Optimal Scheduling of Contract Algorithms for Anytime Problems

Alejandro López-Ortiz
David. R. Cheriton School of Computer Science
University of Waterloo
Waterloo ON Canada, N2L 3G1
alopez-o@uwaterloo.ca

Spyros Angelopoulos
David R. Cheriton School of Computer Science
University of Waterloo
Waterloo ON Canada, N2L 3G1
sangelop@uwaterloo.ca

Angèle M. Hamel
Department of Physics and Computer Science
Wilfrid Laurier University
Waterloo, Ontario, Canada, N2L 3C5
ahamel@wlu.ca

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 compute a solution within the alloted time. An *interruptible* algorithm, in contrast, can be interrupted at an arbitrary point in time and must produce a 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 this simulation. This resolves an open conjecture of Bernstein et al. [IJCAI 2003] who gave an ingenious optimal schedule under the restricted setting of round robin and length-increasing processor schedules, but whose optimality in the general unrestricted case remained open.

Introduction

Anytime algorithms were first considered by Horvitz (Horvitz 1987), (Horvitz 1998) and Dean and Boddy (Dean & Boddy 1998). They occur in settings where a computationally intensive problem is addressed under uncertain resource constraints. These algorithms can produce solutions under arbitrary resource constraints, albeit of potentially different quality depending on the amount of resources alloted. 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. 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 to take advantage of the newly posted price. Another example is given by realtime applications. For instance, consider an motion planning algorithm for a robot in which a solution must be produced within a certain, but varying, amount of time: for certain actions the next step can be carefully computed; for others a move is needed momentarily even if the algorithm is to produce a suboptimal move. In this case the amount of time alloted is given to the algorithm beforehand.

Copyright © 2006, American Association for Artificial Intelligence (www.aaai.org). All rights reserved.

More formally, consider the following scenario: we are given a set P of n different problem instances, and we want to design an algorithm that can be applied, potentially, to any of those n problem instances. The computation time available to the algorithm is not known in advance. More precisely, at a given (unknown) point of time, an interruption occurs, at which point the algorithm is stopped and is queried to report its (partial) solution to any possible instance. 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 before 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 1993; Kao *et al.* 1994), 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.

Algorithms which have the property that they can be interrupted at any time, and thus queried for their solution to a problem instance, are known as *interruptible* algorithms. Such algorithms include versions of local search, e.g., simulating annealing and hill climbing. On the other side, *contract* algorithms are given the amount of allowable computation time (i.e, the intended query time) as part of the input. Clearly, contract algorithms are more specialized than interruptible algorithms. More importantly, contract algorithms are often instrumental in the design of interruptible algorithms. Here, the objective is to define, for each problem instance, a sequence of contracts of increasing lengths (i.e., processing times). Note that, conceptually, the set of required contracts is infinite. Suppose then that a set M of m processors of identical speed is available. The problem we face is scheduling the contracts to each of the available 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* which has been *completed* by the query time.

Several classes of algorithms fall into the paradigm of contract algorithms. For instance, several examples of Fully Polynomial-Time Approximation Schemes (FPTAS) which are based on dynamic programming (DP) techniques can in-

deed be seen as contract algorithms. Here, the general approach is as follows. Given a (usually NP-hard) problem, one first applies an appropriate scaling, then a subsequent rounding, of the input so as to reduce the space of allowable values for the input items. The scaling depends on a parameter ϵ: the smaller the ϵ, the smoother the scaling is. Subsequently, DP is applied on the scaled and rounded input: the running time typically is a function of $1/\epsilon$, whereas the approximation ratio is typically within a factor of $(1-\epsilon)$ from the optimal solution (assuming e.g., a profit-maximization problem). Note that if the time the algorithm is allowed to run, say T, is known in advance, the designer can choose the smallest value of ϵ such that the running time cannot exceed T. However, if T is not known, and an interruption has occurred, it is likely the algorithm will not return any meaningful solution. For a concrete example, see the FPTAS for the well-known knapsack problem due to Ibarra and Kim (Ibarra & Kim 1975)

The efficiency of a scheduling for a set of contracts is determined by the so-called acceleration ratio. A formal definition is provided in the Preliminaries section; 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 for an off-line algorithm that has foreknowledge not only of the query time t but also of the problem instance p of interest; such an 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 as the competitive ratio, which is defined in the context of the analysis of online algorithms.

