Interruptible Algorithms for Multi-Problem Solving

Spyros Angelopoulos
Max Planck Institute for Computer Science
Saarbrücken, Germany
sangelop@mpi-inf.mpg.de

Alejandro López-Ortiz
Cheriton School of Computer Science
University of Waterloo
Waterloo, Canada
alopez-o@uwaterloo.ca

Abstract

In this paper we address the problem of designing an interruptible system in a setting in which n problem instances, all equally important, must be solved. The system involves scheduling executions of contract algorithms (which offer a tradeoff between allowable computation time and quality of the solution) in m identical parallel processors. When an interruption occurs, the system must report a solution to each of the n problem instances. The quality of this output is then compared to the best-possible algorithm that has foreknowledge of the interruption time and must, likewise, produce solutions to all n problem instances. This extends the well-studied setting in which only one problem instance is queried at interruption time. We propose a schedule which we prove is optimal for the case of a single processor. For multiple processors, we show that the quality of the schedule is within a small factor from optimal.

1 Introduction

A designer of real-time systems should anticipate the situation in which there are limitations on the available execution time. Applications such as medical diagnosis systems, automated trading systems and game-playing programs require that the system may be queried at any time during its execution, at which point a solution must be reported. *Anytime algorithms* occur precisely in such settings, namely when a computationally difficult problem is addressed under uncertain running time availability. Such algorithms will produce an output whose quality improves as a function of the available computation time. Anytime algorithms were introduced by [Horvitz, 1987; 1988] and [Dean and Boddy, 1988] and arise in central AI problems such as heuristic search, and planning under uncertainty.

[Russell and Zilberstein, 1991] distinguish between two main classes of anytime algorithms. On one hand, the class of *interruptible algorithms* consists of algorithms that can be interrupted at any point during their execution, and can always report their current (albeit not necessarily optimal) solution. On the other hand, the class of *contract algorithms* consists of algorithms which specify the exact amount of allowable computation time as part of their input. Such algorithms must terminate their execution before a meaningful solution can be produced. Interruptible algorithms offer more flexibility; in contrast, contract algorithms are typically much easier to design, implement and maintain.

A "black-box" method for converting any contract algorithm to its interruptible version consists of running a sequence of executions of the contract algorithm, of increasing execution times. More precisely, in the most general setting, we are presented with a set P of n problem instances which we want to solve, and a system of m identical processors on which we can schedule this sequence of contract algorithms. The goal is to devise an efficient schedule, that is a strategy that assigns each execution of a contract algorithm to a specific processor. In the standard setting, upon an interruption, a query for a problem in P is issued. The algorithm must then report the solution of the (completed) contract algorithm with the longest execution time for the queried problem. Naturally, these "lengths" or durations of completed contracts should be as large as possible, since the longer the execution time, the better the quality of the solution returned by the contract algorithm (and thus by the interruptible system as well).

The standard performance measure for a schedule of contract algorithms is the *acceleration ratio*. Informally, the measure describes the multiplicative increase in processor speed required for the schedule to compensate for the lack of knowledge of the interruption time. More formally, let $l_{p,t}$ denote the length of the largest contract for problem p completed by time t in the schedule. The acceleration ratio is then defined as $\sup_t \max_{p \in P} \frac{t}{l_{p,t}}$. This measure compares the quality of the solution returned by the schedule (that is, the quantity $l_{p,t}$) to an ideal, optimal algorithm that knows the interruption t in advance and dedicates a single processor in order to run a contract of length t for problem p.

Simulating interruptible algorithms by means of schedules of contract algorithms has been a topic of extensive study. [Russell and Zilberstein, 1991] were the first to present such an explicit simulation. For the case of one problem instance and a single processor, they provided a schedule based on iterative doubling of contract lengths for which the corresponding interruptible algorithm has acceleration ratio at most four. [Zilberstein *et al.*, 1999] showed that in this case the schedule is optimal, in the sense that no other schedule of better acceleration ratio exists.

[Zilberstein *et al.*, 1999] addressed the generalization of multiple problem instances (assuming a single available processor), and [Bernstein *et al.*, 2002] studied the generalization in which contracts for a single problem instance must be scheduled on a set of multiple processors. For both cases, optimal schedules are derived. [Bernstein *et al.*, 2003] addressed the problem in its full generality, namely the setting in which n problem instances are given and the schedule is implemented on m processors. In particular, they showed an upper bound of $\frac{n}{m}(\frac{m+n}{n})^{\frac{m+n}{m}}$ on the acceleration ratio; in addition, they showed that the schedule is optimal for the class of *cyclic* schedules. The latter is a somewhat restricted, but still very rich and intuitive class of schedules. This restriction was removed by [López-Ortiz *et al.*, 2006], who showed that this acceleration ratio is indeed optimal among all possible schedules. More recently, Angelopoulos *et al.* [Angelopoulos *et al.*, 2008] studied the setting in which interruptions are soft deadlines (instead of absolute ones), in the sense that the algorithm is allowed an additional window of time. Within this window, the algorithm may have the opportunity to complete the execution of a contract, or inititate a new one.

