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,shlomo}@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 run-time schedule is a worst-case measure of how inefficient the constructed interruptible algorithm is compared to the contract algorithm. The smallest acceleration ratio achievable on a single processor is known. Using multiple processors, smaller acceleration ratios are possible. In this paper, we provide a schedule for m processors and prove that it is optimal for all m. Our results provide general guidelines for the use of parallel processors in the design of real-time systems.

Introduction

The complex reasoning problems faced by both natural and artificial agents can rarely be solved exactly in time for the solution to be useful. Game-playing programs, trading/e-commerce agents, information retrieval systems, and medical diagnosis systems must all act under time pressure from their environments, from other agents, or from users with limited patience. A successful AI system must be able to use whatever time is available for deliberation to maximum advantage, and not miss opportunities or incur costs or disfavor through slow action.

Algorithms that produce solutions of different qualities depending on available computation time are called *anytime* algorithms (Horvitz 1987; Dean & Boddy 1988; Russell & Wefald 1991). A distinction can be made between two types of anytime algorithms. *Interruptible* algorithms, once started, can be interrupted at any time and queried for a solution. For example, local search approaches to optimization such as hill climbing and simulated annealing are interruptible algorithms. *Contract* algorithms need to be given the query time as input. These algorithms set internal parameters so that they produce a solution before the query time. A contract algorithm need not produce any solution before the query time arrives, and may have trouble utilizing spare time if it finds a solution quickly or if the query time is delayed. All other things being equal, interruptible algorithms are more convenient to use and more widely applicable than contract algorithms. But contract algorithms are often more intuitive to design, and typically use simpler data structures and control structures, making them easier to implement and maintain.

An interruptible algorithm can be formed from a contract algorithm by running a sequence of contracts of increasing lengths, returning the last solution produced when an interruption occurs. This sequencing problem has been solved informally in various domains. For example, Dechter and Rish (1998) develop the mini-bucket algorithm for approximate automated reasoning tasks, such as Bayesian inference. Their algorithm allows the user to specify bounds on the number of variables in any dependency that arises during execution (larger bounds lead to better solutions, at the expense of computation time), and they propose a method for scheduling a sequence of bounds so that solutions of improving quality are produced over time. In a similar vein, Munos and Moore (1999) propose various heuristics for incrementally refining discretizations of continuous-state optimal control problems. The incremental refinements produce a sequence of finite-state approximations which are increasingly complex to solve, but which provide solutions of increasing quality for the original, continuous-state problem.

We are interested in developing general contract schedules and providing formal justification for their use. In the case of serial execution of contracts, Russell and Zilberstein (1991) suggest the following sequence of contract lengths: $1, 2, 4, 8, \ldots$ They show 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, a worst-case measure of the loss due to the transformation from contract algorithm to interruptible algorithm. Zilberstein, Charpillet, and Chassaing (1999) show that no sequence of contracts on a single processor has an acceleration ratio of less than four. Only with multiple processors can a more efficient transformation be made.

In this paper, we propose a simple strategy for producing an interruptible algorithm by scheduling a contract algorithm on m processors in parallel. We analyze the strategy, deriving an explicit formula for its acceleration ratio as a function of m. Furthermore, we show that no schedule yields a better acceleration ratio for any m, thus the strategy is optimal. These results provide general guidelines for the use of parallel processors in the design of real-time systems. Finally, we discuss extensions to this work and a connection to a formally similar problem involving multiple robots searching multiple rays for a goal.

Scheduling a Contract Algorithm

An anytime algorithm A, when applied to an optimization problem instance for time t, produces a solution of some real-valued quality $Q_A(t)$. Q_A is called A's *performance profile* on the instance. Performance profiles are defined for both interruptible and contract algorithms. If A is interruptible, then $Q_A(t)$ is the quality of the solution returned by A upon interruption at time t. If A is contract, then $Q_A(t)$ is the quality of the solution returned by A after time t, given that t was specified *in advance*. In general, one does not know an algorithm's performance profile on a problem instance. 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.

An interruptible algorithm can be constructed from a contract algorithm by scheduling a sequence of contracts on m processors in parallel. A schedule is a function $X : \{1, \ldots, m\} \times \mathbb{N} \rightarrow \mathbb{R}$, 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.

We use B to denote the interruptible algorithm formed from the contract algorithm A and schedule X. When B is interrupted, it returns the best solution found by any of the contracts that have completed. Since we assume performance profiles are nondecreasing, this is equivalent to returning the solution of the longest contract that has completed. This is illustrated in Figure 1.

