\tau(1 - H) \leq \tau(1 - \eta) \leq T \leq \tau(1 + \eta) \leq \tau(1 + H).$$

We will first consider the ideal case in which either the prediction is error-free (hence the robustness is evaluated for $\eta = 0$), or it is adversarially generated (hence the consistency is the worst-case acceleration ratio),

Theorem 1. *Suppose that for every $r \geq 4$, there is a sequence for online bidding that has performance (r, s) for prediction equal to the target u. Then there is a contract schedule with the same guarantees for the setting in which the prediction is the interruption, and vice versa.*

From (Angelopoulos et al. 2020), there is a Pareto-optimal bidding sequence, which satisfies the conditions of Theorem 1 with $s = c_r$. This implies the following.

Corollary 1. *For every $r \geq 4$, there is a contract schedule X^*_τ that has performance (r, c_r), and this is Pareto-optimal.*

We also obtain the following corollary.

Corollary 2. *For any r-robust schedule X, and any time t, it holds that $\ell(X, t) \leq t/c_r$. Moreover, for any $\epsilon > 0$, there exists i_0 such that if $x_i = \ell$, with $i \geq i_0$, then the completion time of x_i is at least $c_r\ell - \epsilon$.*

There are two issues here. The first is that the schedule obtained using the reduction to online bidding is fairly complex, because the bidding algorithm in (Angelopoulos et al. 2020) is quite complex. We can give instead, a different schedule, which is more intuitive and has the same performance, hence also Pareto-optimal.

Definition 1. *Consider the exponential schedule $G = (b^i_r)$. Then there exists $\gamma < 1$ such that in the schedule $X^*_\tau \doteq (\gamma b^i_r)$, there is a contract that completes at time precisely equal to τ.*

It is relatively easy to see that X^*_τ has also performance (r, c_r), and thus is also Pareto-optimal, from Corollary 1. The intuition behind this schedule is that it is the r-robust exponential schedule with the largest possible contracts which finishes by time τ.

Example. Suppose $\tau = 100$ and $r = 4.5$, which gives $b_r = 3$. Then we have $b^5_r = 243$, and hence $\gamma = 100/243$.

The second, and more significant issue, is that as in the case of the online bidding algorithm of (Angelopoulos et al. 2020), in the presence of *any* error $\eta \neq 0$, the acceleration ratio of X^*_τ becomes as bad as its robustness r. This is because if $T = \tau - \epsilon$, for infinitessimally small $\epsilon > 0$, a long contract in X^*_τ is not completed.

We will next adapt X^*_τ in order to obtain a more realistic schedule. The idea is to allow some "buffer" so that the schedule can tolerate mispredictions as a function of the buffer size. More precisely, for any $p \in (0, 1)$, consider the schedule $X^*_{\tau(1-p)}$. The following lemma gives an upper bound on the performance of this parameterized, and H-oblivious schedule.

Lemma 1. *For any $p \in (0, 1)$, and $r \geq 4$, $X^*_{\tau(1-p)}$ is r-robust and has consistency $\min\{\frac{c_r}{1-p}, r\}$. It also has acceleration ratio at most $\min\{\frac{c_r(1+\eta)}{(1-p)}, r\}$ for positive error, at most $\min\{\frac{c_r(1-\eta)}{(1-p)}, r\}$ if η is negative error with $\eta \leq p$, and at most r, in every other case.*

The above result provides a tradeoff between the acceleration ratio of $X^*_{\tau(1-p)}$, and the range in which it is sufficiently good, as a function of the error. To illustrate this, consider the case of negative error: If p is relatively small, then the schedule has good acceleration ratio for relatively small η ($\eta < p$), which however can (and will) become as large as r, for a relatively big range of error, i.e, for $\eta > p$.

We now argue that these tradeoffs are unavoidable, in any r-robust and H-oblivious schedule X with prediction τ. Recall that $\ell(X, \tau)$ denotes the largest contract completed in the schedule by time τ in X, and let $p \in [0, 1]$ be such that $\tau(1-p)$ is the completion time of this contract. From Corollary 2 we know that $\ell(X, \tau) \leq \tau(1-p)/c_r$, hence for negative error $\eta \leq p$, the acceleration ratio is at least $\frac{c_r(1-\eta)}{1-p}$, and hence the consistency is at least $\frac{c_r}{1-p}$. Moreover, there exists $x > 0$ such that at time $\tau(1+x)$, the largest completed contract does not exceed $\ell(X, \tau)$. Hence for positive error $\eta < x$, the acceleration ratio is at least $\frac{c_r(1+\eta)}{1-p}$. Last, since the schedule is H-oblivious, T can occur at points right before a contract terminates, for all contracts that completed before τ. In this latter case, the acceleration ratio will inevitably be as large as r, as T becomes large.

