Competitive Sequencing with Query Predictions

Spyros Angelopoulos, Diogo Arsénio, Shahin Kamali

Competitive Sequencing with Query Predictions*

Spyros Angelopoulos[a,*], Diogo Arsénio[b], Shahin Kamali[c]

[a]*Sorbonne University, CNRS, LIP6, Paris, France*
[b]*New York University, Abu Dhabi, UAE*
[c]*Department of Electrical Engineering and Computer Science, York University, Toronto, Canada*

Abstract

Several well-studied online resource allocation problems can be abstracted in terms of an infinite, increasing sequence, where each element is associated with a corresponding allocation value. In Theoretical Computer Science, one such abstraction is known as *online bidding*, in which an algorithm must submit "bids" until an unknown threshold is reached. Another abstraction that has been studied extensively in Artificial Intelligence is known as *contract scheduling*: in this formulation, an algorithm is repeatedly executed with increasing processing times so as to obtain a system with interruptible capabilities.

We study such problems under the *query prediction* model, in which the designer elicits a prediction on the instance via responses to k binary queries so as to improve the algorithm's performance. The queries are answered by an imperfect oracle, and the objective is to obtain efficient algorithms

*Partially supported by the project PREDICTIONS, grant ANR-23-CE48-0010 from the French Research Agency (ANR), and Natural Sciences and Engineering Research Council of Canada (NSERC), grant DGECR-2018-00059.

*Corresponding author

Email addresses: spyros.angelopoulos@lip6.fr (Spyros Angelopoulos), diogo.arsenio@nyu.edu (Diogo Arsénio), kamalis@yorku.ca (Shahin Kamali)

that are also robust to query errors. We first focus on *consistency-robustness* tradeoffs, in which the query responses are either error-free or are generated by a (malicious) adversary: here, we prove a tight information-theoretic lower bound that establishes *Pareto-optimality* with respect to the consistency and the robustness. Next, we consider the more general setting in which some of the query responses can be erroneous: here, we present and analyze an efficient and robust algorithm based on *adaptive* queries. Specifically, we show that small increments in the number of queries lead to substantial improvement in robustness to query errors, in that the performance of our solution approaches the ideal performance of the Pareto-optimal schedule very quickly as k grows, even if as many as $k/4 - o(k)$ responses are adversarially erroneous. Our techniques have applications outside the query model: Namely, we show how to obtain an optimal schedule for a generalization of the *fault tolerant* contract scheduling problem in a multi-processor system, which generalizes the setting of [Kupavskii and Welzl, *Distr. Comp.* 2019].

Keywords: Competitive analysis, algorithms with predictions, consistency, robustness, query models.

---

1. Introduction

Resilience to interruptions is a central requirement in the design of real-time and intelligence systems. For instance, applications such as medical diagnostic systems, robot motion planning, and financial planning and trading require that the system be capable of outputting a reasonably efficient solution at all times [1]. This raises the important issue of designing an interruptible system given, as a building component, an algorithm that does

not necessarily have interruptible capabilities.

The seminal work of Russell and Zilberstein [2] was the first to provide a methodology for the design of interruptible systems using *contract* algorithms as components. The latter are algorithms which, unlike the interruptible ones, require the amount of computational time to be known in advance. Specifically, if a contract algorithm is allowed an execution time that is at least as large as this "contract" time, then its output is guaranteed to be correct; however, if the algorithm is interrupted at any point prior to the contract time, then its output may be totally meaningless. A typical example is algorithms based on dynamic programming (DP); if the algorithm fails to fill the entire DP table, the output may be entirely useless. Despite this lack of flexibility, contract algorithms are often easier to implement and maintain and use relatively simpler data structures [3], which makes them often excellent components for the design of more complex, real-time systems. Note that interruptible and contract algorithms are members of the broader class of *anytime* algorithms, i.e., algorithms whose performance improves as a function of the available computational time.

The approach of [2] is based on iterative deepening and consists of repeatedly executing the contract algorithm with increasing contract times (also called *lengths*). To illustrate with an example, consider a simple doubling rule in which the i-th execution has length 2^i. In this case, even if an interruption occurs at a worst-case time instance T (namely, right before an execution is about to terminate), an execution of length at least $T/4$ has been completed. The factor 4 measures the performance of this doubling rule and describes the tradeoff between resilience to interruptions and processor

speed. That is, a system A based on doubling contract lengths, and with processor speed equal to 4, is at least as efficient as any system B of unit speed, even if B knows beforehand the interruption time (and could thus use only a single execution with contract time T).

More generally, given a contract algorithm A, a contract *schedule* can be described as an increasing sequence $(x_i)_{i\geq 0}$, in which x_i is the length of the i-th execution of A. The work [2] introduced a worst-case, theoretical measure, akin to the competitive ratio, for evaluating the performance of X, which relates an interruption time T to the largest contract length completed by time T. This measure is called the *acceleration ratio* of X. Formally,

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

where $\ell(X,T)$ denotes the length of the largest contract completed in X by time T.

Contract scheduling is a classic resource-allocation problem that has been studied in many settings. For the standard version described above, it is straightforward to show that the doubling rule yields a schedule of optimal acceleration ratio equal to 4, however the problem becomes substantially more involved under more complex settings. We review some related results. Schedules for multi-processor systems were first obtained in [4]. The work [5] studied the setting in which there are several problem instances that must be solved concurrently on a single processor, and [3] as well as [6] provided optimal acceleration ratios for the multi-instance/multi-processor generalization. A study of the setting in which the interruption is not a hard deadline (and the system is allowed some additional time to wrap up its execution was given in [7], whereas [8] introduced performance measures alternative to the

acceleration ratio. [9] considered a setting in which the schedule is deemed complete once a length-related guarantee is reached on its last completed contract. Connections between contract scheduling and searching for a hidden target under the competitive ratio were shown in [3, 10]. We emphasize that all of the above works establish theoretical upper and lower bounds on the acceleration ratio.

Contract scheduling can be considered as an application of a general *sequencing* formulation, in which we seek an increasing sequence that optimizes a given performance measure. Another sequencing problem that has been well studied within TCS is *online bidding* [11, 12, 13, 14, 15, 16]. Here, the objective is to find an increasing sequence $X = (x_i)_{i\geq 0}$ of positive numbers of minimum *competitive ratio*, defined formally as

$$\sup_{u\geq 0} \frac{\sum_{j=1}^{i} x_j}{u} \ : x_{i-1} < u \leq x_i, \tag{2}$$

where u represents an unknown *target* value. Online bidding has applications in problems such as clustering [12], searching on the infinite line [17] and latency minimization [18], see e.g., the survey [11]. The doubling strategy $X = (2^i)_{i\geq 0}$ achieves once again an optimal competitive ratio equal to 4.

1.1. Competitive sequencing with queries

Recently, contract scheduling was studied under a model in which the scheduler leverages predictions on the interruption time, in the form of responses to k *binary queries*, for some given $k \in \mathbb{N}^+$ [19]. For example, a query may be of the form "will the interruption occur before time $t = 100$?", or may be more complex, e.g., "will the interruption occur in $\cup_{i \text{ odd}}[2^i, 2^{i+1}]$?". Formally, a query maps a statement concerning the interruption time to $\{0, 1\}$.

As explained in [19], queries can help improve resource planning in many settings. For instance, in a medical-diagnostic system, the end user may know that the system is likely to be consulted at some critical intervals, e.g., around some anticipated surgery slots, or that it may be more likely that a diagnosis would be needed on a weekday rather than on a weekend.

Figure 1 illustrates an example for the simple case of a single query. Note that in the context of sequencing problems such as contract scheduling and online bidding, the query responses are powerful enough to encode *interval* information about the interruption or the target. Such information cannot be captured by simpler prediction oracles that provide a single value that corresponds to the anticipated interruption or target. The queries are answered by an imperfect oracle; namely, $\eta \leq k$ queries may receive erroneous responses. We refer to η as the *query error*, and we note that η is not known to the system designer ahead of time.

Figure 1: An illustration of the query-based setting for $k = 1$. Here, a single query may answer whether the interruption is in the yellow (lighter) or in the blue (darker) partition. The decision-maker may use the query response to choose one of the schedules S_1 and S_2.

The query prediction model combines two essential aspects of *learning-augmented* optimization: The first aspect is the use of *predictions* in optimization under uncertainty, in which the (typically online) algorithm is augmented with an inherently erroneous prediction oracle, and the objec-

tive is to improve the algorithm's performance via a robust leveraging of the prediction oracle. Such considerations have become very prevalent in recent studies of algorithmic performance, starting with the influential works [20] and [21], see e.g., the survey [22] and the online repository [23] that lists several related works. The second aspect pertains to *query-based* optimization, in which the algorithm recovers the solution to a problem by asking queries. An example is clustering with noisy queries [24, 25], where a query asks whether two elements belong in the same cluster, which is useful in crowdsourcing applications. Query-based oracles can thus help model *parsimonious* predictions, and similar models were recently studied in the context of secretary problems [26] and online paging [27].

As in the analysis of learning-augmented algorithms, e.g. [28, 29], we first evaluate the performance of the algorithms in two extreme situations with respect to the prediction error. In the one extreme, the oracle is perfect, thus $\eta = 0$: in this case, we refer to the acceleration ratio of a schedule as its *consistency*. In the other extreme, the oracle is malicious (i.e., all responses are adversarial): in this case, we refer to the acceleration ratio of a schedule as its *robustness*. Here, the goal is to find *Pareto-optimal* schedules that describe the optimal consistency/robustness tradeoffs, similarly for online bidding under the competitive ratio. Pareto-efficient algorithms with respect to consistency-robustness tradeoffs have become prominent in the context of online optimization problems with untrusted predictions, see e.g., [14, 28, 29, 30, 31, 32, 33, 34].

Beyond these two extremes, we are also interested in efficient algorithms that are robust to query errors. To this end, [19] introduced a *noisy query*

model, specified as follows. The decision maker defines a parameter $\tau \leq k$, which is interpreted as an anticipated *upper bound* on the query error or, alternatively, as the algorithm's desired *tolerance* to the error. This parameter captures the requirement that the algorithm must perform well if this upper bound is met, i.e., if $\eta \leq \tau$, but must also have robustness at most r, for some specified $r \geq 4$, if it so happens that $\eta > \tau$. Such a parameter is common in the analysis of *games with a lying responder*, e.g., [35], in which an upper bound on the erroneous responses is assumed to be known. It is also related to the concept of *weak predictions*, (see, e.g., online knapsack with frequency predictions [36], in which the prediction is an upper bound on the number of items of a given value in the input). We emphasize that unlike the noisy query model of [24], we do not make any probabilistic assumptions on the query responses.

1.2. Contribution

Results. Our first result (Theorem 13) proves the Pareto-optimal trade-off between the consistency c and the robustness r for contract scheduling with k queries, for any value of r. This answers the main problem left open in [19], which showed a tight tradeoff only for the special case of $r = 4$. To prove our result, we give a tight, information-theoretic lower bound on the consistency of r-robust schedules, which matches the known upper bound (Theorem 11). This result also establishes properties useful in the other settings we study. We also show, using a reduction from contract scheduling to online bidding, that the same optimal guarantees carry over to the latter problem as well (Theorem 15). This result answers an open question from [14] that gave a non-tight lower bound assuming $r = 4$.

Our second result concerns the general query model. Here, given a robustness requirement r, we obtain a schedule that is optimal in the ideal case that $\eta = 0$, remains r-robust for adversarial error (i.e., if η is as high as k), and has acceleration ratio that provably degrades gently as a function of the anticipated bound on the error τ, as long as $\eta \leq \tau \leq k/4 - o(k)$[1] (Theorem 20). More importantly, we show that as long as the error is not prohibitively large, namely for any $\eta \leq k/4 - o(k)$, the acceleration ratio converges very quickly to the error-free consistency as k increases. In other words, we prove that small increments in the number of queries lead to significant gains in robustness to errors, even for substantially and unpredictably erroneous queries. This is in contrast to [19], where the acceleration ratio converges to a much larger and sub-optimal value (Observation 21). Our analysis also implies that the scheduler is not constrained by the choice of a particular tolerance parameter τ, even if k is a rather small constant. The same guarantees can be extended to online bidding.

