Journal of Artificial Intelligence Research 51 (2014) 533-554 Submitted 02/14; published 10/14

Optimal Scheduling of Contract Algorithms for Anytime Problem-Solving

Alejandro López-Ortiz ALOPEZ-O@UWATERLOO.CA
Cheriton School of Computer Science
University of Waterloo
Waterloo, Ontario, Canada, N2L 3G1

Spyros Angelopoulos SPYROS.ANGELOPOULOS@UPMC.FR
CNRS and Laboratoire d'Informatique
Université Pierre et Marie Curie
4 Place Jussieu 75252, France

Angèle M. Hamel AHAMEL@WLU.CA
Department of Physics and Computer Science
Wilfrid Laurier University
Waterloo, Ontario, Canada, N2L 3C5

Abstract

A *contract algorithm* is an algorithm which is given, as part of the input, a specified amount of allowable computation time. The algorithm must then complete its execution within the allotted time. An *interruptible* algorithm, in contrast, can be interrupted at an arbitrary point in time, at which point it must report its currently best solution. It is known that contract algorithms can simulate interruptible algorithms using iterative deepening techniques. This simulation is done at a penalty in the performance of the solution, as measured by the so-called acceleration ratio.

In this paper we give matching (i.e., optimal) upper and lower bounds for the acceleration ratio under such a simulation. We assume the most general setting in which n problem instances must be solved by means of scheduling executions of contract algorithms in m identical parallel processors. This resolves an open conjecture of Bernstein, Finkelstein, and Zilberstein who gave an optimal schedule under the restricted setting of round robin and length-increasing schedules, but whose optimality in the general unrestricted case remained open.

Lastly, we show how to evaluate the average acceleration ratio of the class of exponential strategies in the setting of n problem instances and m parallel processors. This is a broad class of schedules that tend to be either optimal or near-optimal, for several variants of the basic problem.

1. Introduction

Anytime algorithms are algorithms whose quality of output improves gradually as the amount of available computation time increases. This class of algorithms was introduced first by Dean and Boddy (1988) in the context of time-depending planning, as well as by Horvitz (1987, 1998) in the context of flexible computation. Anytime algorithms occur naturally in settings where a computationally intensive problem is addressed under uncertainty with respect to the available computation time. An example of this is a problem in which the answer must be provided when determined by an external input over which we have no control. For instance, consider an automated trading program for the stock market. These programs run time-intensive simulations to price various financial instruments. When a change in the bid price of a given stock occurs the algorithm must produce a decision (buy/sell/hold) at that very instant, using whatever information it had garnished over the course of the simulations, so as to take advantage of the newly posted price. Another example is given by real-time applications. For instance, consider a motion planning algorithm for a robot in which a solution must be produced within a certain, but varying, amount of time: for example if a robot is about to collide a move is needed momentarily even if the algorithm is to produce a suboptimal move, while in others, there is sufficient time to carefully compute the next step. In this situation, the amount of allotted time is given to the algorithm beforehand.

According to Russell and Zilberstein (1991), a further distinction can be made between two different types of anytime algorithms. On the one hand, *interruptible* algorithms are algorithms whose allowable running time is not known in advance, and thus can be interrupted (queried) at any given point throughout their execution. Such algorithms typically include versions of local search, e.g., simulated annealing and hill climbing. On the other hand, the more stringent class of *contract* algorithms consists of algorithms which 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 the contract time expires, the algorithm may not return any meaningful results. Such algorithms are thus less flexible than interruptible algorithms, however they tend to be simpler to construct, and it is easier to prove strict guarantees on their performance. A typical example of contract algorithms are polynomial-time approximation schemes (PTAS) based on Dynamic Programming (DP): the bigger the amount of computation time, the better an approximation is achieved. However, the algorithm may require all the available computation time in order to fill in the important entries of the DP table, otherwise the solution returned may be useless.

Consider then the following general scenario: we are given a set P of n different *problem instances*, and we want to design an efficient interruptible algorithm that can be applied, in a concurrent fashion, to all n problem instances. The amount of computation time is not known in advance; instead, it is expected that at some unknown point in time, an interruption will occur, at which point the algorithm is stopped and is queried to report its (partial) solutions to any one among the problem instances. Clearly, the algorithm must make judicious use of its resources and ensure that it can produce a reasonably good solution, despite having no knowledge of the exact time at which interruptions may occur. As indicated earlier, it is hardly surprising that this setting arises very frequently in the design of AI systems, in applications such as game-playing programs (Althöfer, 1997; Kao, Reif, & Tate, 1996; Kao, Ma, Sipser, & Yin, 1998), e-trading agents, and medical diagnosis systems. Essentially the problem captures a fundamental trade-off between the quality of the solution returned by the algorithm and the amount of available computation time. We refer the reader to the survey of Zilberstein (1996) for the importance of anytime algorithms in the design of intelligent systems.

PTAS (Polynomial Time Approximation Schemes) provide a rich source of contract algorithms with guaranteed performance. This motivates the study of general constructions for creating interruptible versions for any given contract algorithm. This provides the flexibility of focusing only on the design of contract algorithms, then converting them to interruptible algorithms by applying a standard, "black-box" transformation. Indeed, a standard method for simulating an interruptible algorithm is by repeatedly executing the contract algorithm using increasing execution times. Consider the general setting in which a set M of m processors of identical speed is available for the execution of this simulation, and each problem instance has a corresponding (single-thread) contract algorithm. The problem we face is how to schedule the executions of the various contract algorithms to the processors in a way that guarantees an efficient interruptible algorithm. In this setting, at query time, the algorithm will report for each problem instance the solution of the corresponding contract of *longest length* (i.e., contract time) which has been completed by query time.

It becomes obvious that a formal measure of the quality of this simulation is required. While one may think of several possible ways of evaluating the quality of the simulation, in our work we adopt the standard measure in this area, namely the *acceleration ratio* (Russell & Zilberstein, 1991). A formal definition is provided in Section 2; informally, the acceleration ratio indicates how much faster the processors in M should be in order to guarantee a solution as good as the one of an ideal algorithm that has foreknowledge not only of the query time t but also of the problem instance p of interest. Such an ideal algorithm would simply utilize a single processor to run solely a contract for instance p, up to time t. In a sense, the acceleration ratio reflects the loss in optimality due to lack of future knowledge about the query times and the problem instance in question, and is motivated by similar considerations to the competitive ratio in the context of the analysis of online algorithms.

In the case of one problem instance and a single processor, Russell and Zilberstein (1991) showed that iterative doubling of contract lengths gives rise to an interruptible algorithm of acceleration ratio at most four. Zilberstein, Charpillet, and Chassaing (2003) showed that this is the optimal acceleration ratio, in the sense that any scheduling strategy defined over any set of contracts has acceleration ratio at least four.

Zilberstein et al. (2003) also studied the generalization of the problem in which multiple problem instances are considered (assuming $|M| = 1$), and Bernstein, Perkins, Zilberstein, and Finkelstein (2002) studied the generalization in which contracts for a single problem instance must be scheduled in a set of multiple processors. For both versions, algorithms of optimal acceleration ratios were derived. The problem, in its full generality, involves a set of processors and a set of problems, both of cardinality greater than one. Bernstein et al. (2003) showed an upper bound of $\frac{n}{m}(\frac{m+n}{n})^{\frac{m+n}{m}}$ on the acceleration ratio; in addition, using elegant techniques, they showed that this bound is optimal for a restricted, though natural and intuitive, class of schedules that use a *round robin* and *length-increasing* strategy. Such strategies are known as *cyclic* strategies. Bernstein et al. leave open the question of whether this bound is tight among *all* possible schedules. In this paper we answer this question in the affirmative.

In general, it has been observed that the theoretical analysis of geometric searches and robot motion planning is closely linked to the scheduling of heuristics and algorithms for problem solving. This connection was first established by Kao et al. (1996, 1998) in the context of randomized algorithms. The work of Bernstein et al. (2003) drew a similar connection between scheduling contract algorithms and robot searching on a set of rays (Alpern & Gal, 2003). In the latter problem, p robots search for a target that is located in one of m concurrent rays. We seek a search strategy that minimizes the *competitive ratio*, namely the worst-case ratio of the search cost over the distance from the starting position to the target.

