Real-Time Problem-Solving with Contract Algorithms

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

François Charpillet
LORIA-INRIA
B.P. 239
54506 Vandoeuvre-lès-Nancy
charp@loria.fr

Philippe Chassaing
Institut Elie Cartan-INRIA
B.P. 239
54506 Vandoeuvre-lès-Nancy
chassaing@iecn.u-nancy.fr

Abstract

This paper addresses the problem of building an interruptible real-time system using contract algorithms. Contract algorithms offer a tradeoff between computation time and quality of results, but their run-time must be determined when they are activated. Many AI techniques provide useful contract algorithms that are not interruptible. We show how to optimally sequence contract algorithms to create the best interruptible system with or without stochastic information about the deadline. These results extend the foundation of real-time problem-solving and provide useful guidance for embedding contract algorithms in applications.

1 Introduction

Since the mid 1980's, the AI community has produced a large body of work on anytime algorithms for solving such problems as real-time search, constraint satisfaction, planning and scheduling, and diagnosis (see, for example, the collection of papers in [Horvitz and Zilberstein, 1996]). Much of this work has focused on the meta-level control problem posed by anytime algorithms, that is, the problem of deciding when to stop deliberation and act on the best available solution. The resulting techniques have proved useful in addressing the high computational complexity of AI problems and the inherent uncertainty associated with AI problem-solving techniques.

Contract algorithms are a special type of anytime algorithms that require the amount of run-time to be determined prior to their activation. In other words, contract algorithms offer a tradeoff between computation time and quality of results, but they are not interruptible. Once activated with a particular contract time, a contract algorithm may not produce any useful result before the end of the contract.

Contract algorithms are easier to construct because they do not have to continually generate solutions of increasing quality. In fact, many existing AI problem-solving techniques produce contract rather than interruptible anytime algorithms. Examples include depth-bounded search or cost-bounded search techniques, design-to-time [Gravey and Lesser, 1993], approximate evaluation of relational database queries, and scheduling [Gallone and Charpillet, 1997]. What is common to these algorithms is that for any given contract time they select a set of parameters that limit the amount of search or computation so as to guarantee returning a solution within the available time. Another important class of systems that yields contract rather than interruptible algorithms includes systems composed of several interruptible or contract algorithms. Composition, even simple sequencing, destroys interruptibility; the resulting system is therefore a contract algorithm. Zilberstein [1995] shows how to divide the contract time among the components so as to maximize the overall quality of the solution for a given contract.

This paper addresses the problem of building an interruptible real-time system using contract algorithms. The key question is how to use prior knowledge about the performance of the algorithm and the deadline in order to produce an optimal sequence of solutions with a contract algorithm. This is done by activating the contract algorithm multiple times with different contracts. Section 2 provides a formal definition of the contract sequencing problem. Section 3 describes an optimal solution to the problem when no information is available about the deadline. Section 4 shows how to handle quality uncertainty and stochastic deadlines. Section 5 provides an optimal solution to the sequencing problem when a time-dependent utility function is given instead of a strict deadline. We conclude with a summary of the contribution of this work.

2 Using contract algorithms in interruptible domains

Suppose that a contract algorithm, $\mathcal{A}$, is used in an interruptible domain in which the amount of time available for problem solving is unknown in advance. At a particular time, the system receives a *deadline* signal indicating that the computation must be terminated and the best available result must be returned. Examples of interruptible domains include a diagnosis system in an intensive care unit; a data visualization program that

may be interrupted by its user; and a scheduling program that may need to return a new schedule once processors are ready to accept new tasks.

We use in this paper prior information about the performance of the contract algorithm. We begin with a simple form of performance profile [Dean and Boddy, 1988] and generalize it in Section 4.

Definition 1 *A* **performance profile**, $Q_{\mathcal{A}}(t)$, *of a contract algorithm* $\mathcal{A}$, *denotes the output quality as a function of contract time* t.

The performance profile is assumed to be a monotone increasing and continuous function of time.

$$t_1 < t_2 \Leftrightarrow Q_{\mathcal{A}}(t_1) < Q_{\mathcal{A}}(t_2)$$

Monotonicity is a standard property of anytime algorithms that can be guaranteed if the best result rather than the most recently generated one is returned. Strict monotonicity and continuity are assumed in order to simplify the analysis in Section 3.

