Scheduling Contract Algorithms on Multiple Processors

Daniel S. Bernstein, Theodore J. Perkins, Shlomo Zilberstein
Department of Computer Science
University of Massachusetts
Amherst, MA 01003
{bern,perkins,zilberstein}@cs.umass.edu

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

Abstract

Anytime algorithms offer a tradeoff between computation time and the quality of the result returned. They can be divided into two classes: contract algorithms, for which the total run time must be specified in advance, and interruptible algorithms, which can be queried at any time for a solution. An interruptible algorithm can be constructed from a contract algorithm by repeatedly activating the contract algorithm with increasing run times. The "acceleration ratio" of a schedule is a worst-case measure of how inefficient the constructed interruptible algorithm is compared to the contract algorithm. When the contracts are executed serially, i.e., on one processor, it is known how to choose contract lengths to minimize the acceleration ratio. We study the problem of scheduling contracts to run on m processors in parallel. We derive an upper bound on the best possible acceleration ratio for m processors, providing a simple exponential scheduling strategy that achieves this acceleration ratio. Further, we show that no schedule can yield a better acceleration ratio.

Introduction

In solving optimization problems, we are often faced with situations in which there is not enough time to determine an optimal solution. We desire approximation algorithms that can trade off computation time for quality of results. Algorithms with this property have been called *anytime algorithms*, and have been studied by researchers in artificial intelligence concerned with designing real-time systems (Horvitz, 1987; Russell & Zilberstein, 1991). Anytime algorithms have been designed for a range of problems, including planning (Dean & Boddy, 1988) and Bayesian inference (Wellman & Liu, 1994). Also, general-purpose search algorithms such as local search and simulated annealing are naturally viewed as anytime algorithms.

A useful distinction has been made between two types of anytime algorithms: *contract algorithms* and *interruptible algorithms*. Contract algorithms require that the total computation time be given in advance. Once activated, a contract algorithm may not produce a useful result until the prespecified amount of time has elapsed. This characteristic distinguishes them from interruptible algorithms, which do not need to know the deadline a priori. Contract algorithms can be easier to design because they have access to more information. Some problem-solving techniques that can be viewed as contract algorithms include depth-bounded heuristic search and solving continuous control problems by discretizing the state space. What is common to these techniques is that for a given contract time they can select parameters (e.g., the depth bound or the coarseness of the discretization) that limit the amount of computation so as to guarantee returning a solution within the available time. However, if a contract algorithm is given more time than it expects, it may have to be started from scratch with new parameters in order to improve upon its current result. Interruptible algorithms are generally more flexible and widely applicable than contract algorithms.

An interruptible algorithm can be formed by repeatedly running a contract algorithm with increasing contract lengths, returning the last result produced in the case of an interruption. In the case of serial execution of contracts, (Russell & Zilberstein, 1991) suggested the sequence of contract lengths: $1, 2, 4, 8, \ldots$. They showed that for any interruption time $t > 1$, the last contract completed is always of length at least $t/4$. This factor of four is the acceleration ratio of the schedule. In (Zilberstein et al., 1999), it was shown that no sequence of contracts on a single processor can reduce the acceleration to below four.

By scheduling the contract algorithm on parallel processors, it is possible to achieve an acceleration ratio of less than four. In this paper, we describe a simple exponential strategy for scheduling a contract algorithm on m processors. By analyzing this strategy, we derive an explicit formula for an upper bound on the optimal acceleration ratio in terms of m. This bound approaches 1 as m approaches infinity. Furthermore, we show that no schedule yields a better acceleration ratio, and thus the bound is tight. Finally, we discuss extensions to this work and the connection between our problem and a problem involving multiple robots searching for a point on multiple rays.

Scheduling a contract algorithm on multiple processors

An anytime algorithm A, when applied to an optimization problem instance for time t, produces a solution of some quality $Q_A(t)$. The function Q_A is called the *performance profile* of the algorithm A on the instance. In general, one does not know the performance profile of an algorithm on a problem instance. But 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 monotonically non-decreasing function of t.

We wish to construct an interruptible algorithm from a contract algorithm by scheduling a sequence of contracts on m processors in parallel. A schedule is a function $X : \{1, \dots, m\} \times \mathbb{N} \rightarrow \mathbb{R}$, where $X(i,j)$ is the length of the jth contract run on processor i. We assume, without loss of generality, that $X(1,1) = 1$ and that $X(i,j) \geq 1$ for all i and j.

