Contract Algorithms and Robots on Rays: Unifying Two Scheduling Problems

Daniel S. Bernstein
Dept. of Computer Science
University of Massachusetts
Amherst, MA 01003
bern@cs.umass.edu

Lev Finkelstein
Computer Science Department
Technion—IIT
Haifa 32000, Israel
lev@cs.technion.ac.il

Shlomo Zilberstein
Dept. of Computer Science
University of Massachusetts
Amherst, MA 01003
shlomo@cs.umass.edu

Abstract

We study two apparently different, but formally similar, scheduling problems. The first problem involves contract algorithms, which can trade off run time for solution quality, as long as the amount of available run time is known in advance. The problem is to schedule contract algorithms to run on parallel processors, under the condition that an interruption can occur at any time, and upon interruption a solution to any one of a number of problems can be requested. Schedules are compared in terms of acceleration ratio, which is a worst-case measure of efficiency. We provide a schedule and prove its optimality among a particular class of schedules. Our second problem involves multiple robots searching for a goal on one of multiple rays. Search strategies are compared in terms of time-competitive ratio, the ratio of the total search time to the time it would take for one robot to traverse directly to the goal. We demonstrate that search strategies and contract schedules are formally equivalent. In addition, for our class of schedules, we derive a formula relating the acceleration ratio of a schedule to the time-competitive ratio of the corresponding search strategy.

1 Introduction

In this paper, we demonstrate a connection between two problems that initially seem unrelated. The first involves computing solutions to multiple problems, under the condition that a solution to any one of the problems can be requested at any time. Challenges of this type arise in the design of intelligent user interfaces, information prefetching systems, and medical diagnosis systems. The second problem involves multiple robots searching an unknown environment for a goal. Problems of this nature arise in robotics and space exploration. In the following paragraphs we describe the problems, along with our contribution, in more detail.

The first problem concerns *anytime* algorithms [Horvitz, 1987; Dean and Boddy, 1988; Russell and Zilberstein, 1991], which produce solutions of different qualities depending on available computation time. More specifically, we focus on *contract* algorithms, which are anytime algorithms that require the deadline as input prior to the start of execution. With contract algorithms, no assumptions can be made about results produced before the given deadline. This is in contrast to the familiar *interruptible* algorithms, which can be queried at any point during execution. Although less flexible than interruptible algorithms, contract algorithms typically use simpler data structures, making them easier to implement and maintain. An example in AI is game playing programs based on heuristic search. For these programs, the allowed deliberation time is usually known in advance, and is used to set internal parameters. Another example is planning algorithms that perform state-space abstraction. With these algorithms, the run time can be controlled by setting the abstraction level at the start of execution.

Our problem can be stated as follows. We are given n instances of an optimization problem, along with a contract algorithm for the problem, and we have an m-processor machine on which to run the algorithm. An interruption can occur at any time, and a solution can be requested for any one of the problem instances. Given these constraints, we want a good general strategy for scheduling runs of the algorithm on the processors.

In the case of one problem instance and one processor, Russell and Zilberstein [1991] suggested iteratively doubling the contract lengths. With this schedule, for any interruption time t, the last contract completed (if one exists) is always of length at least $t/4$. This factor of four is the *acceleration ratio* of the schedule, a worst-case measure of its efficiency. Zilberstein *et al.* [1999] showed that no schedule can achieve an acceleration ratio less than four.

The generalization to multiple problem instances has been considered [Zilberstein *et al.*, 1999], as has the generalization to scheduling contracts on parallel processors [Bernstein *et al.*, 2002]. Optimal acceleration ratios have been derived in both cases. The more general multi-processor, multi-instance case has not previously been studied. In this paper, we provide a schedule for this case, and we prove that this schedule is optimal among a restricted, though still interesting, class of schedules. The optimality proof is a nontrivial extension of the previous proofs, and contains as a lemma a generalization of the monotone convergence principle.

This work is most closely related to Horvitz's continual computation framework [Horvitz, 2001]. In his framework,

as in ours, computation is performed with limited knowledge about the deadline or desired result. However, the assumptions underlying the two frameworks are different. In the continual computation framework, the limited knowledge comes in the form of probability distributions. In contrast, our use of acceleration ratio does not require probabilistic information. Furthermore, contract algorithms and parallel processing are both not considered in continual computation.