The interruptible algorithm's performance profile, Q_B, depends on A's profile and the schedule X. When B is interrupted, it has spent some time running contracts that are superseded by later results. Also, when contracts are interrupted, the time spent on them does not contribute to the final solution. In general, $Q_B(t) \leq Q_A(t)$, and often the inequality is strict.

We wish to find the schedule that is optimal for any given number of processors, with no assumptions about the query time or the contract algorithm's performance profile. The metric that we use to compare schedules is the *acceleration ratio*, which is a measure similar to the competitive ratio for on-line algorithms (Sleator & Tarjan 1985). The acceleration ratio tells us how much faster the interruptible algorithm would need to run in order to ensure the same quality as the contract algorithm for any interruption time.

Before formally defining acceleration ratio, we establish a formula for B's performance profile in terms of A's performance profile and the schedule X. 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 all times t, we define a function that specifies which contracts finish before t by

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

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 can now give a precise definition of acceleration ratio.

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

Although acceleration ratio is defined in terms of a property of solution quality, the following lemma allows us to formulate it without reference to Q.

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

Proof: From the definitions given above, we have $Q_B(t) = Q_A(L_X(t)) \geq Q_A(t/R(X))$ 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 $L_X(t) \geq t/R(X) \Rightarrow R(X) \geq t/L_X(t)$ for all $t > 1$. This implies $R(X) \geq \sup_{t>1} t/L_X(t)$. To show that equality holds, assume the contrary and derive a contradiction with the fact that $R(X)$ is defined as the smallest constant enforcing the inequality between Q_B and Q_A. □

Intuitively, the worst time to interrupt a schedule is just before a contract ends, because the contract time has been spent, but the solution is not yet available. The following lemma formalizes this notion and consequently enables us to consider only finishing times from here on.

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 $t > 1$, $t/L_X(t)$ is left-continuous and piecewise linear and increasing. Thus, the local maxima of $t/L_X(t)$ occur at the points $G_X(i,j)$, $(i,j) \neq (1,1)$; no other times may play a role in the supremum. □

We define the minimal acceleration ratio for m processors to be

$$R^*_m = \inf_X R(X).$$

Zilberstein, Charpillet, and Chassaing (1999) prove that $R^*_1 = 4$. In the following sections, we derive an expression for R^*_m for general m and provide a schedule that achieves this ratio.

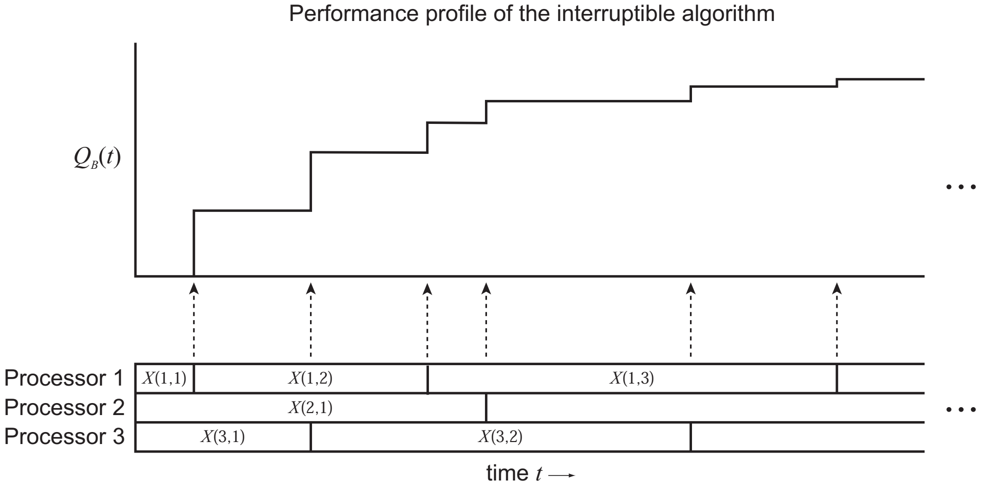

Figure 1: Constructing an interruptible algorithm by scheduling a contract algorithm on three processors.

An Exponential Schedule

A simple approach to scheduling a contract algorithm is to have the contract lengths increase exponentially. Russell and Zilberstein (1991) study the one-processor schedule $X(1,j) = 2^{j-1}$. We consider a generalization of this schedule to m processors in which contracts are assigned to processors in a round-robin fashion, with each contract being $(m+1)^{1/m}$ times longer than the previous one. Formally, the schedule is expressed as $E(i,j) = (m+1)^{(i-1+m(j-1))/m}$.

Theorem 1 $R(E) = \frac{(m+1)^{\frac{m+1}{m}}}{m}$.