In the case of one problem instance and a single processor, Russell and Zilberstein (Russell & Zilberstein 1991) showed that iterative doubling of contract lengths gives rise to an interruptible algorithm of acceleration ratio at most four. Zilberstein *et al.* (Zilberstein, Charpillet, & Chassaing 1999) 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.* (Zilberstein, Charpillet, & Chassaing 1999) studied the generalization of the problem in which multiple problem instances are considered (assuming $|M| = 1$), and Bernstein *et al.* (Bernstein *et al.* 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 with 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.* (Bernstein, Finkelstein, & Zilberstein 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. 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.

The work of Bernstein *et al.* (Bernstein, Finkelstein, & Zilberstein 2003) drew a connection between the problem of scheduling contract algorithms and another well-studied problem in AI, namely that of robot searching on a set of rays (Alpern & Gal 2003). In this problem, p robots search for a target that is located in one of m concurrent rays. We seek search strategies for the robots that minimize the *competitive ratio*, namely the maximum of the ratio of the search cost using the strategy, and the (optimal) distance from the starting position to the target, over all possible positions of the target. Note that the two problems have striking similarities; 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 (López-Ortiz & Schuierer 2004) showed an optimal strategy that achieves competitive ratio $1 + 2\frac{m-p}{p}(\frac{m}{m-p})^{\frac{m}{p}}$. Bernstein *et al.* (Bernstein, Finkelstein, & Zilberstein 2003) later derived the same bound by directly translating their solution in the context of contract scheduling to the context of robot searching in rays.

The paper is organized as follows. In the Preliminaries section we present formal definitions and preliminaries. Our main result, namely Theorem 3, is presented in the Lower Bound section. In particular, we show how to adapt, in a non-trivial way, the approach of López-Ortiz and Schuierer to show the optimality of the schedule of Bernstein *et al.* without any restrictions on the scheduling strategy for contracts.

Preliminaries

Let P denote the set of *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 assume all processors have the same speed). 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 a the set $\{(c_i, m_i, x_i) : c_i \in C\}$, where $m_i \in \{0, \ldots, m-1\}$ denotes the processor to which c_i is scheduled, and x_i denotes the time its processing begins. The schedule X must be feasible, in the sense that for every two contracts $c_i = (p_i, d_i), c_j = (p_j, d_j)$ in C, which are assigned to the same processor by X, and with c_j scheduled immediately after c_i in the said processor, we have that $x_i + d_i \leq x_j$. Namely, c_j does not start before c_i has been completed.

Observation 1 *Without loss of generality we consider only schedules in which $x_i + d_i = x_j$ for i and j as defined above; that is, the processors are never idle.*

To compare schedules we use the standard *acceleration ratio* metric. Following (Bernstein, Finkelstein, & Zilberstein 2003), we assume that when a contract is completed at time t its solution is available when the interruption occurs at any time after t, including 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. 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.

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 all possible interruption times t. Let ϵ denote an infinitesimally small positive value. Then it is easy to see that it suffices to consider interruptions that occur only at times $T - \epsilon$, where T belongs in the set of finish times of all contracts in X, excluding the first contract for each problem. 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, $l(X, p, t'-\epsilon) = l(X, p, t)$; in other words the algorithm has not made any progress on any problem on the time interval $[t, t' - \epsilon]$, thus $t/l(X, p, t) < (t' - \epsilon)/l(X, p, t' - \epsilon)$.

Figure 1 illustrates an example of a schedule of a set of contracts. Note how the acceleration ratio peaks just before each contract is completed.

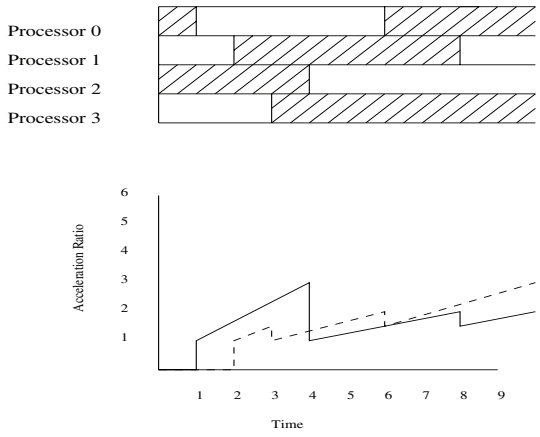

Figure 1: Bar diagram depicting the scheduling of two problems on four processors, and a plot of time vs. acceleration ratio for each problem. Problem 1 (solid line) has contracts of lengths 1 on processor 0, 4 on processor 2, and 6 on processor 1. Problem 2 (dashed line) has contracts of lengths 2 on processor 1, 3 on processor 3, and 5 on processor 0.