The central observation that motivates our work is that the acceleration ratio becomes problematic, as a performance measure, when at interruption time the algorithm is required to return a solution to *all* n *problems in* P instead of only to a specific queried problem. This arises, for instance, in systems that involve parallel executions of different heuristics (and at interruption time, the best heuristic is chosen). A similar setting has been considered in the context of portfolios of algorithms: see, e.g., the work of [Gomes and Selman, 2001]. Another example is a medical diagnostic system which must perform concurrent evaluations for a number of medical issues. Here, the decision of the expert may very well have to take into account all such evaluations.

In particular, suppose that $n > m$, and for the purposes of illustrating our argument, say that $n \gg m$. Note that it is not feasible for any ideal, optimal algorithm (that is, an algorithm with advance knowledge of the interruption time t) to schedule n contracts of length t to m processors, since $n > m$. In a sense, if we applied the acceleration ratio to this domain, we would compare the performance of an interruptible algorithm which is expected to make progress on all n problems to an algorithm which only makes optimal progress on at most m: such a comparison is simply not fair. This shortcoming was noticed by [Zilberstein *et al.*, 2003], who defined the acceleration ratio for the case of n problems and one processor as $\sup_t \max_{p \in P} \frac{t/n}{l_{p,t}}$. This measure describes then an even distribution of the processor time among the n problem instances for the optimal (offline) schedule.

In this paper we address the problem of designing interruptible algorithms using schedules of executions of contract algorithms, assuming that m identical processors are available, and n problem instances, all equally significant must be solved. We begin by considering measures alternative to the acceleration ratio (Section 2), and we propose the *deficiency* of a schedule as our measure of choice. In Section 3 we present a schedule whose deficiency is very small and rapidly decreasing in the ratio n/m (in contrast, the acceleration ratio of every schedule is known to approach infinity when $n/m \to \infty$). Moreover, for the case of a single processor ($m = 1$), we prove in Section 4 that our schedule is optimal.

Due to space limitations, throughout the paper we omit (or only sketch) certain technical proofs.

2 Problem formulation and comparison of measures

Consider an (infinite) schedule X of executions of contract algorithms (also called *contracts*). For an interruption time t we denote by $l(X, p, t)$ the length of the longest contract for problem $p \in P$ which is finished by time t in X (or simply $l_{p,t}$ when X is implied from context). We make the canonical assumption that at interruption time at least one contract per problem has already completed its execution.

We need to formalize the question: what is the best way to exploit the available resources (i.e., processors), in order to solve the set of problems P? Towards this end, consider a schedule Y, which, in contrast to X, is finite: more precisely, Y schedules n distinct contracts, one for each problem in P (if Y schedules more than one contract per problem, than we can transform Y to a schedule Y' which is at least as good as Y by keeping only the largest contract per problem that appears in Y). Each contract in Y is scheduled in one of the m processors in M. We require that Y is *feasible with respect to* t, in the sense that the *makespan* of Y, namely the total sum of contract lengths on the most loaded processor used by Y does not exceed t. Let $\mathcal{Y}_t$ denote the class of all schedules Y with the above properties. We will be calling Y an *offline solution* since it relies on advance knowledge of t.

Having defined $\mathcal{Y}_t$, we need a measure of how a schedule $Y \in \mathcal{Y}_t$ compares to X, which will also dictate which is the *best* schedule in $\mathcal{Y}_t$ compared to X. First observe that the acceleration ratio is not an appropriate measure for our setting, since under it the optimal offline schedule Y for interruption t dedicates all its resources to the contract that is worked on the least by X while failing to produce an answer for all other problems. This results in a large acceleration ratio which however does not truly reflect the quality of X (effectively, the optimal solution "cheats" by ignoring all but one problem instances, which is not acceptable in our setting).

Another alternative would be to compare the smallest contract completed by X to the smallest contract completed by Y, by time t. We will need some preliminary definitions first. Let S^t_X denote the set $\{l(X, p_i, t) | i \in [1, \ldots, n]\}$, and let $S^t_X(i)$ denote the i-th smallest element of S^t_X. Similarly, for $Y \in \mathcal{Y}_t$ let S^t_Y denote the set of n contracts in Y, and $S^t_Y(i)$ be the i-th smallest contract in Y, respectively (ties are resolved arbitrarily).

Formally, we define the *performance ratio of* X *with respect to* Y *at time* t as:

$$\mathit{perf}(X, Y, t) = \frac{\min_i S^t_Y(i)}{\min_i S^t_X(i)} = \frac{\min_i S^t_Y(i)}{S^t_X(1)}. \qquad (1)$$

The performance ratio of X at time t is then defined as $\mathit{perf}(X, t) = \max_Y \mathit{perf}(X, Y, t)$ where Y is a feasible

schedule. Lastly, the performance ratio of X is then defined as $\mathit{perf}(X) = \sup_t \mathit{perf}(X,t)$.