Our contract scheduling results can be directly applied to a robot search problem. In this problem, m robots search for a goal that is located on one of p intersecting rays. The aim is to minimize the *time-competitive ratio*, which is the worst-case ratio of the total time spent searching to the time for a single robot to traverse directly to the goal. The optimal ratio for the one-robot case was derived previously [Baeza-Yates *et al.*, 1993].

We address the general multi-robot case, for which search strategies are formally equivalent to contract schedules. For our class of schedules, we derive a formula relating the acceleration ratio of a schedule to the time-competitive ratio of the corresponding search strategy. The optimal time-competitive ratio is derived as a corollary.

This work is the first to draw a precise connection between contract scheduling and multi-robot search, and the first to provide nontrivial results for the multi-robot case. Kao *et al.* [1998] studied the multi-robot problem, but used a different performance measure. They minimized the *distance-competitive ratio*, which is the worst-case ratio of the total distance traveled during the search to the distance between the origin and the goal.

2 Scheduling a Contract Algorithm

2.1 Problem Description

An anytime algorithm A, when applied to an optimization problem instance i for time t, produces a solution of some real-valued quality $Q_A(t, i)$. The function Q_A is called A's *performance profile*. In general, one does not know an algorithm's performance profile. Nevertheless the concept of a performance profile is useful in reasoning about anytime algorithms. We assume that the performance profile of an anytime algorithm on any problem instance is defined for all $t \geq 0$ and is a nondecreasing function of t.

The distinctions among different types of anytime algorithms arise from different assumptions about which parameters are known prior to execution. When both t and i are known in advance, the algorithm is called *contract*. When only i is known in advance, the algorithm is called *interruptible*. For the case where both are unknown, we will say that the algorithm is *multi-interruptible*, because it acts like multiple interruptible algorithms running in parallel.

Suppose we have a contract algorithm A, which we can run on a machine with m processors. At some unknown deadline, a solution to one of n problem instances will be requested. This setup requires a multi-interruptible algorithm, which we can create by scheduling contracts in such a way that progress is continually made on each problem instance. Upon interruption and query, the result returned is that of the longest completed contract dedicated to the desired problem instance.

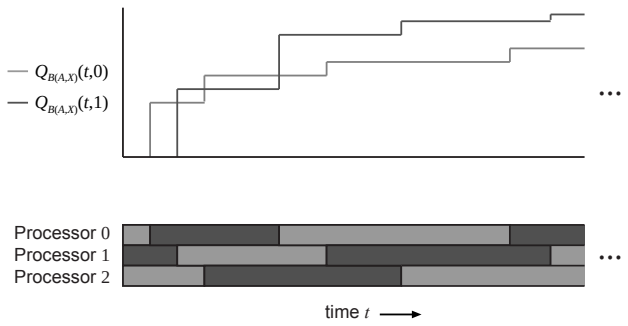

Figure 1: Scheduling a contract algorithm on three processors to create a two-problem multi-interruptible algorithm.

The multi-interruptible algorithm's performance profile depends on A's profile and the schedule, in a way to be formalized shortly. A particular schedule for the case of $m = 3$ and $n = 2$ is illustrated in Figure 1.

Formally, a *schedule* is a triple $X = \langle \{P_k\}, \{I_k\}, \{L_k\} \rangle$, where $P_k \in \{0, \ldots, m-1\}$ is the processor of contract k, $I_k \in \{0, \ldots, n-1\}$ is the problem instance worked on by contract k, and $L_k \in \mathbb{R}^+$ is the length of contract k. The contract index will be assumed to start at zero. A schedule must satisfy some minor conditions, which require a couple more definitions to state. First, let δ be defined as

$$\delta(x, y) = \begin{cases} 1 & \text{if } x = y \\ 0 & \text{otherwise} \end{cases} .$$

Next, for a given schedule X, the *completion time* of the kth contract is defined as

$$G_k = \sum_{i=0}^{k} \delta(P_i, P_k) L_i.$$

Note that although G depends on X, we omit the subscript. Dependence on the schedule will be made implicit throughout the paper for ease of notation. For every schedule, the index ordering must correspond to the completion time ordering. Furthermore, no two contracts may complete at the exact same time. The formal statement of these two conditions is that $k < l$ is equivalent to $G_k < G_l$ for all k, l. Also, a schedule must have $I_k = k$ for $0 \leq k \leq n-1$. This ensures that after time $t = G_{n-1}$, a solution is available for each problem instance.