A contract algorithm A along with a schedule X defines an interruptible algorithm B. When B is interrupted, it returns the best solution found by any of the contracts that have completed. Since we assume performance profiles are monotonic, this is equivalent to returning the solution of the longest contract that has completed. This is illustrated in Figure 1.

The algorithm B has a performance profile which depends on the profile of A and the schedule X. Before describing B's performance profile, we need to make a few definitions. We define the total time spent by processor i executing its first j contracts as:

$$G_X(i,j) = \sum_{k=1}^{j} X(i,k).$$

For a given time t, we define a function that specifies which contracts finish before that time:

$$\Phi_X(t) = \{(i,j) | G_X(i,j) < t\}.$$

We take the view that when a contract completes at time t, its solution is available to be returned upon interruption at any time $\tau > t$. The length of the longest contract to complete before time t is:

$$L_X(t) = \begin{cases} \max_{(i,j) \in \Phi_X(t)} X(i,j) & \text{if } \Phi_X(t) \neq \emptyset \\ 0 & \text{if } \Phi_X(t) = \emptyset \end{cases}$$

Thus, the performance profile for the interruptible algorithm B is

$$Q_B(t) = Q_A(L_X(t)).$$

We wish to find the schedule X that is optimal for a given number of processors m, independent of the particular contract algorithm being used or the problem being solved. We compare schedules based on their acceleration ratios, which is a measure similar to the competitive ratio for on-line algorithms (Sleator & Tarjan, 1985).

Definition 1 *The acceleration ratio, $R_m(X)$, for a given schedule X on m processors is the smallest constant r for which $Q_B(t) \geq Q_A\left(\frac{t}{r}\right)$ for all $t > 1$ and any contract algorithm A.*

The acceleration ratio tells us how much longer the interruptible algorithm has to work to ensure the same quality as the contract algorithm. The following lemma will be useful in the later proofs.

Lemma 1 *For all X, $R_m(X) = \sup_{t>1} \frac{t}{L_X(t)}$.*

Proof: By the definitions above, $Q_B(t) = Q_A(L_X(t)) \geq Q_A\left(\frac{t}{R_m(X)}\right)$ for all $t > 1$. Since this holds for any algorithm A, we can suppose an algorithm A with performance profile $Q_A(t) = t$. Thus $L_X(t) \geq \frac{t}{R_m(X)} \Rightarrow R_m(X) \geq \frac{t}{L_X(t)}$ for all $t > 1$. This implies $R_m(X) \geq \sup_{t>1} \frac{t}{L_X(t)}$. To show that equality holds, assume the contrary and derive a contradiction with the fact that $R_m(X)$ is defined as the smallest constant enforcing the inequality between Q_B and Q_A. □

We define the minimal acceleration ratio for m processors to be

$$R_m^* = \inf_X R_m(X).$$

In (Zilberstein et al., 1999), it was shown that $R_1^* = 4$. In the following sections, we provide tight bounds on this value for arbitrary m.

Upper bound

We first prove a lemma formalizing the idea that the worst time to interrupt the schedule is just as a contract ends.

Lemma 2 *For all X,*

$$\sup_{t>1} \frac{t}{L_X(t)} = \sup_{(i,j)\neq(1,1)} \frac{G_X(i,j)}{L_X(G_X(i,j))}.$$

Proof: $L_X(t)$ is left-continuous everywhere and piecewise constant, with the pieces delimited by the time points $G_X(i,j)$. For $t > 1$, $\frac{t}{L_X(t)}$ is piecewise linear, increasing, and left-continuous. Thus, the extrema of $\frac{t}{L_X(t)}$ can only occur at the points $G_X(i,j)$, $(i,j) \neq (1,1)$; no other points in time may play a role in the supremum. □

Theorem 1 $R_m^* \leq \frac{(m+1)^{\frac{m+1}{m}}}{m}$.