For these reasons we will next consider *H-aware* schedules in which $\eta \leq H$ and H is known. A natural schedule then is $X^*_{\tau(1-H)}$, in which the buffer p is determined by H. Its performance is described in the following lemma, whose proof follows similarly to Lemma 1, by setting $p = H$.

Lemma 2. *$X^*_{\tau(1-H)}$ is r-robust, and has acceleration ratio at most $\min\{\frac{c_r(1+\eta)}{(1-H)}, r\}$ for positive error, and at most $\min\{\frac{c_r(1-\eta)}{(1-H)}, r\}$ for negative error.*

Since $\eta \leq H$, we have $\mathtt{acc}(X^*_{\tau(1-H)}) \leq \min\{\frac{c_r(1+H)}{1-H}, r\}$. The next lemma shows that H can take values in a certain range, as function of c_r, for which no other r-robust schedule can be better.

Lemma 3. *For any H that satisfies the condition $\frac{1+H}{1-H} < \sqrt{\frac{c_r+1}{c_r}} - \delta$, for any fixed $\delta > 0$, the acceleration ratio of any H-aware r-robust schedule is at least $\min\{\frac{c_r(1+H)}{1-H}, r\}$.*

Proof. By way of contradiction, let X denote an H-aware schedule that has acceleration ratio at most $\min\{\frac{c_r(1+H)}{1-H}, r\}$. Then given prediction τ, X must complete by time $\tau(1-H)$ a contract, say x, of length at least

$$\frac{\tau(1-H)^2}{c_r(1+H)}.$$

From Corollary 2, the completion time of x must be at least

$$c_r \cdot \frac{\tau(1-H)^2}{c_r(1+H)} - \epsilon = \frac{\tau(1-H)^2}{1+H} - \epsilon,$$

for arbitrarily small $\epsilon > 0$, since T can be arbitrarily large. We now claim that x is also the largest contract completed

by time $\tau(1+H)$ in X. By way of contradiction, suppose that there is a contract y that follows x, and which completes by time $\tau(1+H)$. Note that y must be at least as big as x. Then it must be that

$$\frac{\tau(1-H)^2}{1+H} - \epsilon + \frac{\tau(1-H)^2}{c_r(1+H)} \leq \tau(1+H),$$

and since ϵ can be arbitrarily small and smaller than δ, we arrive at a contradiction, concerning the assumption on H. Thus, if $T = \tau(1+H)$ (i.e., for positive $\eta = H$, the largest contract completed is x, and thus the acceleration ratio is at least $c_r(\frac{1+H}{1-H})^2 \geq c_r\frac{1+H}{1-H}$. □

We can also show that there is an even larger range for H than that of Lemma 3 for which no other schedule can *dominate* $X^*_{\tau(1-H)}$, in the sense that no schedule can have as good an acceleration ratio as $X^*_{\tau(1-H)}$ on all possible values of $\eta \leq H$, and strictly better for at least one such value.

Lemma 4. *For any H such that $\frac{1+H}{1-H} < \frac{c_r+1}{c_r} - \delta$, no r-robust H-aware schedule dominates $X^*_{\tau(1-H)}$.*

Example. To put the above results into perspective, let us consider the case $r = 4$ (best robustness). Then $c_r = 2$, and X^*_τ is 2-consistent, but can have acceleration ratio 4 for any $\eta \neq 0$. For given bound H, $X^*_{\tau(1-H)}$ has acceleration ratio at most $\min\{\frac{2(1+\eta)}{(1-H)}, 4\}$ for positive error, and at most $\min\{\frac{2(1-\eta)}{1-H}, 4\}$ if η is negative error. Thus, an absolute upper bound on its acceleration ratio is $\min\{\frac{2(1+H)}{1-H}, 4\}$, whereas its consistency is $\min\{\frac{2}{(1-H)}, 4\}$. For any $H < 0.101$, no 4-robust H-aware schedule has better acceleration ratio. Last, for $H < 0.2$, there is no 4-robust H-aware schedule that dominates $X^*_{\tau(1-H)}$.

