SUPPLEMENT OF "BEYOND IMPLICIT-DEADLINE OPTIMALITY: A MULTIPROCESSOR SCHEDULING FRAMEWORK FOR CONSTRAINED-DEADLINE TASKS"

Hyeongbook Baek\(^1\), Hoon Sung Chwa\(^2\) and Jinkyu Lee\(^3\)

\(^1\)Department of Computer Science and Engineering, Sungkyunkwan University (SKKU), Republic of Korea.
\(^2\)Department of Electrical Engineering and Computer Science, The University of Michigan, U.S.A.

A. Execution-time bound under TL(\(\tau\), Fluid, Fluid-FP)

In the following lemma, we prove that the amount of execution of jobs of \(\tau^\text{Class}\) in an interval of length \(\ell\) is upper-bounded by \(R^\text{Class} \cdot L^\text{Class}(\ell)\) even if \(\tau^\text{Class}\) executes with a rate lower than \(R^\text{Class}\) (i.e., executing up to \(R^\text{Class}\)).

**Lemma 8:** For given \(\ell\), the amount of execution of jobs of \(\tau^\text{Class}\) in an interval of length \(\ell\) is upper-bounded by \(R^\text{Class} \cdot L^\text{Class}(\ell)\) if a job of \(\tau^\text{Class}\) executes with a rate up to \(R^\text{Class}\) (which coincides with the scheduling policy of TL(\(\tau\), Fluid, Fluid-FP)).

**Proof:** Suppose that there exist job(s) of \(\tau^\text{Class}\) executing with a rate lower than \(R^\text{Class}\) in an interval of length \(\ell\), and the amount of execution of jobs of \(\tau^\text{Class}\) in the interval is larger than \(R^\text{Class} \cdot L^\text{Class}(\ell)\). We focus on the base situation where the jobs of \(\tau^\text{Class}\) execute with exactly \(R^\text{Class}\) rate shown in Fig. 5, resulting in \(R^\text{Class} \cdot L^\text{Class}(\ell)\) amount of execution in an interval of length \(\ell\). Focusing on jobs of \(\tau_1\) in the interval of interest (e.g., three jobs in Fig. 5), we will show that (i) the earliest-released job, (ii) the latest-released job, and (iii) other jobs (i.e., body jobs) in the interval execute with less than \(R^\text{Class}\) rate, we need at least \(\ell\) interval length for jobs of \(\tau^\text{Class}\) in the interval to have \(R^\text{Class} \cdot L^\text{Class}(\ell)\) amount of execution, which contradicts the supposition.

(Case i and ii) If a job of \(\tau^\text{Class}\) executes with a rate lower than \(R^\text{Class}\), the execution length of the job increases compared with exactly \(R^\text{Class}\) rate. Therefore, in order to have \(R^\text{Class} \cdot L^\text{Class}(\ell)\) amount of execution of jobs of \(\tau^\text{Class}\) in the interval, the interval should be extended to the left (Case i) and right (Case ii) directions compared to the base situation.

(Case iii) Since body jobs fully execute anyway regardless of their execution rates, we need the same interval as the base situation, in order to have \(R^\text{Class} \cdot L^\text{Class}(\ell)\) amount of execution of jobs of \(\tau^\text{Class}\) in the interval.

In the three cases, in order to have \(R^\text{Class} \cdot L^\text{Class}(\ell)\) amount of execution of jobs of \(\tau^\text{Class}\) in the interval, the interval length should be larger than or equal to that of the base situation. This contradicts the supposition.

B. Property of NC-ORA

We prove that binary search in Line 11 of Algo. 3 can be done without backtrack in the following lemma.

**Lemma 9:** Consider two execution rate assignments for \(\tau^\text{LO}\): \(R^\text{LO}_k = A/C_k\) and \(R^\text{LO}_k = A'/C_k\), where \(A > A'\). Then, if Lemma 5 deems \(\tau^\text{LO}\) schedulable (i) with given \(R^\text{LO}_k\) and the execution rate and task priority assignment by Lemma 6, the same holds (ii) for \(\tau^\text{LO}\) with \(R^\text{LO}_k\).