The first observation is that under this measure there exists an "optimal" offline schedule in which all contracts have the same length: here, by "optimal" we mean a schedule $Y \in \mathcal{Y}_t$ against which the performance ratio of X is maximized. Indeed given any offline schedule Y, consider any schedule Y' such that the length of all its contracts is $\min_i S^t_Y(i)$. Note that such a feasible Y' exists, since all contracts in Y are at least that long, and Y itself is feasible. Then it follows from the definition that $\mathit{perf}(X,Y,t) = \mathit{perf}(X,Y',t)$. We can show a close relationship of the performance ratio to the acceleration ratio for the standard setting (namely when we seek the solution only to the queried problem).

Lemma 1. *There is a schedule X such that for any arbitrary interruption time t*

$$\mathit{perf}(X,t) = \begin{cases} \frac{n}{m}\left(\frac{m+n}{n}\right)^{\frac{m+n}{m}} & \text{for } m \geq n \\ \frac{n}{m}\frac{1}{\lceil n/m \rceil}\left(\frac{m+n}{n}\right)^{\frac{m+n}{m}} & \text{for } m < n \end{cases}$$

Furthermore, X is optimal with respect to this measure.

Proof. Consider first the case $m \geq n$, then an optimal offline schedule consists of executing one contract of length t per problem, each on its own processor and hence $\mathit{perf}(X,t) = \frac{t}{\min_i S_X(i)}$. Note then that $\mathit{perf}(X) = \sup_t \mathit{perf}(X,t)$ is precisely the definition of the acceleration ratio, for which the results of Bernstein *et al.* and López-Ortiz *et al.* show the optimal value of $\frac{n}{m}\left(\frac{m+n}{n}\right)^{\frac{m+n}{n}}$.

On the other hand, if $m < n$ then every offline schedule Y is such that there exists at least one processor in which at least $\lceil n/m \rceil$ contracts are scheduled. As argued earlier, there exists an optimal offline schedule in which all contracts have the same length. It follows then that there is an optimal offline schedule with contract lengths equal to $t/\lceil n/m \rceil$. Therefore, $\mathit{perf}(X,t) = \frac{t/\lceil n/m \rceil}{\min_i S_X(i)}$ and $\mathit{perf}(X) = \sup_t \mathit{perf}(X,t)$ can then be described as $\frac{1}{\lceil n/m \rceil}$ times the expression of the acceleration ratio, and the lemma follows. Note that for the case of one processor and n problems, this matches the measure proposed by [Zilberstein *et al.*, 2003], as mentioned in the introduction. □

Interestingly, we can show that under this refined measure, the schedule in the proof of Lemma 1 is very efficient.

Lemma 2. *The performance ratio of the schedule of Lemma 1 is bounded by $2e$, if $n > m$, and by 4 , if $n \leq m$.*

Proof. In the case $m \geq n$, we have

$$\mathit{perf}(X) = \left(1+\frac{n}{m}\right)\left(1+\frac{m}{n}\right)^{\frac{n}{m}} \leq 2 \cdot 2 = 4.$$

In the case $m < n$ we have that

$$\mathit{perf}(X) \leq \left(1+\frac{m}{n}\right)\left(1+\frac{m}{n}\right)^{\frac{n}{m}} \leq 2 \cdot e = 2e.$$

In either case, $\mathit{perf}(X) \leq 2e$. □

Figure 1 shows a plot of $\mathit{perf}(X)$ for $n > m$ assuming for simplicity that m divides n. Note how the performance of the schedule is between 4 and e, and decreases rapidly as n/m increases. When $m \geq n$, $\mathit{perf}(X)$ decreases in a similar manner, as m/n increases, and takes values between 4 and 1.

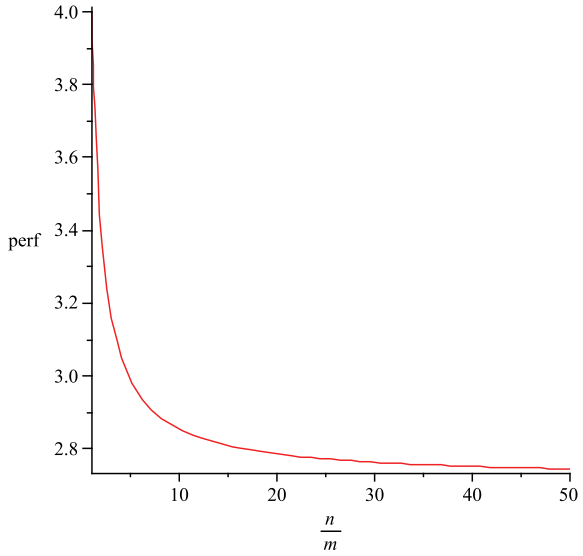

Figure 1: Plot of function $perf(X)$, assuming m divides n.

While the performance ratio seems a better candidate for a measure in the context of our problem, it is far from being the best possible choice. Note that according to this measure, every contract in the optimal offline solution may have fixed length, namely $t/\lceil n/m \rceil$. While it is guaranteed that the smallest contract in S^t_X indeed does not exceed $t/\lceil n/m \rceil$, the solution produced by X may be such that there exist several contracts in S^t_X which exceed this length. More formally, there may exist j such that $S^t_X(j) > S^t_Y(j)$ for the optimal offline solution Y. It then becomes difficult to argue that the optimal offline solution Y at time t is indeed better than the solution produced by the interruptible algorithm at time t, even though, supposedly, Y is optimal!