If the amount of time available for computation is known in advance, the best strategy to maximize the quality of the result is to run the contract algorithm once giving it all the available time. What happens when we have no information (or only stochastic information) about the deadline? The contract algorithm should be activated with some contract time x_1. If the algorithm completes its execution before the deadline, it should be reactivated with a new contract x_2 and so on. Because of the monotonicity of $Q_{\mathcal{A}}$, it is never beneficial to use a short contract following a longer one. Therefore, we get the following sequence of contracts:

$$x_1 < x_2 < \cdots < x_i < x_{i+1} \cdots$$

Suppose that an interruptible anytime algorithm $\mathcal{B}$ is constructed using $\mathcal{A}$ with the sequence of contracts $X = (x_1, x_2, \ldots)$. Whenever $\mathcal{B}$ is interrupted, it should return the result obtained by the most recently completed contract. No solution is available before the termination of the first contract (which is arbitrarily small). Therefore, the performance profile of the interruptible algorithm, $\mathcal{B}$, is as follows.

$$Q_{\mathcal{B}}(t) = \begin{cases} 0 & \text{if } t < x_1 \\ Q_{\mathcal{A}}(x_1) & \text{if } x_1 \le t < x_1 + x_2 \\ \vdots & \\ Q_{\mathcal{A}}(x_i) & \text{if } \sum_{j=1}^{i} x_j \le t < \sum_{j=1}^{i+1} x_j \end{cases} \quad (1)$$

What is the sequence of contracts that produces the best anytime algorithm? To answer this question we must first formalize the notion of "best". If the deadline, d, is known in advance, the best quality $Q_{\mathcal{A}}(d)$ can be guaranteed. Suppose now that an interruptible algorithm is created by a sequence of contracts. We want the interruptible algorithm to guarantee the *same* quality as the contract algorithm, if it runs on a processor that is accelerated by a factor of $r \ge 1$. This definition follows the notion of *bounded optimality* defined by Russell, Subramanian and Parr [1993]. In fact, the results presented in Section 3 can be interpreted as the construction of a bounded-optimal interruptible algorithm from the contract one. Moreover, we prove that the minimal acceleration needed by *any* interruptible algorithm that matches the quality of $\mathcal{A}$ is 4. The acceleration ratio is defined as follows.

Definition 2 *Let* $\mathcal{A}$ *be a contract algorithm and* $\mathcal{B}$ *an interruptible algorithm produced by the sequence of contracts* $X = (x_1, x_2, ...)$, *then the* **acceleration ratio** *of* X, $r \ge 1$, *is the smallest constant* c *for which:*

$$\forall t \ge \frac{x_1}{c} : \quad Q_{\mathcal{B}}(ct) \ge Q_{\mathcal{A}}(t) \quad (2)$$

The acceleration ratio is the minimal acceleration that guarantees that the interruptible algorithm will always have a solution ready from the previous contract that is at least as good as the one produced by the contract algorithm without acceleration. The condition $t \ge \frac{x_1}{c}$ is needed because no solution is available before the termination of the first contract. Note that the acceleration ratio does not imply that the application *must* utilize a faster processor; it is only a performance measure of a sequence of contracts.

3 Optimal sequencing of contracts

Zilberstein and Russell [1996] show that a *particular* sequence of contracts requires an acceleration ratio of 4. The sequence of contracts is a geometric series with runtime being doubled at each activation. They prove the following theorem.

Theorem 1 *For any contract algorithm* $\mathcal{A}$, *an interruptible algorithm* $\mathcal{B}$ *can be constructed such that* $Q_{\mathcal{B}}(4t) \ge Q_{\mathcal{A}}(t)$.

In this section, we prove that the acceleration ratio of 4 is the best possible over any sequence of contracts. We also generalize the result to the case of multiple problem instances.

3.1 Solving a single problem instance

We first show that the construction proposed by Zilberstein and Russell is in fact optimal in the sense that it requires the minimal acceleration ratio. In general, the acceleration ratio must hold in the worst possible case: when the interruptible algorithm is stopped just before the end of the current contract x_{i+1} and it must return the result produced by the previous run (with contract time x_i). This leads to the following property.