It turns out that interesting parallels can be drawn for the two problems: informally, the rays correspond to problem instances, the robots to processors, and the (unknown) location of the target corresponds to the (also unknown) query time. For the general problem of p robots and m rays López-Ortiz and Schuierer (2004) showed an optimal strategy that achieves a competitive ratio of $1 + 2\frac{m-p}{p}(\frac{m}{m-p})^{\frac{m}{p}}$. Bernstein et al. (2003) independently derived the same bound by directly translating their approach in the context of contract scheduling to a solution for robot searching in parallel rays. It should be noted that the upper bounds in the work of López-Ortiz and Schuierer and Bernstein et al. are based on *exponential* strategies. Informally, these are cyclic strategies where the lengths to which the rays are searched (respectively, the lengths of contracts in the schedule) form a geometric sequence (see Section 2 for the precise definition).

At an intuitive level, the problem of scheduling contract algorithms on multiple processors is more involved than the problem of multi-robot searching in concurrent rays. This difficulty stems from the fact that multiple searchers on the same ray are of no benefit, whereas multiple contract algorithms (of different lengths) on the same problem could very well improve the performance. However, in this paper we show that the ideas behind the solution of López-Ortiz and Schuierer (2004) can be adapted, in a non-trivial manner, so as to show the optimality of the schedule of Bernstein et al. (2003) without any restrictions (such as cyclicality) on the scheduling strategy.

The problem of removing the assumption of cyclicality has a long history in search-related problems. For instance, Jaillet and Stafford (2001) argue rigorously that cyclicality can be waived for single-searcher problems on m rays (whereas in the original work Baeza-Yates, Culberson, & Rawlins, 1993, cyclicality is implicit). Along the same lines, López-Ortiz and Schuierer (2004) remove cyclicality assumptions in the context of multi-searcher problems. Similar conclusions are harder to establish concerning randomized algorithms (for instance, see Schuierer, 2003). In a sense, cyclic strategies are simple to analyze and provide relatively easy upper bounds; however, the difficulty lies in establishing optimality within the space of all possible strategies.

The remainder of this paper is organized as follows. In Section 2 we present a formal discussion of the problem setting. Our main result, namely Theorem 3, is presented in Section 3 where we show the optimality of the exponential schedule of Bernstein et al. (2003) without any restrictions. In Section 4 we present an average-case analysis of the acceleration ratio of exponential strategies in the multi-processor setting, assuming a uniform distribution of the interruption times.

2. Preliminaries

Let P denote the set of n *problem instances* or simply *problems*. A *contract* c is a pair (p, d), where $p \in P$ denotes the problem instance to which c is assigned (also called the *problem tag* of c) and d the *duration* or *length* of the contract, which specifies the processing time required to complete c. We denote by M the set of m identical processors. Let C denote a (potentially infinite) set of contracts. Define a *schedule* X for a set of contracts C as a feasible assignment of all contracts in C to the set M of m processors. In particular, X can be described as the set $\{(c_i, m_i, s_i) : c_i \in C\}$, where $m_i \in \{0, \ldots, m-1\}$ denotes the processor to which c_i is scheduled, and s_i denotes the time its processing begins. The schedule X must be feasible, in the sense that for any two contracts $c_i = (p_i, d_i), c_j = (p_j, d_j)$ in C that are assigned to the same processor by X, and such that c_j is scheduled immediately after c_i, we have that $s_i + d_i \leq s_j$. In other words, c_j does not start before c_i has been completed. Note that we assume non-preemptive schedules, in the sense that we cannot interrupt and later resume a contract.

Observation 1 *Without loss of generality we consider only schedules in which the processors are never idle, i.e., the start time of a contract is always the finish time of another contract.*

For n problem instances and m identical processors, Bernstein et al. (2003) define the class of *cyclic* schedules as schedules which have the following natural properties:

1. Property 1 (Problem Round Robin) If $c_i = (p_i, d_i)$ is the ith contract in the cyclic schedule order, the problem instance p_i is such that $p_i = i \bmod n$.

2. Property 2 (Length Increasing) For all $c_i = (p_i, d_i)$ and $c_j = (p_j, d_j)$ if $p_i = p_j$ and $i < j$, then $d_i < d_j$.

3. Property 3 (Processor Round Robin) $m_i = i \bmod m$ for all i.

An *exponential* schedule is a cyclic schedule in which the lengths of contracts in the round-robin order increase exponentially. More formally, the i-th contract in the order has length b^i for some fixed number b.

We use the *acceleration ratio* as the standard measure of evaluating the quality of a schedule. Following Bernstein et al. (2003), we assume that when a contract is completed at time t its solution is available when the interruption occurs at any time after, or including time t. We also limit the interruptions to occur only after at least one contract for each problem in P has completed, otherwise the problem is vacuous (this is again a canonical assumption). Denote by $l_X(p, t)$ the length of the longest contract for problem p that has been completed by or at time t in X (if there is no source of confusion we omit the subscript.)

Definition 1 *Given a set P of n problem instances and a set M of m processors of identical speed, the acceleration ratio of a schedule X for P, denoted by $R_{m,n}(X)$ is defined as the smallest value r, with $r \geq 1$ such that for any allowable interruption time t, and any problem $p \in P$, we have that $l_X(p, t) \geq t/r$. Then the acceleration ratio for P and a set M of processors of identical speed is defined as*

$$R^*_{m,n} = \inf_X R_{m,n}(X).$$

A schedule X is optimal *if $R_{m,n}(X) = R^*_{m,n}$.*

We argue that for a given schedule X, the acceleration ratio $R_{m,n}(X)$ can be determined by looking at a discrete subset of the timeline, instead of at all possible interruption times t. Let ϵ denote an infinitesimally small positive value, and let F denote the set of finish times of all contracts in X. Then it is easy to see that it suffices to consider interruptions that occur only at times $t-\epsilon$, for all $t \in F$. To see this, consider a certain interruption t that does not conform with the above rule, and let t' be the earliest time in which a contract finishes in X such that $t'-\epsilon > t$. Then for all problems p, $l(p,t'-\epsilon) = l(p,t)$; in other words the algorithm has not made any progress on any problem on the time interval $[t, t'-\epsilon]$, thus $t/l(p,t) < (t'-\epsilon)/l(p,t'-\epsilon)$.

The following is essentially an alternative definition of the acceleration ratio, based on the above observation.

Observation 2 (Bernstein et al., 2003) *Let F denote the set of all finish times of contracts in the schedule X. The acceleration ratio of a schedule X for P is*

$$R_{m,n}(X) = \sup_{p,t} \left\{ \frac{t}{l(p,t)} \right\} = \sup_{p,t\in F,\epsilon\to 0} \left\{ \frac{t-\epsilon}{l(p,t-\epsilon)} \right\}.$$

For a given interruption time t, let $\alpha_p(t)$ denote the function $t/l(p,t)$. In other words, the acceleration ratio of X is simply the maximum value of $\alpha_p(t)$, over all possible interruption times t and all problem instances p. Figure 1 illustrates an example of a schedule for 2 problem instances and 4 processors. Note how the value of α_p peaks just before each contract is completed, for each problem instance.

Given a schedule X with associated set of contracts C, and two problem instances $p_1, p_2 \in P$, let C_1, C_2 denote two (potentially infinite) subsets of C, such that all contracts in C_1 have problem tag p_1, and all contracts in C_2 have problem tag p_2. Consider a new set of contracts C' which is identical to C, with the exception that every contract in C_1 acquires problem tag p_2 instead of p_1 and every contract in C_2 acquires problem tag p_1 instead of p_2. Consider also the schedule X' which is otherwise identical to X (except for the problem tag swaps described above). We say that X' is obtained from X by a *swap* for sets C_1 and C_2.

Following the convention of López-Ortiz and Schuierer (2004), given two schedules X and X', we say that X is *contained* in X' up to time T, denoted by $X \subseteq_T X'$ if the two schedules are identical up to time T. Given a sequence of schedules $\mathcal{V} = (X_1, X_2, \ldots)$ we say that $\mathcal{V}$ *converges* to a limit schedule X if for all $T > 0$ there exists N such that for all $m \geq N$, $X_m \subseteq_T X_{m+1}$. The limit schedule X is defined in the obvious way.