To prove the above results, we develop techniques that are also applicable to multi-criteria contract scheduling unrelated to the query setting. As an example, in our third main result, we introduce and study a *robust* generalization of *parallel*, *fault-tolerant* contract scheduling. In the original problem studied by Kupavskii and Welzl [37], the objective is to optimize the acceleration ratio of a contract schedule in a parallel p-processor system, assuming that at most f processors may be faulty, for some given $f < p$; however, the schedule may be arbitrarily bad if the number of faulty processors exceeds f.

[1]Here, the term $o(k)$ describes any fixed, and slowly increasing function $f(k) \in o(k)$, e.g., $f(k) = \log^*(k)$.

In our setting, instead, we additionally require that the system remains efficient if all but a single processor are faulty. We show how to obtain a schedule with the best-possible tradeoff between the two performance objectives (Theorems 22 and 24). Last, in Section 6 we provide an experimental evaluation of our algorithms that demonstrates the performance improvements that are attained in practice.

Techniques. In regards to Pareto-optimality, we give a lower bound on the consistency of any r-robust schedule with k queries by treating it as a "virtual" multi-processor scheduling problem in 2^k identical parallel processors, where each processor executes an r-robust schedule. The robustness requirement adds significant complexity to our setting, which is more involved than the standard multi-processor contract scheduling studied by López-Ortiz et al. [6]. Specifically, while the algorithm and the analysis in [6] allow for schedules that are non-robust, in our setting, each processor is required to implement an r-robust schedule. This necessitates a more careful analysis by looking deeper into the linear recurrence relations (equalities and inequalities) that formulate the concept of robustness, as we will show in Section 3. In particular, in order to establish Pareto-optimality, we relate inequalities expressed via linear recurrences to the solution of linear recurrence relations (e.g., in the statement and proof of Proposition 6) and rely on upper-limit calculus (e.g., in the proof of Lemma 9).

To achieve robustness to query errors, we first define a space of 2^k schedules that include the Pareto-optimal one and which has a "nice" structure: namely, there exists an ordering such that if the j-th schedule has the best performance (for a given interruption), then we can bound the acceleration

ratio of the $(j-x)$-th schedule in the ordering as a function of x. We then use an error-tolerant binary search algorithm inspired by Disser and Kratsch [38] based on the query responses, so as to find a schedule close to the best one, even in the presence of errors, and without any knowledge of the ordering.

We emphasize that our approach is based on *adaptive* queries, in that the i-th query is a function of the responses to queries $1 \dots i-1$. Adaptive queries allow searching an exponential space of candidate schedules, unlike [19] which relies on static queries that can only help search a linear space of schedules (as a function of the number of queries k). We thus demonstrate that adaptivity is important for optimality.

2. Preliminaries

We introduce notation for the contract scheduling problem; for online bidding, our results carry over via a reduction shown in Theorem 15. In a single-processor system, a schedule X is defined as an increasing sequence of the form $X = (x_i)_{i \geq 0}$. We make the standing assumption that the interruption occurs after at least the first contract has completed its execution; otherwise, no schedule has a bounded acceleration ratio. Without queries, the acceleration ratio of X is given by (1). It is well-known that the worst-case interruptions that maximize $\mathtt{acc}(X)$ are infinitesimally prior to the completion times of contracts, namely $x_i - \epsilon$, for $\epsilon \to 0$. Hence the equivalent expression (where x_{-1} is defined to be equal to 1)

$$\mathtt{acc}(X) = \sup_i \frac{\sum_{j=0}^{i} x_j}{x_{i-1}}. \tag{3}$$

For a schedule X with k queries, [19] defines the *consistency* of X as

its acceleration ratio assuming no response errors, and its *robustness* as its acceleration ratio assuming worst-case, adversarial responses. A schedule is r-*robust* if its robustness is at most r (similarly for consistency), and it is *Pareto-optimal* if its consistency and robustness are in a best-possible relation. Note that the robustness of a schedule is equal to its acceleration ratio in the standard setting (since both definitions involve worst-case, adversarial settings). Hence, it is always the case that $r \geq 4$, and we will thus use these two terms interchangeably.

A schedule G_b is called *geometric with base* $b > 1$ if it is of the form $G_b = (b^i)_{i \geq 0}$. Geometric schedules are significant since they are often efficient for several variants of contract scheduling. It is known that the acceleration ratio of G_b is equal to $b^2/(b-1)$ [5], which is minimized for $b = 2$, thus $G_2 = (2^i)_{i \geq 0}$ has optimal acceleration ratio equal to 4. More generally, one can easily identify the geometric schedules that are r-robust, for any given r.

Definition 1. *For any given r, define $\zeta_{1,r}$ and $\zeta_{2,r}$ as the smallest and largest real roots, respectively, of the function $f(x) = \frac{x^2}{x-1} - r$. It also holds that $1 < \zeta_{1,r} \leq \zeta_{2,r}$.*

This definition implies the following useful property:

Property 2. *The schedule G_b with $b > 1$ has acceleration ratio at most $b^2/(b-1)$. In particular, for any $r \geq 4$, G_b has acceleration ratio at most r if and only if $b \in [\zeta_{1,r}, \zeta_{2,r}]$.*

The roots of Definition 1 are easily computable, i.e., $\zeta_{1,r} = (r - \sqrt{r^2 - 4r})/2$ and $\zeta_{2,r} = (r + \sqrt{r^2 - 4r})/2$, and are such that $\zeta_{1,r} \in (1, 2]$, and $\zeta_{2,r} \geq 2$, for any $r \geq 4$.

Example 3. *Suppose that $r = 6$, then $\zeta_{1,6} = 3-\sqrt{3}$, and $\zeta_{2,6} = 3+\sqrt{3}$. Any geometric schedule G_b, with $b \in [3-\sqrt{3}, 3+\sqrt{3}]$ is 6-robust.*

The above definitions assume a single processor. In our work, we show and exploit connections between single-processor schedules with k queries and multi-processor schedules without queries on 2^k parallel processors. Hence, we present some definitions and notation concerning the setting of $p > 1$ parallel processors, labelled $\{0, \ldots, p-1\}$. In the p-processor setting, each processor j defines its own *strategy* of the form $X_j = (x_{i,j})_{i \geq 0}$. We call the set $X = \{X_j\}_{j=0}^{p-1}$ a *p-processor* schedule, or equivalently, we say that X is *defined by the set* $\{X_j\}_{j=0}^{p-1}$. The acceleration ratio of a p-processor schedule is defined as in (1), with the difference that $\ell(X, T)$ denotes the largest contract length completed by time T among all p processors [3].

Let $Y = (y_i)_{i=0}^{\infty}$ denote a positive sequence, and define $\alpha_Y = \limsup_{n\to\infty} y_n^{1/n}$. E.g., if $Y = (2^i)_{i=0}^{\infty}$, then $\alpha_Y = 2$. This notion appears in the statement of a theorem by Gal [39], which is the basis of the analysis in the multiprocessor setting of [6]. This theorem, informally, gives a lower bound on the supremum of a set of functionals by the supremum of these functionals over geometrically increasing sequences. Given a set (or sequence) Y of positive reals, we denote by $\bar{Y}$ the sequence of all elements in Y in non-decreasing order. We refer to Appendix Appendix A for the formal statement of Gal's theorem, which will not be of direct use in our work, though the notion of α_Y will figure prominently. Table 1 summarizes some important notation used throughout this work.

Notation	Description
T	interruption time (unknown to the scheduler)
$\ell(X,T)$	largest contract completed in schedule X by time T
$\mathtt{acc}(X)$	acceleration ratio of schedule X
k	number of queries
G_b	geometric strategy with base $b>1$, i.e., of the form $(b^i)_{i=0}^{\infty}$
r	robustness ($r \geq 4$)
$\zeta_{1,r}, \zeta_{1,r}$	the smallest and largest positive roots of the function $f(x) = \frac{x^2}{x-1} - r$.
α_Y	$\limsup_{n\to\infty} y_n^{1/n}$ of the sequence $Y = (y_i)_{i=0}^{\infty}$
$\mathcal{X}_{b,l}$	the set of strategies $\{X_0, \ldots X_{l-1}\}$, where $X_i = (b^{i+jl})_{j=0}^{\infty}$ ($i \in [0, 2^k-1]$) (see Definition 12.)
η	(unknown) number of erroneous query responses ($\eta \leq k$)
τ	anticipated upper bound on η, or tolerance
U	defined as $2^{\lfloor k/2 \rfloor + 2\tau}$
$c(k,r)$	consistency of the Pareto-optimal schedule of Theorem 11
$R(k,r,\tau)$	acceleration ratio of the schedule RQS of Theorem 18

Table 1: Summary of notation and definitions.

3. Pareto-optimality of contract scheduling and online bidding

In this section, we give the optimal consistency-robustness tradeoff for both sequencing problems, with access to k queries. Our main result is a tight information-theoretic lower bound that applies to any schedule with information encoded as a k-bit string, which we call the *response string*.

Overview of the proof. We first give an overview and the intuition of the proof. The starting observation is that any r-robust schedule X with k queries must be selected from a set $\mathcal{X}$ of at most 2^k r-robust schedules. This implies that the consistency of X is the acceleration ratio of the 2^k-processor schedule defined by $\mathcal{X}$. We can use a lower bound on this acceleration ratio as a function of the parameter $\alpha_{\bar{\mathcal{X}}}$ (Lemma 4), and recall that $\bar{\mathcal{X}}$ is the sequence of all contract lengths of schedules in $\mathcal{X}$, in non-decreasing order.

There is, however, a substantial complication, in that it is imperative to show that $\alpha_{\bar{\mathcal{X}}}$ is within a certain range in order to establish the tightness of the lower bound; this is to capture the requirement that each strategy in $\mathcal{X}$ must be r-robust. To this end, we show that $\alpha_{\bar{\mathcal{X}}} \in [\zeta_{1,r}^{1/2^k}, \zeta_{2,r}^{1/2^k}]$ (Corollary 10). This is accomplished by first showing upper and lower bounds on the contract lengths of any r-robust schedule (Theorem 5), then applying the definitions of the sorted sequence of contract lengths $\bar{\mathcal{X}}$ and the properties of upper limits (Lemma 9). Combining the above yields the lower bound (Theorem 11). The tightness of the result will follow by directly comparing to the upper bound of [19] (Theorem 13).

Analysis. We now proceed with the technical analysis. The following lemma is due to [6] and is a special case of a more general result that incorporates fault tolerance and which we will prove later, namely Lemma 23.

Lemma 4. [6] *Let X be a p-processor schedule, as defined by a set $\mathcal{X}$ of p single-processor strategies, each of which has a finite acceleration ratio. Then* $\mathtt{acc}(X) \geq \frac{\alpha_{\bar{\mathcal{X}}}^{p+1}}{\alpha_{\bar{\mathcal{X}}}^{p}-1}$.

In the next step, we show upper and lower bounds on the contract lengths of any r-robust schedule. We emphasize that these bounds apply to any

schedule of acceleration ratio r without queries. This will be instrumental in bounding the range of $\alpha_{\bar{\mathcal{X}}}$.