Proof: It is straightforward to show that for $(i,j) \neq (1,1)$

$$L_E(G_E(i,j)) = \begin{cases} E(i-1,j) & \text{if } i \neq 1 \\ E(m,j-1) & \text{if } i = 1 \end{cases}.$$

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

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

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

$$\begin{aligned} \frac{G_E(i,j)}{L_E(G_E(i,j))} &= \frac{G_E(i,j)}{E(i-1,j)} \\ &= \frac{(m+1)^{\frac{i-1+mj}{m}}}{m(m+1)^{\frac{i-2+mj-m}{m}}} - \frac{(m+1)^{\frac{i-1}{m}}}{m(m+1)^{\frac{i-2+mj-m}{m}}} \\ &= \frac{(m+1)^{\frac{m+1}{m}}}{m} - \frac{(m+1)^{\frac{m-mj+1}{m}}}{m}, \end{aligned}$$

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

$$\begin{aligned} \frac{G_E(i,j)}{L_E(G_E(i,j))} &= \frac{G_E(i,j)}{E(m,j-1)} \\ &= \frac{(m+1)^j}{m(m+1)^{\frac{mj-m-1}{m}}} - \frac{1}{m(m+1)^{\frac{mj-m-1}{m}}} \\ &= \frac{(m+1)^{\frac{m+1}{m}}}{m} - \frac{(m+1)^{\frac{m-mj+1}{m}}}{m}. \end{aligned}$$

We see that the same expression is derived in each case. Note also that the expression is independent of i and increases with j. Thus,

$$\begin{aligned} R(E) &= \sup_{(i,j)\neq(1,1)} \frac{G_E(i,j)}{L_E(G_E(i,j))} \\ &= \lim_{j\to\infty} \frac{(m+1)^{\frac{m+1}{m}}}{m} - \frac{(m+1)^{\frac{m-mj+1}{m}}}{m} \\ &= \frac{(m+1)^{\frac{m+1}{m}}}{m}. \end{aligned}$$

□

Optimality of the Exponential Schedule

In this section, we show that the exponential schedule is optimal by proving that no schedule can achieve a smaller acceleration ratio. It is convenient to index contracts by their relative finishing 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 without loss of generality that no two contracts 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 a bijection. 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

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

We further define a finishing-time function that takes as input a global index:

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

Without loss of generality, we can assume that $Y_X(k) < Y_X(k+1)$ for all $k \geq 1$. Any schedule that doesn't have this property is dominated by one that does.

Lemma 3 *For all X and all $k \geq 1$,*

$$H_X(k+1) \leq R(X) Y_X(k).$$

Proof: This follows directly from the assumption above and the definition of acceleration ratio. □

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

$$H'_X(k) = \sum_{l=1}^{k} Y_X(l).$$

The following lemma expresses a property of this function.

Lemma 4 *For all X and all $k \geq 1$,*

$$H'_X(k+m+1) \leq R(X)(H'_X(k+m) - H'_X(k)).$$

Proof: Consider the contract with global index $k+m+1$. We define the set S such that $s \in S$ if and only if s is the global index of the last contract on some processor to finish no later than contract $k+m+1$ finishes. It follows that $H'_X(k+m+1) = \sum_{s \in S} H_X(s)$. Note that S contains at most m distinct integers, each between 1 and $k+m+1$. Since H_X is increasing,

$$\sum_{s \in S} H_X(s) \leq \sum_{l=1}^{m} H_X(k+l+1).$$

Using Lemma 3, we get

$$\begin{aligned} \sum_{l=1}^{m} H_X(k+l+1) &\leq R(X) \sum_{l=1}^{m} Y_X(k+l) \\ &= R(X) \left(H'_X(k+m) - H'_X(k) \right). \end{aligned}$$

□

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

Proof: Let us define $P(k) = H'_X(k+1)/H'_X(k)$ for all $k \geq 1$. From Lemma 4, we have

$$H'_X(k+m+1) \leq R(X)(H'_X(k+m) - H'_X(k)),$$

and thus

$$R(X) \left(1 - \frac{H'_X(k)}{H'_X(k+m)} \right) \geq \frac{H'_X(k+m+1)}{H'_X(k+m)},$$

so

$$R(X) \left(1 - \frac{1}{P(k) \cdots P(k+m-1)} \right) \geq P(k+m). \quad (1)$$

We let

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

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

$$R(X) \left(1 - \frac{1}{P(k'+m)^m} \right) \geq P(k'+m).$$

Thus

$$R(X) \geq \frac{P(k'+m)^{m+1}}{P(k'+m)^m - 1}. \quad (2)$$

We are interested in how small $R(X)$ can be. Let $c = P(k'+m)$. 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

$$\begin{aligned} \frac{d}{dc} \frac{c^{m+1}}{c^m - 1} &= \frac{(m+1)c^m}{c^m - 1} - \frac{c^{m+1} m c^{m-1}}{(c^m - 1)^2} = 0 \\ &\Rightarrow (m+1)c^m(c^m - 1) - mc^{2m} = 0 \\ &\Rightarrow c^{2m} - (m+1)c^m = 0. \end{aligned}$$