Proof: Consider the schedule $X(i,j) = (m+1)^{\frac{i-1+m(j-1)}{m}}$. Note that in the one-processor case this reduces to $X(i,j) = 2^{j-1}$. It is straightforward to show that for $(i,j) \neq (1,1)$

$$L_X(G_X(i,j)) = \begin{cases} X(i-1,j) & \text{if } i \neq 1 \\ X(m,j-1) & \text{if } i = 1 \end{cases}$$

Also, the following is true for all $(i,j) \neq (1,1)$:

$$\begin{aligned} G_X(i,j) &= \sum_{k=1}^{j} X(i,k) \\ &= \sum_{k=1}^{j} (m+1)^{\frac{i-1+m(k-1)}{m}} \\ &= (m+1)^{\frac{i-1-m}{m}} \sum_{k=1}^{j} (m+1)^k \\ &= (m+1)^{\frac{i-1-m}{m}} \left(\frac{(m+1)^{j+1} - (m+1)}{m} \right) \\ &< \frac{(m+1)^{\frac{i-1+mj}{m}}}{m}. \end{aligned}$$

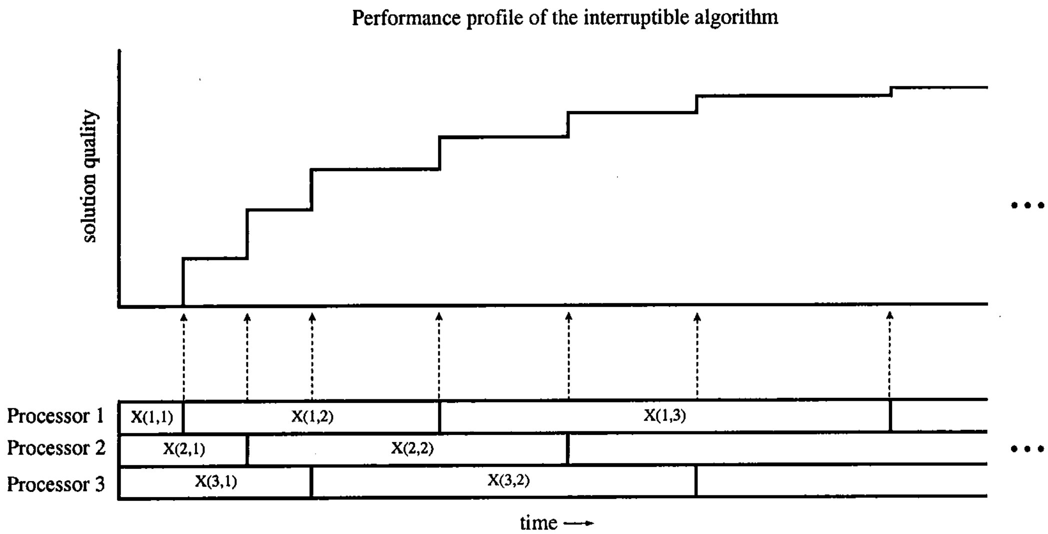

Figure 1: Constructing interruptible algorithm B by scheduling contract algorithm A on three processors.

So for all i, j such that $i \neq 1$,

$$\begin{aligned}\frac{G_X(i,j)}{L_X(G_X(i,j))} &= \frac{G_X(i,j)}{X(i-1,j)} \\ &< \frac{(m+1)^{\frac{i-1+mj}{m}}}{m(m+1)^{\frac{i-2+mj-m}{m}}} = \frac{(m+1)^{\frac{m+1}{m}}}{m},\end{aligned}$$

and for all i, j such that $i = 1$ and $j \neq 1$,

$$\begin{aligned}\frac{G_X(i,j)}{L_X(G_X(i,j))} &= \frac{G_X(i,j)}{X(m,j-1)} \\ &< \frac{(m+1)^{j}}{m(m+1)^{\frac{mj-m-1}{m}}} = \frac{(m+1)^{\frac{m+1}{m}}}{m}.\end{aligned}$$

Therefore

$$\begin{aligned}R_m^* &\leq R_m(X) \\ &= \sup_{(i,j)\neq(1,1)} \frac{G_X(i,j)}{L_X(G_X(i,j))} \leq \frac{(m+1)^{\frac{m+1}{m}}}{m}.\end{aligned}$$

□

Lower bound

In this section, it will be convenient to index contracts by their relative finish times. The following function counts how many contracts finish no later than the jth contract on the ith processor finishes. For a schedule X, let

$$\Psi_X(i,j) = |\{(i',j')|G_X(i',j') \leq G_X(i,j)\}|.$$

We assume w.l.o.g. that no two contracts can finish at exactly the same time—it is straightforward to show that any schedule that doesn't satisfy this condition is dominated by a schedule that does. This assumption guarantees that Ψ_X is one-to-one; it is also onto and thus an isomorphism. We refer to $\Psi_X(i,j)$ as the *global index* of the jth contract run on processor i.