Theorem 5. *Let $Z = (z_i)_{i\geq 0}$ be an r-robust single-processor schedule, for a fixed, finite $r \geq 4$. Then there exist c, d which are only functions of r, such that $z_i \leq c\cdot\zeta_{2,r}^i$ and $z_i \geq d\cdot\zeta_{1,r}^i$, if $r > 4$. Moreover, $z_i \leq c\cdot i\cdot 2^i$ and $z_i \geq d\cdot 2^i$, if $r = 4$.*

To prove Theorem 5, we first show the following technical result that relies on linear recurrence relations. For some intuition, Proposition 6 relates a general r-robust strategy X to another r-robust strategy Y, which satisfies all constraints with equality. This will be helpful not only in establishing the upper bounds in Theorem 5 but, more importantly, the lower bounds, which is the harder part.

Proposition 6. *Let $(x_i)_{i=0}^{\infty}$ and $(y_i)_{i=0}^{\infty}$ be sequences of positive numbers such that*

$$\sum_{i=0}^{n} x_i \leq r x_{n-1} \quad \text{and} \quad \sum_{i=0}^{n} y_i = r y_{n-1}, \tag{4}$$

for all $n \geq 2$, where $r \geq 4$ is a fixed constant. If $x_0 \geq y_0$ and $x_1 \leq y_1$, then $x_i \leq y_i$, for all $i \geq 2$.

Proof. First, we introduce the sequence of coefficients $(A_t, B_t)_{t=1}^{\infty}$ defined recursively by

$$\begin{pmatrix} A_{t+1} \\ B_{t+1} \end{pmatrix} = M \begin{pmatrix} A_t \\ B_t \end{pmatrix},$$

where

$$M = \begin{pmatrix} r-1 & -1 \\ 1 & 1 \end{pmatrix},$$

with initial values $(A_1, B_1) = (r-1, 1)$. We claim that

$$A_t, B_t \geq 0, \quad \text{for every } t \geq 1. \tag{5}$$

Let us momentarily assume the validity of (5) and complete the proof of the proposition. To that end, we show that

$$\begin{aligned} x_n &\leq A_t x_{n-t} - B_t \sum_{i=0}^{n-t-1} x_i \\ \text{and} \quad y_n &= A_t y_{n-t} - B_t \sum_{i=0}^{n-t-1} y_i, \end{aligned} \tag{6}$$

for all $1 \leq t < n$. This follows from an induction argument on t. Indeed, for a given $n \geq 2$, the case $t = 1$ is a mere reformulation of (4). Then, in view of (5), assuming that (6) holds for some $1 \leq t \leq n-2$, we obtain

$$\begin{aligned} x_n &\leq A_t x_{n-t} - B_t \sum_{i=0}^{n-t-1} x_i \\ &\leq A_t (r x_{n-t-1} - \sum_{i=0}^{n-t-1} x_i) - B_t \sum_{i=0}^{n-t-1} x_i \\ &= (A_t(r-1) - B_t) x_{n-t-1} - (A_t + B_t) \sum_{i=0}^{n-t-2} x_i \\ &= A_{t+1} x_{n-t-1} - B_{t+1} \sum_{i=0}^{n-t-2} x_i \end{aligned}$$

and similarly for y_n, thereby establishing (6) for all $1 \leq t < n$.

Now, observe that setting $t = n-1$ in (6) yields

$$x_n \leq A_{n-1} x_1 - B_{n-1} x_0 \leq A_{n-1} y_1 - B_{n-1} y_0 = y_n, \tag{7}$$

for all $n \geq 2$, which completes the proof.

Thus, there only remains to justify the bound (5), which will easily follow from an explicit representation formula for (A_t, B_t) based on an eigenvector decomposition of M. More precisely, straightforward calculations establish that the eigenvalues of M are the two roots $\zeta_{1,r} \leq \zeta_{2,r}$ of the characteristic polynomial $p(\zeta) = \zeta^2 - r\zeta + r$, as defined also in Definition 1. These roots are both positive if $r \geq 4$ and distinct whenever $r > 4$. In fact, it is readily seen that $\zeta_{2,r} \geq \zeta_{1,r} > 1$. Moreover, it holds that $\zeta_{2,r} + \zeta_{1,r} = r$, $\zeta_{2,r}\zeta_{1,r} = r$ and $(\zeta_{2,r} - 1)(\zeta_{1,r} - 1) = 1$.

When $r > 4$, we obtain the eigenvector decomposition

$$\begin{pmatrix} A_t \\ B_t \end{pmatrix} = \frac{\zeta_{2,r}^t}{\zeta_{2,r} - \zeta_{1,r}} \begin{pmatrix} \zeta_{2,r} - 1 \\ 1 \end{pmatrix} - \frac{\zeta_{1,r}^t}{\zeta_{2,r} - \zeta_{1,r}} \begin{pmatrix} \zeta_{1,r} - 1 \\ 1 \end{pmatrix} \tag{8}$$

for every $t \geq 1$, which implies (5) because $\zeta_{2,r} \geq \zeta_{1,r} > 1$. Finally, further letting $r \to 4$ yields the representation[2].

$$\begin{pmatrix} A_t \\ B_t \end{pmatrix} = \begin{pmatrix} 2^t + t2^{t-1} \\ t2^{t-1} \end{pmatrix} \tag{9}$$

in the case $r = 4$, which also validates (5) and thus concludes the proof of the proposition. □

Observation 7. *Observe from* (7), (8) *and* (9) *in the preceding proof that one has the convenient representation formulas, for every* $n \geq 2$,

[2]We include in the analysis the case $r = 4$ so as to demonstrate that the approach applies to all robustness values.

$$\begin{aligned} y_n &= \frac{\zeta_{2,r}^{n-1}(\zeta_{2,r}-1)-\zeta_{1,r}^{n-1}(\zeta_{1,r}-1)}{\zeta_{2,r}-\zeta_{1,r}}y_1 - \frac{\zeta_{2,r}^{n-1}-\zeta_{1,r}^{n-1}}{\zeta_{2,r}-\zeta_{1,r}}y_0 \\ &= \zeta_{2,r}^{n-1}\frac{(\zeta_{2,r}-1)y_1-y_0}{\zeta_{2,r}-\zeta_{1,r}} - \zeta_{1,r}^{n-1}\frac{(\zeta_{1,r}-1)y_1-y_0}{\zeta_{2,r}-\zeta_{1,r}}, \end{aligned} \tag{10}$$

if $r > 4$, *and*

$$\begin{aligned} y_n &= (2^{n-1}+(n-1)2^{n-2})y_1 - (n-1)2^{n-2}y_0 \\ &= 2^{n-1}y_1 + (n-1)2^{n-2}(y_1-y_0), \end{aligned} \tag{11}$$

when $r = 4$. *If one further requires that* (4) *hold for* $n = 1$, *whereby* $y_1 = (r-1)y_0$, *then one finds that*

$$y_n = \frac{\zeta_{2,r}^{n}(\zeta_{2,r}-1)-\zeta_{1,r}^{n}(\zeta_{1,r}-1)}{\zeta_{2,r}-\zeta_{1,r}}y_0,$$

if $r > 4$, *and*

$$y_n = \left(3\cdot 2^{n-1}+(n-1)2^{n-1}\right)y_0,$$

when $r = 4$.

We are now ready to formally prove Theorem 5.

Proof of Theorem 5. First, observe that if $X = (x_i)_{i=0}^{\infty}$ is r-robust, then from (3) it follows that $\sum_{i=0}^{n} x_i \leq r x_{n-1}$. The upper bounds then follow directly from Proposition 6, with $x_0 = y_0 = z_0$ and $x_1 = y_1 = z_1$, and the representation formulas (10) and (11).

The lower bounds, in contrast, are more subtle. In order to establish their validity, we define, for any given integer $j \geq 0$, an auxiliary sequence $(x_i)_{i=0}^{\infty}$ by

$$x_0 = \sum_{k=0}^{j} z_k \quad \text{and} \quad x_i = z_{j+i}, \text{ if } i \geq 1.$$

In particular, it holds that $\sum_{i=0}^{n} x_i \leq r x_{n-1}$, for all $n \geq 2$. Therefore, by Proposition 6 combined with the formulas for y_n from Observation 7 we

deduce that, for all $n \geq 2$,

$$x_n \leq \zeta_{2,r}^{n-1} \frac{(\zeta_{2,r}-1)x_1 - x_0}{\zeta_{2,r}-\zeta_{1,r}} - \zeta_{1,r}^{n-1} \frac{(\zeta_{1,r}-1)x_1 - x_0}{\zeta_{2,r}-\zeta_{1,r}},$$

if $r > 4$, and $x_n \leq 2^{n-1}x_1 + (n-1)2^{n-2}(x_1 - x_0)$, for $r = 4$. Since $(x_i)_{i=0}^{\infty}$ is also nonnegative and, as $n \to \infty$, the dominant terms above are $\zeta_{2,r}^{n-1}$ and $(n-1)2^{n-2}$, we conclude that necessarily $x_0 \leq (\zeta_{2,r}-1)x_1$, for all values $r \geq 4$. In terms of the original sequence $(z_i)_{i=0}^{\infty}$, observing that $(\zeta_{2,r}-1)(\zeta_{1,r}-1) = 1$, this yields that

$$(\zeta_{1,r}-1)\sum_{k=0}^{j} z_k \leq z_{j+1},$$

for every $j \geq 0$. In particular, if $z_{i+1} \geq (\zeta_{1,r}-1)\zeta_{1,r}^{i} z_0$ holds for every $0 \leq i \leq j$, then

$$\begin{aligned} z_{j+2} &\geq (\zeta_{1,r}-1)\sum_{k=0}^{j+1} z_k \geq (\zeta_{1,r}-1)z_0 + (\zeta_{1,r}-1)^2 z_0 \sum_{k=1}^{j+1} \zeta_{1,r}^{k-1} \\ &= (\zeta_{1,r}-1)z_0 + (\zeta_{1,r}-1)^2 z_0 \frac{\zeta_{1,r}^{j+1}-1}{\zeta_{1,r}-1} = (\zeta_{1,r}-1)\zeta_{1,r}^{j+1} z_0, \end{aligned}$$

thereby completing the proof of the lower bounds by induction. □

Observation 8. *The bounds of Theorem 5 are asymptotically tight. This is because Property 2 states that any geometric schedule G_b with $b \in [\zeta_{1,r}, \zeta_{2,r}]$ is r-robust.*

Using the calculus of upper limits, we prove a property of merged sequences which allows us to bound the range of $\alpha_{\bar{\mathcal{X}}}$. The technical proof is given in Appendix Appendix B.

Lemma 9. *Let $(x_{1,i})_{i=1}^{\infty}$, $(x_{2,i})_{i=1}^{\infty}$, ..., $(x_{N,i})_{i=1}^{\infty}$ be N positive nondecreasing sequences satisfying the bounds $c_A A^i \leq x_{j,i} \leq c_B i B^i$, for all i and j, where*

$c_A, c_B > 0$ *and* $A, B > 1$ *are given constants. Let* $(y_i)_{i=1}^{\infty}$ *be the sequence obtained by merging all* $(x_{j,i})_{i=1}^{\infty}$ *and sorting the resulting set of values in nondecreasing order. Then,*

$$A^{\frac{1}{N}} \leq \liminf_{i\to\infty} y_i^{\frac{1}{i}} \leq \limsup_{i\to\infty} y_i^{\frac{1}{i}} \leq B^{\frac{1}{N}}.$$

Theorem 5, Lemma 9, and the definition of $\alpha_{\bar{\mathcal{X}}}$ yield:

Corollary 10. *Let* X *be a* p*-processor schedule defined by the set* $\mathcal{X} = \{X_0, X_1, \ldots, X_{p-1}\}$*, where each* X_j *is an* r*-robust strategy, for a given* $r \geq 4$*. Then* $\alpha_{\bar{\mathcal{X}}} \in [\zeta_{1,r}^{1/p}, \zeta_{2,r}^{1/p}]$.

We now state the main result of this section. Its proof formalizes the intuition given at the beginning of the section.