Observation 2 *The acceleration ratio of a schedule X for P is*

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

Given a schedule X, and two problem instances $p, p' \in P$, let C_1, C_2 denote two (potentially infinite) subsets of X, such that all contracts in C_1 have problem tag p, and all contracts in C_2 have problem tag p'. 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' instead of p and every contract in C_2 acquires problem tag p instead of p'. 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 (López-Ortiz & 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 there is a strictly increasing function $T(n)$ with $\lim_{n\to\infty} T(n) = \infty$ such that for each n, $X_m \subseteq_{T(n)} X_{m+1}$ for all $m \geq n$. The limit schedule X is defined in the obvious way.

Lower Bound

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 the conditions of Lemma 1, Lemma 2, and Theorem 1 in that order. These transformations either preserve or reduce the acceleration ratio. Interestingly, the last transformation can possibly produce an object that is not necessarily a schedule, but whose acceleration ratio is nevertheless well defined. We then lowerbound the acceleration ratio of these objects and show that it matches the upper bound of Bernstein *et al.* (Bernstein, Finkelstein, & Zilberstein 2003).

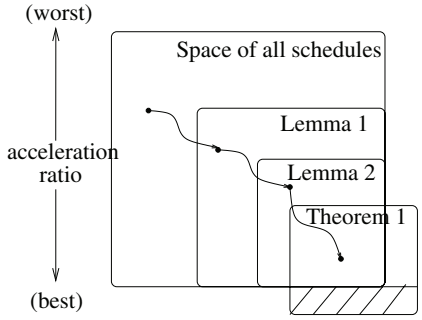

Figure 2: Illustration of the proof technique. The shaded region corresponds to non-valid strategies, which could, in principle, have acceleration ratio strictly smaller than the optimal acceleration ratio; however, we show that this never happens, thus the region collapses to the line of all strategies of optimal acceleration ratio.

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

We are given a schedule X of contracts that includes the contract (p_i, d_i) and the next contract in the schedule involving p_i that is completed is (p_i, D_i). We will follow this convention throughout of denoting by lower case d and upper case D the durations of a pair of consecutively completed contracts on a given problem p, respectively.

Definition 2 *Given a contract c of length D_c we define the* acceleration ratio *at D_c immediately before completion of c as $r(c) = (T_c + D_c)/d_c$ (assuming $d_c \neq 0$).*

In particular, let C' denote the set of all contracts in the schedule, excluding the first completed contract for each problem. Then from Observation 1,

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

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

Suppose time T_i is the time when a processor is to start working on contract (p_i, D_i) and there is another problem p_j such that the terms (p_j, d_j) and (p_j, D_j) appear in X. Suppose a processor is to start working on contract (p_j, D_j) at time T_j.

Lemma 1 *Given a schedule X with two problems p_i and p_j as described above with $d_j < d_i$ 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) as described above to obtain a new schedule X'. Then we will show the acceleration ratio of the new schedule X' 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 after the swap is

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

Observe that everywhere else the acceleration ratio of the schedule remains unchanged. Then since $T_j > T_i$ and $D_j > D_i$, $\frac{T_j + D_j}{d_j} \geq \frac{T_i + D_i}{d_j}$, and since $d_j < d_i$, we have $\frac{T_j + D_j}{d_j} \geq \frac{T_j + D_j}{d_i}$. Hence the acceleration ratio of the original schedule is greater than or equal to the acceleration ratio of the alternative schedule. □

Corollary 1 *Given a schedule X there is another schedule in which for any two problems p_i and p_j, with $d_j < d_i$, and $T_j > T_i$, it is always the case that $D_j < D_i$.*

Proof. We apply the process, i.e., appropriate problem switching, 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 lengths for all problems in X. 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 to f_ℓ. Now starting with f_1 and for each f_ℓ with $\ell = 1, 2 \ldots$ 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 Lemma 2. Let S_T denote the set of all contracts completed by time T. Also let S^T be the complement of S_T, namely all contracts in $X - S_T$. For any problem p_j let $D_j = \min\{D : (p_j, D) \in S^{T_0+D_0}\}$. Observe that $d_j = \max\{d : (p_j, d) \in S_{T_0+D_0}\}$.