To compare schedules, we use a worst-case metric called *acceleration ratio*. The acceleration ratio tells us how much faster our constructed algorithm would need to run in order to ensure the same quality as if the query time and problem were known, and a dedicated processor was assigned to producing a result. Intuitively, it measures how well a schedule handles the uncertainty about the problem instance and interruption time.

Before formally defining acceleration ratio, we must state some more technical details of the problem and present some more definitions. First, we take the view that when a contract completes at time t, its solution is available to be returned upon interruption at any time $\tau > t$. Second, we assume that

no interruptions occur until after time $t = G_{n-1}$, so that there is always at least one result available.

The length of the longest contract for instance i to complete before some time $t > G_{n-1}$ is

$$s(i,t) = \max\{L_k | G_k < t \text{ and } I_k = i\}.$$

The performance profile for the multi-interruptible algorithm $B(A,X)$ is defined for all i and all $t > G_{n-1}$ as

$$Q_{B(A,X)}(t,i) = Q_A(s(i,t),i).$$

We can now give a precise definition of acceleration ratio.

Definition 1 *The* acceleration ratio, $R_{m,n}(X)$, *is the smallest constant r for which $Q_{B(A,X)}(t,i) \geq Q_A(t/r,i)$ for all contract algorithms A, all problem instances i, and all times $t > G_{n-1}$.*

In the next section, we provide a schedule that is optimal within a restricted, though still interesting, class of schedules. We state below in precise terms the properties that delineate the class of schedules under consideration. Schedules having the three properties below will be called *cyclic* schedules.

The first property states that problem instances are completed in a round-robin manner. This seems sensible, as the desired problem instance is unknown. However, we cannot yet prove that for every non-problem-round-robin schedule, there is an equally good problem-round-robin schedule.

Property 1 (Problem-round-robin) *$I_k = k \bmod n$ for all k.*

The next property states that the lengths of contracts for each problem instance must increase with time. Given that performance profiles are nondecreasing, it seems that it would never be beneficial to use a schedule that doesn't satisfy this property. However, as with the first property, we cannot yet prove this. One difficulty lies in having to satisfy the problem-round-robin property mentioned above. We would like to be able to "remove" useless contracts from a schedule, but we have not found a way to do this while guaranteeing that the resulting schedule will be problem-round-robin.

Property 2 (Length-increasing) *For all k, l, if $I_k = I_l$ and $k < l$, then $L_k < L_l$.*

The final property states that processors return results in a round-robin manner. This property does not play a part in our lower bound derivation, but it is used in drawing a connection to the robot search problem. We introduce it at this point only for ease of exposition.

Property 3 (Processor-round-robin) *$P_k = k \bmod m$ for all k.*

We can prove a lemma that allows us to cast acceleration ratio in simpler terms when we are considering only cyclic schedules. In the proof, the following facts are established: acceleration ratio can be stated without reference to performance profiles; the only interruption times that need to be considered are completion times; and upon interruption, the result returned is from the contract with index exactly n less than index of the current contract.

Lemma 1 *For all cyclic schedules X,*

$$R_{m,n}(X) = \sup_k \frac{G_{k+n}}{L_k}.$$

Proof: We first argue that

$$R_{m,n}(X) = \max_i \sup_{t > G_{n-1}} \frac{t}{s(i,t)}.$$

From the definitions given above, we have

$$Q_A(s(i,t),i) = Q_{B(A,X)}(t,i) \geq Q_A(t/R_{m,n}(X),i),$$

for all i and $t > G_{n-1}$. Since this holds for any algorithm A, we can suppose an algorithm A with performance profile $Q_A(t,i) = t$ for all i. Thus $s(i,t) \geq t/R_{m,n}(X)$, and hence $R_{m,n}(X) \geq t/s(i,t)$ for all i and $t > G_{n-1}$. This implies that

$$R_{m,n}(X) \geq \max_i \sup_{t > G_{n-1}} \frac{t}{s(i,t)}.$$

To show that equality holds, assume the contrary and derive a contradiction with the fact that $R_{m,n}(X)$ is defined as the smallest constant enforcing the inequality between $Q_{B(A,X)}$ and Q_A.