We introduce a contract length function that takes as input a global index. For all i, j, let

$$Y_X(\Psi_X(i,j)) = X(i,j).$$

For notational simplicity, we will hereafter write Y in place of Y_X. We further define a finish time function that takes as input a global index:

$$G_Y(\Psi_X(i,j)) = G_X(i,j).$$

Given this definition and the definition of acceleration ratio, it follows that $G_Y(k+1) \leq R_m(Y)Y(k)$ for all k.

Finally, we define a quantity to represent the sum of the lengths of all the contracts finishing no later than contract k finishes:

$$G'_Y(k) = \sum_{l=1}^{k} Y(l).$$

Lemma 3 *For an arbitrary schedule, for all* $k \geq 1$,

$$G'_Y(k+m+1) \leq R_m(Y)(G'_Y(k+m) - G'_Y(k)).$$

Proof: We first relate G_Y and G'_Y. Consider the contract with global index $k+m+1$. $\sum_{l=1}^{m} G_Y(k+l+1)$ is the sum of the finishing times for the last m contracts to finish no later than contract $k+m+1$ finishes. $G'_Y(k+m+1)$ is the sum over all processors of the finish time for the last contract to finish *on that processor* no later than contract $k+m+1$ finishes. It is straightforward to show that $G'_Y(k+m+1) \leq \sum_{l=1}^{m} G_Y(k+l+1)$ (and they are equal if the last m contracts to finish include one from each processor). Furthermore

$$\begin{aligned}\sum_{l=1}^{m} G_Y(k+l+1) &\leq R_m(Y)\sum_{l=1}^{m} Y(k+l) \\ &= R_m(Y)\left(G'_Y(k+m) - G'_Y(k)\right).\end{aligned}$$

□

Theorem 2 $R_m^* = \frac{(m+1)^{\frac{m+1}{m}}}{m}$.

Proof: Let us define $H(k) = G'_Y(k+1)/G'_Y(k)$ for all $k \geq 1$. From Lemma 3, we have

$$G'_Y(k+m+1) \leq R_m(Y)(G'_Y(k+m) - G'_Y(k)),$$

and thus

$$R_m(Y)\left(1 - \frac{G'_Y(k)}{G'_Y(k+m)}\right) \geq \frac{G'_Y(k+m+1)}{G'_Y(k+m)},$$

so

$$R_m(Y)\left(1 - \frac{1}{H(k)\cdots H(k+m-1)}\right) \geq H(k+m).$$

We denote

$$H^*(k) = \max\{H(k), \ldots, H(k+m)\}.$$

There are two cases to consider. In the first case, there exists some $k' \geq 1$ such that $H^*(k') = H(k'+m)$. Then we have $H(k')\cdots H(k'+m-1) \leq H(k'+m)^m$, and

$$R_m(Y)\left(1 - \frac{1}{H(k'+m)^m}\right) \geq H(k'+m).$$

Thus

$$R_m(Y) \geq \frac{H(k'+m)^{m+1}}{H(k'+m)^m - 1}.$$

We are interested in how small $R_m(Y)$ can be. Let $C = H(k'+m)$. Suppose we minimize the right-hand side with respect to the only free variable, C, over the region $C > 1$. Setting the derivative to zero, we find

$$\begin{aligned}
\frac{d}{dC}\frac{C^{m+1}}{C^m - 1} &= \frac{(m+1)C^m}{C^m - 1} - \frac{C^{m+1}mC^{m-1}}{(C^m-1)^2} = 0 \\
&\Rightarrow\ (m+1)C^m(C^m - 1) - mC^{2m} = 0 \\
&\Rightarrow\ C^{2m} - (m+1)C^m = 0
\end{aligned}$$

The only solution is $C = (m+1)^{\frac{1}{m}}$. At the boundaries $C = 1$ and $C = \infty$, the value goes to infinity, so this solution is the one and only minimum. Substituting into the inequality, we find

$$R_m(Y) \geq \frac{(m+1)^{\frac{m+1}{m}}}{(m+1)^{\frac{m}{m}} - 1} = \frac{(m+1)^{\frac{m+1}{m}}}{m}.$$