The above illustrates the need for defining a further measure, one which takes into account the intuitive expectation that "the optimal offline solution should perform better than the interruptible algorithm *on each problem*". To accomplish this, we allow the offline solution to observe the behavior of X at each point in time t, and then produce an appropriate "optimal" offline solution, tailored to the specifications of our problem. In a sense we allow the offline solution vast powers for choosing its schedule, while at the same time we still require it to produce solutions to all problem instances. This yields a measure which we call *deficiency*, and which is: i) consistent with the requirements of the problem; and ii) powerful enough, in the sense that if an algorithm performs well with respect to the new measure, then there are very strict guarantees about its performance.

Formally, we say that Y is *at least as good as* X, denoted by $Y \geq X$, if and only if $S^t_Y(i) \geq S^t_X(i)$, for all $i \in [1,n]$.

Then for a schedule $Y \in \mathcal{Y}_t$ with $Y \geq X$, we say that the *deficiency of X wrt Y* for interruption t is defined as

$$\mathit{def}(X, Y, t) = \min_i \frac{S_Y^t(i)}{S_X^t(i)} \tag{2}$$

The *deficiency of X given t* is then defined as

$$\mathit{def}(X, t) = \max_{Y \in \mathcal{Y}_t, Y \geq X} \mathit{def}(X, Y, t) \tag{3}$$

We define the deficiency of X simply as

$$\mathit{def}(X) = \sup_t \mathit{def}(X, t) \tag{4}$$

Not surprisingly, one can draw a connection between the problem of minimizing the deficiency of a schedule, and the problem of makespan minimization. In the latter, given m identical processors and n jobs $j_1, \ldots, j_n$, each of a certain size (length), the objective is to assign each job to a processor such that the maximum load among all processors, is minimized. The following lemma establishes this relation.

Lemma 3. *$def(X,t) = \frac{t}{OPT(S_X^t)}$, where $OPT(S_X^t)$ is the minimum makespan for scheduling n jobs in m processors, with job i having size (length) equal to $S_X^t(i)$.*

For an informal interpretation of the lemma, consider the set of the n largest contract lengths (one per problem), completed by X at time t. Then the largest $d > 1$ such that if we multiply each of the n contract lengths by d, there exists a schedule of the "blown-up" contracts in m processors that has makespan at most t, is the deficiency of X given t.

3 A near-optimal schedule for general m

In this section we propose an *exponential* schedule of very small deficiency. The overall schedule is as follows. Let $b > 1$ be a real number,

- problems are assigned to processors in a cyclic, round-robin order, i.e. the ith time a processor $j \in \{0, \ldots m-1\}$ executes a contract it does so on problem $(im + j) \bmod n$ and for a length of time b^{im+j}.
- the sequence of contract lengths is $\{b^i\}_{i=0,1,2,\ldots}$,
- the length of the k-th contract in this round-robin sequence is b^k for some $b > 1$.
- the value b is the *base* of the schedule,

The remainder of this section is devoted on finding the best value of b, i.e., the value that minimizes the deficiency of the exponential schedule. We denote by G_k the finish time of the k-th contract of X, in the cyclic order, whereas $L_k = b^k$ denotes the length of the k-th contract.

The following lemma shows that it suffices to evaluate the deficiency of any schedule (not necessarily exponential) at interruption times right before a contract terminates. Let G_c^- denote a time infinitesimally smaller than the finish time G_c of contract $c \in X$. Recall also that $OPT(\{\cdot\})$ denotes the optimal makespan for a schedule of a given set of jobs in m processors.

Lemma 4. *$\mathit{def}(X) = \sup_{c \in X} \frac{G_C^-}{OPT(S_X^{G_c^-})}$.*

In the context of an exponential schedule it is easy to see that Lemma 4 becomes

$$\mathit{def}(X) = \sup_{k \geq 0} \frac{G_{n+k}^-}{OPT(\{L_k, \ldots L_{n+k-1}\})} \tag{5}$$

where $L_i = b^i$ is the length of the i-th contract in the cyclic order.

The next lemma gives a lower bound on the optimal makespan, which will be useful in deriving an upper bound for the deficiency, as given by (5).