Next we show that

$$\max_i \sup_{t > G_{n-1}} \frac{t}{s(i,t)} = \max_i \sup_{k \geq n} \frac{G_k}{s(i,G_k)}.$$

For each i, $s(i,t)$ is left-continuous everywhere and piecewise constant, with the pieces delimited by time points G_k. So for all i, $t/s(i,t)$ is left-continuous and piecewise linear and increasing. Thus, the local maxima of $t/s(i,t)$ occur at the points G_k; no other times may play a role in the supremum.

Finally,

$$\max_i \sup_{k \geq n} \frac{G_k}{s(i,G_k)} = \sup_k \frac{G_{k+n}}{L_k}$$

follows from the problem-round-robin and length-increasing properties, and the fact that no two contracts can finish at the exact same time. □

To conclude this section, we define the minimal acceleration ratio for m processors and n problems to be

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

where the infimum is taken over the set of cyclic schedules. In the following sections, we provide tight bounds for this ratio.

2.2 An Exponential Schedule

A simple approach to scheduling contract algorithms is to have the contract lengths increase exponentially. We consider the schedule

$$E = \langle k \bmod m, k \bmod n, ((m+n)/n)^{k/m} \rangle.$$

It is easily verified that this is a cyclic schedule. The following theorem gives an expression for this schedule's acceleration ratio.

Theorem 1 *The acceleration ratio for the exponential schedule is*

$$R_{m,n}(E) = \left(\frac{n}{m}\right)\left(\frac{m+n}{n}\right)^{\frac{m+n}{m}}.$$

Proof: Let $b = ((m+n)/n)^{1/m}$. The following is true for all k:

$$\begin{aligned}
\frac{G_{k+n}}{L_k} &= \frac{\sum_{i=0}^{k+n}\delta(P_i,P_k)L_i}{L_k} \\
&= \frac{\sum_{i=0}^{\lfloor (k+n)/m\rfloor} L_{(k+n) \bmod m+mi}}{L_k} \\
&= \frac{\sum_{i=0}^{\lfloor (k+n)/m\rfloor} b^{(k+n) \bmod m+mi}}{b^k} \\
&= \frac{b^{(k+n) \bmod m}\sum_{i=0}^{\lfloor (k+n)/m\rfloor} b^{mi}}{b^k} \\
&= \frac{b^{(k+n) \bmod m}\left(\frac{n}{m}\right)\left(b^{m(\lfloor (k+n)/m\rfloor+1)}-1\right)}{b^k} \\
&= \left(\frac{n}{m}\right)\frac{b^{k+m+n}-b^{(k+n) \bmod m}}{b^k} \\
&= \left(\frac{n}{m}\right)b^{m+n} - \left(\frac{n}{m}\right)b^{(k+n) \bmod m-k}.
\end{aligned}$$

Note that this expression is nondecreasing with k. Thus

$$\begin{aligned}
R_{m,n}(E) &= \lim_{k\to\infty}\left(\frac{n}{m}\right)b^{m+n} - \left(\frac{n}{m}\right)b^{(k+n) \bmod m-k} \\
&= \left(\frac{n}{m}\right)b^{m+n} \\
&= \left(\frac{n}{m}\right)\left(\frac{m+n}{n}\right)^{\frac{m+n}{m}}.
\end{aligned}$$

□

There are a few things to note about the ratio we just derived. As the number of processors approaches infinity, it tends to one. This is intuitive; by adding processors, we can get arbitrarily close to the omniscient algorithm. As the number of problems approaches infinity, the ratio tends to infinity. Finally, the ratio depends only on the ratio of problems to processors, and not on the absolute numbers.

We turn now to showing that no cyclic schedule can achieve a smaller ratio.

2.3 Lower Bound

We define a function to represent the sum of the lengths of all the contracts finishing no later than contract k finishes:

$$H_k = \sum_{i=0}^{k} L_i.$$

We can derive an inequality involving only the acceleration ratio and $\{H_k\}$.