Lemma 2 *Let $C_1 = (p_0, D_0)$ be a contract scheduled by X at time T_0 and $C_2 = (p_j, D_j)$ be any contract in S^{T_0}. Then there exists another schedule of no worse acceleration ratio such that if $d_0 \geq d_j$ for a problem $p_j \neq p_0$ then $T_0 + D_0 \geq T_j + D_j$.*

Proof. Assume this is not the case, i.e. the schedule X is such that $d_0 \geq d_j$ for at least one problem $p_j \neq p_0$ and $T_0 + D_0 \leq T_j + D_j$. Consider then the switch in which all contracts for problems p_0 and p_j in $S^{T_0+D_0}$ switch problem tags. We will argue that the switch gives rise to a new schedule 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_1 and C_2 are completed. First, note that before the switch, the contribution of those two points to the acceleration ratio is

$$\alpha = \max\left\{\frac{T_0 + D_0}{d_0}, \frac{T_j + D_j}{d_j}\right\}$$

and after the switch, the corresponding contribution becomes

$$\beta = \max\left\{\frac{T_0 + D_0}{d_j}, \frac{T_j + D_j}{d_0}\right\}.$$

Since $d_j \leq d_0$ and $T_0 + D_0 \leq T_j + D_j$ we get that $\beta \leq \alpha$, 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 to the smallest contract time f_ℓ with $\ell = 1, 2 \ldots$ and checking that no larger contract $d_0 \geq d_j$ is such that $T_0 + D_0 \leq T_j + D_j$. If there are one or more such contracts, we select the D_0 with the smallest completion time $T_0 + D_0$ and switch 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 $d_0 \geq d_j$ with $p_j \neq p_0$ we have that $T_0 + D_0 \geq T_j + D_j$.

Corollary 2 *Let $C = (p_i, D_i)$ denote the contract scheduled by X at time T_i. Then there exists another schedule of no worse acceleration ratio in which there is no contract $C_k = (T_k, D_k)$ with $d_k > d_i$ such that $T_k + D_k < T_i + D_i$.*

Proof. Assume the contrary, and then apply Lemma 2 with $T_0 = T_k$ and $T_j = T_i$. This shows then that there exists another schedule in which $T_k + D_k \geq T_i + D_i$ as claimed and no worse acceleration ratio. □

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

Theorem 1 *The acceleration ratio $R_{m,n}(X)$ of an optimal normalized schedule $X = ((c_0, m_0, x_0), (c_1, m_1, x_1), \ldots)$ for n problems with m processors is at least*

$$R_{m,n}(X) \geq \sup_{k \geq 0} \left\{ \sum_{i=0}^{k+n} x_i^s \Big/ \sum_{i=k-m+1}^{k} x_i^s \right\} \quad (1)$$

where $X^s = (x_0^s, x_1^s, \ldots)$ is the sequence of the sorted x values of X and $x_i^s := 0$ 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 allotted time that M_0 will spend on P_0 at time T_0. Let the longest completed contract for problem P_0 at time $T_0 + D_0$ be d_0. In general, let M_j be the processor responsible for the longest completed contract on problem P_j at time $T_0 + D_0$, for $0 \leq j \leq n-1$.

Now consider the sequence of contract lengths of processor M_j completed up to $T_0 + D_0$ inclusively. These time spans are elements in the sequence X^s; let I_j be the set of indices in X^s of these scheduled time spans for processor M_j.

Note that $d_0 = x_{k_0}^s$, for some $k_0 \geq 0$. Furthermore, let d_j be the longest completed contract for problem P_j at time just before $T_j + D_j \leq T_0 + D_0$. Note that $d_j = x_{k_j}^s$, for some $k_j \geq 0$. Then by Lemma 2 we have $d_j < d_0$. The acceleration ratio for the problem P_j at D_j is given by

$$\frac{\sum_{i \in I_j} x_i^s}{x_{k_j}^s}$$

according to Observation 2, for $0 \leq j \leq m-1$. Hence, the worst case acceleration ratio that has occurred up to time $T_0 + D_0$ is at least

$$R_{m,n}(X) \geq \max_{0 \leq j \leq m-1} \left\{ \frac{\sum_{i \in I_j} x_i^s}{x_{k_j}^s} \right\} \geq \frac{\sum_{j=0}^{m-1} \sum_{i \in I_j} x_i^s}{\sum_{j=0}^{m-1} x_{k_j}^s}.$$

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 $A = \sum_{j=0}^{m-1} \sum_{i \in I_j} x_i^s$ contains as summands all x_i^s that have been completed up to time $T_0 + D_0$. In particular we know that A includes all x_j^s that are smaller than $x_{k_0}^s$, as Lemma 2 guarantees that any problem completed to a contract d_j smaller than a problem completed to a contract d_0 will complete the next time around before $T_0 + D_0$ and hence the summation given at time $T_0 + D_0$ contains all x_k's (i.e. d_j's) that are smaller than x_{k_0} (i.e. d_0). In other words, every element up to $x_{k_0}^s$ in the sorted schedule X^s appears in A. Since there are n problems all of which have been explored to lengths exceeding $x_{k_0}^s$ then we have that A contains all sorted values up to $x_{k_0}^s$ plus at least n larger values corresponding to the finished contracts in each of the n problems. The smallest choices for these n values are $x_{k_0+1}^s, \ldots, x_{k_0+n}^s$. Hence, we obtain

$$\sum_{j=0}^{m-1} \sum_{i \in I_j} x_i^s \geq \sum_{i=0}^{k_0+n} x_i^s.$$

Now consider the values $x_{k_j} = d_j$, for $1 \leq j \leq m-1$. 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 Corollary 2 d_0 is the largest time among the d_i's. The $m-1$ largest d_i values are $x_{k_0-m+1}^s, \ldots x_{k_0-1}^s$ and