Binary Predictions

In this section, we study the setting in which the prediction is in the form of answers to n binary queries $Q_1, \ldots, Q_n$, for some given n. Hence, the prediction P is an n-bit string, where the i-th bit is the answer to Q_i. It is worth pointing out that even a single binary query can be quite useful. For example, it can be of the form "Is $T \leq B$, for some given bound B"?, or "Is $T \in [a, b]$, for some given a, b"? The prediction error $\eta \in [0, 1]$ is the fraction of erroneous bits in P. We will assume, for simplicity, that the total number of erroneous bits, that is ηn, is an integer.

Our approach to this problem is as follows. Let $\mathcal{X}$ be a set of r-robust schedules. The prediction P will help choose a good schedule from this class. For positive results, we need to define $\mathcal{X}$, and show how the prediction can help us choose an efficient schedule from X; moreover the prediction must have a practical interpretation, and must tolerate errors. For negative (i.e., impossibility) results, we need to show that any choice of 2^n r-robust schedules in $\mathcal{X}$ cannot guarantee consistency below a certain bound. Note that in this scheme, all schedules in $\mathcal{X}$ must be r-robust, because *any* schedule in $\mathcal{X}$ can be chosen, if the prediction is adversarially generated.

We begin with a negative result, for the simple, but important case $r = 4$, i.e., for optimal robustness. The following theorem gives a lower bound on the consistency.

Theorem 2. *For any binary prediction P of size n, any schedule with performance $(4, s)$ is such that $s \geq 2^{1+\frac{1}{2^n}}$.*

Proof sketch. The proof is based on an information-theoretic argument. With n binary queries, the prediction P can only help us choose a schedule from a class $\mathcal{X}$ of at most 2^n 4-robust schedules. Let $X_1, X_2, \ldots, X_{2^n}$ describe these schedules. By way of contradiction, suppose we could guarantee consistency $S = 2^{1+\frac{1}{2^n}} - \delta$, with $\delta > 0$. We show that there exists an ordering of these schedules with the following property, which we prove by induction: there is a set of $2^n - 1$ interruptions, $T_2, \ldots, T_{2^n-1}$ such that, for interruption T_i, with $i \in [2, 2^n - 1]$, no schedule of rank at most $i+1$ in the ordering can guarantee consistency S. This means that for interruption T_{2^n-1}, no schedule in $\mathcal{X}$ can guarantee robustness S, a contradiction. □

We complement Theorem 2 with the following positive result. Consider the set $\mathcal{X} = \{X_i, i \in [0, 2^n - 1]\}$ of schedules, in which $X_i = (x_{j,i})_{j\geq 1}$ is defined by $x_{j,i} = d^{j+\frac{i}{2^n}}$, for $d > 1$ that we will choose later. In words, X_i is a near-exponential schedule with base d, and a scaling factor equal to $d^{\frac{i}{2^n}}$. The prediction P then chooses an index, in $[0, 2^n - 1]$, of the schedule in this $\mathcal{X}$. We call IDEAL the schedule obtained from $\mathcal{X}$ with prediction P.

Theorem 3. *For every $r \geq 4$, define $d = b_r$, if $r \leq \frac{(1+2^n)^2}{2^n}$, and $d = 1 + 2^n$, otherwise. Then* IDEAL *has performance $(r, d^{1+\frac{1}{2^n}}/(d-1))$.*

For example, for $r = 4$, IDEAL has performance $(2^{1+\frac{1}{2^n}}, 4)$, which matches Theorem 2, and is, therefore, Pareto-optimal.

IDEAL, as its name suggests, is not a practical schedule: a single error in one of the queries can make its acceleration ratio as bad as its robustness. Intuitively, this occurs because the n queries implement a type of "binary search" in the space of all 2^n schedules in $\mathcal{X}$, and which is not robust to errors. We will instead propose a family of schedules, which we call $Robust_p$, where $p \in [0, 1]$ is a parameter that defines the range of error that the schedule can tolerate (this will become more clear shortly). More precisely, we will define a class of schedules $\mathcal{X}$, and the prediction P will be the index of one of these schedules. However, this time there are only n schedules in $\mathcal{X}$ instead of 2^n, as in the case of IDEAL. Each $X_i \in \mathcal{X}$ is defined as $X_i = (x_{j,i})_{j\geq 0} = d^{j+\frac{i}{n}}$, with $i \in [0, n-1]$, and again $d > 1$ to be determined later.