Lemma 2 *For all cyclic schedules X and all k,*

$$H_{k+m+n} \le R_{m,n}(X)(H_{k+m} - H_k).$$

Proof: Consider the contract with index $k+m+n$. We define the set U such that $u \in U$ if and only if u is the index of the last contract on some processor to finish no later than contract $k+m+n$ finishes. It follows that $H_{k+m+n} = \sum_{u\in U} G_u$. Note that U contains at most m distinct integers, each between 1 and $k+m+n$. Since G is increasing,

$$\sum_{u\in U} G_u \le \sum_{i=1}^{m} G_{k+i+n}.$$

Using Lemma 1, we get

$$\begin{aligned}
\sum_{i=1}^{m} G_{k+i+n} &\le R_{m,n}(X)\sum_{i=1}^{m} L_{k+i} \\
&= R_{m,n}(X)\left(H_{k+m} - H_k\right).
\end{aligned}$$

□

To derive our lower bound, we also need the following lemma, which is a generalization of the monotone convergence principle. Its proof is deferred to the appendix.

Lemma 3 *Let m and n be relatively prime, and let $\{p_k\}$ be a sequence of real numbers that is bounded from below by 1. If $(p_k \cdots p_{k+m-1})^{n/m} \ge p_{k+m}\cdots p_{k+m+n-1}$ is satisfied for all k, then $\{p_k\}$ converges.*

We can now prove our lower bound theorem.

Theorem 2 *The optimal acceleration ratio for m processors and n problems is*

$$R^*_{m,n} = \left(\frac{n}{m}\right)\left(\frac{m+n}{n}\right)^{\frac{m+n}{m}}.$$

Proof: Consider an arbitrary cyclic schedule X. From Lemma 2, we have

$$R_{m,n}(X)(H_{k+m} - H_k) \ge H_{k+m+n},$$

and thus

$$R_{m,n}(X)\left(1 - \frac{H_k}{H_{k+m}}\right) \ge \frac{H_{k+m+n}}{H_{k+m}}$$

for all k.

Now let $v = \gcd(m,n)$. We define $m' = m/v$ and $n' = n/v$. Note that m' and n' are relatively prime. Further, let $p_k = H_{v(k+1)}/H_{vk}$. Note that $p_k > 1$ for all k. Then for all k,

$$\begin{aligned}
&R_{m,n}(X)\left(1 - \frac{1}{p_k\cdots p_{k+m'-1}}\right) \\
&= R_{m,n}(X)\left(1 - \frac{H_{vk}\cdots H_{v(k+m'-1)}}{H_{v(k+1)}\cdots H_{v(k+m')}}\right) \\
&= R_{m,n}(X)\left(1 - \frac{H_{vk}}{H_{vk+m}}\right) \\
&\ge \frac{H_{vk+m+n}}{H_{vk+m}} \\
&= \frac{H_{v(k+m'+1)}\cdots H_{v(k+m'+n')}}{H_{v(k+m')}\cdots H_{v(k+m'+n'-1)}} \\
&= p_{k+m'}\cdots p_{k+m'+n'-1}.
\end{aligned}$$

This implies that for all k,

$$R_{m,n}(X) \geq \frac{p_k \cdots p_{k+m'+n'-1}}{p_k \cdots p_{k+m'-1} - 1}.$$

There are two cases to consider.

Case 1: There exists some k' such that

$$(p_{k'} \cdots p_{k'+m'-1})^{\frac{n}{m}} \leq p_{k'+m'} \cdots p_{k'+m'+n'-1}.$$

Then we have

$$R_{m,n}(X) \geq \frac{(p_{k'} \cdots p_{k'+m'-1})^{\frac{m+n}{m}}}{p_{k'} \cdots p_{k'+m'-1} - 1}.$$

We are interested in how small $R_{m,n}(X)$ can be. Let $d = p_{k'} \cdots p_{k'+m'-1}$. Then

$$R_{m,n}(X) \geq \frac{d^{\frac{m+n}{m}}}{d-1}.$$

The value $d = (m+n)/n$ minimizes the right-hand side over the region $d > 1$. Substituting into the previous inequality, we find

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

Case 2: The inequality

$$(p_k \cdots p_{k+m'-1})^{\frac{n}{m}} > p_{k+m'} \cdots p_{k+m'+n'-1}$$

holds for all k.

