Addendum to Tardiness Bounds for Global EDF with Deadlines Different from Periods

Jeremy Erickson[1], Nan Guan[2], and Sanjoy Baruah[1]

[1] The University of North Carolina at Chapel Hill, Chapel Hill, NC
{jerickso,baruah}@cs.unc.edu
[2] Uppsala University, Uppsala, Sweden, Nan.Guan@it.uu.se

Abstract. In *Tardiness Bounds for Global EDF with Deadlines Different from Periods* [1], provided tardiness bounds for the global Earliest Deadline First (EDF) scheduling algorithm which are the tightest known in the case of arbitrary deadlines, in which deadlines need not be equal to periods. However, in some cases it is possible to develop even tighter bounds. In this technical report we describe this technique and provide an initial experimental appraisal.

In [1] we provided, with proof, tardiness bounds that could be used to analyze the global Earliest Deadline First (EDF) scheduler even in the case in which deadlines and periods differ. We did so by providing an extra term S_i for each task which can be roughly indicated as the extra tardiness that could be created due to a task being due early. In [1] we statically defined this number purely based on the task parameters. However, this technique is more conservative than necessary in some cases. In this report, we will refer heavily to the theorems, lemmas, and proofs presented in [1].

Observe that the values U_i and S_i are used in Theorem 1 only for the purpose of upper-bounding $\text{DBF}(\tau_i, t)$. Lemma 1 proved that using $U_i = \frac{C_i}{T_i}$ (i.e., utilization) and $S_i = C_i \times \max\left\{0, 1 - \frac{D_i}{T_i}\right\}$ are sufficient to produce tardiness bounds. However, any values of V_i and S_i such that

$$\text{DBF}(\tau_i, t) \leq V_i t + S_i \tag{1}$$

holds can play the same role in the proof of Theorem 1, and provide bounded tardiness under the assumption that $\forall \tau_i, V_i \leq 1$ and $\sum_{\tau_i \in \tau} V_i \leq m$. We can then obtain tardiness bounds by replacing U_i with V_i in Definition 1 and finding a minimal compliant vector as before.

It is trivial to observe that if $V_i < U_i$, then (1) cannot hold. Thus, the bounds cannot be improved by decreasing V_i. However, if we *increase* V_i, we can reduce the corresponding S_i term. Of course, due to the conditions for bounded tardiness above, this is only possible if the system is not fully utilized. Using a method similar to the proof of Lemma 1, one can demonstrate that $S_i = \max\{0, C_i - V_i D_i\}$ provides the smallest value such that (1) holds.

We have performed simple experiments which demonstrate that in some cases, tardiness bounds can be improved by increasing V_i and decreasing S_i.

The most obvious case is when a value does not contribute to $\mathbf{L}(\boldsymbol{x})$ but does have a nonzero S_i value. In this case, we may increase V_i at least to the minimum value it would require to contribute to $\mathbf{L}(\boldsymbol{x})$, and $\mathbf{S}(\tau)$ will be decreased with no increase to $\mathbf{L}(\boldsymbol{x})$, resulting in smaller bounds for all tasks. In some other cases, the increase to $L(\boldsymbol{x})$ caused by increasing V_i is less than the decrease to $\mathbf{S}(\tau)$, which also leads to smaller tardiness bounds. Therefore, for many task systems which are not fully utilized, improvements to the bound are possible.

However, we have not yet found an efficient method for determining the smallest possible tardiness bounds for a given task set. We instead use Algorithm 1 below to find an improvement which is not guaranteed to be optimal. We observe in light of Lemma 5 that minimizing $\mathbf{L}(\boldsymbol{x}) + \mathbf{S}(\tau)$ is sufficient to minimize the overall tardiness bounds. Algorithm 1 starts by using the initial $\mathbf{L}(\boldsymbol{x})+\mathbf{S}(\tau)$ as in Theorem 1, which can be computed exactly using Algorithm 1. We then iterate, using the variable i to track whether an improvement has been made. We use one parameter, j, which specifies the desired step size by which to increase each V_i value. We increase by less than j in cases where increasing by j either violates the conditions for bounded tardiness or is clearly suboptimal (i.e., when $V_i = \frac{C_i}{D_i}$, $S_i = 0$ so no further increase is desirable.) We attempt to increase V_i for each task independently, and actually increase V_i for whichever task leads to the largest decrease in $\mathbf{L}(\boldsymbol{x}) + \mathbf{S}(\tau)$. Iteration terminates whenever no improvement was made during one iteration, or when $\sum_{\tau_i \in \tau} V_i$ reaches m.

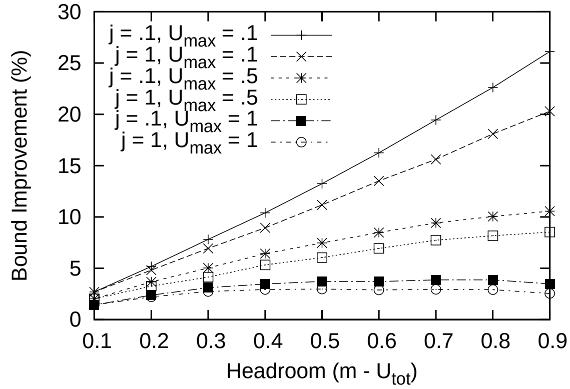
In order to determine the validity of this approach, we generated a random set of constrained-deadline ($D_i \leq T_i$) task systems. Each set of 1000 tasks was determined by parameters m, the number of CPUs, U_{max}, the maximum possible utilization for a given task, and U_{tot}, the total utilization of all tasks. Task utilizations U_i were selected uniformly from the range $[0, U_{max}]$, periods T_i uniformly from $[5, 30]$, and deadlines uniformly from $[0, T_i]$. Tasks were generated until $\sum U_i > U_{tot} - U_{max}$, at which point a task of utilization $U_{tot} - \sum U_i$ was created to achieve U_{tot} exactly. Tasks were generated with U_{tot} ranging from $m - .9$ to $m - .1$ in increments of 0.1, with U_{max} values of 0.1, 0.5, and 1, and with m values of 4, 8, and 16.