Theorem 11 (Lower Bound)**.** *Any* r*-robust contract schedule with* k *queries has consistency at least* $c(k, r)$*, where*

$$c(k,r) = \min_x \frac{x^{2^k+1}}{x^{2^k}-1} \quad \textit{subject to} \quad \zeta_{1,r}^{1/2^k} \leq x \leq \zeta_{2,r}^{1/2^k}. \tag{12}$$

Proof. Any schedule X with k queries will choose a schedule among a set of at most 2^k schedules, say $\mathcal{X} = \{X_0, \ldots, X_{2^k-1}\}$. For X to be r-robust, it must be that each X_i, with $i \in [0, 2^k-1]$ is likewise r-robust; otherwise, maliciously generated responses would lead to choosing a schedule of robustness greater than r. Note that the consistency of X is identical to the acceleration ratio of the 2^k-processor schedule that is defined by $\mathcal{X}$. Let X' denote this multi-processor schedule. From Lemma 4, we have that $\texttt{acc}(X') \geq \frac{\alpha_{\bar{\mathcal{X}}}^{2^k+1}}{\alpha_{\bar{\mathcal{X}}}^{2^k}-1}$. Last,

since every single-processor schedule X_i must be r-robust, from Corollary 10 it follows that $\alpha_{\bar{\mathcal{X}}} \in [\zeta_{1,r}^{1/2^k}, \zeta_{2,r}^{1/2^k}]$. □

We will argue that the lower bound of Theorem 11 matches the upper bound of [19]. Specifically, consider the following class of single-processor schedules.

Definition 12. *For given $b > 1$, and $l \in \mathbb{N}^+$ define $\mathcal{X}_{b,l}$ as the set of schedules $\{X_0, \dots X_{l-1}\}$, in which $X_i = (b^{i+jl})_{j\geq 0}$, for every $i \in [0, l-1]$.*

In other words, each schedule X_i in $\mathcal{X}_{b,l}$ is a geometric schedule with base b^l, whose lengths are multiplied by a factor b^i. In [19], a $b > 1$ is chosen so as to guarantee that all strategies in $\mathcal{X}_{b,2^k}$ are r-robust, and if the response string is error-free, then the acceleration ratio of $\mathcal{X}_{b,2^k}$ is at most $\frac{b^{2^k+1}}{b^{2^k}-1}$. Thus, there is an r-robust schedule of consistency at most

$$\min_b \frac{b^{2^k+1}}{b^{2^k}-1} \quad \text{subject to} \quad \zeta_{1,r}^{1/2^k} \leq b \leq \zeta_{2,r}^{1/2^k} \text{ and } b > 1. \tag{13}$$

The tightness of our lower bound follows from directly comparing (13) with Theorem 11 and the fact that $\zeta_{1,r} > 1$. We can also express the optimal consistency given by (13), and thus obtain the following result.

Theorem 13 (Pareto-optimality)**.** *The optimal consistency of an r-robust schedule with k queries is equal to*

$$c(k,r) = \frac{b^{2^k+1}}{b^{2^k}-1}, \text{ with } b = \begin{cases} \zeta_{2,r}^{1/2^k}, & \text{if } r \leq \frac{(1+2^k)^2}{2^k} \\ (1+2^k)^{1/2^k}, & \text{otherwise.} \end{cases} \tag{14}$$

Figure 2 depicts the optimal consistency-robustness tradeoff as expressed by (14) for various values of k.

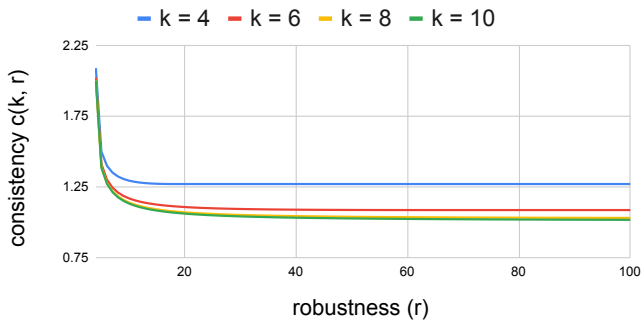


Figure 2: Illustration of the optimal consistency-robustness tradeoff.

Example 14. *For $r = 4$, the optimal consistency is $2^{1+\frac{1}{2^k}}$, thus rapidly converging to 2, as function of k. For large values of r, the consistency converges to $(1+2^k)^{1+1/2^k}/2^k$. E.g., for $k = 4$, the consistency converges to 1.26833, as $r \to \infty$.*

Last, we show that the same Pareto-optimal trade-off can be obtained for online bidding via a reduction from contract scheduling.

Theorem 15. *Any r-robust online bidding strategy, with k queries, has consistency at least $c(k, r)$, as expressed by (14). Furtermore, this result is tight.*

Proof. For the lower bound, we will show a reduction from contract scheduling with k queries. Let T be the unknown interruption time for a given contract scheduling instance. Let $\mathcal{X}$ be an r-robust, c-consistent bidding strategy with k queries, for some given r, c. We will show how to obtain a schedule with the same guarantees. To this end, we can interpret $\mathcal{X}$ as a set of 2^k sequences $\{X_0, \ldots, X_{2^k-1}\}$, each of which must be r-robust. We define the bidding target to be equal to T, and let $Y \in \mathcal{X}$ be the sequence returned

by the query responses for this bidding instance $u = T$.

By definition, Y is r-robust. Let m denote the smallest index for which $y_m \geq T$. Since $\mathcal{X}$ is also c-consistent, we have $\sum_{j=0}^{i} y_j \leq r y_{i-1}$, for all i, and $\sum_{j=0}^{m} y_j \leq cT$. Define $Z = (z_i)_{i \geq 0}$ such that $z_i = y_i/c$. It follows that

$$\sum_{j=0}^{i} z_j \leq r z_{i-1}, \text{ for all } i, \text{ and } \quad z_m \geq \frac{T}{c}, \quad \sum_{j=0}^{m} z_j \leq T. \tag{15}$$

If we interpret Z as a contract schedule, the first equation in (15) shows that Z is r-robust, whereas the last two equations show that it is c-consistent. Therefore, from Theorem 11, it follows that $c \geq c(k, r)$. □

Last, we note that [14] gave an explicit strategy for online bidding of consistency equal to $c(k, r)$, which establishes the tightness of the result.

4. Sequencing with noisy queries

In this section, we extend our study to the noisy query model. We focus on contract scheduling, but we note that the same upper bounds can be easily extended to online bidding using the same techniques. Recall that in this model, $\eta \leq k$ query responses may be erroneous, and the scheduler specifies a parameter $\tau \leq k$ that describes its desired tolerance to errors or, alternatively, an anticipated upper bound on the query error.

We first discuss the intuition behind our approach. The starting observation is that the Pareto-optimal schedule can be found in the class $\mathcal{X}_{b,2^k}$ (recall Definition 12), as shown in Section 3). This class has a nice structural property, as we show in Property 16: if j_T is the index of the best schedule in this class, for interruption T, then any schedule of index close, but smaller

than j_T is likewise quite efficient. In the next step, we show how to apply an *error-tolerant* binary-search algorithm, using k binary queries in the presence of errors, so as to find a schedule in $\mathcal{X}_{b,2^k}$ that has index close to (but no larger than) j_T; Property 16 implies that this schedule should also perform well. In Theorem 18, we quantify this statement and optimize the choice of b. Our main result is Theorem 20: here, we prove that the obtained schedule is extremely robust to errors and very close to the Pareto-optimal one as k increases. It is worth noting that the resulting schedule is determined by *adaptive* queries since the underlying binary search algorithm builds upon adaptive queries. Adaptivity, in particular, helps us find an efficient schedule within the exponential-sized class $\mathcal{X}_{b,2^k}$.

We start with the structural property that follows immediately from Definition 12. See also Figure 3 for an illustration.

Property 16. *Consider the set $\mathcal{X}_{b,l} = \{X_0, \ldots, X_{l-1}\}$ of the l single-processor strategies in Definition 12. For a given interruption time T, let $j_T \in [0, l-1]$ be such that X_{j_T} completes a contract of largest length among all schedules in $\mathcal{X}_{b,l}$, and specifically of length L_0. Then, for all $j \in [0, j_T]$, X_j completes a contract of length at least $b^{j-j_T} L_0$ by T.*

We emphasize that j_T depends on the interruption time T, and is thus unknown to the scheduler. We would like to find a schedule of index close, but no larger than j_T. To model this situation, we define a problem which we call MinSearch. The input to this problem is an unknown array $A[0 \ldots 2^k - 1]$ that stores a permutation of $\{0, \ldots, 2^k - 1\}$, such that there exists a $j^* \in [0, 2^k - 1]$ for which $A[j^* - i] = i$, for all $i \in [0, j^*]$. For example, A can be of the form $A = [3, 2, 1, 0, 6, 4, 7, 5]$, where $j^* = 3$. The objective is to

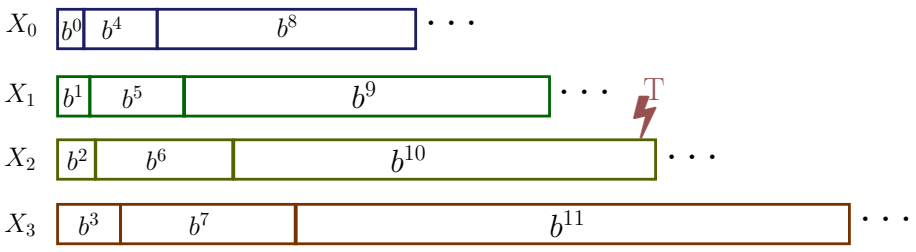


Figure 3: An illustration of the class of schedules $\mathcal{X}_{b,2^k}$, for $k = 2$. Given an interruption T, the schedule that completes the largest contract by time T is X_1 (a contract of length b^9) hence $j_T = 1$.

use k binary queries, at most τ of which can receive erroneous responses, for a given τ, so as to identify an index $i \in [0, j^*]$ such that $A[i]$ is as small as possible, without knowing j^*. In [40], an algorithm called *Robust Binary Interval Search* (RBIS) was given in the context of the online *time-series search* problem, based on an algorithm for a related error-tolerant setting due to [38]. The same algorithm can be applied to MinSearch:

Observation 17. *There is an algorithm for* MinSearch *with k binary queries that outputs an index $i \in [0, j^*]$ such that $A[i] \leq 2^{\lfloor k/2 \rfloor + 2\tau}$, for all $\tau \leq k/4$, if there are at most τ erroneous query responses.*

Thus, in our schedule, each binary query is of the form "is $j_T \leq x$?", where x is chosen in $[0, 2^k - 1]$. Note, however, that from the structure of $\mathcal{X}_{b,2^k}$, one can express equivalently such query as a partition query of the form "is the interruption in A or B?", where A, B form a disjoint partition of the timeline. We refer to [19] for the details on the implementation of such partition queries.

Combining Property 16 and Remark 17, we obtain the following perfor-

mance guarantees. Define $U = 2^{\lfloor k/2 \rfloor + 2\tau}$.

Theorem 18. *For any $r \geq 4$ and $\eta \leq \tau \leq k/4$, there exists a schedule that has acceleration ratio at most*

$$R(k, r, \tau) = \frac{b^{2^k+U+1}}{b^{2^k}-1}, \tag{16}$$

where

$$b = \begin{cases} \zeta_{2,r}^{1/2^k}, & \text{if } \ r \leq \frac{(1+2^k/(1+U))^2}{2^k/(1+U)} \\ (1+2^k/(1+U))^{1/2^k}, & \text{otherwise,} \end{cases}$$

and where $U = 2^{\lfloor k/2 \rfloor + 2\tau}$. Otherwise, the acceleration ratio of the schedule is at most r.