We now describe the n queries that comprise the prediction P. Each query Q_i, for $i \in [0, n-1]$ is of the form "Is the best schedule, for the given interruption in $\{X_0, \ldots, X_i\}$?". Note that the queries obey a monotonicity property: if Q_i is "no", and Q_{i+1} is "yes", we know an error has occurred in one of these queries. It is also important to note that each of the queries Q_i has an equivalent statement of the following form: "Does the interruption T belong to a subset S_i of

the timeline?". Thus each query asks whether T falls in a certain partition of the timeline, which has a more natural, and practical interpretation. This holds for both IDEAL and $Robust_p$.

If there were no errors (i.e., for $\eta = 0$), then the best schedule in $\mathcal{X}$ would be the number of "no" responses to the n queries, minus one to account for indexing from 0. However, in the presence of errors, one needs to be careful, because, once again, a single error can have an enormous impact. For this reason, $Robust_p$ uses the parameter p. In particular, it chooses schedule X_m, where m is defined as $(N-1-pn) \bmod n$ and N is the number of "no" responses (again, for convenience we will assume that pn is integral). In words, $Robust_p$ chooses a schedule of index "close and above", in the cyclic order of indices, to an index that would correspond to an error-free prediction. The following theorem bounds the performance of $Robust_p$, and shows how to choose the base d. We make two assumptions: that $\eta \leq p$ (thus $Robust_p$ can only tolerate up to p fraction of query errors), and that $p \leq 1/2$ (otherwise, in the worst case, the queries are too "corrupt" to be of any use).

Theorem 4. *For every $r \geq 4$, define K to be equal to $\frac{2pn+1}{n}$, and d to be equal to b_r, if $r \leq (1+K)^2/K$, and $1+K$, otherwise. Then $Robust_p$ is r-robust and has acceleration ratio at most $\frac{d^{1+\frac{1}{n}+2p}}{d-1}$, assuming $\eta \leq p \leq 1/2$.*

Proof. For interruption T, let l denote the index of the best schedule in $\mathcal{X}$. From the structure of $\mathcal{X}$, this means that, in worst-case, T occurs right before the completion of a contract, say j, in the schedule $X_{(l+1) \bmod n}$. We will consider the case $l \neq n-1$, thus $(l+1) \bmod n = l+1$; the outlier case $l = n-1$ follows similarly, but with a slightly different argument (namely, the worst case interruption occurs right before the completion time of contract $j+1$ of X_0). We express this interruption as

$$T = \sum_{i=1}^{j} x_{i,l+1} = \sum_{i=1}^{j} d^{i+\frac{l+1}{n}} \leq \frac{d^{j+1+\frac{l+1}{n}}}{d-1}.$$

Let m denote the index chosen by $Robust_p$, as defined earlier. The crucial observation is that in a cyclic ordering of the indices, m and l are within a distance at most $(\eta+p)n$. Here, a distance of at most ηn is due to the maximum number of erroneous queries, and an additional distance of at most pn is further incurred by the algorithm. Since $\eta \leq p$, they are within a distance at most $2pn$.

We will give a lower bound on the largest contract length, say L completed by time T in $Robust_p$. We consider two cases. First, suppose that $m \leq l$, then by the structure of $\mathcal{X}$, L is at least the length $x_{j,l-2pn} = d^{j+\frac{l-2pn}{n}}$. Next, suppose that $m > l$. In this case, L is at least the length of $x_{j-1,n+l-2pn} = d^{j-1+\frac{n+l-2pn}{n}} = d^{j+\frac{l-2p}{n}}$. In both cases we conclude that $L \geq d^{j+\frac{l-2pn}{n}}$. Therefore the acceleration ratio is at most $T/L \leq \frac{d^{1+\frac{1}{n}+2p}}{d-1}$. We now want to find d such that $d^2/(d-1) \leq r$ and $\frac{d^{1+\frac{1}{n}+2p}}{d-1}$ is minimized. Using standard calculus, it follows that the best choice of d is as in the statement of the theorem. □

For example if $r = 4$, then for any given $p \leq 1/2$, $Robust_p$ is 4-robust, can tolerate at most a $p \leq 1/2$ fraction of erroneous responses, and has acceleration ratio at most $2^{1+\frac{1}{n}+2p}$. We can interpret the result of the theorem in two ways. First, one can use p as a hedging parameter: with larger p, better tolerance to errors can be achieved, at the expense however of the acceleration ratio (akin to Lemma 1 and the discussion following it). Second, the acceleration ratio improves rapidly as a function of the numbers (not as rapidly as in IDEAL, but still very fast).