3. A Matching Lower Bound on the Acceleration Ratio

In this section we prove a lower bound on the acceleration ratio which applies to all schedules. Before we proceed with the proof of the main result, we present the intuition behind our approach which is illustrated in Figure 2. Given an arbitrary schedule, we implement transformations that successively transform it into schedules complying with certain regularization conditions (see Lemma 1, Lemma 2, and Theorem 1 below). These transformations either preserve or reduce the acceleration ratio, and their purpose is to infuse a certain amount of structure into the schedule. It is important to observe that the transformation

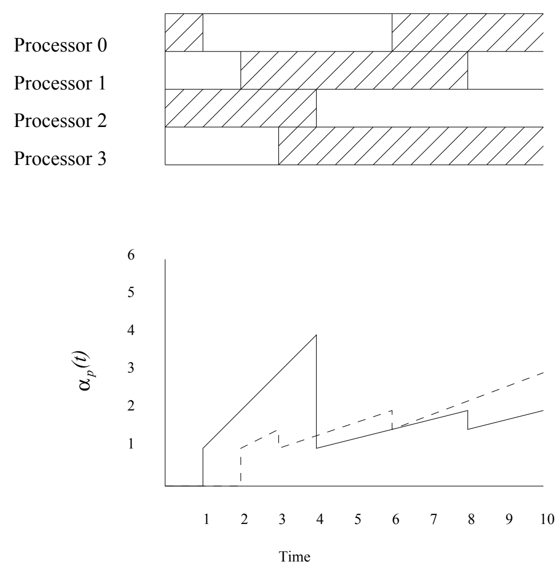

Figure 1: The top figure depicts a schedule of contracts for the case of 4 processors and 2 problems, for the first ten time units. Here, the white and hatched rectangles correspond to executions of contracts for the two problems, respectively, with no idle time in the schedule. The bottom figure depicts the plots of the function $\alpha_p(t)$ vs time for the two problems ($p \in \{1, 2\}$). The hatched (white) area on top corresponds to the solid (hashed) line plot on the bottom (respectively). The acceleration ratio is the maximum value, on the y axis, attained by either curve, and in this example it is equal to 4.

of Theorem 1 might actually result in an object which is a non-feasible schedule, but with a well defined acceleration ratio which is, once again, shown to be no greater than that of the original schedule. We then lower-bound the acceleration ratio of all the obtained (i.e., normalized) schedules, and show that it matches the upper bound of Bernstein et al. (2003).

A similar approach in showing optimality of cyclic strategies was applied by López-Ortiz and Schuierer (2004), in the context of parallel robot searching in concurrent rays. We need to emphasize, however, that the series of transformations for the problem we consider in this paper differ significantly from the ones by López-Ortiz and Schuierer. This is due to the fact that for the ray-searching problem it is easier to argue about structural properties of the optimal algorithm. Consider the following example. Suppose that by time t, a given ray r has been explored up to distance d. Then an optimal algorithm (and indeed every "reasonable" search algorithm) must be such that at any time $t' > t$, if a robot explores ray r, then it will proceed *at least* up to distance d from the origin (otherwise, the algorithm gains nothing from this exploration). In contrast, in the scheduling problem we consider in this paper, it may be the case that in a certain processor M_1, a contract of length l is scheduled with start time t and finish time $t + l$, whereas in a different processor M_2, a different contract for the same problem is scheduled with start time bigger than t and finish time smaller than $t + l$ (i.e., of length smaller than l). At first sight, the latter contract appears to be redundant; however if l is too large, and an interruption occurs right before $t + l$, then the schedule may gain from the smaller-length contract. In fact, such schedules are indeed possible in an optimal algorithm, as they are not ruled out by our transformation techniques. In particular, the above example illustrates that it is difficult to give a "black-box" transformation of the proof by López-Ortiz and Schuierer to our problem of interest (although it would be very interesting to obtain such an explicit reduction).

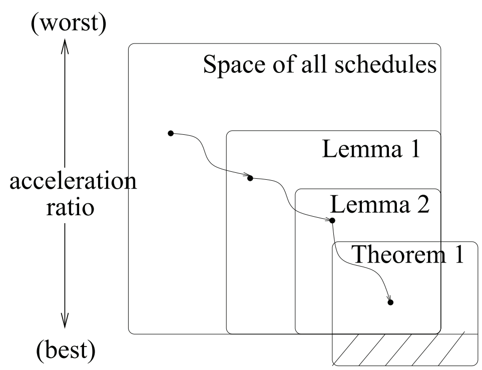

Figure 2: Illustration of the proof technique. The shaded region corresponds to artificial, and possibly infeasible, strategies which are by-products of the proof and could in principle have acceleration ratio strictly smaller than the optimal acceleration ratio; however, we show that this never happens, thus the shaded region collapses to the line of all strategies and non-strategies with acceleration ratio exactly equal to the optimal value.

Note that we can assume, without loss of generality, that the schedule does not start a contract that is smaller than or equal to one that has already been completed on a given problem.

Let X be a given schedule of contracts. We will follow the convention of denoting by lower-case d and upper-case D the lengths of a pair of consecutively completed contracts, for a given problem p, respectively. More precisely, if (p_i, d_i) denotes a contract of length d_i for problem p_i, then the earliest contract in X for p_i which is completed after contract (p_i, d_i) finishes will be denoted by (p_i, D_i).

Definition 2 *For a schedule X of contracts, given a contract c of length D_c we define the* acceleration ratio *at D_c (that is, immediately before the completion of c in X) as $r(c) = (T_c + D_c)/d_c$ (assuming $d_c \neq 0$), where T_c denotes the time when a processor is to start working on the contract (p_c, D_c).*

In particular, let C' denote the set of all contracts in the schedule, excluding the first completed contract for each problem. Then from Observation 2,

$$R_{m,n}(X) = \sup_{c \in C', \epsilon \to 0} \frac{T_c + D_c - \epsilon}{d_c}$$

which converges to $\sup_{c \in C'} r(c)$.

The following lemma establishes a quasi-cyclic property of an optimal strategy schedule. It states that a problem with a shorter-length completed contract should generally be given precedence over another problem with a longer completed contract; there is one possible exception: if the smaller-length contract is passed over during a processor assignment then the next contract in its schedule must be shorter than the one assigned to any problem that went ahead instead.

Lemma 2 establishes another facet of the quasi-cyclicality property: a problem with a shorter-length contract, whether favored or not in its next processor assignment, must complete its next contract before any other problem with a longer completed contract, thus re-establishing a quasi-cyclic property.

Lemma 1 *Consider a schedule X, and suppose that at time T_i a processor is to start working on contract (p_i, D_i) for which there is another problem p_j with contracts (p_j, d_j) and (p_j, D_j) in X. Let T_j denote the time at which a processor is to start working on contract (p_j, D_j).*

Given any two problems p_i and p_j as described above for which $d_j < d_i$ and $T_j > T_i$, then either $D_j < D_i$ or we can define a new schedule such that $D_j < D_i$, and whose acceleration ratio is no worse than that of the original schedule.

Proof. Suppose X is such that $d_j < d_i$ and $T_j > T_i$, and suppose that $D_j > D_i$. Execute a swap of program tags for all contracts on p_i that complete after (p_i, d_i) and all contracts on p_j that complete after (p_j, d_j), so as to obtain a new schedule X' (recall the definition of problem swapping, as given in Section 2).

Observe that all contracts before the swapping remain untouched. Likewise, contracts for problems not involved in the swap are also unaffected. Therefore, the contribution of the acceleration ratio for those is unchanged.

Consider now the contracts for the swapped problems once their very first swapped contract is completed. That is, let c_r for $r > i$ be a contract for problem p_i in the original schedule. In that schedule its contribution to the acceleration ratio is $(T_r + D_r)/d_r$. After the swap this expression still denotes the acceleration ratio but now corresponding to problem p_j, since contracts d_r and D_r are now run on problem p_j, which implies that the schedule remains unaffected by the swapping for those contracts as well.

Thus the only place where the acceleration ratio changes is right at the time of the swapping when the previously longest completed contract (the denominator) comes from the old schedule while the new contract about to be completed comes from the new schedule (the numerator). We will show that the acceleration ratio of the new schedule X' on those contracts is no worse than that of the original schedule.

