Continuity and Approximability of Response Time Bounds

S.K Baruah
Department of Computer Science, University of North Carolina, Chapel Hill, USA.

E. Bini
Scuola Superiore Sant'Anna, Pisa, Italy

T.H.C. Nguyen, P. Richard
Laboratoire d'Informatique Scientifique et Industrielle, Ensma, Poitiers,France.

Abstract

Since worst-case response times must be determined repeatedly during the interactive design of real-time application systems, repeated exact computation of such response times would slow down the design process considerably. In this research, we identify three desirable properties of estimates of the exact response times: continuity *with respect to system parameters;* efficient computability*; and* approximability. *We demonstrate that a recently-proposed technique for estimating the worst-case response time of sporadic task systems that are scheduled using static priority upon a preemptive uniprocessor possesses these properties.*

1. Introduction and Motivation

In preemptive uniprocessor sporadic task systems that are scheduled according to static priorities, the well-known technique of *response-time analysis* (RTA) allows for the exact computation of the worst-case response time of each task in time pseudo-polynomial in the representation of the task system. Consider a system of n tasks $\tau_1, \tau_2, \ldots, \tau_n$, with the i'th task τ_i characterized by a worst-case execution time C_i and a minimum inter-arrival separation parameter T_i, and with the additional constraint that each task τ_i have a worst-case response time $\leq T_i$. Without loss of generality, assume that the tasks are indexed in decreasing order of priority. Let R_i denote the the worst-case response time of τ_i; RTA asserts that the the value of R_i is equal to the smallest t satisfying the following equality:

$$t = C_i + \sum_{j<i} \left\lceil \frac{t}{T_j} \right\rceil C_j \qquad (1)$$

One of the features of the worst-case response time that is easily seen from Equation 1 above is that it is *not a continuous function of system parameters*. For instance in Equation 1, if decreasing some T_j by an infinitesimally small amount causes $\lceil R_i/T_j \rceil$ to increase by one, it can be shown that τ_i's response time increases by an amount $\geq C_j$.

Such discontinuities are a major hurdle to a process of incremental, interactive system design. Ideally, such an design process would allow for the interactive exploration of the state space of possible designs; this would be greatly facilitated if making minor changes to a design (equivalently, moving small distances in the state space of possible system designs) results in minor changes to system properties.

Now these discontinuities are unavoidable, since they are a feature of worst-case response time per se, and not just of the RTA technique for comput-

ing them. Nevertheless, we believe that there is some benefit to studying *upper bounds* on worst-case response times that do not suffer such discontinuities with system parameters. If we were to use these continuous upper bounds (rather than the discontinuous exact bounds) during the system design process, then we would know that neighboring points in the design state space would have similar response time bounds. (Thus for example if we were at a point in the design state space where every task except one had its response-time bound well within acceptable limits, we could safely consider making small changes to the task parameters in order to search for neighboring points in the design state space in which the response time bound of the non-compliant task is decreased, without needing to worry that the response time of some currently compliant task would increase by a large amount.)

Any such continuous upper bound on response-time is necessarily not tight (since as stated above, the property of worst-case response-time is itself not continuous in the system parameters). In other words, by choosing a continuous upper bound we would be trading off accuracy for continuity: it is desirable that this loss of continuity be *quantified* in some manner in order that the system designer may determine whether the loss of accuracy is worth the benefits that continuous bounds provide to the system design process.

An additional requirement on such bounds, if they are to be used repeatedly during an interactive design process, is that they be *efficiently computable*, preferably in fast polynomial time. (Efficient computation could be another reason for preferring to use some approximation – continuous or not – rather than the exact worst-case response-time of Equation 1, which has a pseudo-polynomial computation time.)

To summarize the points made above: we are seeking upper bounds on the worst-case response time that are **continuous** functions of system parameters; in addition, we would like these bounds to be **efficiently computable**, and to have **quantifiable deviation** from the exact bounds.

2. Approximation results

The theory of approximation algorithms for combinatorial optimization problem can be used to quantify the performance guarantee of response time bounds. The performance guarantee of an algorithm is analyzed through its *approximation ratio*. Let a be the value obtained by an algorithm A that solves a minimization problem, and opt be the exact (i.e., minimum) value; algorithm A has an approximation ratio of c, where $c \leq 1$, if and only if $opt \leq a \leq c \times opt$ for all inputs to the algorithm A. (If such a c does not exist, then algorithm A is said to have no approximation ratio — this means that the bound computed by A can be very far away from the optimal one.) If A is a polynomial time algorithm and has a bounded performance guarantee, then it is called an approximation algorithm.