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

Hence,

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

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 (Gal 1980) and Schuierer (Schuierer 2001) which we state here without proof and in a simplified form for completeness. 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

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

In particular, in our case it is easy to see that, if we set

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

then F_k satisfies all conditions of Theorem 2. Hence,

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

Note that if $a \leq 1$, then the above ratio tends to infinity as $k \to \infty$. Hence, we can assume that $a > 1$ and obtain

$$R_{m,n}(X) \geq \sup_{0 \leq k < \infty} \left\{ \frac{(a^{k+n+1} - 1)/(a-1)}{(a^{k+1} - a^{k-m+1})/(a-1)} \right\}$$

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

The above expression is minimized for $a = ((m+n)/n)^{1/m}$ and the acceleration ratio is bounded from below by

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

Theorem 3 *Given n problems and m processors every schedule simulating an interruptible algorithm using contract algorithms has an acceleration ratio not less than*

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

Optimal Schedule

We now observe that, by construction of the functional F_k, the schedule which explores contracts in round robin order at lengths $1, a, a^2, ...$ matches the lower bound for the acceleration ratio. In other words, the round robin and length-increasing schedule proposed by Bernstein *et al.* (Bernstein, Finkelstein, & Zilberstein 2003) 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 non-round robin schedules for the case when m is a multiple of n. More formally,

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

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

Conclusions

We resolved an open question in Bernstein *et al.* (Bernstein, Finkelstein, & Zilberstein 2003) regarding the optimal acceleration ratio of a schedule of contract algorithms for simulation of interruptible algorithms. This bridges the gap between the acceleration ratio of round robin and length-increasing and general schedules.

An open problem is to investigate whether the bounds generalize to randomized or average case schedules. In addition, while the optimal schedule presented here parallels search strategies in the case of p robots and m rays, the exact correspondence remains to be shown. Such a correspondence would extend the one established by Bernstein *et al.* (Bernstein, Finkelstein, & Zilberstein 2003) for round robin, length-increasing schedules and strategies.

References

Alpern, S., and 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):187.

Bernstein, D.; Perkins, T. J.; Zilberstein, S.; and Finkelstein, L. 2002. Scheduling contract algorithms on multiple processors. In *American Association for Artificial Intelligence Conference*, 702–706.

Bernstein, D. S.; Finkelstein, L.; and Zilberstein, S. 2003. Contract algorithms and robots on rays: Unifying two scheduling problems. In *IJCAI*, 1211–1217.

Dean, T., and Boddy, M. S. 1998. An analysis of time-dependent planning. In *American Association for Artificial Intelligence Conference*, 49–54.

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

Horvitz, E. 1987. Reasoning about beliefs and actions under computational resource constraints. In *Workshop on Uncertainty in Artificial Intelligence*, 301–324.

Horvitz, E. 1998. Reasoning under varying and uncertain resource constraints. In *American Association for Artificial Intelligence Conference*, 111–116.

Ibarra, O. H., and Kim, C. E. 1975. Fast approximation algorithms for the knapsack and sum of subset problems. *Journal of the ACM* 22(4):463–468.

Kao, M.-Y.; Ma, Y.; Sipser, M.; and Yin, Y. 1994. Optimal constructions of hybrid algorithms. In *Proc. 5th ACM-SIAM Sympos. Discrete Algorithms*, 372–381.

Kao, M.-Y.; Reif, J. H.; and Tate, S. R. 1993. Searching in an unknown environment: An optimal randomized algorithm for the cow-path problem. In *Proc. 4th ACM-SIAM Sympos. Discrete Algorithms*, 441–447.

López-Ortiz, A., and Schuierer, S. 2004. On–line parallel heuristics, processor scheduling and robot searching under the competitive framework. *Theoretical Computer Science* 310:527–537. Extended abstract appeared in 8th Scandinavian Workshop on Algorithms and Theory, SWAT, 2002, pp. 260-269.

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

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

Zilberstein, S.; Charpillet, F.; and Chassaing, P. 1999. Real-time problem-solving with contract algorithms. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, 1008–1015.