The acceleration ratio of the original schedule at D_i, D_j is

$$\max\left\{\frac{T_i + D_i}{d_i}, \frac{T_j + D_j}{d_j}\right\}$$

whereas the acceleration ratio of X' at D_i, D_j is

$$\max\left\{\frac{T_i + D_i}{d_j}, \frac{T_j + D_j}{d_i}\right\}.$$

Then since $T_j > T_i$ and $D_j > D_i$, we have that $\frac{T_j+D_j}{d_j} \geq \frac{T_i+D_i}{d_j}$; moreover since $d_j < d_i$, we have $\frac{T_j+D_j}{d_j} \geq \frac{T_j+D_j}{d_i}$. Therefore, the acceleration ratio of X is greater than or equal to the acceleration ratio of the alternative schedule. □

Corollary 1 *Given a schedule X there is a schedule X' of no worse acceleration ratio with the property that for any two problems p_i and p_j, with $d_j < d_i$ in X, and $T_j > T_i$, it is always the case that $D_j < D_i$ in X'.*

Proof. If X already satisfies this condition then set $X' := X$ and there is nothing to show. Otherwise we apply the process, i.e., appropriate problem swapping, as argued in the proof of Lemma 1 in the following way: Let $F = \{f_1, f_2, \ldots, \}$ be the sorted sequence of contract finish times for all problems in X. For a given f_ℓ define $p(f_\ell)$ to be the problem associated with finishing time f_ℓ in X. Let $p_j = p(f_\ell)$ and let d_j be the length of the contract associated with f_ℓ. Now starting with f_1 and letting f_ℓ range over $\ell = 1, 2 \ldots$ we check that for each $d_j < d_i$ and $T_j > T_i$ we have $D_j < D_i$ and if not, we swap the contracts as described in the proof of Lemma 1. □

We introduce some notation that will be needed in the statement and proof of Lemma 2. For a schedule X, let S_T denote the set of all contracts completed by time T inclusive. Also let S^T be the complement of S_T, namely all contracts in $X - S_T$. Fix a contract $C_0 = (p_0, D_0)$, which is scheduled to start at time T_0. For any problem p_j observe that because of Lemma 1 we have $D_j = \min\{D : (p_j, D) \in S^{T_0+D_0}\}$. Observe that in the context of the above definitions, we have $d_j = \max\{d : (p_j, d) \in S_{T_0+D_0}\}$.

Lemma 2 *Let $C_i = (p_i, D_i)$ be a contract scheduled by X at time T_i and $C_j = (p_j, D_j)$ be any contract in S^{T_i}. Then there exists a schedule X' of no worse acceleration ratio such that if $d_i \geq d_j$ for a problem $p_j \neq p_i$ then $T_i + D_i \geq T_j + D_j$.*

Proof. If X itself satisfies the conditions then we set $X' := X$ and the lemma holds. Otherwise if this is not the case, the schedule X is such that $d_i \geq d_j$ for at least one problem $p_j \neq p_i$ and $T_i + D_i < T_j + D_j$. Consider then the swap in which all contracts for problems p_i and p_j in $S^{T_i+D_i}$ swap problem tags as defined in Section 2. We will argue that the swap gives rise to a new schedule X' which has no worse acceleration ratio. One can see that it suffices to look at how the acceleration ratio is affected at the points right before C_i and C_j are completed. First, note that before the swap, the acceleration ratio at these two points is equal to the quantity

$$\alpha = \max\left\{\frac{T_i + D_i}{d_i}, \frac{T_j + D_j}{d_j}\right\}$$

and after the swap, the acceleration ratio at the same two points becomes

$$\beta = \max\left\{\frac{T_i + D_i}{d_j}, \frac{T_j + D_j}{d_i}\right\}.$$

Since $d_j \leq d_i$ and $T_i + D_i \leq T_j + D_j$ we have $\beta \leq \frac{T_j+D_j}{d_j}$ and thus $\beta \leq \alpha$, which means that the acceleration ratio does not worsen at those two specific points. □

As in Lemma 1 we can apply this process repeatedly, starting with d_j which is the length of the contract associated with the smallest finish time f_ℓ (with $\ell = 1, 2 \ldots$) and checking that no larger contract length $d_i \geq d_j$ is such that $T_i + D_i \leq T_j + D_j$. If there are one or more such contracts, we select the D_i with the smallest completion time $T_i + D_i$ and swap tags with (p_j, D_j). We then proceed to the next finishing time $f_{\ell+1}$. This produces a schedule in which for all contracts such that $d_i \geq d_j$ (and $p_j \neq p_i$) we have that $T_i + D_i \geq T_j + D_j$.

Definition 3 *A schedule X is said to be* normalized *if it satisfies the conditions of Corollary 1 and Lemma 2.*

Lemma 3 *There exists an optimal normalized schedule.*

Proof. Since Lemma 2 is stronger than Lemma 1, applying Lemma 2 cannot violate the conditions of Lemma 1 on the pair of contracts being swapped. It is indeed possible, however, that after the swap some other contract is now in conflict with the earliest of the recently modified contracts. To be more precise, consider the configuration of Figure 3(a) where in the original schedule after the next scheduled contracts after the completion of contracts d_i and d_j, were such that $T_j + D_j$ was originally larger than $T_i + D_i$. After the swap we get new completion times $T'_j + D'_j := T_i + D_i$ and $T'_i + D'_i := T_j + D_j$. Observe, however that the new completion time $T'_i + D'_i$ can create a new conflict with another schedule d_k as illustrated in Figure 3. It is not hard to verify that this figure represents the general case and that no other type of conflict can be created.

Figure 3: Situation before and after a single application of Lemma 2.

The key now is to observe that in the original setting the pairs $\langle d_j, d_i \rangle$ and $\langle d_j, d_k \rangle$ formed an *inversion*, i.e. each constituted a violation of Lemma 2. However after the application of the lemma both inversions disappeared while a new inversion $\langle d_i, d_k \rangle$ is created for a net decrease of one in the number of inversions in the schedule. Hence this process must eventually resolve all inversions involving contracts up to an index N for any fixed value of N and this process converges to a well defined strategy. $\square$

Theorem 1 *The acceleration ratio $R_{m,n}(X)$ of an optimal normalized schedule X for n problems with m processors is at least*

$$R_{m,n}(X) \quad \geq \quad \sup_{k \geq 0} \frac{\sum\limits_{i=0}^{k+n} x_i^s}{\sum\limits_{i=k-m+1}^{k} x_i^s} \tag{1}$$

where $X^s = (x_0^s, x_1^s, \ldots)$ is the sorted sequence of contract lengths (in increasing order, ties broken arbitrarily) in the schedule X and we define $x_i^s = 0$ if $i < 0$.

Proof. Let X be an optimal normalized schedule. Consider a time T_0 such that processor M_0 is about to begin a new contract. Since X is a normalized schedule, M_0 will choose a problem p_0 in a way that satisfies the conditions of Corollary 1. Let D_0 be the alloted processing time that M_0 will devote to p_0 starting at time T_0. Let the longest completed contract for problem p_0 at time just before $T_0 + D_0$ be d_0.

Now observe that, because of Lemmas 1 and 2, every contract of length strictly smaller than d_0 must complete within the open interval $(0, T_0 + D_0)$, and hence at the end of this interval every processor is engaged in a contract of length at least d_0 and every problem has completed a contract to length at most d_0 in the previous step.

The lengths of these contracts $d_\ell \geq d_0$ for $0 \leq \ell \leq n-1$ are elements in the sequence X^s. Similarly let M_j denote a processor and denote as I_j the set of indices in X^s of all the contracts executed on processor M_j up to time $T_0 + D_0$, not inclusively, for $0 \leq j \leq m-1$. Note that $d_0 = x^s_{k_0}$, for some $k_0 \geq 0$.

Furthermore, let D_j be the last completed contract on processor M_j, say for a problem p_ℓ, such that the previous completed contract d_j for p_ℓ, is less than d_0. Then the acceleration ratio for the problem p_ℓ at D_j is given by

$$\frac{\sum_{i \in I_j} x^s_i}{x^s_{k_j}}$$

according to Observation 2, for $0 \leq j \leq m-1$. Hence, the worst case acceleration ratio that has occurred up to the time when all the contracts first exceeding d_0 are completed is at least

$$R_{m,n}(X) \geq \max_{0 \leq j \leq m-1} \left\{ \frac{\sum_{i \in I_j} x^s_i}{x^s_{k_j}} \right\} \geq \frac{\sum_{j=0}^{m-1} \sum_{i \in I_j} x^s_i}{\sum_{j=0}^{m-1} x^s_{k_j}}. \tag{2}$$

Here we make use of the fact that $\max\{a/c, b/d\} \geq (a+b)/(c+d)$, for all $a, b, c, d > 0$. Note that the sum $A = \sum_{j=0}^{m-1} \sum_{i \in I_j} x^s_i$ contains as summands all x^s_i that have been completed up to time $T_0 + D_0$. In particular we know that A includes all x^s_ℓ that are smaller than $x^s_{k_0}$, as Lemma 2 guarantees that any problem completed to a contract d_j will complete another contract D_j before a problem completed to a contract $d_0 \geq d_j$ before $T_0 + D_0$ and hence the summation given at time $T_0 + D_0$ contains all x^s_ℓ's (i.e. d_j's) that are smaller than $x^s_{k_0}$ (i.e. d_0). In other words, every element up to $x^s_{k_0}$ in the sorted schedule X^s appears in A.