An efficiently-computable continuous upper bound on response time is easily derived from the technique of [3] (this technique was originally proposed in [3] for computing a lower bound):

$$R_i \leq \frac{\sum_{j<i} C_j}{1 - \sum_{j<i} U_j} \tag{2}$$

(Here and henceforth, U_i denotes the *utilization* C_i/T_i of task τ_i.)

An improved efficiently-computable continuous upper bound was recently proposed in [1]:

$$R_i \leq \frac{C_i + \sum_{j<i} C_j(1 - U_j)}{1 - \sum_{j<i} U_j} \stackrel{\text{def}}{=} ub_i \tag{3}$$

In [2] it was shown that the upper bound of [3] (Equation 2 above) does not have an approximation ratio. The next result states that neither does the upper bound ub_i of [1] (Equation 3 above).

Theorem 1 *The upper bound ub_i of [1] (Equation 3 above) does not have an approximation ratio.*

Proof: We prove this by demonstrating a task system and a task τ_i for which ub_i/R_i tends to ∞. Let us represent the parameters of a task τ_i by an ordered pair (C_i, T_i). Consider the following task set: $\tau_1 = (K, 2K + \epsilon)$, $\tau_2 = (K, 2K + \epsilon)$

and $\tau_3 = (\epsilon, 2K + \epsilon)$, where ϵ is an arbitrarily small positive number and K is an arbitrary number greater than ϵ. The utilization $U_1+U_2+U_3$ is 1 and since tasks have all periods equal to $2K + \epsilon$, then $R_3 = 2K+\epsilon$ using the Rate Monotonic scheduling policy. The upper bound is:

$$\begin{aligned} ub_i &= \frac{\epsilon + 2K(1 - \frac{K}{2K+\epsilon})}{1 - \frac{2K}{2K+\epsilon}} \\ &= \frac{\epsilon(2K+\epsilon) + 2K(2K+\epsilon-K)}{2K+\epsilon-2K} \\ &= \frac{(2K+\epsilon)\epsilon + 2K(K+\epsilon)}{\epsilon} \\ &= 4K + \epsilon + \frac{2K^2}{\epsilon} \end{aligned}$$

Thus,

$$\lim_{\epsilon \to 0} ub_i = \lim_{\epsilon \to 0} \left(4K + \epsilon + \frac{2K^2}{\epsilon} \right) = \infty$$

and the theorem is proved. ■

3. Resource Augmentation analysis

The results in Section 2 reveal that the upper bound ub_i on response-time of [1] does not offer any quantifiable performance guarantee, according to the conventional approximation ratio measure that is used in optimization theory. However, an alternative approach towards approximate analysis is sometimes used in real-time scheduling theory – the technique of **resource augmentation**. In this technique, the performance of the algorithm under analysis is compared with that of an optimal algorithm *that runs on a slower processor.* In this section, we apply this resource augmentation technique to quantify the deviation of ub_i from optimality.

In obtaining upper bounds to the worst-case response time, the first step is typically to replace the expression within the ceiling function — i.e., the expression $(\lceil t/T_j \rceil C_j)$ — in the exact computation of the worst-case response time (Equation 1) by a linear approximation $\text{LA}(\tau_j, t)$. The approximation introduced in [1] is as follows:

$$\text{LA}(\tau_j, t) = U_j \times t + C_j(1 - U_j) \qquad (4)$$

(See [1] for details, and a proof of correctness.)