Lemma 5. *$OPT(L_k, \ldots L_{n+k-1}) \geq \kappa \cdot b^k \frac{b^{n+m-1} - b^{(n-1) \bmod m}}{b^m - 1}$, where $\kappa = \max\left\{\frac{1}{2-1/m}, \frac{b^m-1}{b^m}\right\}$.*

Proof. We will evaluate the makespan of the well-known greedy algorithm which schedules jobs of sizes $L_k, \ldots L_{n+k-1}$, in m identical processors numbered $0, \ldots m-1$, assuming the jobs are considered in this particular order (i.e., in increasing order of size). More precisely, the algorithm will assign each job to the machine of least current load. It is easy to see that the decisions of the greedy algorithm are such that the job of size L_{k+i} is scheduled on processor $i \bmod m$. Moreover, the incurred makespan is determined by the total load of jobs scheduled on the same processor as job L_{n+k-1}, namely processor $(n-1) \bmod m$. Denote by $Gr(L_k, \ldots L_{n+k-1})$ the makespan of the greedy algorithm. We obtain

$$\begin{aligned} Gr(L_k, \ldots L_{n+k-1}) &= \sum_{i=0}^{\lfloor (n-1)/m \rfloor} b^{k+mi+(n-1) \bmod m} \\ &= b^k b^{(n-1) \bmod m} \sum_{i=0}^{\lfloor (n-1)/m \rfloor} b^{mi} \\ &= b^k \frac{b^{n+m-1} - b^{(n-1) \bmod m}}{b^m - 1}. \end{aligned} \tag{6}$$

To complete the proof, we need to argue that the greedy scheduling policy does not achieve a makespan worse that $(1/\kappa)$ times the optimal makespan. Graham's fundamental theorem on the performance of the greedy scheduling policy [Graham, 1966] states that the greedy algorithm has an approximation ratio of $2 - 1/m$. Moreover, we know that the optimal makespan is at least b^{n+k-1}, which in conjunction with (6) yields that the greedy algorithm is also a $b^m/(b^m-1)$ approximation (for our specific instance). The lemma follows by combining the above two approximation guarantees. □

We now proceed to bound the deficiency of the exponential schedule X. It is easy to show that for exponential schedules, $G_{n+k} = \frac{b^{k+n+m} - b^{(k+n) \bmod m}}{b^m - 1}$. Let $\lambda = 1/\kappa = \min\left\{2 - \frac{1}{m}, \frac{b^m}{b^m-1}\right\}$. Combining with (5) and Lemma 5 we

obtain

$$\begin{aligned} def(X) &\leq \lambda \cdot \sup_{k\geq 0} \frac{b^{k+n+m} - b^{(k+n) \bmod m}}{b^k(b^{n+m-1} - b^{(n-1) \bmod m})} \\ &= \lambda \cdot \sup_{k\geq 0} \frac{b^{n+m} - (b^{(k+n) \bmod m})/b^k}{b^{n+m-1} - b^{(n-1) \bmod m}} \\ &\leq \lambda \cdot \frac{b^{n+m}}{b^{n+m-1} - b^{\gamma}}, \cdot \end{aligned} \quad (7)$$

where γ is defined as $(n-1) \bmod m$.

We thus seek the value of b that minimizes the RHS of (7). It is not clear how to do this analytically, since the factor λ depends on b, and the derivative of the RHS does not have roots that can be identified analytically. Instead, let $f(b)$ denote the function $b^{n+m}/(b^{n+m-1} - b^{\gamma})$. Is is easy to show that $f(b)$ is minimized for a value of b equal to $\beta = (n+m-\gamma)^{\frac{1}{n+m+\gamma-1}}$. Let ρ be such that $n-1 = \rho m + \gamma$, then $\beta = (m(\rho+1)+1)^{\frac{1}{m(\rho+1)}}$. Observe also that $f(b) = 1/(b^{-1} - b^{\gamma-n-m}) = 1/(b^{-1} - b^{-m(\rho+1)-1})$. Summarizing, for $b = \beta$ we obtain a schedule of deficiency

$$def(X) \leq \min\left\{2 - \frac{1}{m}, \frac{\beta^m}{\beta^m - 1}\right\} \cdot \frac{1}{\beta^{-1} - \beta^{-m(\rho+1)-1}}. \quad (8)$$

Figure 2 shows a plot of the deficiency of the schedule (or, more accurately, the RHS of (8) for the interesting case where $n > m$) as function of m and ρ. Note that for fixed m, the deficiency increases as a function of n, until a point at which it becomes relatively stable with n (this can be explained by the factor λ that is the minimum of two functions, one of which does not depend on n). For large m, and even larger n, the deficiency is close to 2. From the plot, the maximum value of deficiency is $3/8 \cdot 5^{5/4} \sim 2.803$.

We will now provide an analytical bound on the maximum value of the deficiency, by showing that $def(X)$ is at most 3.74 when $n > m$ and at most 4 when $n \leq m$. Substituting $m(\rho+1)$ with y in (8) we obtain

$$def(X) \leq \left(2 - \frac{1}{m}\right) \frac{1}{(y+1)^{-\frac{1}{y}} - (y+1)^{-\frac{y+1}{y}}}. \quad (9)$$

Using standard calculus it is straightforward to show that the functions $(y+1)^{-1/y}$ and $(y+1)^{-\frac{y+1}{y}}$ are increasing and decreasing functions of y, respectively. Therefore, the denominator of the RHS of (9) is an increasing function of $y = m(\rho+1)$. It turns out that for $n > m$ this upper bound on $def(X)$ is maximized when $m = 2$ and $\rho = 1$, hence $def(X) \leq 3.74$. When $n \leq m$, the $(2 - 1/m)$ factor vanishes, since the greedy schedule achieves optimal makespan (in other words, $\lambda = 1$). In this case $def(X)$ is at most 4. These upper bounds are not tight as the figure below shows. The plot shows tight numerical bounds for the plotted ranges of n and m. The analytical method in contrast gives an exact but not tight upper bound for all n and m. The two results differ slightly because the true exact maximum of the function could not be found analytically.