Lemma 1 *Equation* (2) *is equivalent to*

$$\forall i \ge 1 : \quad Q_{\mathcal{A}}(x_i) \ge Q_{\mathcal{A}}(\sum_{j=1}^{i+1} x_j \ /c) \quad (3)$$

Proof: We first show that (2) $\Rightarrow$ (3). Consider some $i \ge 1$ and let

$$s(i) = \frac{1}{c} \sum_{j=1}^{i} x_j.$$

Let $t \to x_-$ denote the fact that t approaches x from below. Given that Condition (2) holds in particular for $t \to s(i+1)_-$, we get:

$$\lim_{t \to s(i+1)_-} Q_{\mathcal{B}}(ct) \geq \lim_{t \to s(i+1)_-} Q_{\mathcal{A}}(t)$$

From (1) we know that for any t, $s(i) \leq t < s(i+1)$:

$$Q_{\mathcal{B}}(ct) = Q_{\mathcal{A}}(x_i).$$

Therefore we obtain:

$$\forall i \geq 1: \quad Q_{\mathcal{A}}(x_i) \geq \lim_{t \to s(i+1)_-} Q_{\mathcal{A}}(t).$$

Finally, by the continuity of $Q_{\mathcal{A}}$:

$$\forall i \geq 1: \quad Q_{\mathcal{A}}(x_i) \geq Q_{\mathcal{A}}(s(i+1)),$$

that is:

$$\forall i \geq 1: \quad Q_{\mathcal{A}}(x_i) \geq Q_{\mathcal{A}}(\sum_{j=1}^{i+1} x_j \ /c).$$

The converse, (3) $\Rightarrow$ (2), is straightforward. Consider any t, $s(i) \leq t < s(i+1)$, we have

$$\begin{aligned} Q_{\mathcal{B}}(ct) &= Q_{\mathcal{A}}(x_i) \\ &\geq Q_{\mathcal{A}}(\sum_{j=1}^{i+1} x_j \ /c) \\ &\geq Q_{\mathcal{A}}(t). \end{aligned}$$ □

To summarize, in order to show that a particular sequence of contracts is optimal, it is sufficient to show that its acceleration ratio is the smallest required to satisfy Equation (3).

Theorem 2 *The minimal acceleration ratio needed to construct an interruptible algorithm from a given contract algorithm is $r = 4$.*

Proof: From Lemma 1 we know that for any sequence of contracts, $X = (x_1, x_2, ...)$, r must satisfy:

$$\forall i \geq 1: \quad Q_{\mathcal{A}}(\frac{x_1 + x_2 + \cdots + x_{i+1}}{r}) \leq Q_{\mathcal{A}}(x_i)$$

From the strict monotonicity of $Q_{\mathcal{A}}$ we get:

$$\forall i \geq 1: \quad \sum_{j=1}^{i+1} x_j \leq r x_i$$

Similarly,

$$\forall i \geq 1: \quad \sum_{j=1}^{i+2} x_j \leq r x_{i+1}$$

The difference between the last two equations gives:

$$\forall i \geq 1: \quad x_{i+2} \leq r(x_{i+1} - x_i) \tag{4}$$

The sequence $(x_1, x_2, ...)$ is an increasing sequence of positive numbers. Let ρ be defined as follows.

$$\rho = \inf\{\frac{x_2}{x_1}, \frac{x_3}{x_2}, ..., \frac{x_{i+1}}{x_i}, ...\}$$

It is clear that $\rho \geq 1$. From Equation 4 we obtain:

$$\forall i \geq 1: \quad x_{i+1} \geq x_i + \frac{x_{i+2}}{r},$$

and thus, for any $i \geq 1$ we have

$$\frac{x_{i+1}}{x_i} \ \geq \ 1 + \frac{x_{i+2}}{r x_i} \ \geq \ 1 + \frac{\rho^2}{r}.$$

Because $\{\forall i \ x_i \geq a\} \Rightarrow \{\inf_i x_i \geq a\}$, we conclude that

$$\rho \geq 1 + \frac{\rho^2}{r}.$$

Equivalently, because $\rho > 1$, we get

$$r \geq \frac{\rho^2}{\rho - 1}.$$

The function $f(\rho) = \frac{\rho^2}{\rho - 1}$ has a minimum of 4 (when $\rho = 2$) over the interval $(1, +\infty)$. Therefore, $r \geq 4$. □