Because $n/m = n'/m'$, we can apply Lemma 3 to show that $\{p_k\}$ converges, and hence $\lim_{k\to\infty} p_k$ is well defined. Then we have

$$\begin{aligned} R_{m,n}(X) &\geq \lim_{k\to\infty} \frac{p_k \cdots p_{k+m'+n'-1}}{p_k \cdots p_{k+m'-1} - 1} \\ &= \frac{\lim_{k\to\infty} p_k \cdots p_{k+m'+n'-1}}{\lim_{k\to\infty} p_k \cdots p_{k+m'-1} - 1} \\ &= \frac{(\lim_{k\to\infty} p_k)^{m'+n'}}{(\lim_{k\to\infty} p_k)^{m'} - 1}. \end{aligned}$$

We can proceed as in the previous case but with $d = \lim_{k\to\infty} p_k$. This gives

$$R_{m,n}(X) \geq \frac{d^{m'+n'}}{d^{m'} - 1}.$$

As in the previous case, we find that

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

Combining this inequality with Theorem 1, we get the desired result. □

3 Multi-Robot Search on Rays

The results from the previous section can be directly applied to a formally similar problem involving multiple robots searching for a goal. The problem is described as follows. Initially, m robots stand at the intersection of p rays (with $m < p$). The robots, all moving in a continuous fashion and at the same speed, search for a goal at an unknown location on one of the rays (see Figure 2). The search ends as soon as one of the robots finds the goal. Because there are not enough robots to cover all of the rays, finding a good search strategy is nontrivial. At this level of detail, the search problem may seem very different from the contract scheduling problem. Below we provide a more precise description of the search problem and show how the contract scheduling results can be applied.

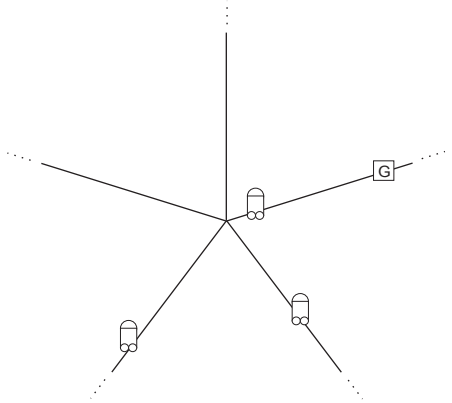

Figure 2: The search problem with three robots and five rays.

A search is an infinite sequence of search extents, or return trips departing from the origin. Formally, a *search strategy* is a triple $X = \langle\{P_k\}, \{I_k\}, \{L_k\}\rangle$, where $P_k \in \{0, \ldots, m-1\}$ is the robot executing search extent k, $I_k \in \{0, \ldots, p-1\}$ is the ray on which search extent k takes place, and $L_k \in \mathbb{R}^+$ is the length of search extent k. By adding the same conditions as in the contract scheduling case, we get an exact correspondence between schedules and search strategies. A *cyclic* search strategy is defined in the same way as a cyclic schedule.

A natural metric for the efficiency of a search strategy is the *time-competitive ratio*, the worst-case ratio of the total time spent searching to the time required for a single robot to traverse directly to the goal. We assume that the goal is not considered discovered until a robot actually moves beyond it. We assume also that the location of the goal is such that it cannot be found on the first search extent on any of the rays. This means that for each ray a, we consider only locations $t \geq L_a$.

Before formally defining time-competitive ratio, we need to introduce a new function and explain its use in the definition. Let us define $s(a,t)$ to be the index of the first search extent that goes past point t on ray a. This means that if the goal is located at point t on ray a, then it is found during search extent $s(a,t)$. The total search time up through extent $s(a,t)$ is $2G_{s(a,t)}$. (The factor of two results from search extents going out and back on rays.) However, since the search ends as soon as the goal is found, the last extent does not go to completion, and thus we must subtract out $2L_{s(a,t)} - t$. This leads us to the following formal definition for time-competitive ratio.

Definition 2 *The time-competitive ratio,*

$$T_{m,p}(X) = \max_a \sup_{t \geq L_a} \frac{2G_{s(a,t)} - (2L_{s(a,t)} - t)}{t}.$$

We can now give a formula for the relationship between acceleration ratio and time-competitive ratio for cyclic schedules and search strategies.

Theorem 3 *For all cyclic search strategies (schedules) X,*

$$T_{m,p}(X) = 2R_{m,p-m}(X') + 1,$$

where X' is the same as X, but with $I_k = k \bmod (p-m)$.