Proof. We apply the error-tolerant binary search algorithm of Remark 17 for MinSearch on the set of indices of all schedules in $\mathcal{X}_{b,2^k}$, where $b > 1$ will be chosen later. Assuming that $\eta \leq \tau$, then the output is the index of a strategy in $\mathcal{X}_{b,2^k}$ which is ranked at most U among the schedules in $\mathcal{X}_{b,2^k}$, in regards to its largest completed contract (where lower ranking indicates better performance). From Property 16, this means that the selected schedule has length at least L_0/b^U, where L_0 is the length of the largest contract completed by time T among all schedules in $\mathcal{X}_{b,2^k}$. It is easy to evaluate the latter formally (see the details in the proof of Theorem 22), and we infer that the acceleration ratio of the selected schedule is at most $\frac{b^{2^k+1+U}}{b^{2^k}-1}$.

Furthermore, given the desired robustness $r \geq 4$, we require that each schedule in $\mathcal{X}_{b,2^k}$ must be r-robust. From Property 2, this is equivalent to the constraint $\zeta_{1,r} \leq b^{2^k} \leq \zeta_{2,r}$. Therefore, the acceleration ratio of the schedule is equal to

$$\min \frac{b^{2^k+1+U}}{b^{2^k}-1}, \text{ subject to } \ \zeta_{1,r} \leq b^{2^k} \leq \zeta_{2,r}.$$

Optimizing the above expression yields the result. □

The schedule of Theorem 18 is very robust to query errors. More precisely, in Theorem 20, we show that as long as $\eta \leq \tau \leq k/4 - o(k)$, not only does the acceleration ratio degrade gently as a function of τ, but, more importantly, it improves rapidly as k increases, and approaches the ideal performance of the Pareto-optimal schedule. Recall that $c(k, r)$ and $R(k, r, \tau)$ are given by (14) and (16), respectively. We first define a function that will help us express and prove the performance guarantee.

Definition 19. *For all $x \geq 1$, define the function $g(x) = \frac{(1+x)^{1+\frac{1}{x}}}{x}$.*

Note that g is a decreasing function of x and converges to 1 as $x \to \infty$.

Theorem 20. *For all $k \in \mathbb{N}^+$, $r \geq 4$ and $\eta \leq \tau \leq k/4$, it holds that*

$$R(k, r, \tau) \leq f(k, r, \tau) \cdot c(k, r),$$

where $f(k, r, \tau) = \frac{g(\frac{2^k}{1+U})}{g(2^k)} \cdot \zeta_{2,r}^{\frac{U}{2^k}}$.

Proof. We consider cases, depending on whether $r \leq \frac{(1+2^k)^2}{2^k}$ or not (see (14)) and whether $r \leq \frac{(1+2^k/(1+U))^2}{2^k/(1+U)}$ or not (see (16)). To simplify notation, define

$$\rho_0 = \frac{(1+2^k)^2}{2^k} \qquad \text{and} \qquad \rho_1 = \frac{(1+2^k/(1+U))^2}{2^k/(1+U)},$$

and note that it is always the case $\rho_0 \geq \rho_1$, since the function $(1+x^2)/x$ is increasing for all $x \geq 1$. In the discussion that follows, recall that the function is defined in Definition 19.

Case 1: $r \leq \rho_1$ (thus $r \leq \rho_0$ as well). In this case,

$$R(k, r, \tau) = \frac{\zeta_{2,r}^{1+\frac{1+U}{2^k}}}{\zeta_{2,r} - 1}, \text{ and } c(k, r) = \frac{\zeta_{2,r}^{1+\frac{1}{2^k}}}{\zeta_{2,r} - 1}.$$

Therefore, we have that $R(k,r,\tau) = \zeta_{2,r}^{\frac{U}{2^k}} c(k,r)$, and the theorem follows .

Case 2: $r > \rho_0$ (and thus $r > \rho_1$ as well). In this case,

$$R(k,r,\tau) = g(2^k/(1+U)), \quad \text{and} \quad c(k,r) = g(2^k),$$

therefore we have that

$$\frac{R(k,r,\tau)}{c(k,r)} = \frac{g(\frac{2^k}{1+U})}{g(2^k)},$$

and the theorem follows.

Case 3: $r \in (\rho_1, \rho_0]$. For this case to apply, it must be that

$$\zeta_{2,r} - 1 \leq 2^k \quad \text{and} \quad \zeta_{2,r} - 1 \geq \frac{2^k}{1+U}. \tag{17}$$

We have that

$$R(k,r,\tau) = g(\frac{2^k}{1+U}) \quad \text{and} \quad c(k,r,\tau) = \frac{\zeta_{2,r}^{1+\frac{1}{2^k}}}{\zeta_{2,r} - 1}, \tag{18}$$

Define $y = \frac{2^k}{1+U}$ and $z = \zeta_{2,r} - 1$, then by (17) we have that $z \geq y$. Furthermore, we can express $c(k,r)$ equivalently as

$$c(k,r,\tau) = g(z) \frac{\zeta_{2,r}^{\frac{1}{2^k}}}{\zeta_{2,r}^{\frac{1}{z}}}. \tag{19}$$

Combining the above, we obtain

$$\begin{aligned}
\frac{R(k,r,\tau)}{c(k,r)} &= \frac{g(y)}{g(z)} \cdot \zeta_{2,r}^{\frac{1}{z} - \frac{1}{2^k}} && \text{(From (18) and (19))} \\
&= \frac{g(y)}{g(z)} \cdot \zeta_{2,r}^{\frac{U}{2^k}} && \text{(From (17))} \\
&= \frac{g(\frac{2^k}{1+U})}{g(\zeta_{2,r} - 1)} \cdot \zeta_{2,r}^{\frac{U}{2^k}} && \text{(def of } z, y\text{)} \\
&\leq \frac{g(\frac{2^k}{1+U})}{g(2^k)} \cdot \zeta_{2,r}^{\frac{U}{2^k}}. && \text{(From monotonicity of } g \text{ and (17))}
\end{aligned}$$

□

Theorem 20 leads to interesting conclusions in regard to the robustness of the schedule to errors. Specifically, simple calculus shows that the function rapidly converges to 1, as $k \to \infty$, even if τ is as large as $k/4 - o(k)$. This implies that the scheduler is not constrained by the choice of a particular tolerance parameter τ, even if k is a fairly small constant. Namely, even if we choose a very pessimistic value for τ such as $\tau = k/4 - o(k)$, and even if the error is as high as $k/4 - o(k)$, the schedule has an acceleration ratio very close to the ideal case of error-free queries (i.e., as good as its consistency). This improves considerably upon the state-of-the-art schedule:

Observation 21. *The schedule of [19] does not exhibit comparable robustness. To see this, consider the case $r = 4$, for which their acceleration ratio is $2^{1+\frac{1}{k}+2\frac{\tau}{k}}$. Suppose that $\tau = \Theta(k)$, i.e., $\tau = c \cdot k$ for constant $c < 1/4$, then this is equal to $2^{1+\frac{1}{k}+2c}$, which not only can be much larger than the ideal consistency, if c is large, but is also practically unaffected by the number of queries k.*

5. Robust fault-tolerant contract scheduling

The techniques we introduced in Sections 3 and 4 can be applied to other multi-criteria optimization settings unrelated to query predictions. We illustrate, using as an example the *robust, fault-tolerant contract scheduling* problem, defined as follows. Suppose we are given a system of p identical parallel processors, a *robustness* requirement r, and a *fault-tolerance* parameter $f < p$. The objective is to obtain a schedule of contact algorithms in the system of p parallel processors that has a minimum acceleration ratio if up to f processors can be faulty but also has an acceleration ratio at most

r if all but a single processor are faulty. Here, a fault is byzantine, in that a processor may cease to work at any point in time. We denote this problem as $\texttt{RFT}(r, p, f)$. We emphasize that there is no concept of predictions in this formulation.

This problem generalizes the fault-tolerant setting studied by Kupavskii and Welzl [37]; in their model, there is no robustness requirement, and one aims to minimize the acceleration ratio of the system in the presence of at most f faults. In our setting, instead, we treat f as a "soft" bound on the number of faults that may occur and would still like the schedule to perform well if this bound is exceeded. Note that if $r = \infty$, RFT reduces to the setting of [37].

We first show how to obtain a schedule for $\texttt{RFT}(r, p, f)$ by analyzing an explicit schedule. Namely, for a $b > 1$ to be fixed later, we will use the family of schedules $\mathcal{X}_{b,p}$ (recall Definition 12), with each strategy X_i in this collection scheduled on processor i. From Property 16, the acceleration ratio of this schedule is $b^{p+f+1}/(b^p - 1)$, in the presence of at most f faults. Moreover, since each X_i is near-geometric, we can enforce the requirement that its individual acceleration ratio is at most r by appealing to Property 2. This yields the following result.

Theorem 22. *There is a schedule for* $\texttt{RFT}(r, p, f)$ *of acceleration ratio at most*

$$\min \frac{b^{p+f+1}}{b^p - 1} \quad \textit{subject to} \quad b \in [\zeta_{1,r}^{1/p}, \zeta_{2,r}^{1/p}].$$

Proof. Consider an interruption T that occurs right before the contract of length b^{jp+l} terminates, for some $j \in \mathbb{N}$, and $l \in [0, p-1]$. Then, from the

definition of $\mathcal{X}_{b,p}$, a contract of length $b^{jp+l-f-1}$ has completed, even if f processor faults have occurred. We have

$$\frac{T}{b^{jp+l-f-1}} = \frac{\sum_{i=0}^{j} b^{ip+l}}{b^{jp+l-f-1}} \leq \frac{b^{p+f+1}}{b^p - 1}.$$

which, together with Property 2 completes the proof. □

We will show that this result is tight, and thus, our schedule is optimal. To this end, we need the following generalization of Lemma 4, to the fault-tolerant setting. The proof closely follows an approach from [41], which studied a fault-tolerant version of the online bidding problem. The technical proof is given in Appendix Appendix B for completeness.

Lemma 23. [Appendix] *Let X be a p-processor schedule, as defined by a set $\mathcal{X}$ of p single-processor strategies, assuming that each strategy in $\mathcal{X}$ has finite acceleration ratio and that at most f processors may be faulty. Then* $\mathtt{acc}(X) \geq \frac{\alpha_{\bar{\mathcal{X}}}^{p+f+1}}{\alpha_{\bar{\mathcal{X}}}^{p}-1}$.

We can now show that the upper bound of Theorem 22 is tight.

Theorem 24. *Any schedule for* $\mathtt{RFT}(r, p, f)$ *of acceleration ratio at most*

$$\min \frac{b^{p+f+1}}{b^p - 1} \quad \textit{subject to} \quad b \in [\zeta_{1,r}^{1/p}, \zeta_{2,r}^{1/p}].$$

Proof. Given a p-processor schedule X, and from the robustness requirement of the problem, the schedule of each individual processor must be r-robust. Thus, Corollary 10 applies, and in combination with Lemma 23, we obtain that the acceleration ratio of any schedule is at least $\min \frac{\alpha_{\bar{\mathcal{X}}}^{p+f+1}}{\alpha_{\bar{\mathcal{X}}}^{p}-1}$, subject to the condition $\alpha_{\bar{\mathcal{X}}} \in [\zeta_{1,r}^{1/p}, \zeta_{2,r}^{1/p}]$, which completes the proof. □

We can obtain a closed-form expression of the acceleration ratio of Theorem 22 along the lines of the result of Theorem 18, by replacing 2^k with p, and U with f.

Example 25. *Consider* $\texttt{RFT}(r, p, f)$ *with* $r = 4$, $p = 3$ *and* $f = 1$. *Then Theorem 22 shows that there is a 3-processor schedule which has acceleration ratio* $2^{5/3} \approx 3.175$*, if at most one processor fault occurs, and has acceleration ratio 4 if all but one processor may be faulty.*

5.1. A note on fault-tolerant contract scheduling

We emphasize that our results provide rigorous proofs not only for RFT, but also for the standard fault-tolerant contract scheduling problem introduced in [37] (recall that in the latter, we have $r = \infty$). We note that [37] did not provide explicit acceleration ratios but gave an optimal strategy for the problem of searching for a hidden target in the line in a setting in which there are p searchers, f of which may be faulty. The work implies that the same approach gives a solution to the problem of fault-tolerant contract scheduling, however we argue bellow that the two problems are fundamentally different in multi-processor/multi-searcher settings, and it is not obvious how to reduce one to the other.