3.2 Solving multiple problem instances

We now generalize the above result to the case in which a contract (or interruptible) anytime algorithm with performance profile $Q_{\mathcal{A}}$ is used to solve m independent problem instances. The goal is to maximize the *minimal* solution quality over all problem instances. When the deadline d is known in advance, the best strategy is to divide all the available time equally among the problem instances guaranteeing a minimal quality of $Q_{\mathcal{A}}(\frac{d}{m})$. The question is how to construct the best interruptible anytime algorithm when the deadline is unknown. As with a single task, we want to find the minimal acceleration needed to match the performance of the contract algorithm with a known deadline.

There are many situations in which multiple instances of a problem must be solved in real-time. Examples include search problems with multiple possible starting states; optimizing the code of several tasks to be executed on a parallel machine; and compression of a file divided into m blocks to be transmitted through m equivalent communication channels. In all these examples the objective is to maximize the minimal quality over all problem instances by the deadline. Another application is the continual computation model developed by Horvitz [1997] in which an interactive system uses idle time in order to solve possible future problems. When the user interrupts the computation by making a specific choice, the system responds using the best available solution for that choice.

For a given sequence of contracts $X = (x_1, x_2, ...)$, the performance profile of the interruptible algorithm solving m problem instances is as follows.

$$Q_{\mathcal{B}}(t) = \begin{cases} 0 & \text{if } t < x_1 + x_2 + ... + x_m \\ Q_{\mathcal{A}}(x_1) & \text{if } \sum_{j=1}^{m} x_j \leq t < \sum_{j=1}^{m+1} x_j \\ \vdots & \\ Q_{\mathcal{A}}(x_i) & \text{if } \sum_{j=1}^{i+m-1} x_j \leq t < \sum_{j=1}^{i+m} x_j \end{cases}$$

This is due to the fact that the first (and shortest) among the last m contracts defines the overall quality.

We want to find the minimal acceleration ratio c such that $Q_{\mathcal{B}}(ct)$ delivers a result at least as good as the one returned by $\mathcal{A}$ when the deadline is known (i.e., $Q_{\mathcal{A}}(\frac{t}{m})$).

Lemma 2 *For any $c \geq 1$ the condition:*

$$\forall t \geq \frac{x_1 + ... + x_m}{c} : \quad Q_{\mathcal{B}}(ct) \geq Q_{\mathcal{A}}(\frac{t}{m})$$

holds if and only if:

$$\forall i \geq 1 : \quad Q_{\mathcal{A}}(x_i) \geq Q_{\mathcal{A}}(\sum_{j=1}^{i+m} x_j \ /cm)$$

Proof: This is a straightforward generalization of Lemma 1. The complete proof is omitted.

Theorem 3 *The minimal acceleration ratio needed to construct an interruptible algorithm to solve m problem instances with a given contract algorithm is $r = (\frac{m+1}{m})^{m+1}$.*

Proof: The proof is similar to the proof of Theorem 2 and is given in Appendix A. Note that the ratio $(\frac{m+1}{m})^{m+1}$ is obtained by a sequence of contracts defined by a geometric series with run-times being multiplied by a factor of $\frac{m+1}{m}$. In other words:

$$X = (1, \frac{m+1}{m}, ..., (\frac{m+1}{m})^i, ...).$$

Therefore, the construction described in Zilberstein and Russell [1996] is the solution for the special case in which $m = 1$.

4 Handling quality uncertainty and stochastic deadlines

In this section we generalize the problem of sequencing contract algorithms to situations in which some information about the deadline is available. We assume that there is uncertainty about both the deadline and the quality produced by the contract algorithm. It is not surprising that there is no closed-form solution to the sequencing problem in this case. Instead, the optimal sequence of contracts is determined using dynamic programming. To be able to apply dynamic programming, the problem is formalized using a discrete representation of time (as opposed to the continuous representation used in the previous sections).

Definition 3 *A* **stochastic deadline** *is a probability distribution, $P_{\mathcal{D}}(t)$, of the deadline over time.*

A stochastic deadline is the prior probability that the deadline will occur at time t measured relative to the activation time of the contract algorithm. The representation of performance profiles is generalized as follows.