Now observe that the other $n-1$ problems $p_1, \ldots, p_{n-1}$ must have been completed to durations exceeding $x^s_{k_0}$ as otherwise the current contract of length D_0 would have been assigned to that problem instead of p_0. Then we have that A contains all sorted values in X^s up to $x^s_{k_0}$ plus at least $n-1$ larger values corresponding to the finished contracts in each of the $n-1$ problems. The smallest choices for these $n-1$ values together with the D_0 itself are $x^s_{k_0+1}, \ldots, x^s_{k_0+n}$. Hence, we obtain

$$\sum_{j=0}^{m-1} \sum_{i \in I_j} x^s_i \geq \sum_{i=0}^{k_0+n} x^s_i. \tag{3}$$

Consider now the values $x_{k_j} = d_j$, for $1 \leq j \leq m-1$. Recall that the value D_j is the time to which problem p_j will be completed at time $T_0 + D_0$ by processor M_j and d_j is the longest completed contract for p_j just before time $T_j + D_j$. Then by Lemma 2 d_0 is the largest time among the d_i's. The $m-1$ largest d_i values are $x^s_{k_0-m+1}, \ldots, x^s_{k_0-1}$ and

$$\sum_{j=0}^{m-1} d_j \leq \sum_{i=k_0-m+1}^{k_0} x_i^s. \tag{4}$$

Combining (2),(3) and (4) we have

$$R_{m,n}(X) \geq \frac{\sum_{j=0}^{m-1} \sum_{i \in I_j} x_i^s}{\sum_{j=0}^{m-1} x_{k_j}^s} \geq \frac{\sum\limits_{i=0}^{k_0+n} x_i^s}{\sum\limits_{i=k_0-m+1}^{k_0} x_i^s},$$

for all $k_0 \geq n$. □

In order to prove a lower bound on the right hand side of Inequality (1) we make use of the results by Gal (1980) and Schuierer (2001) which we state here without proof and in a simplified form for completeness; in particular, we follow the work of Schuierer (2001, Thm. 1)[1]. Define $G_a = (1, a, a^2, \ldots)$ to be the geometric sequence in a and $X^{+i} = (x_i, x_{i+1}, \ldots)$ the suffix of sequence X starting at x_i.

Theorem 2 (Schuierer, 2001) *Let $X = (x_0, x_1, \ldots)$ be a sequence of positive numbers, r an integer, and $a = \overline{\lim}_{n\to\infty}(x_n)^{1/n}$, for $a \in \mathbb{R} \cup \{+\infty\}$. If F_k, $k \geq 0$, is a sequence of functionals which satisfy*

1. *$F_k(X)$ only depends on $x_0, x_1, \ldots, x_{k+r}$,*
2. *$F_k(X)$ is continuous, for all $x_i > 0$, with $0 \leq i \leq k+r$,*
3. *$F_k(\alpha X) = F_k(X)$, for all $\alpha > 0$,*
4. *$F_k(X+Y) \leq \max\{F_k(X), F_k(Y)\}$, and*
5. *$F_{k+i}(X) \geq F_k(X^{+i})$, for all $i \geq 1$,*

then

$$\sup_{0 \leq k < \infty} F_k(X) \geq \sup_{0 \leq k < \infty} F_k(G_a).$$

For our case, the sequence X represents the lengths of the contracts in the schedule. The acceleration ratio at the completion of each contract forms a sequence of functionals. The value of each functional depends only on the prefix of contracts whose start time is before the current time. The sequence G_a represents a geometric sequence, for which we wish to show that it describes the optimal schedule.

1. Theorem 1 as proven by Schuierer (2001) applies in a broad setting, and for the purposes of our proof it suffices to consider only the case "$p = 1$".

Proposition 1 *Let $F_k(X^s)$ be a sequence of functionals defined as follows:*

$$F_k(X^s) = \sum_{i=0}^{k+n} x_i^s \Big/ \sum_{i=k-m+1}^{k} x_i^s,$$

then $F_k(X^s)$ satisfies the conditions of Theorem 2.

Proof. It is straightforward to see that $F_k(X^s)$ satisfies conditions (1)-(3). To verify condition (4), let $X_T = \sum_{i=0}^{k+n} x_i^s$, $X_B = \sum_{i=k-m+1}^{k} x_i^s$ and define Y_T and Y_B analogously (i.e., by substituting x_i^s with y_i^s). Observe that $Y_T = Y_B + Q$, where $Q = \sum_{i=0}^{k-m} y_i^s + \sum_{i=k+1}^{k+n} y_i^s$. Now, we wish to show that $F_k(X+Y) \leq \max\{F_k(X), F_k(Y)\}$ or equivalently

$$\frac{X_T + Y_T}{X_B + Y_B} = \frac{X_T + Y_B + Q}{X_B + Y_B} \leq \max\left\{\frac{X_T}{X_B}, \frac{Y_T}{Y_B}\right\} \tag{5}$$

which follows from the previously noted inequality $\max\{a/c, b/d\} \geq (a+b)/(c+d)$, for all $a, b, c, d > 0$.

To verify Condition (5) of Theorem 2, first note that from the definition of $F_k(X^s)$ we have

$$F_{k+j}(X) = \frac{\sum_{i=0}^{k+n+j} x_i^s}{\sum_{i=k-m+1+j}^{k+j} x_i^s}$$

while

$$F_k(X^{+j}) = \frac{\sum_{i=0}^{k+n} x_{i+j}^s}{\sum_{i=k-m+1}^{k} x_{i+j}^s} = \frac{\sum_{i=j}^{k+n+j} x_i^s}{\sum_{i=k-m+1+j}^{k+j} x_i^s}.$$

Lastly we observe that since all the terms in X^s are positive and hence

$$\sum_{i=0}^{j-1} x_i^s \geq 0 \quad \Longrightarrow \quad \sum_{i=0}^{j-1} x_i^s + \sum_{i=j}^{k+n+j} x_i^s \geq \sum_{i=j}^{k+n+j} x_i^s \quad \Longrightarrow \quad \frac{\sum_{i=0}^{j-1} x_i^s + \sum_{i=j}^{k+n+j} x_i^s}{\sum_{i=k-m+1+j}^{k+j} x_i^s} \geq \frac{\sum_{i=j}^{k+n+j} x_i^s}{\sum_{i=k-m+1+j}^{k+j} x_i^s}$$

as required. □

Therefore, combining Theorem 1, Proposition 1 and Theorem 2 we have:

$$R_{m,n}(X) \geq \sup_{0 \leq k < \infty} F_k(X^s) \geq \sup_{0 \leq k < \infty} F_k(G_a) = \sup_{0 \leq k < \infty} \left\{ \sum_{i=0}^{k+n} a^i \Big/ \sum_{i=k-m+1}^{k} a^i \right\}.$$

Note that for $a \neq 1$, we have $\frac{\sum_{i=0}^{k+n} a^i}{\sum_{i=k-m+1}^{k} a^i} = \frac{a^{k+n+1}-1}{a^{k+1}-a^{k-m+1}} = \frac{a^n - \frac{1}{a^{k+1}}}{1-a^{-m}}$. Therefore, if $a < 1$, we deduce that $R_{m,n}(X)$ tends to infinity as $k \to \infty$. Moreover, if $a = 1$, we obtain that

$R_{m,n}(X) = \frac{k+n+1}{m}$, which, likewise, tends to infinity as $k \to \infty$. Hence, we can assume that $a > 1$ and obtain

$$\begin{aligned} R_{m,n}(X) &\geq \sup_{0 \leq k < \infty} \left\{ \frac{(a^{k+n+1}-1)/(a-1)}{(a^{k+1}-a^{k-m+1})/(a-1)} \right\} \\ &= \sup_{0 \leq k < \infty} \left\{ \frac{a^{k+n+1}-1}{a^{k+1}-a^{k-m+1}} \right\} \overset{(a>1)}{=} \frac{a^n}{1-a^{-m}} = \frac{a^{n+m}}{a^m-1}. \end{aligned}$$