In the second case, we have $H^*(k) \neq H(k+m)$ for all $k \geq 1$. Thus

$$\begin{aligned}
&H^*(k+1) \\
&\quad = \max\{H(k+1), \ldots, H(k+m), H(k+m+1)\} \\
&\quad = \max\{H(k+1), \ldots, H(k+m)\} \\
&\quad \leq H^*(k),
\end{aligned}$$

which means that the $H^*(k)$ form a non-increasing sequence. This sequence must be limited by 1, so

$$\lim_{k\to\infty} H^*(k) = D,$$

for some $D \geq 1$. Therefore $\lim_{k\to\infty} H^*(k)\cdots H^*(k+m-1) = \lim_{k\to\infty} H^*(k)^m = D^m$. Then

$$\begin{aligned}
&R_m(Y)\left(1 - \frac{1}{D^m}\right) \\
&= R_m(Y)\left(1 - \frac{1}{\lim_{k\to\infty} H^*(k-m)\cdots H^*(k-1)}\right) \\
&= R_m(Y)\left(1 - \frac{1}{\limsup_{k\to\infty} H(k-m)\cdots H(k-1)}\right) \\
&= \limsup_{k\to\infty} R_m(Y)\left(1 - \frac{1}{H(k-m)\cdots H(k-1)}\right) \\
&\geq \limsup_{k\to\infty} H(k) \\
&= \lim_{k\to\infty} H^*(k) \\
&= D.
\end{aligned}$$

Using the same analysis as in the previous case, we have that

$$R_m(Y) \geq \frac{(m+1)^{\frac{m+1}{m}}}{m}.$$

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

Discussion

We described a simple exponential strategy for scheduling contract algorithms on multiple processors to form an interruptible algorithm. In addition, we proved that this schedule achieves the minimal acceleration ratio among the set of all schedules.

In this work, we assumed no knowledge of the deadline or of the contract algorithm's performance profile. In (Zilberstein et al., 1999), the authors study the problem where the performance profile is known and the deadline is drawn from a known distribution. In this case, the problem of sequencing runs of the contract algorithm on one processor to maximize the expected quality of results at the deadline can be framed as a Markov decision process. It still remains to extend this work to the multiple processor case.

We note that the results presented in this paper are also applicable to a problem involving multiple robots searching for a goal on multiple rays. In this problem, k robots start at the intersection of m rays and move along the rays until the goal is found. An optimal search strategy is defined to be one that minimizes the *competitive ratio*, which is the worst-case ratio of the time spent searching to the time that would have been spent if the goal location was known initially. For $k \geq m$, the problem is trivial; the strategy that simply assigns one robot to each ray achieves a ratio of one. If $k < m$, however, robots may have to return to the origin so as not to neglect rays.

The problem with $k = 1$ and $m = 2$ is studied in (Ricardo et al., 1993), and it is shown that the optimal competitive ratio is 9. The general problem is briefly mentioned in (Kao et al., 1998), where a related problem is studied. It turns out that the analysis in this paper applies to the restricted case where $k = m - 1$. A sequence of contract lengths for a processor is analogous to a sequence of search extents for a robot, where a search extent is the distance a robot goes out on a ray before returning to the origin. It can be shown that the competitive ratio for a multi-robot schedule of search extents is $1 + 2r$, where r is the acceleration ratio for the schedule.

Acknowledgements

This work was supported in part by the National Science Foundation under grants IRI-9624992 and INT-9612092, and by NASA under grants NAG-2-1394 and NAG-2-1463. Daniel Bernstein was also supported by a National Science Foundation Graduate Fellowship. Theodore Perkins was supported by a graduate fellowship from the University of Massachusetts. 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.

References

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

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

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

Ricardo, B.-Y., Culberson, J. & Rawlins, G. (1993). Searching in the plane. *Information and Computation, 106*, 234–252.

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

Sleator, D. D. & Tarjan, R. E. (1985). Amortized efficiency of list update and paging rules. *Communications of the ACM, 28*, 202–208.

Wellman, M. & Liu, C.-L. (1994). State-space abstraction for anytime evaluation of probabilistic networks. In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence.*

Zilberstein, S., Charpillet, F. & Chassaing, P. (1999). Real-time problem-solving with contract algorithms. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence.*