Consider first the case $f = 0$, i.e., no searchers or processors are faulty. Then, with only two searchers, the competitive ratio of searching on the line is 1: one can dedicate a searcher to each direction of the line. In contrast, [6] shows that in any p-processor contract schedule, the acceleration ratio is always strictly larger than one. It is also instructive to see that the proof of [6] is very much different than, say, [42], which studied multi-robot searching.

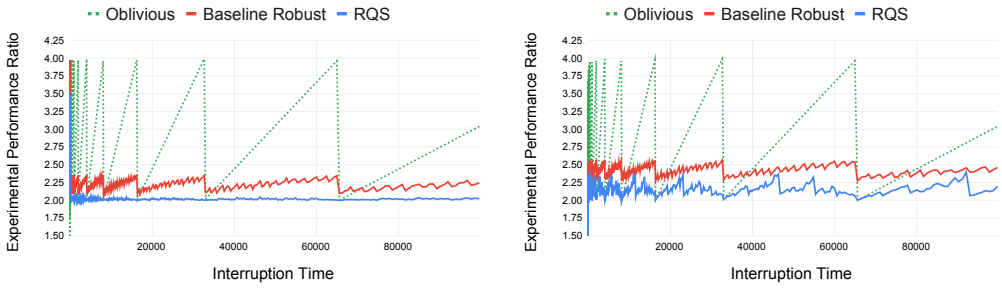
This observation extends to the fault-tolerant setting: while it is possible to explore each of the two branches of the line in an optimal manner, by dedicating $f + 1$ searchers to search that branch exclusively, this action has no counterpart in the domain of contract scheduling, since infinite length executions of contract algorithms lead to schedules of unbounded acceleration ratio. Nevertheless, our results in Section 5 apply not only to robust, fault-tolerant contract scheduling but also provide explicit formulas for the vanilla fault-tolerant model of [37].

6. Experimental evaluation

We present an experimental analysis of the query-based schedule we introduced in Section 4; we refer to this schedule as ROBUST QUERY-BASED SCHEDULE, or RQS for brevity (and recall its analysis in Theorem 18). This is the most general setting studied in this work and the only one that incorporates noisy data and randomness.

We evaluate RQS for interruptions $T \in (1, 10^5]$. More precisely, we consider all T of the form $T = \lceil 1.01^i \rceil$ for $i \in [1, 1157]$. For each such potential interruption, there is a unique, error-free bit string of size k, which identifies the best schedule among the 2^k candidates in $\mathcal{X}_{b,2^k}$ (b chosen according to (16)), and whose bits are responses to binary partition queries, as explained in Section 4. Given this string and a tolerance parameter $\tau \leq k/4$, we generate the noisy response to the k queries by choosing η uniformly at random in $[0, \tau]$, then flipping η bits, again chosen uniformly at random[3].

[3]Since RQS is determined by a noisy search over a space of 2^k schedules, the experimental evaluation needs to assume a reasonably small value of k.



(a) $\tau = 2$ (b) $\tau = 4$

Figure 4: Experimental performance ratios of various schedules for the setting $r = 4$, $k = 16$, and $\tau \in \{2, 4\}$.

For each interruption T and its associated noisy query responses, we evaluate the *experimental* performance ratio, namely the ratio between T and the longest contract completed in the schedule at hand by time T. More precisely, for each T, we generate 100 random noisy query responses (each a k-bit string) as described above, and the longest completed contract in the schedule is the average one over these 100 runs. For a given robustness guarantee $r \geq 4$, we compare RQS against the schedule $Robust_\tau$ of [19], to which we refer as *Baseline Robust*. We also compare RQS to two *oblivious* schedules that do not rely to queries, namely the geometric schedules with base $\zeta_{1,r}$ and $\zeta_{2,r}$, respectively. Recall that, from Property 2, these are the two extreme base values for which a geometric schedule without queries is r-robust, and note that for $r = 4$, $\zeta_{1,4} = \zeta_{2,4} = 2$ (i.e., there is a single oblivious schedule).

Figure 4 depicts the experimental performance ratio of the various schedules for $k = 16$, $r = 4$, and $\tau \in \{2, 4\}$. The peaks of the plots are the empirical acceleration ratios of the corresponding schedules. RQS improves

upon Baseline-Robust and Oblivious in 94.76% and 94.94% of interruption times, respectively, for $\tau = 2$, and 95.77% and 94.94%, respectively, for $\tau = 4$. For $\tau = 2$, we observe that RQS and Baseline-Robust have experimental acceleration ratios 2.054 and 2.355, whereas for $\tau = 4$, their acceleration ratios are 2.402 and 2.596, respectively.

The results demonstrate that for small values of error, i.e., for $\tau = 2$, the empirical performance ratio of RQS is very close to 2, which is consistent with Theorems 11 and 20, since the ideal consistency (assuming no error) is equal to $2^{1+\frac{1}{2^k}}$. As the error increases, i.e., for $\tau = 4$, the performance of both schedules becomes more noisy, as expected; however, RQS still performs better than Baseline-Robust, even for $\tau = k/4$. Hence, as long as the query error is not prohibitive, the improvements over Baseline-Robust are considerable.

We refer to Appendix Appendix C for many additional experimental results. In particular, we show that the performance improvement becomes more pronounced as r increases. For example, for $r = 6$ and $r = 10$, RQS performs better than Baseline-Robust in at least 98% of interruptions, and the acceleration ratio improvement is at least 16%. This is due to the fact that RQS has greater leeway to optimize the base b, given that the worst-case effect of the query error is less significant than in Baseline-Robust. We also show experiments with different numbers of queries, ranges of query errors, and query errors either below or beyond the tolerance parameter. These results show that, as expected, RQS performs much better than Baseline-Robust if η is much smaller than τ, and it is comparable to Baseline-Robust, even if η exceeds τ.

7. Discussion

We studied a classic problem from sequential decision-making in the query-based prediction model. Our approach exploits connections between the quality of responses, parallelism, error-tolerant search, and fault-tolerance at the processor level. We gave Pareto-optimal schedules, and schedules based on adaptive queries in which the performance degradation due to error is negligible, as the number of queries grows. We also showed that our techniques are applicable to multi-criteria scheduling beyond the query setting, such as the robust, fault-tolerant contract scheduling problem.

The techniques of Section 3 can be useful in *best-of-both-worlds* analysis and multi-criteria optimization for other problems such as searching for a hidden target under the competitive ratio in an unbounded line environment, a classic search problem in TCS, AI and OR that has been studied in a variety of settings. More precisely, we know from [3, 10] that any contract schedule of acceleration ratio r can be interpreted as a search strategy of competitive ratio $1 + 2r$, and vice versa. Theorem 5 then solves the following problem: it characterizes the $(1 + 2r)$-competitive strategies whose search lengths are as large as possible (the upper bound in the statement of the theorem), or as small as possible (the lower bound, respectively). This answers the question of finding the most "aggressive" and the most "conservative" strategies for any desired competitive ratio. Such characterizations help obtain strategies that simultaneously optimize multiple performance measures; see., e.g., [43], which answered this question only for the special case $r = 9$.

The broader future objective is to obtain learning-augmented algorithms in settings in which predictions are elicited via queries. For instance, [11]

showed connections between sequencing problems (such as contract scheduling) and problems such as hierarchical clustering and minimum latency, which opens the possibility of applying our approaches to other important problems.

References

[1] S. Zilberstein, Using anytime algorithms in intelligent systems, AI Magazine 17 (1996) 73–83.

[2] S. J. Russell, S. Zilberstein, Composing real-time systems, in: Proc. International Joint Conference on Artificial Intelligence (IJCAI), 1991, pp. 212–217.

[3] D. S. Bernstein, L. Finkelstein, S. Zilberstein, Contract algorithms and robots on rays: Unifying two scheduling problems, in: Proc. International Joint Conference on Artificial Intelligence (IJCAI), 2003, pp. 1211–1217.

[4] D. S. Bernstein, T. J. Perkins, S. Zilberstein, L. Finkelstein, Scheduling contract algorithms on multiple processors, in: Proc. AAAI Conference on Artificial Intelligence, 2002, pp. 702–706.

[5] S. Zilberstein, F. Charpillet, P. Chassaing, Optimal sequencing of contract algorithms., Ann. Math. Artif. Intell. 39 (2003) 1–18.

[6] A. López-Ortiz, S. Angelopoulos, A. Hamel, Optimal scheduling of contract algorithms for anytime problem-solving, J. Artif. Intell. Res. (2014) 533–554.

[7] S. Angelopoulos, A. López-Ortiz, A. M. Hamel, Optimal scheduling of contract algorithms with soft deadlines, J. Sched. 20 (2017) 267–277.

[8] S. Angelopoulos, A. López-Ortiz, Interruptible algorithms for multi-problem solving, J. Sched. 23 (2020) 451–464.

[9] S. Angelopoulos, S. Jin, Earliest-completion scheduling of contract algorithms with end guarantees, in: Proc. International Joint Conference on Artificial Intelligence, (IJCAI), 2019, pp. 5493–5499.

[10] S. Angelopoulos, Further connections between contract-scheduling and ray-searching problems, in: Proc. International Joint Conference on Artificial Intelligence (IJCAI), 2015, pp. 1516–1522.

[11] M. Chrobak, C. Kenyon-Mathieu, SIGACT news online algorithms column 10: Competitiveness via doubling, SIGACT News 37 (2006) 115–126.

[12] M. Chrobak, C. Kenyon, J. Noga, N. E. Young, Incremental medians via online bidding, Algorithmica 50 (2008) 455–478.

[13] L. Epstein, A. Levin, Randomized algorithms for online bounded bidding, Information processing letters 110 (2010) 503–506.

[14] S. Angelopoulos, C. Dürr, S. Jin, S. Kamali, M. P. Renault, Online computation with untrusted advice, in: Proc. International Conference on Innovations in Theoretical Computer Science (ITCS), 2020, pp. 52:1–52:15.

[15] S. Im, B. Moseley, C. Xu, R. Zhang, Online state exploration: Competitive worst case and learning-augmented algorithms, in: Proc. European Conference on Machine Learning and Knowledge Discovery in Databases (ECML/PKDD), 2023, pp. 333–348.

[16] K. Anand, R. Ge, A. Kumar, D. Panigrahi, A regression approach to learning-augmented online algorithms, in: Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, 2021, pp. 30504–30517.

[17] A. Beck, D. Newman, Yet more on the linear search problem, Israel Journal of Mathematics 8 (1970) 419–429.

[18] M. X. Goemans, J. M. Kleinberg, An improved approximation ratio for the minimum latency problem, Mathematical Programming 82 (1998) 111–124.

[19] S. Angelopoulos, S. Kamali, Contract scheduling with predictions, J. Artif. Intell. Res. 77 (2023) 395–426.

[20] T. Lykouris, S. Vassilvitskii, Competitive caching with machine learned advice, J. ACM 68 (2021) 24:1–24:25.

[21] M. Purohit, Z. Svitkina, R. Kumar, Improving online algorithms via ML predictions, in: Proc. Conference on Neural Information Processing Systems (NIPS), 2018, pp. 9661–9670.

[22] M. Mitzenmacher, S. Vassilvitskii, Algorithms with predictions, in: Beyond the Worst-Case Analysis of Algorithms, Cambridge University Press, 2020, pp. 646–662.