The above expression is minimized for $a = ((m+n)/n)^{1/m}$, which implies that the acceleration ratio of X is bounded from below by

$$R_{m,n}(X) \geq \frac{\left(\frac{m+n}{n}\right)^{(m+n)/m}}{\frac{m+n}{n}-1} = \left(\frac{n}{m}\right)\left(\frac{m+n}{n}\right)^{\frac{m+n}{m}}.$$

We have thus shown the following theorem:

Theorem 3 *Given n problem instances and m processors, every schedule that simulates an interruptible algorithm using executions of contract algorithms has an acceleration ratio no less than*

$$\left(\frac{n}{m}\right)\left(\frac{m+n}{n}\right)^{\frac{m+n}{m}}.$$

It is worth pointing out that the round-robin schedule of contract lengths $1, a, a^2, \ldots$ for a value of $a = \frac{m+n}{n}^{\frac{1}{m}}$ has acceleration ratio that is precisely $\left(\frac{n}{m}\right)\left(\frac{m+n}{n}\right)^{\frac{m+n}{m}}$, as shown by Bernstein et al. (2003), and thus matches our lower bound. In other words, the round robin and length-increasing schedule proposed by Bernstein et al. is optimal among *all* possible schedules whether round robin and length-increasing, or not. This is not to say that all optimal schedules are round robin and length-increasing, in fact one can easily construct optimal non round-robin schedules for the case when n is a multiple of m. More formally, we obtain the following theorem.

Theorem 4 *The optimal schedule for n problems and m processors that simulates an interruptible algorithm using executions of contract algorithms has an acceleration ratio of*

$$\left(\frac{n}{m}\right)\left(\frac{m+n}{n}\right)^{\frac{m+n}{m}}.$$

Theorem 4 provides a tight bound on the *worst-case* acceleration ratio. More precisely, we assume that interruptions are issued by a malicious adversary, typically right before a contract algorithm terminates its execution. In this sense, the measure is extremely pessimistic, and reflects only the performance of a schedule in an adversarial setting. In the next section we show how to upper-bound the *average-case* acceleration ratio of exponential schedules. The issue of stochastic deadlines has been addressed by Zilberstein et al. (2003) in the case of a single processor and n problem instances. In their setting, there is uncertainty about both the interruption and the quality of the output of the contract algorithm. Similar types of analysis have been applied by Kao and Littman (1997) to the ray-search problem on two rays, assuming some probabilistic knowledge on the placement of the target.

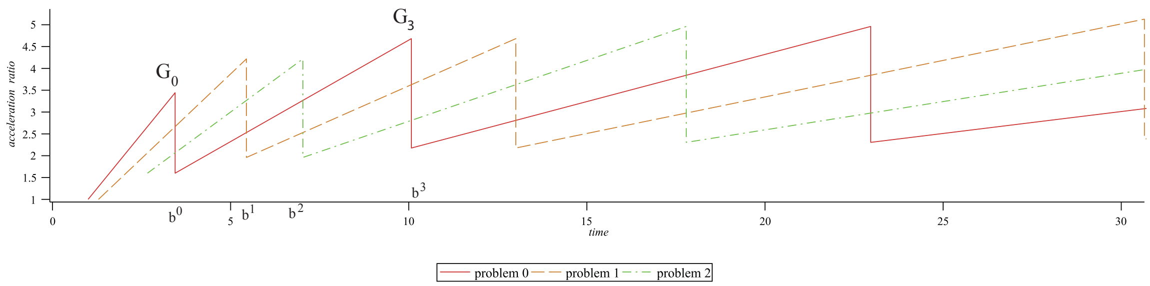

Figure 4: Example with $m = 2$, $n = 3$. Each line corresponds to the acceleration ratio on that problem if interrupted at time t.

4. Average-Case Analysis of Exponential Strategies

In this section we present an average-case analysis of the acceleration ratio of exponential strategies (recall the formal definition given in Section 2). In this scenario, the interruption occurs at a time which is not chosen adversarially, rather it is chosen uniformly at random in the interval $[0, U]$, for some $U > 0$. Likewise, the problem instance which is queried is also chosen uniformly at random among the n problems.

A similar problem has been considered in the context of robots searching for a target on m rays. A natural scenario is a hiker who becomes injured in the woods. The searchers must then efficiently explore the forest trails as to find the injured person as soon as possible. In this setting there is no reason to expect that the hiker would locate itself adversarially. Hence while the worst case bound provides an exact upper bound on the worst possible delay in reaching the target, the average case analysis gives a more realistic estimate of the expected amount of time before the target is found. Kao et al. (1996) showed that on the average the target is found nearly twice as fast as in the worst case scenario.

Here we consider the analogous setting in which the interruption time is chosen independently at random, rather than adversarially. Hence, we expect that the randomly-chosen interruption will most likely not coincide with interruptions that yield high values of the acceleration ratio in the worst-case scenario.

For the remainder of this section, we consider an exponential schedule X with base b; that is, the length of the k-th contract in the cyclic ordering is b^k. Let G_k denote the finish time of this contract. We also assume that the problems are numbered $0, \ldots, n-1$; this ordering makes the presentation easier to follow.

Before formalizing the concept of the average acceleration ratio we illustrate the intuition using an example. Refer to Figure 4, which depicts an example with $n = 3$ problems and $m = 2$ processors. Here, each of the three jagged line segments corresponds to the acceleration ratio of each problem as a function of interruption time. More precisely, each line is a plot of the function $\alpha_p(t)$, as formally defined in Section 2. Consider for example problem 0 as shown by the solid line in Figure 4. The first contract completed for this problem is contract 0, of length b^0, and has completion time G_0. After this contract is completed, the available processors are occupied computing contracts of length b^1 and b^2

for problems 1 and 2 respectively. Eventually, a processor becomes available and computes a contract of length b^3 for problem 0, which is completed at time G_3. Note that the acceleration ratio for problem 0 degrades linearly (i.e., increases) in the time interval between G_0 and G_3. Similar observations can be made for the remaining linear segments of the plot, as well as for the other two problems.

The average acceleration ratio for a specific problem is then described by the area under its respective acceleration ratio curve, normalized by the length U of the sampled space $[0, U]$. The overall average acceleration ratio is then given by the average of the individual acceleration ratio averages for each of the n problems[2]. Since the acceleration ratio for each problem is a piece-wise linear function, in order to obtain the average we compute the integral under each line segment, and sum over all segments. To normalize, we need to divide by the length of the interval as well as by the number n of problems.

Consider now an interruption T in the interval $[G_k, G_{n+k})$, for some index k in the cyclic order, and let p_k denote the problem that corresponds to the contract with finish time G_k. Since the schedule is exponential, it follows that $\alpha_{p_k}(T) = T/b^k$. Since T is a random variable, α_{p_k} is also a random variable. Thus, we can compute the average acceleration ratio for problem p_k within the interval $[G_k, G_{n+k})$ as

$$
\begin{aligned}
E_{[G_k,G_{k+n})}[\alpha_{p_k}] &= \frac{1}{G_{k+n}-G_k}\int_{G_k}^{G_{k+n}} \alpha_{p_k}(x)\,dx \\
&= \frac{1}{G_{k+n}-G_k}\int_{G_k}^{G_{k+n}} \frac{x}{b^k}\,dx \\
&= \frac{1}{G_{k+n}-G_k}\int_{0}^{G_{k+n}-G_k} \frac{G_k+x}{b^k}\,dx \\
&= \frac{1}{G_{k+n}-G_k}\cdot\left(\frac{G_k(G_{k+n}-G_k)}{b^k}+\frac{(G_{k+n}-G_k)^2}{2\,b^k}\right) \\
&= \frac{G_{k+n}+G_k}{2b^k}.
\end{aligned}
$$

To compute the average acceleration ratio of problem p_k over the entire span $[0, U]$ we need to add up the area below the entire jagged line that corresponds to this problem. Observe that the area below a single sawtooth is given by the quantity $(G_{k+n} - G_k) \cdot E_{[G_k,G_{k+n})}[\alpha_{p_k}]$. This allows us to give expressions for the acceleration ratio of each one of the n problems.