Note that the strategy proposed in this work outperforms known strategies proposed for other settings. For example, the optimal schedule for the acceleration ratio of [Bernstein *et al.*, 2003; López-Ortiz *et al.*, 2006] has unbounded deficiency for $m \gg n$. When $n > m$, this schedule has constant deficiency, but larger than ours (4.24 in the worst case). Moreover, it is provably non-optimal for $m = 1$ as it does not achieve the bounds of Theorem 9 in Section 4. For a single problem and processor, Eq. (7) becomes $def(X) \leq b^2/(b-1)$, which equals the acceleration ratio of exponential schedules with base b, as expected.

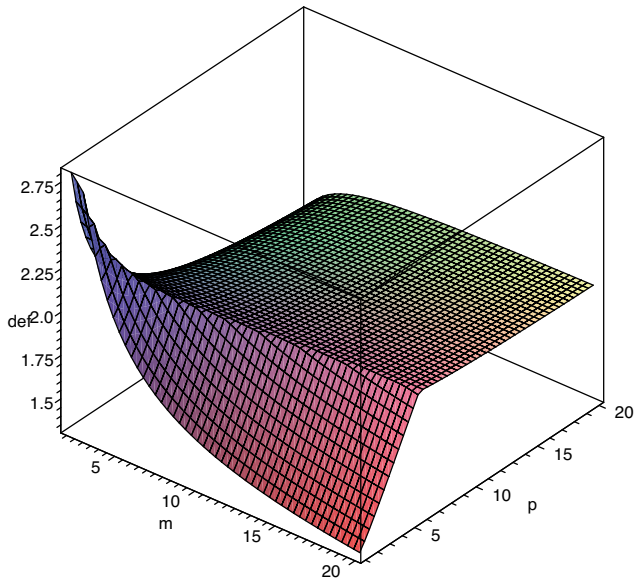

Figure 2: Plot of $def(X)$ as a function of m and ρ.

4 Optimality for the case of a single processor

In this section we show that the exponential schedule we described in Section 3 is optimal for the case of a single processor. Denote by X any given schedule of contracts on a single processor. For a fixed problem $p \in P$, we will denote by lower-case d and upper case D the lengths of consecutively completed contracts for problem p in X, respectively. More formally, if (p_i, d_i) is a contract in X, then the next contract for p_i scheduled after (p_i, d_i) will be denoted by (p_i, D_i).

We begin the proof by introducing some "normalization" in the schedules. In particular, the following technical lemma demonstrates that any schedule (hence the optimal schedule as well) can be transformed to a schedule of no worse deficiency, which also observes certain useful properties. Ideally, we would like the resulting schedule to have as simple a structure as a cyclic (or even better, an exponential) schedule, because the analysis would then become very easy. However, this not possible in general. Instead, we derive schedules which although not as "canonical" as cyclic schedules, are much more amenable to analysis than the original schedule (which may observe no properties at all). We call the resulting schedules *normalized*. Similar normalization reductions have

been applied in [López-Ortiz *et al.*, 2006] and [Angelopoulos *et al.*, 2008].

Lemma 6. *For every schedule of contracts X in a single processor, there exists a schedule X' such that $def(X') \leq def(X)$, and furthermore X' satisfies the following properties: Let (p_i, d_i) and (p_j, d_j) denote two contracts in X', with $i \neq j$. Let also T_i and T_j denote the start times of contracts (p_i, D_i) and (p_j, D_j) in X'. Then*

(i) If $d_i < d_j$ then $T_i + D_i < T_j + D_j$, which also implies that $T_i < T_j$.

(ii) If $d_i < d_j$ then $D_i < D_j$.

Proof sketch. If X is not normalized, then we can transform it to a normalized one by defining an appropriate series of *contract swaps*, similar to the ones of [López-Ortiz *et al.*, 2006]. These swaps introduce the normalized proprties and do not worsen the deficincy of the original schedule. □

In the next step, we establish a lower bound on the deficiency of any schedule, as a function of contract lengths.

Lemma 7. *For any normalized schedule X for n problems on a single processor, we have*

$$def(X) \geq \sup_{k \geq 0} \left\{ \sum_{i=0}^{k+n} x_i^s \Big/ \sum_{i=0}^{n-1} x_{k+i}^s \right\}, \qquad (10)$$

where $X^s = (x_0^s, x_1^s, \ldots)$ is the sequence of contract lengths in X, in increasing order, and $x_i^s := 0$ if $i < 0$.

Proof. Suppose that at time T_0, a contract C_0 for problem $p_0 \in P$ is about to start in X; let D_0 denote the length of this contract. Denote by C_j, with $j \in [1, n-1]$, the latest contract for problem p_j, which completes before time T_0 in X, and by D_j its corresponding length.