The only solution is $c = (m+1)^{1/m}$. At the boundaries $c = 1$ and $c = \infty$, the value goes to infinity, so this solution is the one and only minimum. Substituting into inequality (2), we find

$$R(X) \geq \frac{(m+1)^{\frac{m+1}{m}}}{(m+1)^{\frac{m}{m}} - 1} = \frac{(m+1)^{\frac{m+1}{m}}}{m}.$$

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

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

which means that the $P^*(k)$ form a nonincreasing sequence. Since $P^*(k) \geq 1$ for all k, the sequence must have a limit. We use d to denote this limit. By the definition of $P^*(k)$, it follows that $\limsup_{k \to \infty} P(k) = d$. Applying $\limsup_{k \to \infty}$ to both sides of inequality (1), we get

$$R(X) \left(1 - \frac{1}{\limsup_{k \to \infty} P(k) \cdots P(k+m-1)} \right) \geq \limsup_{k \to \infty} P(k+m).$$

Thus

$$R(X) \left(1 - \frac{1}{d^m} \right) \geq d.$$

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

$$R(X) \geq \frac{(m+1)^{\frac{m+1}{m}}}{m}.$$

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

Discussion

We studied the problem of constructing an interruptible algorithm by scheduling a contract algorithm to run on multiple processors. Our proposed schedule was shown to be optimal for any number of processors. As the number of processors increases, the optimal acceleration ratio approaches one and thus the distinction between contract and interruptible algorithms becomes less important. These results provide insight into the role of parallelism in the design of real-time systems.

In this work, we assumed no knowledge of the deadline or of the contract algorithm's performance profile. With problem-specific knowledge, more sophisticated scheduling strategies become appropriate. Zilberstein, Charpillet, and Chassaing (1999) consider the case where the performance profile is known and the deadline is drawn from a known distribution. In this case, the problem of scheduling a contract algorithm on a single processor to maximize the expected quality of results at the deadline can be framed as a Markov decision process. It remains to extend this work to the multiple processor case.

Another avenue for future research is to extend the contract algorithm model to include a broader class of algorithms. A number of algorithms have time complexity that scales with the values of input parameters, but in a fairly unpredictable way. For example, the worst-case time complexity of depth-bounded search scales with the depth bound and the maximum branching factor. However, the actual search time can be much less than the upper bound. Also, some contract-like algorithms are able to save information from one instantiation and use the information to accelerate the next instantiation. Finally, it may be interesting to consider contract algorithms that are themselves parallelizable, rather than just scheduling inherently sequential contracts on different processors. One challenge is to produce a contract algorithm model that takes these characteristics into account while still allowing for insightful analysis.

Finally, we note that the results presented in this paper may shed light on an abstract robotics problem involving multiple robots searching for a goal. In this problem, m robots start at the intersection of $m+1$ 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. The problem with $m = 1$ is considered in (Baeza-Yates, Culberson, & Rawlins 1993), where it is shown that the optimal competitive ratio is 9. It turns out that this problem is nearly identical to that of scheduling contracts on a single processor (Zilberstein, Charpillet, & Chassaing 1999). A contract schedule corresponds to a sequence of search extents for the robot, where a search extent is the distance the robot goes out on a ray before returning to the origin. If we let r denote the acceleration ratio for a schedule, then $1 + 2r$ is the competitive ratio for the corresponding sequence of search extents. In the multiprocessor case, each processor corresponds to a different robot. We conjecture that the same relationship between acceleration ratio and competitive ratio holds in this case.

Acknowledgments

We thank François Charpillet and Philippe Chassaing for fruitful discussions of this work. Support was provided in part by the National Science Foundation under grants ECS-0070102, ECS-9980062, INT-9612092, and IIS-9907331, 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.

References

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

Dean, T., and Boddy, M. 1988. An analysis of time-dependent planning. In *Proceedings of the Seventh National Conference on Artificial Intelligence*.

Dechter, R., and Rish, I. 1998. Mini-buckets: a general scheme for approximating inference. Technical report, Department of Information and Computer Science, University of California, Irvine.

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

Munos, R., and Moore, A. 1999. Variable resolution discretization for high-accuracy solutions of optimal control problems. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*.

Russell, S., and Wefald, E. H. 1991. *Do the Right Thing: Studies in Limited Rationality*. Cambridge, MA: MIT Press.

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

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

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