2. Normally, a standard assumption is that interruptions occur after at least one contract for each problem is completed, namely after time G_{n-1}. We make the simplifying assumption that interruptions can occur earlier than G_{n-1}, in which case the acceleration ratio is 0. A refinement to interruptions only after G_{n-1} follows along the lines of the discussion in this section. In this case the average is over the sampled space $[G_{n-1}, U)$ and has a net zero effect on the asymptotic acceleration ratio (the extra term goes to zero as U goes to infinity), but it results in much more complicated expressions.

In particular, the average acceleration ratio of problem 0 is given by:

$$\begin{aligned}
E_{[0,U)}[\alpha_0] &= \frac{1}{U}\left(\sum_{k=0}^{\lfloor r/n \rfloor} \left(G_{(k+1)n} - G_{kn}\right) E_{[G_{kn},G_{(k+1)n})}[\alpha_0] + (U - G_{r+n}) E_{[G_{r+n},U)}[\alpha_0]\right) \\
&= \frac{1}{U}\left(\sum_{k=0}^{\lfloor r/n \rfloor} \frac{G^2_{(k+1)n} - G^2_{kn}}{2b^{kn}} + \frac{U^2 - G^2_{r+n}}{2b^{r+n}}\right) \qquad (6)
\end{aligned}$$

where r is such that $U \in [G_{r+n}, G_{r+n+1})$. Note that the last term corresponds to the truncated sawtooth in the interval $[G_{r+n}, U]$.

The expression for the average acceleration ratio for problem $i \in \{1, \ldots n-1\}$ is similar; however, instead of endpoints at $G_0, G_n, G_{2n}, \ldots$ we need to consider the endpoints $G_i, G_{n+i}, G_{2n+i}, \ldots$; More precisely, we obtain the expression

$$E_{[0,U)}[\alpha_i] = \frac{1}{U}\left(\sum_{w=0}^{\lfloor \frac{r-i}{n} \rfloor} \frac{G^2_{(w+1)n+i} - G^2_{wn+i}}{2b^{wn+i}} + \frac{U^2 - G^2_{r+i}}{2b^{r+i}}\right). \qquad (7)$$

The overall acceleration ratio α is a random variable defined as $\alpha = \frac{1}{n}\sum_{i=0}^{n-1} \alpha_i$. Using (6) and (7) we obtain

$$\begin{aligned}
E_{[0,U]}[\alpha] &= \frac{1}{n}\sum_{i=0}^{n-1} E_{[0,U]}[\alpha_i] \\
&= \frac{1}{nU}\left(\sum_{k=0}^{r} (G_{k+n} - G_k)\, E_{[G_k,G_{k+n})}[\alpha_0] + \sum_{i=1}^{n-1} (U - G_{r+i})\, E_{[G_{r+i},U]}[\alpha_i]\right) \\
&= \frac{1}{nU}\left(\sum_{k=0}^{r} \frac{G^2_{k+n} - G^2_k}{2b^k} + \sum_{i=1}^{n-1} \frac{U^2 - G^2_{r+i}}{2b^{r+i}}\right). \qquad (8)
\end{aligned}$$

It is easy to compute G_k (Bernstein et al., 2003, Proof of Theorem 1). Namely,

$$G_k = \sum_{i=0}^{\lfloor k/m \rfloor} b^{mi+(k \bmod m)} = b^{k \bmod m} \sum_{i=0}^{\lfloor k/m \rfloor} b^{mi} = \frac{b^{k+m} - b^{k \bmod m}}{b^m - 1}.$$

For simplicity of presentation we consider only the case $U = G_{r+n}$. The more general case is analogous, though contains slightly more complicated expressions.

$$\begin{aligned}
E_{[0,G_{r+n}]}[\alpha] &= \frac{1}{nG_{r+n}}\left(\sum_{k=0}^{r} \frac{G^2_{k+n} - G^2_k}{2b^k} + \sum_{i=1}^{n-1} \frac{G^2_{r+n} - G^2_{r+i}}{2b^{r+i}}\right) \\
&= \frac{1/(b^m - 1)}{2n(b^{r+n+m} - b^{[\![r+n]\!]_m})}\left(\sum_{k=0}^{r} \frac{(b^{k+n+m} - b^{[\![k+n]\!]_m})^2 - (b^{k+m} - b^{[\![k]\!]_m})^2}{b^k}\right. \\
&\qquad \left. + \sum_{i=1}^{n-1} \frac{(b^{r+n+m} - b^{[\![r+n]\!]_m})^2 - (b^{r+i+m} - b^{[\![r+i]\!]_m})^2}{b^{r+i}}\right). \qquad (9)
\end{aligned}$$

where $[\![x]\!]_m$ denotes $x \bmod m$ We now evaluate separately the two summations involved in (9). First,

$$\begin{aligned}
&\sum_{k=0}^{r} \frac{(b^{k+n+m} - b^{[\![k+n]\!]_m})^2 - (b^{k+m} - b^{[\![k]\!]_m})^2}{b^k} = \\
= &\sum_{k=0}^{r} \left(b^{2(n+m)+k} - 2b^{n+m+[\![k+n]\!]_m} + \frac{b^{2[\![k+n]\!]_m}}{b^k} - b^{2m+k} + 2b^{m+[\![k]\!]_m} - \frac{b^{2[\![k]\!]_m}}{b^k} \right) \\
= &\sum_{k=0}^{r} \left((b^{2(n+m)} - b^{2m})b^k - 2b^{n+m+[\![k+n]\!]_m} + \frac{b^{2[\![k+n]\!]_m}}{b^k} + 2b^{m+[\![k]\!]_m} - \frac{b^{2[\![k]\!]_m}}{b^k} \right) \\
\leq &\sum_{k=0}^{r} \left((b^{2(n+m)} - b^{2m})b^k + 2b^{n+2m} + \frac{b^{2m}}{b^k} + 2b^{2m} + \frac{b^{2m}}{b^k} \right) \\
= &\left(b^{2(n+m)} - b^{2m} \right) \frac{b^{r+1}-1}{b-1} + O\big(rb^{n+2m}\big) + O\left(\frac{b^{2m+1}}{b-1} \right), \qquad (10)
\end{aligned}$$

where in the penultimate step we used that $[\![x]\!]_m < m$. Second, we get

$$\begin{aligned}
&\sum_{i=1}^{n-1} \frac{(b^{r+n+m} - b^{[\![r+n]\!]_m})^2 - (b^{r+i+m} - b^{[\![r+i]\!]_m})^2}{b^{r+i}} = \\
= &\sum_{i=1}^{n-1} \left(\frac{b^{2(n+m)+r} - 2b^{n+m+[\![r+n]\!]_m}}{b^i} + \frac{b^{2[\![r+n]\!]_m}}{b^{r+i}} - b^{2m++i} + 2b^{m+[\![r+i]\!]_m} - \frac{b^{2[\![r+i]\!]_m}}{b^{r+i}} \right) \\
= &\; b^{2m+r+n+1} \frac{b^{n-1}-1}{b-1} + O\left(\frac{2b^{2m+1}(b^{n-1}-1)}{b^r(b-1)} \right) - b^{2m+r+1} \frac{b^{n-1}-1}{b-1} + \\
+ &\; O\big(nb^{2m}\big) + O\left(\frac{b^{2m+1-n}(b^{n-1}-1)}{b^r(b-1)} \right). \qquad (11)
\end{aligned}$$

Substituting (10) and (11) in (9) and using simple algebraic expansion and simplification we obtain

$$\begin{aligned}
E_{[0,G_{r+n}]}[\alpha] \;=\;& \frac{(b^m-1)^{-1}}{2n(b^{n+m} - b^{[\![r+n]\!]_m - r})} \left[\left(b^{2(n+m)} - b^{2m} \right) \frac{b}{b-1} + \left(b^{2m+n+1} - b^{2m+1} \right) \frac{b^{n-1}-1}{b-1} \right] \\
+\;& \frac{(b^m-1)^{-1}}{2n(b^{n+m} - b^{[\![r+n]\!]_m - r})} \left[O\left(rnb^{n+2m} \cdot \frac{1}{b^r} \right) + O\left(\frac{b^{2m+1}(b^{n-1}-1)}{b-1} \cdot \frac{1}{b^r} \right) \right]. \qquad (12)
\end{aligned}$$