[23] A. Lindermayr, N. Megow, Repository of works on algorithms with predictions, https://algorithms-with-predictions.github.io/about, 2023. Accessed: 2024-04-26.

[24] A. Mazumdar, B. Saha, Clustering with noisy queries, in: Proc. Conference on Neural Information Processing Systems (NIPS), volume 30, 2017, pp. 5788–5799.

[25] R. Addanki, S. Galhotra, B. Saha, How to design robust algorithms using noisy comparison oracle, Proc. VLDB Endow. 14 (2021) 1703–1716.

[26] Z. Benomar, V. Perchet, Advice querying under budget constraint for online algorithms, in: Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, 2023, pp. 75026 – 75047.

[27] S. Im, R. Kumar, A. Petety, M. Purohit, Parsimonious learning-augmented caching, in: International Conference on Machine Learning, ICML, volume 162 of *Proceedings of Machine Learning Research*, PMLR, 2022, pp. 9588–9601.

[28] B. Sun, R. Lee, M. Hajiesmaili, A. Wierman, D. Tsang, Pareto-optimal learning-augmented algorithms for online conversion problems, in: Proc. Conference on Neural Information Processing Systems (NeurIPS), 2021, pp. 10339–10350.

[29] A. Wei, F. Zhang, Optimal robustness-consistency trade-offs for

learning-augmented online algorithms, Advances in Neural Information Processing Systems 33 (2020) 8042–8053.

[30] T. Li, R. Yang, G. Qu, G. Shi, C. Yu, A. Wierman, S. H. Low, Robustness and consistency in linear quadratic control with untrusted predictions, Proc. ACM Meas. Anal. Comput. Syst. 6 (2022) 18:1–18:35.

[31] R. Lee, J. Maghakian, M. H. Hajiesmaili, J. Li, R. K. Sitaraman, Z. Liu, Online peak-aware energy scheduling with untrusted advice, in: Proc. International Conference on Future Energy Systems (e-Energy), 2021, pp. 107–123.

[32] N. Christianson, J. Shen, A. Wierman, Optimal robustness-consistency tradeoffs for learning-augmented metrical task systems, in: Proc. International Conference on Artificial Intelligence and Statistics (AISTATS), 2023, pp. 9377–9399.

[33] R. Lee, B. Sun, M. Hajiesmaili, J. C. S. Lui, Online search with predictions: Pareto-optimal algorithm and its applications in energy markets, in: e-Energy, ACM, 2024, pp. 50–71.

[34] S. Angelopoulos, Online search with a hint, Inf. Comput. 295 (2023) 105091.

[35] R. L. Rivest, A. R. Meyer, D. J. Kleitman, K. Winklmann, J. Spencer, Coping with errors in binary search procedures, J. Comput. Syst. Sci. 20 (1980) 396–404.

[36] S. Im, R. Kumar, M. M. Qaem, M. Purohit, Online knapsack with

frequency predictions, in: Proc. Conference on Neural Information Processing Systems (NeurIPS), 2021, pp. 2733–2743.

[37] A. Kupavskii, E. Welzl, Lower bounds for searching robots, some faulty, Distributed Computing 34 (2019) 229–237.

[38] Y. Disser, S. Kratsch, Robust and adaptive search, in: Proc. Symposium on Theoretical Aspects of Computer Science (STACS), volume 66, 2017, pp. 26:1–26:14.

[39] S. Gal, Search Games, Academic Press, 1980.

[40] S. Angelopoulos, S. Kamali, D. Zhang, Online search with best-price and query-based predictions, in: Proc. AAAI Conference on Artificial Intelligence, 2022, pp. 9652–9660.

[41] S. Angelopoulos, S. Kamali, Rényi-Ulam games and online computation with imperfect advice, in: Proc. International Symposium on Mathematical Foundations of Computer Science, (MFCS), 2023, pp. 13:1–13:15.

[42] A. López-Ortiz, S. Schuierer, On-line parallel heuristics, processor scheduling and robot searching under the competitive framework, Theoretical Computer Science 310 (2004) 527–537.

[43] S. Angelopoulos, C. Dürr, S. Jin, Best-of-two-worlds analysis of online search, in: Proceedings of the 36th International Symposium on Theoretical Aspects of Computer Science (STACS), 2019, pp. 7:1–7:17.

[44] S. Gal, A general search game, Israel Journal of Mathematics 12 (1972) 32–45.

[45] S. Alpern, S. Gal, The theory of search games and rendezvous, Kluwer Academic Publishers, 2003.

Appendix

Appendix A. Statement of Gal's theorem

Below, we give the formal statement of Gal's theorem.

Theorem 26 (Gal [44])**.** *Let q be a positive integer, and $X = (x_i)_{i=0}^{\infty}$ a sequence of positive numbers with $\sup_{n\geq 0} x_{n+1}/x_n < \infty$ and $\alpha_X > 0$. Suppose that F_i is a sequence of functionals that satisfy the following properties:*

(1) $F_i(X)$ depends only on $x_0, x_1, \ldots x_{i+q}$,

(2) $F_i(X)$ is continuous in every variable, for all positive sequences X,

(3) $F_i(aX) = F_i(X)$, for all $a > 0$,

(4) $F_i(X+Y) \leq \max(F_i(X), F_i(Y))$, for all positive sequences X, Y, and

(5) $F_{i+j}(X) \geq F_i(X^{+j})$, for all $j \geq 1$, where $X^{+j} = (x_j, x_{j+1}, \ldots)$.

Then

$$\sup_{0\leq q<\infty} F_q(X) \geq \sup_{0\leq q<\infty} F_q(G_{\alpha_X}),$$

where G_a is defined as the geometric sequence $(a^i)_{i=0}^{\infty}$.

Appendix B. Omitted proofs

Proof of Lemma 9. For each $j = 1, \ldots, N$, we define the function $f_j(t) \in \mathbb{N}$, where $t > 0$, as the number of $x_{j,i}$'s such that $x_{j,i} \leq t$. More precisely, for any $t > 0$, the value $f_j(t)$ is the unique nonnegative integer such that

$$x_{j,i} \leq t, \text{ for all } 1 \leq i \leq f_j(t) \quad \text{and} \quad x_{j,i} > t, \text{ for all } i > f_j(t).$$

In particular, if $c_B i B^i \leq t$, then $f_j(t) \geq i$. Therefore, if $c_B i B^i \leq t < c_B(i+1)B^{i+1}$, using that $\log(i+1) \leq \sqrt{i}$ for all $i \geq 0$, we conclude that

$$\begin{aligned} f_j(t) \geq i &> \frac{\log t - \log c_B}{\log B} - 1 - \frac{1}{\log B}\sqrt{i} \\ &\geq \frac{\log t - \log c_B}{\log B} - 1 - \frac{1}{\log B} f_j(t)^{\frac{1}{2}}. \end{aligned}$$

Similarly, noticing that, if $t < c_A A^i$, then $f_j(t) < i$, we conclude that

$$f_j(t) < i \leq \frac{\log t - \log c_A}{\log A} + 1,$$

whenever $c_A A^{i-1} \leq t < c_A A^i$.

Now, consider any large $t > 0$ such that $y_i \leq t < y_{i+1}$, for some large i. It must then hold that

$$i = \sum_{j=1}^{N} f_j(t),$$

whence

$$N\left(\frac{\log t - \log c_B}{\log B} - 1 - \frac{1}{\log B}\sqrt{i}\right) < i < N\left(\frac{\log t - \log c_A}{\log A} + 1\right).$$

In particular, setting $t = y_i$ yields that

$$c_A A^{\frac{i}{N}-1} < y_i < c_B e^{\sqrt{i}} B^{\frac{i}{N}+1},$$

which implies

$$A^{\frac{1}{N}} = \lim_{i\to\infty} c_A^{\frac{1}{i}} A^{\frac{1}{N}-\frac{1}{i}} \leq \liminf_{i\to\infty} y_i^{\frac{1}{i}} \leq \limsup_{i\to\infty} y_i^{\frac{1}{i}}$$
$$\leq \lim_{i\to\infty} c_B^{\frac{1}{i}} e^{\frac{1}{\sqrt{i}}} B^{\frac{1}{N}+\frac{1}{i}} = B^{\frac{1}{N}},$$

thereby completing the proof of the lemma. □

Proof of Lemma 23. Consider a schedule X for RFT. For $j \in [0, \ldots p-1]$, define $l_X(t, j)$ as the length of the largest contract in S that has completed in processor j by time t. We also define by $\ell_{X,f}(t)$ as the $(f+1)$-largest length in the set $\{l_X(t, j)\}_{j=0}^{p-1}$.

Following the notation of [6], we denote each contract c_j in X as a pair of the form (T_j, D_j), where T_j is the start time of c_j, and D_j its length (as we will see, the specific processor to which the contract is assigned will not be significant for our analysis). We also define d_j to be equal to $\ell_{X,f}(T_j + D_j - \epsilon)$, for $\epsilon \to 0$. In words, d_j is the longest contract length that has been completed right before c_j is about to terminate, assuming a worst-case scenario in which f processors have been faulty, and they also happened to be the processors that have completed the longest contracts in the schedule by the said time: we call this contract length the $(f+1)$ *length relative to* D_j.

Recall that $\bar{X}$ denotes the sequence of all contract lengths in X, in non-decreasing order. Hence, each contract in X is mapped via its length to an element of this sequence (breaking ties arbitrarily).

Fix a time, say t, at which a contract $c_{j_0} = (T_{j_0}, D_{j_0})$ terminates, say on processor 0 i.e., $t = T_{j_0} + D_{j_0}$. For all $m \in [1, p-1]$, let $c_{j_m} = (T_{j_m}, D_{j_m})$ denote the longest contract length that has completed on processor m by time t. For every $m \in [0, p-1]$, define I_m as the set of indices in $\mathbb{N}$ such

that $i \in I_m$ if and only if the contract of length x_i has completed by time t in processor m. From the definition of the acceleration ratio, we have that

$$\mathtt{acc}(X) \geq \frac{\sum_{i \in I_m} x_i}{d_{j_m}}, \qquad \text{for all } m \in [0, p-1].$$

Therefore,

$$\mathtt{acc}(X) \geq \max_{0 \leq m \leq p-1} \frac{\sum_{i \in I_m} x_i}{d_{j_m}},$$

and using the property $\max\{a/b, c/d\} \geq \frac{a+b}{c+d}$, for all $a, b, c, d > 0$, we obtain that

$$\mathtt{acc}(X) \geq \frac{\sum_{m=0}^{p-1} \sum_{i \in I_m} x_i}{\sum_{m=0}^{p-1} d_{j_m}}. \tag{B.1}$$

Next, we will bound the numerator of the fraction in (B.1) from below and its denominator from above. We begin with a useful observation: we can assume, without loss of generality, that by time t (defined earlier), every contract of length d_{j_0} or smaller has completed its execution. This follows from the definition of d_{j_0}: if X completed a contract of length at most d_{j_0} later than time t, then one could simply "remove" this contract from X, and obtain a schedule of no worse acceleration ratio (in other words, such a contract is useless, and one can derive a schedule of no larger acceleration ratio than X that does not contain it).