Experimental Results

In this section, we present the experimental evaluation of our schedules. We use exponential schedules (without any prediction) as the baseline for our comparisons. Recall that for any $r \geq 4$, any exponential schedule (a^i) with base $a \in [c_r, b_r]$ has robustness at most r. For the special but important case of $r = 4$, there is only one such schedule with $a = 2$. We report results for $r = 4$, but we note that for $r > 4$ the experiments show the same trends.

Interruption Time as Prediction

We model $\tau \in [T-H, T+H]$ to be a random (truncated) normal variable with mean T and standard deviation 1, such that $\eta \leq H$. Recall that an H-aware schedule knows H, whereas an H-oblivious one does not. Figure 1 depicts the average acceleration ratio (y-axis) of the schedule $X^*_{\tau(1-p)}$ for different values of p, as a function of the interruption time T (x-axis), for fixed $H = 0.1$. The plot depicts the performance of four schedules: the H-aware schedule, in which $p = H$, and three H-oblivious schedules for $p = 0.05$, $p = 0.2$ and $p = 0.3$. We run the experiment over 1,000 evenly spaced values of the interruption time in the interval $[2, 2^{20}]$. For each value of $T \in [2, 2^{20}]$, we compute the acceleration ratio of the schedule for 1,000 random values of $\tau \in [T-H, T+H]$, and report the average.

The figure shows that the H-aware schedule has an advantage over the schedules with different values of p. In particular, the expected value of the acceleration ratio of this schedule is around 2.23 for all values of the interruption T, compared to acceleration ratios of 2.41 for the schedule with buffer smaller than H ($p = 0.05$) and ratios 2.49 and 2.85 of the schedule whose buffer is larger than H ($p = 0.2, 0.3$, respectively). As p decreases, the fluctuation of the acceleration ratio due to the noise increases, since the interruption becomes closer to the completion time of a contract.

As Figure 1 shows, our schedules with predictions do not outperform the baseline algorithm for every interruption. This is to be expected, since there is no schedule that can dominate any other schedule. More precisely, even a schedule of very bad robustness (e.g., a schedule with a huge contract early on) will have excellent acceleration ratio for some range of interruptions (e.g., for certain interruptions before the completion time of the huge contract). Nevertheless, we can quantify the advantage of the schedules with predictions, as shown in Table 1. The table depicts the percentage of interruptions in $[2, 2^{20}]$ for which $X^*_{\tau(1-p)}$ outperforms

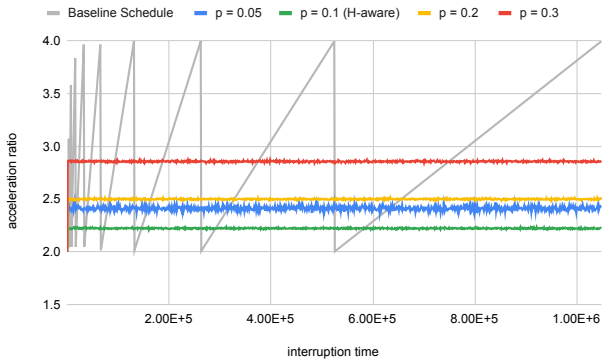

Figure 1: Acceleration ratios of $X^*_{\tau(1-p)}$, for $H = 0.1$.

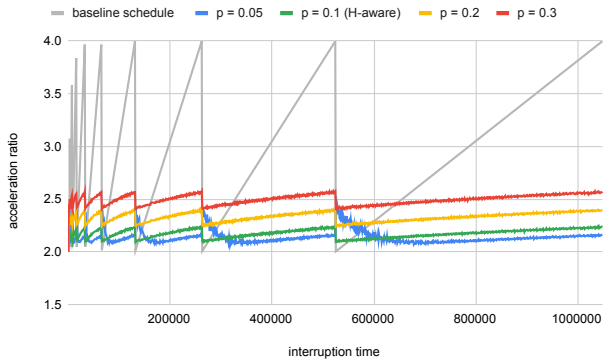

Figure 2: Acceleration ratios of $Robust_p$, for $H = 0.1$.