Note that the lengths of contracts which are completed by time $T_0 + D_0$ in X are elements in the sequence X^s. In particular, it must be that $d_0 = x_k^s$ for some $k > 0$. Let I denote the set of indices in X^s of these lengths. We can then say that $def(X)$ (for an interruption right before $T_0 + D_0$) is

$$def(X) \geq \frac{\sum_{i \in I} x_i^s}{\sum_{i=1}^{n} l(X, p_i, T_0 + d_0^-)} = \frac{\sum_{i \in I} x_i^s}{d_0 + \sum_{i=1}^{n-1} D_i}. \qquad (11)$$

We will proceed by lower-bounding the numerator and upper-bounding the denominator of (11). First, observe that the numerator contains as summands all x_i^s which have been completed by time $T_0 + D_0$. Lemma 6 implies then that every contract smaller than $d_0 = x_k^s$ has been scheduled by time $T_0 + D_0$, and appears as a summand in the numerator. The same lemma guarantees also that for all $j \in [1, n-1]$, D_j cannot be smaller than d_0, otherwise the next contract for problem j to complete after D_j would also complete before time $T_0 + D_0$, which would contradict the definition of D_j as the earliest contract for problem j to finish before time $T_0 + D_0$. Hence we conclude that the numerator contains all contract lengths smaller than x_k^s, as well as at least $n - 1$ contract lengths (one for each $j \in [1, n-1]$) larger than x_k^s, and, last, contains a contract length equal to $D_0 > d_0$. The smallest values the latter n lengths can attain are then $x_{k+1}^s, \ldots x_{k+n}^s$. We thus obtain that $\sum_{i \in I} x_i^s \geq \sum_{i=0}^{k+n} x_i^s$.

We will now upper bound the denominator of (11). By definition of T_0, for every $j \neq 0$, we have $T_j + D_j \leq T_0 + D_0$. Hence Lemma 6 implies that $d_j < d_0$, and also $D_j < D_0$. We also know (as argued earlier) that $D_j > d_0$. Let $l > k$ be an index such that $D_0 = x_l^s > x_k^s = d_0$. We will argue that for every $j \in [1, n-1]$, there is at most one contract for problem j of length equal to x_i^s, for $i \in (k, l)$. Indeed, if this was not the case, then for problem j there would exist two consecutive contracts (relative to problem j) of lengths greater than d_0 and smaller than D_0, which contradicts Lemma 6. We conclude that the largest values $D_1, \ldots D_{n-1}$ can attain are $x_{k+1}^s, \ldots x_{k+n-1}^s$, hence $d_0 + \sum_{i=1}^{n-1} D_i \leq \sum_{i=0}^{n-1} x_{k+i}^s$.

The lemma follows from the above bounds on the numerator and the denominator of (11). □

As a last main step in the proof, we will apply the following result due to [Gal, 1980] and [Schuierer, 2001], which provides a lower bound for the supremum of a sequence of functionals. This result will help us relate the deficincy of normalized schedules to the deficiency of exponential ones. Define $G_a = (1, a, a^2, \ldots)$ to be the geometric sequence in a and $X^{+i} = (x_i, x_{i+1}, \ldots)$ the suffix of sequence X starting at x_i.

Theorem 8 ([Schuierer, 2001],[Gal, 1980]). *Let $X = (x_0, x_1, \ldots)$ be a sequence of positive numbers, r an integer, and $a = \overline{\lim}_{n \to \infty}(x_n)^{1/n}$, for $a \in \mathbb{R} \cup \{+\infty\}$. If F_k, $k \geq 0$, is a sequence of functionals which satisfy*

1. *$F_k(X)$ only depends on $x_0, x_1, \ldots, x_{k+r}$,*
2. *$F_k(X)$ is continuous, for all $x_i > 0$, with $0 \leq i \leq k+r$,*
3. *$F_k(\alpha X) = F_k(X)$, for all $\alpha > 0$,*
4. *$F_k(X + Y) \leq \max(F_k(X), F_k(Y))$, and*
5. *$F_{k+i}(X) \geq F_k(X^{+i})$, for all $i \geq 1$,*

then

$$\sup_{0 \leq k < \infty} F_k(X) \geq \sup_{0 \leq k < \infty} F_k(G_a).$$

Define now $F_k(X^s) := \sum_{i=0}^{k+n} x_i^s \Big/ \sum_{i=0}^{n-1} x_{k+i}^s$. It is easy to show that F_k satisfies the conditions of Lemma 8. Therefore,

$$\begin{aligned} def(X) &\geq \sup_{0 \leq k < \infty} F_k(X^s) \geq \sup_{0 \leq k < \infty} F_k(G_{\tilde{a}}) \\ &= \sup_{0 \leq k \leq \infty} \left\{ \sum_{i=0}^{k+n} a^i \Big/ \sum_{i=0}^{n-1} a^{k+i} \right\}. \end{aligned}$$