Proof: Because of the length-increasing and problem-round-robin properties, we know that the first search extent to pass a point on a ray comes p extents after the last extent on that same ray. Formally, we have that for all a and all $t \geq L_a$,

$$L_{s(a,t)-p} \leq t < L_{s(a,t)}.$$

Thus we have

$$\begin{aligned} T_{m,p}(X) &= \max_a \sup_{t \geq L_a} \frac{2G_{s(a,t)} - (2L_{s(a,t)} - t)}{t} \\ &= \max_a \sup_{t \geq L_a} \frac{2G_{s(a,t)} - 2L_{s(a,t)}}{t} + 1 \\ &= \max_a \sup_{t \geq L_a} \frac{2G_{s(a,t)} - 2L_{s(a,t)}}{L_{s(a,t)-p}} + 1. \end{aligned}$$

The last equation above expresses the intuitive notion that the worst place for the goal to be is just out of reach of one of the robots.

Now we have

$$\begin{aligned} &\max_a \sup_{t \geq L_a} \frac{2G_{s(a,t)} - 2L_{s(a,t)}}{L_{s(a,t)-p}} + 1 \\ &= \sup_k \frac{2G_{k+p} - 2L_{k+p}}{L_k} + 1 \\ &= \sup_k \frac{2(G_{k+p} - L_{k+p})}{L_k} + 1 \\ &= \sup_k \frac{2G_{k+p-m}}{L_k} + 1. \end{aligned}$$

The last equation requires some explanation. Because of the processor-round-robin property, the robot that performs search extent $k+p$ does so immediately after completing search extent $k+p-m$. Thus $G_{k+p} = G_{k+p-m} + L_{k+p}$.

Finally,

$$\begin{aligned} \sup_k \frac{2G_{k+p-m}}{L_k} + 1 &= 2 \sup_k \frac{G_{k+p-m}}{L_k} + 1 \\ &= 2R_{m,p-m}(X') + 1. \end{aligned}$$

□

We define the minimal time-competitive ratio for m robots and p rays as

$$T^*_{m,p} = \inf_X T_{m,p}(X),$$

where the infimum is taken over the set of cyclic search strategies. This has the following immediate consequence.

Corollary 1 $T^*_{m,p} = 2R^*_{m,p-m} + 1$.

4 Conclusion

In this paper, we addressed two apparently different scheduling problems, one involving contract algorithms, and the other involving robots searching on rays. For the contract scheduling problem, we provided a schedule and proved its optimality among the class of cyclic schedules. We further showed how contract scheduling results can be applied to the robot search problem, thus unifying the two problems.

A natural direction for future research is to study less restricted classes of schedules and search strategies. One intriguing question is whether lower acceleration ratios can be achieved with schedules that are not problem-round-robin or length-increasing. It would also be interesting to know whether the contract scheduling and robot search problems have similarities beyond those that result from using cyclic schedules and search strategies.

Acknowledgments

We thank William Hesse and László Babai for providing key insights into the proof of Lemma 3. We also thank Theodore Perkins and Charles Sutton for comments on earlier drafts. Support for this work was provided in part by the NSF under grant IIS-0219606 and by NASA under grants NAG-2-1394 and NAG-2-1463. Daniel Bernstein was supported by a NASA GSRP Fellowship. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not reflect the views of the NSF or NASA.

A Proof of Lemma 3

This section contains a proof of the central lemma used for our lower bound result. It is repeated below.

Lemma 3 *Let m and n be relatively prime, and let $\{p_k\}$ be a sequence of real numbers that is bounded from below by 1. If $(p_k \cdots p_{k+m-1})^{n/m} \geq p_{k+m} \cdots p_{k+m+n-1}$ is satisfied for all k, then $\{p_k\}$ converges.*

Proof: We first define $q_k = \ln p_k$. Since the convergence of $\{q_k\}$ implies the convergence of $\{p_k\}$, we can hereafter focus our attention on $\{q_k\}$.