	$p = 0.05$	$p = 0.1$	$p = 0.2$	$p = 0.3$
improvement	79.22%	88.71%	74.73%	57.04%
strong improv.	55.24%	66.43%	50.05%	28.47%

Table 1: Percentage of interruptions in $[2, 2^{20}]$ for which $X_{\tau(1-p)}$ outperforms the baseline schedule.

	$p = 0.05$	$p = 0.1$	$p = 0.2$	$p = 0.3$
improvement	89.81%	94.25%	86.07%	77.07%
strong improv.	74.33%	70.98%	60.94%	49.95%

Table 2: Percentage of interruptions in $[2, 2^{20}]$ for which $Robust_p$ outperforms the baseline schedule.

the baseline schedule, as well as the percentage of interruptions for which the improvement is significant (at least by 20%). As expected, the H-aware schedule yields the best improvements, but even the H-oblivious schedules tend to perform much better than the baseline schedule. Thus, while H-awareness yields improvements, it is not indispensable.

Similar conclusions can be drawn for different values of H, but as H increases, the acceleration ratios of the schedules $X^*_{\tau(1-p)}$ also smoothly increase, as expected. In addition, similar results are obtained for τ uniformly at random in $[T - H, T + H]$.

Binary Predictions

We evaluate experimentally the performance of $Robust_p$ (as mentioned earlier, IDEAL is not a practical schedule, and thus we do not implement it). We fix the number n of queries to be equal to 100, and as in the previous setting, we also set $H = 0.1$. Given a binary prediction of size 100, we generate a noisy prediction by flipping a fraction η of the 100 bits (rounded down) where η is chosen uniformly at random in $[0, H]$. Figure 2 depicts the average acceleration ratio (y-axis) of $Robust_p$ for different values of the parameter p, as a function of the interruption time T (x-axis). As earlier, the expectation is taken over 1,000 random values of the error, and the interruption time takes values in the interval $[2, 2^{20}]$.

We consider $Robust_p$ with four values of the parameter p, namely $p \in \{0.05, 0.1, 0, 2, 0.3\}$. Note that the theoretical upper bound of Theorem 4 applies only if $p \geq 0.1$ in this setting. For such values of p, the acceleration ratio is a "saw-like" function of the interruption. There are some "critical" interruptions at which the acceleration ratio drops, then gently increases until the next critical interruption, as shown in Figure 2. The acceleration ratio of $Robust_p$ also increases with p, as predicted by Theorem 4, but is much smaller than the baseline acceleration ratio; for instance, for $p = 0.3$, it fluctuates in the interval $[2.4, 2.6]$. Note also that even for $p = 0.05 < H$, $Robust_p$ performs better than the baseline schedule, which is interesting because such a case is not captured by Theorem 4. This implies that $Robust_p$ may work in practice for a wider range of values of p than predicted by the theorem, and that $Robust_p$ need not be H-aware to perform well.

To quantify the above observation, in Table 2 we report the performance gain of $Robust_p$ for different values of p, and fixed $H = 0.1$. Once again, the table shows the percentage of interruptions in the range $[2, 2^{20}]$ for which $Robust_p$ outperforms the baseline schedule, as well as the percentage in which the performance gain is significant (at least 20%).

Conclusion

It is intriguing that a problem with a very simple statement, namely contract scheduling under the acceleration ratio, turns out to be quite challenging in the setting of predictions. We explored the tradeoffs between the prediction accuracy, the acceleration ratio, the consistency and the robustness of schedules in two natural settings of prediction. In future work, we would like to study the multi-instance setting, as discussed in the introduction, for which a lot of work has been done in the standard framework of no predictions. Last, the techniques we developed in this work should be readily applicable to the problem of searching on the line under the competitive ratio, and to the settings studied recently in (Angelopoulos 2021) given the connections between this problem and contract scheduling (Bernstein, Finkelstein, and Zilberstein 2003; Angelopoulos 2015).

Acknowledgments

This research benefited from the support of the FMJH Program PGMO and from the support to this program from EDF-THALES-ORANGE.

References

Angelopoulos, S. 2015. Further Connections Between Contract-Scheduling and Ray-Searching Problems. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI)*, 1516–1522.

Angelopoulos, S. 2021. Online Search with a Hint. In *12th Innovations in Theoretical Computer Science Conference, ITCS*, volume 185 of *LIPIcs*, 51:1–51:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Angelopoulos, S.; Dürr, C.; Jin, S.; Kamali, S.; and Renault, M. P. 2020. Online Computation with Untrusted Advice. In *Proceedings of the 11th International Conference on Innovations in Theoretical Computer Science (ITCS)*, 52:1–52:15.

