Optimal Sequencing of Contract Algorithms Using Multiple Processors

Daniel S. Bernstein and **Shlomo Zilberstein**
Department of Computer Science
University of Massachusetts
Amherst, Massachusetts 01003
{bern,shlomo}@cs.umass.edu

May 7, 2001

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.

1 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 (Horvitz, 1987; Dean & Boddy, 1988; Russell & Zilberstein, 1991) concerned with designing real-time systems. Anytime algorithms are widely used in computer science. For instance, local search, simulated annealing, and genetic algorithms are all 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. 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. In real-time systems, the amount of time available for deliberation is often unknown ahead of time. Interruptible algorithms are generally more flexible and widely applicable than contract algorithms.

We consider the problem of constructing an interruptible algorithm using a contract algorithm. 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.

2 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 $X : \{1, \ldots, m\} \times \mathbb{N} \to \mathbb{R}$, 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.

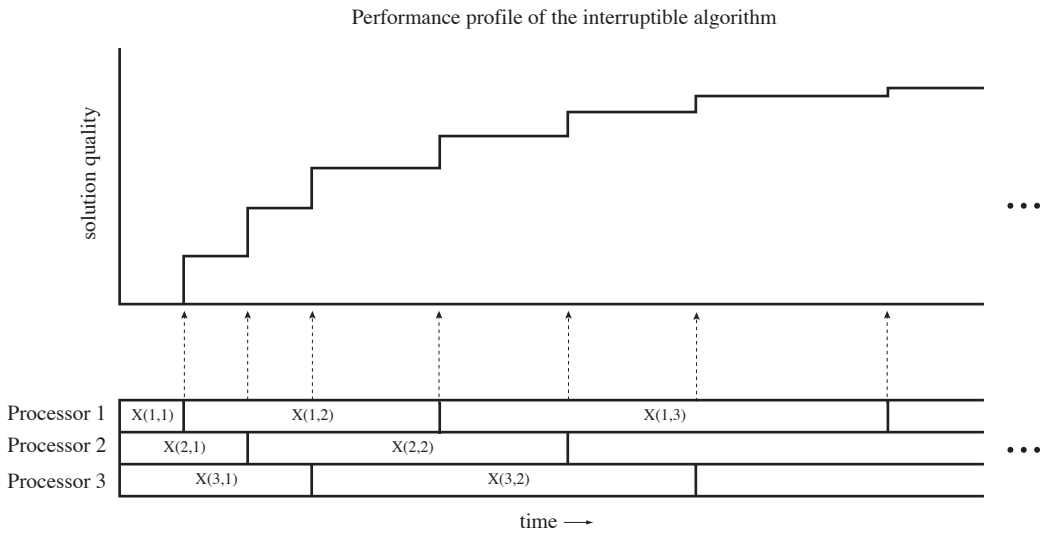

Figure 1: Constructing interruptible algorithm B by scheduling contract algorithm A on three processors.

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 next section, we provide an upper bound on this value for arbitrary m.

3 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 $(i,j) \neq (1,1)$:

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

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

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

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

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

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

Therefore

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

□

4 Discussion

We described a simple exponential strategy for scheduling contract algorithms on multiple processors to form an interruptible algorithm. We conjecture that this schedule achieves the smallest acceleration ratio among the set of all schedules, but this remains an open problem.

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.

Acknowledgements

This work was supported in part by the National Science Foundation under grants IRI-9624992 and INT-9612092, and by the National Aeronautics and Space Administration. Daniel Bernstein was also supported by a National Science Foundation Graduate Fellowship.

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.*

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.

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.*