We have

$$nq_k + \cdots + nq_{k+m-1} \geq mq_{k+m} + \cdots + mq_{k+m+n-1}.$$

Let us define

$$r_k = nq_k + \cdots + nq_{k+m-1} - mq_{k+m} - \cdots - mq_{k+m+n-1}.$$

Note that $r_k \geq 0$ for all k. Now let

$$\begin{aligned} s_k &= nq_k + 2nq_{k+1} + \cdots + mnq_{k+m-1} + \cdots + \\ &\quad 2mq_{k+m+n-3} + mq_{k+m+n-2}. \end{aligned}$$

A simple calculation shows that for all k, $r_k = s_k - s_{k+1}$. Because the r_k are all nonnegative, we have $s_k \geq s_{k+1}$ for all k. Since $\{p_k\}$ is bounded below by 1, $\{q_k\}$ is bounded from below, and hence $\{s_k\}$ is bounded below. By the monotone convergence principle, $\{s_k\}$ converges.

Thus we have

$$\begin{aligned} v + \epsilon_k &= nq_k + 2nq_{k+1} + \cdots + mnq_{k+m-1} + \cdots + \\ &\quad 2mq_{k+m+n-3} + mq_{k+m+n-2}, \end{aligned}$$

where $v = \lim_{k\to\infty} s_k$ and $\lim_{k\to\infty} \epsilon_k = 0$. This equation is a linear nonhomogeneous difference equation [Mickens, 1987]. The behavior of $\{q_k\}$ depends on the roots of the characteristic polynomial,

$$\begin{aligned} f(x) &= nx^{m+n-2} + 2nx^{m+n-3} + \cdots + \\ &\quad mnx^{n-1} + \cdots + 2mx + m. \end{aligned}$$

We know that $\{q_k\}$ is bounded below, and since $\{s_k\}$ converges, $\{q_k\}$ is also bounded above. Thus to demonstrate convergence, it suffices to show that f has no roots on the unit circle.

Consider the polynomial

$$g(x) = (x-1)^2 f(x) = nx^{m+n} - (n+m)x^n + m.$$

Clearly $z = 1$ is a root of g but not of f. To show that f has no roots on the unit circle, we will show that g has no roots on the unit circle other than $z = 1$.

We must show that for all z, if $g(z) = 0$ and $|z| = 1$, then $z = 1$. If $g(z) = 0$, then

$$nz^{m+n} + m = (n+m)z^n.$$

Adding the assumption that $|z| = 1$, we get

$$n + m = |n+m||z|^n = |(n+m)z^n| = |nz^{m+n} + m|.$$

In order for the first and last expressions to be equal, we must have $z^{m+n} = 1$. Substituting back into the original equality, we get

$$n + m = (n+m)z^n,$$

which implies that $z^n = 1$. It is a basic property of complex numbers that if $z^{m+n} = z^n = 1$ and $\gcd(m, n) = 1$, then $z = 1$.

We have thus shown that $\{q_k\}$ converges, and hence $\{p_k\}$ converges. □

References

[Baeza-Yates *et al.*, 1993] Ricardo Baeza-Yates, Joseph Culberson, and Gregory Rawlins. Searching in the plane. *Information and Computation*, 106:234–252, 1993.

[Bernstein *et al.*, 2002] Daniel S. Bernstein, Theodore J. Perkins, Shlomo Zilberstein, and Lev Finkelstein. Scheduling contract algorithms on multiple processors. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence*, 2002.

[Dean and Boddy, 1988] Thomas Dean and Mark Boddy. An analysis of time-dependent planning. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, 1988.

[Horvitz, 1987] Eric Horvitz. Reasoning about beliefs and actions under computational resource constraints. In *Workshop on Uncertainty in Artificial Intelligence*, 1987.

[Horvitz, 2001] Eric Horvitz. Principles and applications of continual computation. *Artificial Intelligence Journal*, 126(1-2):159–196, 2001.

[Kao *et al.*, 1998] Ming-Yang Kao, Yuan Ma, Michael Sipser, and Yiqun Yin. Optimal constructions of hybrid algorithms. *Journal of Algorithms*, 29:142–164, 1998.

[Mickens, 1987] Ronald E. Mickens. *Difference Equations*. Van Nostrand Reinhold Company, Inc., New York, NY, 1987.

[Russell and Zilberstein, 1991] Stuart J. Russell and Shlomo Zilberstein. Composing real-time systems. In *Proceedings of the Twelth International Joint Conference on Artificial Intelligence*, 1991.

[Zilberstein *et al.*, 1999] Shlomo Zilberstein, François Charpillet, and Philippe Chassaing. Real-time problem-solving with contract algorithms. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, 1999.