Angelopoulos, S.; and Jin, S. 2019. Earliest-Completion Scheduling of Contract Algorithms with End Guarantees. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence, (IJCAI)*, 5493–5499.

Angelopoulos, S.; and Kamali, S. 2020. Contract scheduling with predictions. *arXiv 2011.12439* .

Angelopoulos, S.; and López-Ortiz, A. 2009. Interruptible Algorithms for Multi-Problem Solving. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI)*, 380–386.

Angelopoulos, S.; López-Ortiz, A.; and Hamel, A. 2008. Optimal Scheduling of Contract Algorithms with Soft Deadlines. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence (AAAI)*, 868–873.

Antoniadis, A.; Coester, C.; Elias, M.; Polak, A.; and Simon, B. 2020. Online metric algorithms with untrusted predictions. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, 11453–11463.

Bernstein, D.; Finkelstein, L.; and Zilberstein, S. 2003. Contract Algorithms and Robots on Rays: Unifying Two Scheduling Problems. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI)*, 1211–1217.

Bernstein, D.; Perkins, T. J.; Zilberstein, S.; and Finkelstein, L. 2002. Scheduling Contract Algorithms on Multiple Processors. In *Proceedings of the 18th AAAI Conference on Artificial Intelligence (AAAI)*, 702–706.

Boddy, M. S.; and Dean, T. L. 1994. Deliberation Scheduling for Problem Solving in Time-Constrained Environments. *Artif. Intell.* 67(2): 245–285.

Chrobak, M.; and Kenyon-Mathieu, C. 2006. SIGACT news online algorithms column 10: Competitiveness via Doubling. *SIGACT News* 37(4): 115–126.

Gollapudi, S.; and Panigrahi, D. 2019. Online Algorithms for Rent-Or-Buy with Expert Advice. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, 2319–2327.

Horvitz, E. 1988. Reasoning About Beliefs and Actions Under Computational Resource Constraints. *Int. J. Approx. Reasoning* 2(3): 337–338.

Kouvelis, P.; and Yu, G. 2013. *Robust discrete optimization and its applications*, volume 14. Springer Science & Business Media.

Kupavskii, A.; and Welzl, E. 2018. Lower Bounds for Searching Robots, Some Faulty. In *Proceedings of the 37th ACM Symposium on Principles of Distributed Computing (PODC)*, 447–453.

Lattanzi, S.; Lavastida, T.; Moseley, B.; and Vassilvitskii, S. 2020. Online Scheduling via Learned Weights. In *Proceedings of the 30th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 1859–1877.

López-Ortiz, A.; Angelopoulos, S.; and Hamel, A. 2014. Optimal Scheduling of Contract Algorithms for Anytime Problem-Solving. *J. Artif. Intell. Res.* (51): 533–554.

Lykouris, T.; and Vassilvitskii, S. 2018. Competitive Caching with Machine Learned Advice. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, 3302–3311.

Mazumdar, A.; and Saha, B. 2017. Clustering with Noisy Queries. In *Annual Conference on Neural Information Processing Systems (NeurIPS)*, volume 30, 5788–5799.

Mitzenmacher, M. 2020. Scheduling with Predictions and the Price of Misprediction. In *Proceedings of the 11th Innovations in Theoretical Computer Science Conference (ITCS)*, volume 151, 14:1–14:18.

Purohit, M.; Svitkina, Z.; and Kumar, R. 2018. Improving Online Algorithms via ML Predictions. In *Annual Conference on Neural Information Processing Systems (NeurIPS)*, volume 31, 9661–9670.

Rohatgi, D. 2020. Near-Optimal Bounds for Online Caching with Machine Learned Advice. In *Proceedings of the 30th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 1834–1845.

Russell, S. J.; and Zilberstein, S. 1991. Composing real-time Systems. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI)*, 212–217.

Zilberstein, S.; Charpillet, F.; and Chassaing, P. 2003. Optimal Sequencing of Contract Algorithms. *Ann. Math. Artif. Intell.* 39(1-2): 1–18.

Zilberstein, S.; and Russell, S. J. 1996. Optimal Composition of Real-Time Systems. *Artif. Intell.* 82(1-2): 181–213.