Experiments were performed on each task set using Algorithm 1 with a simple binary search algorithm in place of Algorithm 1 to compute $\mathbf{L}(\boldsymbol{x})$ values. The binary search algorithm runs very quickly at the expense of providing only an approximate solution. Algorithm 1 was run both with $j = 1$ (in which case each V_i was increased as much as possible or not at all) and with $j = 0.1$. Experiments ran noticeably faster for $j = 1$, because many fewer iterations were required. However, utilizing $j = 0.1$ produced marginally better improvements. This demonstrates that increasing several V_i values each by less than the full amount can be better than increasing a smaller number of V_i values.

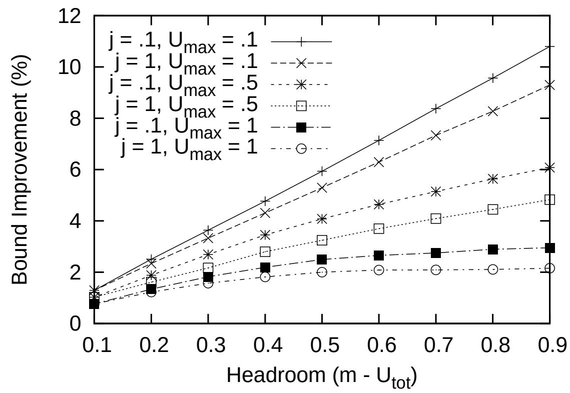
Results are shown in Figures 1(a), 1(b), and 1(c). Several trends immediately appear. For task sets with small values of U_{max}, having a larger difference between m and U_{tot} leads to more significant improvements. This is unsurprising, because larger differences between m and U_{tot} allow for greater increases in V_i values. In the extreme case where $U_{tot} = m$ our technique would provide abso-

for all $\tau_i \in \tau$ **do**
  $V_i = U_i$
  $S_i = C_i \times \max\left\{0, 1 - \frac{D_i}{T_i}\right\}$
end for
$b =$ initial best $\mathbf{L}(\boldsymbol{x}) + \mathbf{S}(\tau)$ value
$i = 1$
while $i = 1$ AND $\sum_{\tau_i \in \tau} V_i < m$ **do**
  $i = 0$
  $k = \min\{j, m - \sum_{\tau_i \in \tau} V_i\}$
  for all $\tau_i \in \tau$ **do**
    $V_i = \min\left\{1, \frac{C_i}{\min\{D_i, T_i\}}, V_i + k\right\}$
    $S_i = \max\{0, C_i - V_i D_i\}$
    $c =$ best $\mathbf{L}(\boldsymbol{x}) + \mathbf{S}(\tau)$ value for τ
    if $c < b$ **then**
      $b = c$
      $i = 1$
    end if
    Restore V_i and S_i to previous values
  end for
  if $i = 1$ **then**
    Update V_i and S_i to match most recent b value
  end if
end while

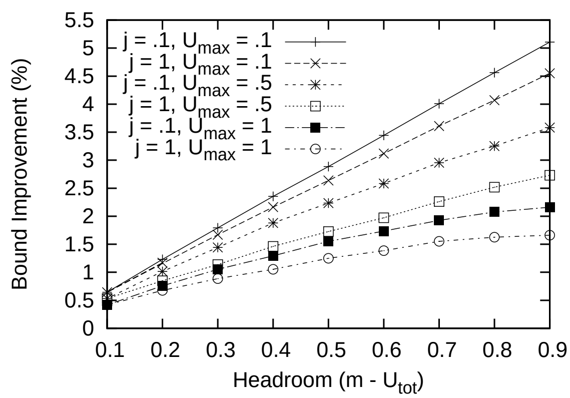
Algorithm 1: Algorithm to determine improved bound by altering V_i and S_i



(a) Improvement to Average Bound for $m = 4$



(b) Improvement to Average Bound for $m = 8$



(c) Improvement to Average Bound for $m = 16$

lutely no improvement to the bounds, because no V_i value could be increased. For task sets with larger values of U_{max}, larger differences between m and U_{tot} do not necessarily lead to larger improvements in the bound. We believe this is primarily a result of the fact that task sets with larger utilizations have larger x_i values, so increasing V_i causes larger increases to $L(\boldsymbol{x})$ than in task systems with smaller utilization. This fact is also likely the reason that improvements to the bound are smaller for task systems with larger utilization.

With larger values of m, the percentage increase of the bounds is smaller. We believe this is due to the fact that $\mathbf{L}(\boldsymbol{x}) + \mathbf{S}(\tau)$ is divided by m in computing the bounds. The scaling factor is not linear due to the C_i component of the bound. Also, for larger values of m, the effect of larger U_{max} values is less pronounced. This may be due to the presence of a greater number of smaller tasks that can have their utilization increased without increasing $L(\boldsymbol{x})$.

Altogether, some level of improvemenet does seem to be possible, particularly on a small number of processors when the system is under-utilized by nearly a full processor and when task utilizations are small.

References

1. Erickson, J.P., Guan, N., Baruah, S.K.: Tardiness bounds for global EDF with deadlines different from periods. In: Principles of Distributed Systems (OPODIS). (2010)