Definition 4 *A* **stochastic performance profile**, *$P_{\mathcal{A}}(q|t)$ of a contract algorithm, $\mathcal{A}$, denotes the probability of getting solution of quality q with contract time t.*

We assume that the quality of the result is observable. That is, once the contract algorithm completes its execution, the *actual* quality of the solution produced by the algorithm can be determined. The quality of a plan, for example, is observable if it is measured by the sum of the costs of the operators. However, when solution quality is defined as the *approximation ratio* with respect to the *optimal* solution it may not be observable. Hansen and Zilberstein [1996] have studied situations in which solution quality is partially observable. Their approach to the problem could be used to augment the framework presented in this paper.

Definition 5 *A* **utility function**, *$U(q)$, is the value of a solution of quality q produced by the contract algorithm.*

The contract sequencing problem in this case is defined as follows. Given a contract algorithm, its stochastic performance profile, a stochastic deadline and an arbitrary utility function, find an optimal policy to activate a sequence of contracts. An optimal policy is one that maximizes the expected utility of the available solution at the deadline.

We approach the problem by defining a corresponding Markov decision process (MDP) and finding an optimal policy by solving it. The states of the MDP are (q, t) indicating the availability of solution of quality q at time t. In addition, there is a terminal state associated with each solution quality q, indicating reaching the deadline with that quality. The action or decision taken at each non-terminal state is to activate the contract algorithm with a new contract time τ. The outcomes of these actions depend on the stochastic performance profile and the likelihood of reaching the deadline, both of which satisfy the Markov assumption (they are independent of previous states, given the current state of the computation).

Definition 6 *Let $P_{\mathcal{D}}(m|n)$ be the probability distribution of the deadline occurring at time m $(m > n)$ given that it has not occurred so far and the current time is n.*

Note that $P_{\mathcal{D}}(m|n)$ can be easily computed from the prior probability $P_{\mathcal{D}}(t)$ as follows.

$$P_{\mathcal{D}}(m|n) = P_{\mathcal{D}}(m)/(1 - \sum_{i=1}^{n} P_{\mathcal{D}}(i)) \qquad (5)$$

This can be easily obtained by observing that n stands for the proposition that the deadline did not occur at time points $\{1, ..., n\}$ and by applying Bayes' rule.

Definition 7 *The* **completion probability** *of a contract τ starting in state (q, t) is*

$$c(t, \tau) = \sum_{i \geq t+\tau} P_{\mathcal{D}}(i|t)$$

The completion probability is the probability that the contract algorithm will terminate before the deadline occurs.

We now define the following value function over states:

$$V(q,t) = \max_{\tau} \begin{cases} c(t,\tau) \ \sum_{q'} P_{\mathcal{A}}(q'|\tau) V(\max(q,q'), t+\tau) \\ + \ (1 - c(t,\tau)) \ U(q) \end{cases} \tag{6}$$

The value of a contract τ at state (q,t) depends on whether the deadline occurs during the contract or not. If the contract algorithm completes its execution (with probability $c(t,\tau)$), then the value depends on the new quality and time. Note that if the new quality is lower than the current solution quality, the better solution is used rather than the most recent one. If the deadline occurs, a final state is reached with a reward that depends on the quality of the existing solution. The value function is defined based on the best action τ in each state.

Theorem 4 *The contracts that maximize the value function defined in* Equation (6) *provide an optimal solution to the contract sequencing problem for a given stochastic deadline.*

Proof: Because of the one-to-one correspondence between the sequencing problem and the MDP and because the stochastic performance profile and the deadline satisfy the Markov assumption, it is obvious that an optimal policy for the MDP provides an optimal solution to the sequencing problem. □

Many existing algorithms can be used to solve Equation (6). Because this is a finite-horizon MDP with no cycles (time always moves forward), dynamic programming can be used to determine the best action for each state in one sweep of the state space. The policy can be constructed off-line once for a given deadline distribution, offering efficient reactive meta-level control. The selection of a time unit affects the size and complexity of the policy. We are confident that a coarse time unit and a compact policy are sufficient in practice for making good meta-level decisions.

5 Optimal sequencing with no strict deadline

In this section we consider the case in which there is no strict deadline. Instead, the value of the solution decreases over time as specified by a time-dependent utility function [Dean and Boddy, 1988].