Note that if $a \leq 1$, then the above ratio tends to infinity as $k \to \infty$. Hence, we can assume that $a > 1$ and obtain

$$\begin{aligned} def(X) &\geq \sup_{0 \leq k < \infty} \left\{ \frac{(a^{k+n+1} - 1)/(a-1)}{a^k(a^n - 1)/(a-1)} \right\} \\ &= \sup_{0 \leq k < \infty} \left\{ \frac{a^{k+n+1} - 1}{a^k(a^n - 1)} \right\} \\ &\overset{(a>1)}{=} \frac{a^{n+1} - a^{-k}}{a^n - 1} \geq \frac{a^{n+1}}{a^n - 1}. \end{aligned} \qquad (12)$$

Recall that from (7), the deficiency of the exponential schedule of Section 3 is bounded by $b^{n+1}/(b^n - 1)$, for a value of b that minimizes this ratio, namely $b = (n+1)^{\frac{1}{n}}$. Comparing this expression to the lower bound of (12), we deduce that the exponential schedule is indeed optimal.

Theorem 9. *For a single processor, the optimal deficiency is* $\frac{(n+1)^{\frac{n+1}{n}}}{n}$*, and is achieved by the exponential schedule.*

5 Conclusion

In this paper we addressed the problem of devising interruptible algorithms in a setting in which solutions to all problem instances are required and all problems are equally important. Even though our proposed solutions are efficient, it would be nevertheless interesting to know whether our schedule is optimal for any number of processors. This appears to be a complicated task: for general m, minimizing makespan is NP-hard. More importantly, the known efficient makespan-scheduling algorithms do not yield a closed-form expression of the makespan as a function of the sizes of jobs (Graham's greedy algorithm is an exception to this, but it is no better than a 2-approximation).

Another direction for further research is to consider the generalization in which at interruption time a subset of $i \leq n$ problems are queried for their solution. We are also interested in a related setting in the context of robot searching in multiple concurrent rays. [Bernstein *et al.*, 2003] showed important connections between this problem and contract scheduling under the acceleration ratio. Suppose, however, that instead of a single target, i targets are located, at unknown positions from the common origin. We have m robots available to explore the rays. What is the best strategy for the robots so as to locate *all* targets? There are interesting parallels between the two problems, e.g., the optimal solution in the multi-target variant can be formulated as a makespan scheduling problem, with job sizes a function of the distances of the targets from the origin. We believe the results of this paper can be useful in addressing this problem.

References

[Angelopoulos *et al.*, 2008] S. Angelopoulos, A. López-Ortiz, and A. Hamel. Optimal scheduling of contract algorithms with soft deadlines. In *Proceedings of the 23rd AAAI conference on Artificial Intelligence (AAAI-08)*, pages 868–873, 2008.

[Bernstein *et al.*, 2002] D.S. Bernstein, T. J. Perkins, S. Zilberstein, and L. Finkelstein. Scheduling contract algorithms on multiple processors. In *Proceedings of the 18th AAAI Conference on Artificial Intelligence (AAAI-02)*, pages 702–706, 2002.

[Bernstein *et al.*, 2003] D. S. Bernstein, L. Finkelstein, and S. Zilberstein. Contract algorithms and robots on rays: Unifying two scheduling problems. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI-03)*, pages 1211–1217, 2003.

[Dean and Boddy, 1988] T. Dean and M. S. Boddy. An analysis of time-dependent planning. In *Proceedings of the 7th AAAI Conference on Artificial Intelligence (AAAI-88)*, pages 49–54, 1988.

[Gal, 1980] S. Gal. *Search Games*. Academic Press, 1980.

[Gomes and Selman, 2001] C. P. Gomes and B. Selman. Algorithm portfolios. *Artificial Intelligence*, 126(1-2):43–62, 2001.

[Graham, 1966] R. Graham. Bounds for certain multiprocessing anomalies. *Bell System Technical Journal*, 45:1563–81, 1966.

[Horvitz, 1987] E. Horvitz. Reasoning about beliefs and actions under computational resource constraints. In *Proceedings of the 3rd Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 301–324, 1987.

[Horvitz, 1988] E. Horvitz. Reasoning under varying and uncertain resource constraints. In *Proceedings of the 7th AAAI Conference on Artificial Intelligence (AAAI-88)*, pages 111–116, 1988.

[López-Ortiz *et al.*, 2006] A. López-Ortiz, S. Angelopoulos, and A. M. Hamel. Optimal scheduling of contract algorithms for anytime problems. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI-06)*, 2006.

[Russell and Zilberstein, 1991] S. J. Russell and S. Zilberstein. Composing real-time systems. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI-91)*, pages 212–217, 1991.

[Schuierer, 2001] S. Schuierer. Lower bounds in online geometric searching. *Computational Geometry: Theory and Applications*, 18(1):37–53, 2001.

[Zilberstein *et al.*, 1999] S. Zilberstein, F. Charpillet, and P. Chassaing. Real-time problem-solving with contract algorithms. In *Proc. of the 16th International Joint Conference on Artificial Intelligence (IJCAI-99)*, pages 1008–1015, 1999.

[Zilberstein *et al.*, 2003] S. Zilberstein, F. Charpillet, and P. Chassaing. Optimal sequencing of contract algorithms. *Annals of Mathematics and Artificial Intelligence*, 39(1-2):1–18, 2003.