Now it would be nice if

$$C_j \left\lceil \frac{t}{T_j} \right\rceil \leq \text{LA}(\tau_j, t)$$

were to hold for all $t \geq 0$; if this were the case, the correctness of ub_i — the fact that it is indeed an upper bound on the exact worst-case response time — would follow immediately. Unfortunately, however, this inequality is not true for all t; indeed, values of t arbitrarily close to 0 bear witness to its falsity. However, it can be shown[1] that $C_j \left\lceil \frac{t}{T_j} \right\rceil \leq \text{LA}(\tau_j, t)$ does indeed hold at all those values of t *which matter*:

Lemme 1 *For all values of* t that are potential values of R_i,

$$C_j \left\lceil \frac{t}{T_j} \right\rceil \leq \text{LA}(\tau_j, t)$$

An upper bound on amount by which $\text{LA}(\tau_j, t)$ deviates from its exact value can also be shown:

Lemme 2 *For all values of* t *that are potential values of* R_i,

$$\text{LA}(\tau_j, t) \leq 2 \times C_j \left\lceil \frac{t}{T_j} \right\rceil$$

It can be shown that the bound ub_i (as defined in Equation 3 above) is equal to the smallest value of t satisfying the following inequality:

$$t = C_j + \sum_{j<i} \text{LA}(\tau_j, t) \qquad (5)$$

We can use this result to prove a resource-augmentation bound on ub_i, as follows. Since ub_i is the smallest value of t to satisfy Equation 5

[1] Many proofs are omitted below, for this WIP submission.

above, it must be the case that for all $t < ub_i$

$$
\begin{aligned}
t &< C_j + \sum_{j<i} \text{LA}(\tau_j, t) \\
&\Rightarrow t < C_j + 2 \sum_{j<i} \left\lceil \frac{t}{T_j} \right\rceil C_j \\
&\Rightarrow t < 2 \left(C_j + \sum_{j<i} \left\lceil \frac{t}{T_j} \right\rceil C_j \right) \\
&\equiv \frac{1}{2} \;\; t < C_j + \sum_{j<i} \left\lceil \frac{t}{T_j} \right\rceil C_j
\end{aligned}
$$

That is, the cumulative workload of jobs with priority greater than or equal to τ_i's that must be scheduled over the interval $[0, t)$ prior to the completion of task τ_i's first job in the critical instant exceeds the total capacity of a processor with computing capacity $1/2$ over the same interval. Hence no such $t < ub_i$ can represent the worst-case response time of τ_i upon a processor of computing capacity one-half. Theorem 2 follows:

Theorem 2 *The bound ub_i of [1] (Equation 3) is*

1. *An upper bound on the worst-case response time of τ_i; and*
2. *A lower bound on the worst-case response time of τ_i* if the system is implemented upon a processor of speed one-half.

How is the systems designer to interpret Theorem 2 above? First, it is guaranteed that ub_i is indeed an upper bound on R_i; hence, it is a safe estimate of the exact worst response time. And while Theorem 2 is unable to bound the amount by which ub_i exceeds the actual value of R_i (indeed, Section 2 has shown that there can, in general, be no such bound), it does assure the designer that [s]he could have obtained a worst-case response time no better than ub_i if the system had instead been implemented upon a processor half as fast. Stated differently, *a processor speedup of two is an upper bound on the price being paid for using an efficiently computable upper bound on response time that is a continuous function of the system parameters.*

4. Conclusions

We have argued that *continuity* with respect to system parameters is a very desirable property of response-time bounds, if such bounds are to be used as an integral part of an incremental, interactive, design process. However, such continuity necessarily comes with a loss of accuracy; in this work, we have attempted to quantify this loss of accuracy in some recently-proposed continuous upper bounds on response time. We have shown that while these upper bounds do not offer non-trivial performance guarantees according to conventional metrics, we were able to show that a performance guarantee can indeed be obtained using the concept of resource augmentation.

Specifically, we considered the response-time bound presented in [1]. This bound is continuous in all system parameters, and is very efficiently computable in time linear in the representation of the task system. We demonstrated that this bound offers the following quantitative guarantee – it is indeed an upper bound on the exact response time, and the exact response time would necessarily be at least as large as this bound if the system were instead implemented upon a processor that is at most only one-half times as fast.

References

[1] E. Bini and S. Baruah. Efficient computation of response time bounds under fixed-priority scheduling. *Int. Conf. on Real-Time and Network Systems (RTNS'07)*, pages 95–104, 2007.

[2] P. Richard. Polynomial time approximate schedulability tests for fixed-priority real-time tasks: some numerical experimentations. *14th Real-Time and Network Systems, Poitiers (France)*, 2006.

[3] M. Sjodin and H. Hansson. Improved response time analysis calculations. *proc. IEEE Int Symposium on Real-Time Systems (RTSS'98)*, 1998.