Definition 8 *A* **time-dependent utility function**, $U(q,t)$, *is the value of a solution of quality q produced by the contract algorithm and returned at time t.*

The contract sequencing problem in this case is defined as follows. Given a contract algorithm, its stochastic performance profile and an arbitrary time-dependent utility function, find the best policy to activate a sequence of contracts. As in the previous section, an optimal policy maximizes the expected utility of the returned solution. Note that in this case the meta-level control problem is not only to determining the next contract, but also when to stop the computation and return the current best result.

Similar to the previous section, we approach the problem by defining a corresponding MDP with states representing the current quality and time. The action taken at each state, however, is either to terminate the computation and return the current solution of quality q or activate the algorithm with a particular contract τ. We define the following value function over states.

$$V(q,t) = \max_{d} \begin{cases} \sum_{q'} P_{\mathcal{A}}(q'|\tau) V(\max(q,q'), t+\tau) \\ \qquad \text{if } d\text{=continue with contract } \tau \\ U(q,t) \\ \qquad \text{if } d\text{=terminate the computation} \end{cases} \tag{7}$$

When a new contract is activated, the value is defined by the distribution of output quality and the contract time. If the computation is terminated, the value is the utility of returning a result of quality q at time t.

Theorem 5 *The policy that maximizes the value function defined in* Equation (7) *provides an optimal solution to the contract sequencing problem for a given time-dependent utility function.*

Proof: Again, this is an immediate result of the one-to-one correspondence between the optimal sequencing problem and the above MDP and because the stochastic performance profile satisfies the Markov assumption. □

Using dynamic programming, the optimal policy can be constructed off-line once for a given time-dependent utility function, offering an efficient, reactive meta-level control.

6 Conclusion

We have analyzed the problem of optimal sequencing of contract algorithms. The problem arises when there is uncertainty about the amount of time available for problem-solving with contract algorithms. When no prior information is available about the deadline, the best sequence can match the performance of the contract algorithm when it runs on a processor that is 4 times faster. No slower acceleration can guarantee that performance. This result is generalized to the case in which an interruptible system must solve multiple problem instances. When stochastic information is available about the deadline and about the performance of the contract algorithm, the optimal sequence of contracts can be constructed using dynamic programming. Finally, we solve the case in which no deadline is specified but the utility function is time-dependent.

Section 3 was inspired by the work of Baeza-Yates *et al.* [1993] on searching in the plane. The analogy between the two problems is not obvious, but once formalized, they present similar mathematical questions. In particular, Lemmas 1 and 2 in this paper provide a foundation to solving the problem of searching in the plane. Furthermore, the analytical work reported here offers a dramatic simplification of the proofs presented in [Baeza-Yates *et al.*, 1993].

Sections 4 and 5 build on a large body of work on meta-level control of computation by Dean and Boddy [1988], Horvitz [1988], Russell and Wefald [1991] and others. By and large, existing meta-level control mechanisms are myopic; the computation is terminated once no *single* computational step has positive value. One exception is the technique developed by Russell, Subramanian, and Parr [1993] for sequencing a set of rules given a stochastic deadline. Similar to their dynamic programming approach, our solution provides a globally optimal policy, taking into account future computations.

To summarize, we show how to optimally sequence contract algorithms so as to maximize their utility in interruptible domains. These results provide useful guidance for embedding contract algorithms in a wide range of practical applications.

Acknowledgments

This work was supported in part by the National Science Foundation under grants No. IRI-9624992 and IRI-9634938, by an NSF/INRIA Cooperative Research grant, and by the LIRE Cooperative Program at INRIA.