Note that in (12) the O-terms tend asymptotically to 0 as r approaches infinity. Hence, these terms have negligible impact on the acceleration ratio when the interruption occurs far ahead in time, and in this sense they can be considered error terms. In particular, we obtain the following simplified expression for the *asymptotic* acceleration ratio:

$$\begin{aligned}
E_{[0,G_{r+n}]}[\alpha] \;\overset{r\to\infty}{=}\;& \frac{(b^m-1)^{-1}}{2nb^{n+m}} \left[\left(b^{2(n+m)} - b^{2m} \right) \frac{b}{b-1} + \left(b^{2m+n+1} - b^{2m+1} \right) \frac{b^{n-1}-1}{b-1} \right] \\
=\;& \frac{b^m(b^n-1)(b+1)}{2n(b^m-1)(b-1)}.
\end{aligned}$$

Theorem 5 *The asymptotic average acceleration ratio of an exponential schedule b^i, for $i = 0, 1, \ldots$, with interruption chosen uniformly at random in $[0, U]$ is bounded by*

$$\frac{(b^{m+n} - b^m)\,(b+1)}{2n\,(b^m - 1)\,(b-1)}.$$

Proof. Follows from the discussion above. □

Corollary 2 *The asymptotic average acceleration ratio of the optimal schedule in Theorem 4 is bounded by*

$$\frac{(m+n)\left(\left(\frac{m+n}{n}\right)^{\frac{n}{m}} - 1\right)\left(\left(\frac{m+n}{n}\right)^{\frac{1}{m}} + 1\right)}{2mn\left(\left(\frac{m+n}{n}\right)^{\frac{1}{m}} - 1\right)}.$$

Corollary 3 *The schedule with optimal acceleration ratio has a non-optimal average acceleration ratio.*

Proof. This can readily be verified by computing the derivative of the expression from Theorem 5 and evaluating it around the value $b^* = ((m+n)/n)^{1/m}$ used by the optimal worst case strategy. One can then observe through algebraic manipulation that the derivative is always negative at this point, which implies that a larger value $b' = b^* + \epsilon$ results in a lower (i.e. better) average acceleration ratio and hence the worst-case optimal schedule is not optimal in the average sense. □

5. Conclusion

In this paper we resolved an open question posed by Bernstein et al. (2003) concerning the optimal acceleration ratio of a schedule of contract algorithms. This is a well-studied problem in artificial intelligence, with several applications in the design of real-time systems. Our main result shows that optimal schedules can be found in the class of cyclic schedules, or, alternatively, that we cannot improve the quality of the simulation by designing more complicated schedules. We also performed an average-case analysis of exponential schedules, assuming a uniform distribution of the interruption times.

In more recent work, Angelopoulos, López-Ortiz, and Hamel (2008) were able to apply similar techniques so as to design optimal schedules for interruptible algorithms at the presence of *soft deadlines.* In this setting, the interruption is not a hard deadline, in the sense that when an interruption occurs, the algorithm is allowed an additional window of time to complete its execution (which can be seen, intuitively, as a "grace period"). The algorithm must then report the solution to the queried problem within the additional time window. In a different example of further work, Angelopoulos and López-Ortiz (2009) addressed refinements of the acceleration ratio which reflect better the performance of schedules when the number of problems is larger than the number of available processors. In both cases, we resorted to the use of normalization techniques along the same lines as Section 3. This provides evidence that our techniques are not tied to this specific variant of the problem, but instead can be applicable in much wider settings.

An interesting open problem is to find tight bounds for randomized strategies, namely schedules in which the length of contracts, as well as the processors to which the contracts are assigned are random variables. Note that randomization is known to be of help in the context of the ray searching problem (Kao et al., 1996). A related (and very challenging) problem is to find schedules that achieve optimal average-case acceleration ratio. Are exponential schedules optimal under this measure? Based on our analysis in Section 4, we expect that an answer to this question will be fairly technical and involved. An even more ambitious problem is to find optimal schedules assuming a certain known probability distribution on the interruption times and problems that are queried.

Exponential schedules can be seen as an example of a *doubling* algorithm, in that the lengths of contracts increase geometrically. The challenging part of this work (as in Angelopoulos et al., 2008; Angelopoulos & López-Ortiz, 2009) is to lower-bound the performance of arbitrary strategies: this is accomplished by lower-bounding the supremum of a sequence of functions by functionals over geometric sequences (Theorem 2). We believe that similar techniques can be applied to many other optimization problems for which doubling algorithms are known to perform well (see, e.g., the survey of Chrobak & Mathieu, 2006). A recent example can be found in the work of Langetepe (2010), concerning the optimality of spiral search for locating a target on the plane.

Last, but not least, while the optimal schedule presented in our work has parallels to search strategies for the problem of searching m rays using p robots, the exact correspondence remains to be shown. Such a correspondence would extend the one established by Bernstein et al. (2003) concerning cyclic schedules and strategies.

6. Acknowledgments

A preliminary version of this paper (López-Ortiz, Angelopoulos, & Hamel, 2006) appeared in the Proceedings of the Twenty-First National Conference on Artificial Intelligence (AAAI).

References

Alpern, S., & Gal, S. (2003). *The Theory of Search Games and Rendezvous*. Kluwer Academic Publishers.

Althöfer, I. (1997). A symbiosis of man and machine beats grandmaster Timoshchenko. *Journal of the International Computer Chess Association.*, *20*(1), 40–47.

Angelopoulos, S., & López-Ortiz, A. (2009). Interruptible algorithms for multi-problem solving. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 380–386.

Angelopoulos, S., López-Ortiz, A., & Hamel, A. (2008). Optimal scheduling of contract algorithms with soft deadlines. In *Proceedings of the 23rd National Conference on Artificial Intelligence (AAAI)*, pp. 868–873.

Baeza-Yates, R., Culberson, J., & Rawlins, G. (1993). Searching in the plane. *Information and Computation*, *106*, 234–252.

Bernstein, D. S., Finkelstein, L., & Zilberstein, S. (2003). Contract algorithms and robots on rays: Unifying two scheduling problems.. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1211–1217.

Bernstein, D., Perkins, T. J., Zilberstein, S., & Finkelstein, L. (2002). Scheduling contract algorithms on multiple processors. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI)*, pp. 702–706.

Chrobak, M., & Mathieu, C. (2006). Competitiveness via doubling. *SIGACT News*, *37*(4), 115–126.

Dean, T., & Boddy, M. S. (1988). An analysis of time-dependent planning. In *Proceedings of the 7th National Conference on Artificial Intelligence*, pp. 49–54.

Gal, S. (1980). *Search Games*. Academic Press.

Horvitz, E. (1987). Reasoning about beliefs and actions under computational resource constraints. In *Proceedings of the Third Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, pp. 301–324.

Horvitz, E. (1998). Reasoning under varying and uncertain resource constraints. In *Proceedings of the 7th National Conference on Artificial Intelligence (AAAI)*, pp. 111–116.

Jaillet, P., & Stafford, M. (2001). Online searching. *Operations Research*, *49*, 501–515.

Kao, M.-Y., & Littman, M. L. (1997). Algorithms for informed cows. In *AAAI workshop on Online Search*.

Kao, M.-Y., Ma, Y., Sipser, M., & Yin, Y. (1998). Optimal constructions of hybrid algorithms. *Journal of Algorithms*, *29*(1), 142–164.

Kao, M.-Y., Reif, J. H., & Tate, S. R. (1996). Searching in an unknown environment: An optimal randomized algorithm for the cow-path problem. *Information and Computation*, *131*(1), 63–79.

Langetepe, E. (2010). On the optimality of spiral search. In *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*.

López-Ortiz, A., Angelopoulos, S., & Hamel, A. (2006). Optimal scheduling of contract algorithms for anytime problems. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI)*.

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

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

Schuierer, S. (2001). Lower bounds in on-line geometric searching. *Computational Geometry: Theory and Applications*, *18*(1), 37–53.

Schuierer, S. (2003). A lower bound for randomized searching on m rays. In *Computer Science in Perspective*, pp. 264–277.

Zilberstein, S. (1996). Using anytime algorithms in intelligent systems. *AI Magazine*, *17*(3), 73–83.

Zilberstein, S., Charpillet, F., & Chassaing, P. (2003). Real-time problem-solving with contract algorithms. *Annals of Mathematics and Artificial Intelligence*, *39*(1–2), 1–18.