Using the above observation, it follows that the numerator in (B.1) includes, as summands, all contracts of length at most d_{j_0}, as well as at least $f+1$ contracts that are at least as large as d_{j_0}. Let q denote an index such that $d_{j_0} = \bar{x}_q$, then we have that

$$\sum_{m=0}^{p-1} \sum_{i \in I_m} x_i \geq \sum_{i=0}^{q+f+1} \bar{x}_i.$$

We now show how to upper-bound the denominator using the monotonicity implied in the definition of the $(f+1)$ length relative to a given contract length. Since c_{j_0} is completed no earlier than any other contract c_{j_m}, with $m \in [1, p-1]$, we have that $d_{j_m} \leq d_{j_0}$. It thus follows that

$$\sum_{m=0}^{p-1} d_{j_m} \leq \sum_{i=q}^{q-(p-1)} \bar{x}_i.$$

Combining the two bounds, it follows that

$$\mathtt{acc}(S) \geq \sup_{0 \leq q < \infty} \frac{\sum_{i=0}^{q+f+1} \bar{x}_i}{\sum_{i=q}^{q-(p-1)} \bar{x}_i}.$$

Define now the functional $F_q(\bar{X}) = \frac{\sum_{i=0}^{q+f+1} \bar{x}_i}{\sum_{i=q}^{q-(p-1)} \bar{x}_i}$, for every q. The functional satisfies the conditions (1)-(5) of Theorem 26 (see Example 7.3 in [45]). Moreover, $\sup_{n \geq 0} \bar{x}_{n+1}/\bar{x}_n < \infty$, otherwise an infinite contract would be scheduled in some processor, which, in turn, would render the corresponding processor "useless", since this contract would never complete. Last, we note that the condition $\alpha_X > 0$ is indeed satisfied, from Theorem 5 and Corollary 10.

By applying Gal's Theorem (Theorem 26) it follows that

$$\mathtt{acc}(X) \geq \sup_{0 \leq q < \infty} \frac{\sum_{i=0}^{q+f+1} \alpha_{\bar{X}}^i}{\sum_{i=q}^{q-(p-1)} \alpha_{\bar{X}}^i}.$$

If $\alpha_{\bar{X}} \leq 1$, then it is easy to show that the above expression shows that $\mathtt{acc}(X) = \infty$; see, e.g. [6]. Otherwise, i.e., if $\alpha_{\bar{X}} > 1$, after some simple calculations along the lines of [6], we arrive at the desired result. □

Appendix C. Additional experimental results

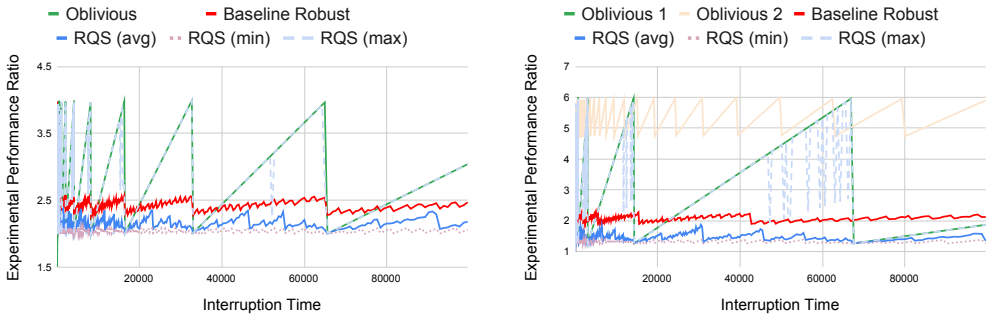
In this section, we provide additional experimental results concerning the RQS schedule.

Appendix C.1. Experiments on the robustness r

In the main paper, we reported experiments for the setting in which $r = 4.0$. Here, we compare RQS against the Oblivious and Baseline-Robust schedules for values of r such that $r > 4$. As before, the number of queries is $k = 16$, and we assume $\tau = 4 = k/4$ (i.e., the largest possible value of error the schedule can tolerate). The query error is generated as discussed in the main paper.

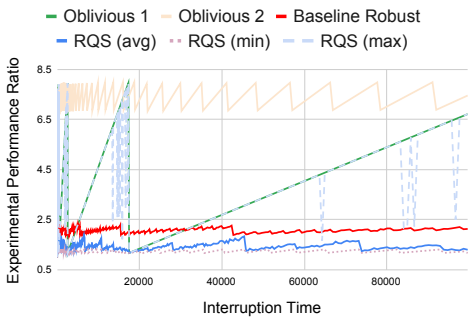
Figure C.5 depicts the experimental performance ratio for $r \in \{4, 6, 8, 10\}$. The results show that RQS is consistently better than Oblivious schedules and Baseline-Robust, in at least 95% of the possible interruption times. We also measured the average percentage improvement of RQS relative to Baseline-Robust: this is defined as the average, over all possible interruptions, of the signed % improvement in each interruption. These improvements are $11.88, 37.54, 42.96$ and 45.48, for $r \in \{4, 6, 8, 10\}$, respectively. The experimental acceleration ratios of RQS are 2.39913, 1.88021, 1.83836, and 1.75828, respectively. We observe that the performance gains of RQS relative to Baseline-Robust increase as a function of r, as explained in the main paper.

In Figure C.5, we also plotted the minimum and maximum performance ratios of RQS, among the 100 random query responses, for each interruption time. As expected, the maximum performance ratio is similar to that of the Oblivious 1 schedule with a large base $\zeta_{1,r}$, since $\tau = k/4$; however, this happens only very rarely, namely when the robust search algorithm finishes in a node that is very high up in the search tree. On average, we observe that RQS performs very close to the best run.

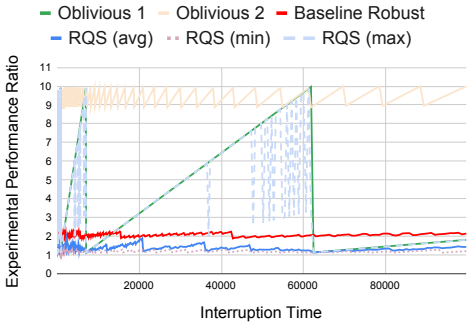


(a) $r = 4.00$

(b) $r = 6.00$



(c) $r = 8.00$



(d) $r = 10.00$

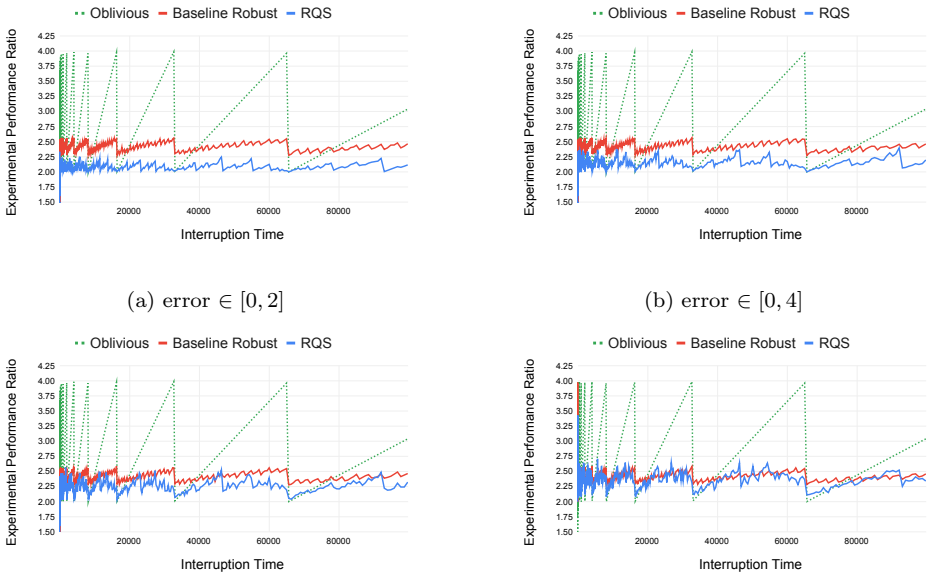
Figure C.5: Experimental performance ratios of various schedules for $r \in \{4.00, 6.00, 8.00, 10.00\}$ when $k = 16$ and $\tau = 4$.

Appendix C.2. Experiments on the query error η

In the main paper, we considered the setting where the error takes values from $[0, \tau]$. Here, we study a setting in which the maximum query error can be either smaller or larger than the tolerance τ. As before, we consider $r = 4.00$, $k = 16$, and $\tau = 4$. Figure C.6 depicts the performance of schedules when η query responses are randomly flipped, and η is uniformly distributed in $[0, \tau/2]$, $[0, \tau]$, $[0, 3\tau/2]$ and $[0, 2\tau]$. The last two ranges correspond to very noisy query responses, for which the theoretical performance guarantees of

Theorem 18 do not necessarily hold.

We observe that, as expected, for relatively small values of τ, and thus small values of query error, RQS performs much better than Baseline-Robust. This is because the robust search algorithm performs very well, if the error is relatively small, and thus the schedule of RQS is very close to the Pareto-optimal schedule of Theorem 18. If the error is too large, namely even if η exceeds τ, RQS is still experimentally comparable to Baseline-Robust. Recall also our observation at the end of Section 4 concerning the worst-case comparison of the two schedules.



(a) error $\in [0, 2]$

(b) error $\in [0, 4]$

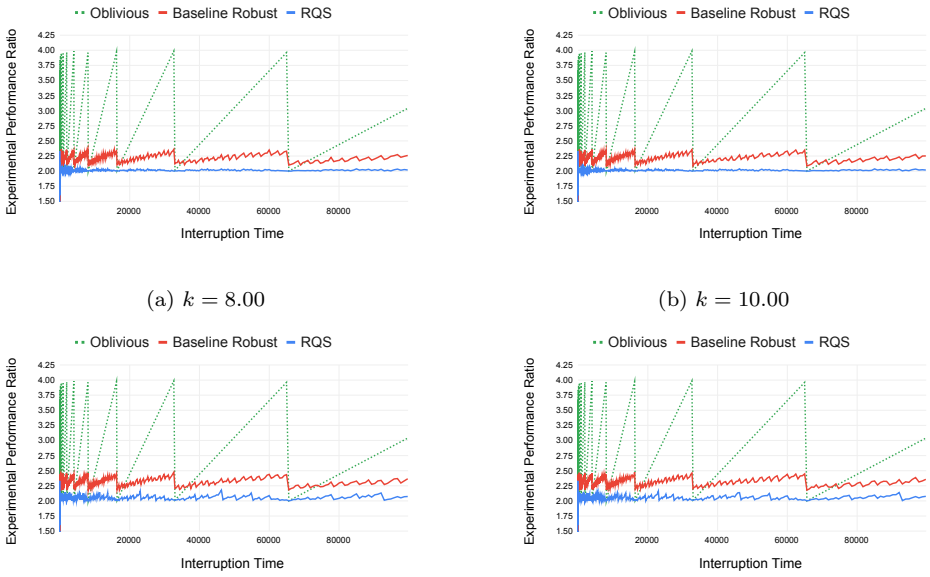
(c) error $\in [0, 6]$

(d) error $\in [0, 8]$

Figure C.6: Experimental performance ratio of various schedules for different ranges of error when $r = 4.00, k = 16$, and $\tau = 4$.

Appendix C.3. Experiments on the number of queries k

We report experiments for different values of the number of queries k, for the setting in which $r = 4.00$, $\tau = \lfloor k/4 \rfloor$ (note that we consider the highest possible tolerance to errors). Figure C.7 summarizes our findings. We observe that RQS is consistently better than Baseline-Robust, and independently on the number of queries, in at least 95% of the possible interruption times. The average percentage improvements of RQS relative to Baseline-Robust are $8.75, 8.75, 11.50$ and 11.51, for $k \in \{8, 10, 12, 14\}$, respectively. The experimental acceleration ratios of RQS are 2.045, 2.043, 2.167, and 2.164 for $k \in \{8, 10, 12, 14\}$, respectively. We observe that RQS has a very stable performance, which is very close to the theoretically ideal acceleration ratio of 2, even for small constant k, in accordance with Theorem 20, and the discussion following the theorem in Section 4. For $k = 12$, both RQS and Baseline-Robust are somewhat more noisy: this is because $k = 12$ is the value that maximizes the ratio k/τ in this set of experiments.



(a) $k = 8.00$

(b) $k = 10.00$

(c) $k = 12.00$

(d) $k = 14.00$

Figure C.7: Experimental performance ratio of various schedules for $k \in \{8, 10, 12, 14\}$ when $r = 4.00$ and $\tau = \lfloor k/4 \rfloor$.