Appendix A: Proof of Theorem 3

From Lemma 2, for any sequence of contracts $X = (x_1, x_2, ...)$, r must satisfy:

$$\forall i \geq 1: \quad Q_{\mathcal{A}}(\frac{x_1 + x_2 + ... + x_{i+m}}{mr}) \leq Q_{\mathcal{A}}(x_i)$$

From the strict monotonicity of $Q_{\mathcal{A}}$ we get:

$$\forall i \geq 1: \quad \sum_{j=1}^{i+m} x_j \leq mrx_i.$$

Similarly:

$$\forall i \geq 1: \quad \sum_{j=1}^{i+m+1} x_j \leq mrx_{i+1}.$$

The difference between the last two equations gives:

$$\forall i \geq 1: \quad x_{i+m+1} \leq mr(x_{i+1} - x_i). \tag{8}$$

The sequence $(x_1, x_2, ...)$ is an increasing sequence of positive numbers. As before, let ρ be defined as:

$$\rho = \inf\{\frac{x_2}{x_1}, \frac{x_3}{x_2}, ..., \frac{x_{i+1}}{x_i}, ...\}.$$

It is clear that $\rho \geq 1$. From Equation 8 we obtain:

$$\forall i \geq 1: \quad x_{i+1} \geq x_i + \frac{x_{i+m+1}}{mr}$$

Therefore,

$$\begin{aligned} \frac{x_{i+1}}{x_i} &\geq 1 + \frac{1}{mr}\frac{x_{i+m+1}}{x_i} \\ &= 1 + \frac{1}{mr}\frac{x_{i+1}}{x_i}\frac{x_{i+2}}{x_{i+1}} \cdots \frac{x_{i+m+1}}{x_{i+m}} \\ &\geq 1 + \frac{\rho^{m+1}}{mr}. \end{aligned}$$

We conclude that:

$$\rho \geq 1 + \frac{\rho^{m+1}}{mr}.$$

Equivalently, because $\rho > 1$, we get:

$$r \geq \frac{\rho^{m+1}}{m(\rho - 1)}$$

The function $\rho \longrightarrow \frac{\rho^{m+1}}{\rho - 1}$ reaches its minimum on the interval $(1, +\infty)$ when $\rho = \frac{m+1}{m}$, therefore $r \geq (\frac{m+1}{m})^{m+1}$.

The ratio $(\frac{m+1}{m})^{m+1}$ can be obtained by a sequence of contracts defined by a geometric series with run-times being multiplied by a factor of $\frac{m+1}{m}$. Thus the best possible acceleration ratio is $r = (\frac{m+1}{m})^{m+1}$. □

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.

[Dean and Boddy, 1988] Thomas Dean and Mark Boddy. An analysis of time-dependent planning. *Seventh National Conference on Artificial Intelligence*, 49–54, 1988.

[Gallone and Charpillet, 1997] Jean-Michel Gallone and François Charpillet. Real-time scheduling with Neurosched. *13th International Conference on Tools with Artificial Intelligence*, 478–479, 1997.

[Gravey and Lesser, 1993] Alan Garvey and Victor Lesser. Design-to-time real-time scheduling. *IEEE Transactions on Systems, Man, and Cybernetics*, 23(6):1491–1502, 1993.

[Hansen and Zilberstein, 1996] Eric A. Hansen and Shlomo Zilberstein. Monitoring the progress of anytime problem-solving. *Thirteenth National Conference on Artificial Intelligence*, 1229–1234, 1996.

[Horvitz, 1988] Eric Horvitz. Reasoning under varying and uncertain resource constraints. *Seventh National Conference on Artificial Intelligence*, 111–116, 1988.

[Horvitz, 1997] Eric Horvitz. Models of continual computation. *Fourteenth National Conference on Artificial Intelligence*, 286–293, 1997.

[Horvitz and Zilberstein, 1996] Eric Horvitz and Shlomo Zilberstein, eds. Fall Symposium on Flexible Computation in Intelligent Systems. Technical Report FS-96-06, AAAI: Menlo Park, CA, 1996.

[Russell *et al.*, 1993] Stuart J. Russell, Devika Subramanian and Ron Parr. Provably bounded optimal agents. *Thirteenth International Joint Conference on Artificial Intelligence*, 338–344, 1993.

[Russell and Wefald, 1991] Stuart J. Russell and Eric H. Wefald. Principles of metareasoning. *Artificial Intelligence*, 49:361–395, 1991.

[Zilberstein, 1995] Shlomo Zilberstein. Optimizing decision quality with contract algorithms. *Fourteenth International Joint Conference on Artificial Intelligence*, 1576–1582, 1995.

[Zilberstein and Russell, 1996] Shlomo Zilberstein and Stuart J. Russell. Optimal composition of real-time systems. *Artificial Intelligence* 82(1-2):181–213, 1996.