**Proof:** We show that if Eqs. (5) and (6) hold for (i), those also hold for (ii). The execution rate and task priority assignment in Lemma 6 guarantees that Eq. (5) is always satisfied for \(\tau^\text{LO}_k\) (regardless of \(R^\text{LO}_k\)) since the left term of the LHS of Eq. (5) is 0, and the right term of that cannot be larger then \(m \cdot (D_k - X^\text{LO}_k) / \delta_{\text{sum}}(\text{\text{HI}}) = m\). When it comes to Eq. (6), the reduction from \(R^\text{LO}_k\) to \(R^\text{LO}'_k\) increases the RHS of Eq. (6) exactly by \(R^\text{LO}_k - R^\text{LO}'_k\), while it increases the LHS of Eq. (6) by less than \(R^\text{LO}_k - R^\text{LO}'_k\), meaning the same amount of execution is extracted from \(\tau^\text{LO}_k\) and added to \(\tau^\text{HI}\) but such execution is performed in \(\tau^\text{HI}\) with an execution rate no larger than that in \(\tau^\text{LO}\) since \(R^\text{HI}_k \leq R^\text{LO}_k\) holds for \(C_k = C^\text{LO}_k\). Thus the lemma holds.

C. Example of NC-ORA

We describe how NC-ORA Algo. 3 works with the following example.

**Example 4:** Recall the same task set as Example 3. After Lines 1–8 in Algo. 3 are executed, for given \(C^\text{HI}_k\) and \(C^\text{LO}_k\) of each task \(\tau_k\), the first execution rate and priority assignment illustrated in Lemma 6 constructs the following:

For \(\tau_1\):
- \(\tau_1^\text{HI} = \{\tau_1, \tau_2, \tau_1^\text{HI}(T_1=10, C^\text{HI}_1=1.8, D_1=6)\}\)
- \(\tau_1^\text{LO} = \{\tau_1^\text{LO}(12,1,2,6)\}\)

For \(\tau_2\):
- \(\tau_2^\text{HI} = \{\tau_2, \tau_3, \tau_2^\text{HI}(T_2=12, C^\text{HI}_2=1.5)\}\)
- \(\tau_2^\text{LO} = \{\tau_2^\text{LO}(12,1,2,5)\}\)

For \(\tau_3\):
- \(\tau_3^\text{HI} = \{\tau_3, \tau_3^\text{HI}(T_3=15, C^\text{HI}_3=1.2)\}\)
- \(\tau_3^\text{LO} = \{\tau_3^\text{LO}(6,1,2,15)\}\)

For \(\tau_4\), it is deemed schedulable by Lemma 5 with above execution rate and priority assignment, indicating that the execution rate of \(\tau_4^\text{HI} = (15, 3, 10)\) is the minimum one to avoid deadline miss for \(\tau_4\). On the other hand, for \(\tau_1, \tau_2\) and \(\tau_3\), NC-ORA cannot find any rate of \(C^\text{LO}_k\) that makes \(\tau^\text{LO}_k\) deemed schedulable using binary search (Lines 11–12), meaning whole amount of \(C_k\) should be assigned to \(\tau^\text{HI}\). Since the summation of minimum execution rates that should be assigned to \(\tau^\text{HI}\) is 36/3+3/5+3/5+3/10 = 2, the example task set \(\tau\) satisfies the necessary condition in Lemma 7.

D. Task set generation method

We randomly generate 100,000 constrained-deadline task sets for each \(m \in \{2, 4, 8, 16\}\), based on a technique proposed in [27] used in many studies, e.g., [13, 23]. For task set generation, we consider two input parameters: the number of processor \(m = 4\) or 8) and task utilization parameter. For a task \(\tau\), \(T_1\) is uniformly chosen in \([1, 1000]\), and \(C_i\) is determined by a binomial and exponential utilization parameter. For a given binomial parameter \(p\), a value for \(C_i/T_i\) is uniformly chosen in \([0, 0.5]\) with probability \(p\), and for a given exponential parameter \(1/\lambda\), that is chosen according to the exponential distribution whose probability density function is \(\lambda \cdot \exp(-\lambda \cdot x)\), where each parameter can be a value of 0.1, 0.3, 0.5, 0.7 or 0.9. Then, \(D_i\) is uniformly chosen in \((C_i, T_i)\) as we consider constrained deadline task model.

For each task utilization, we conduct the following steps until 10,000 task sets are generated.
1) We generate a task set containing $m + 1$ tasks.
2) We check whether the generated task set passes a necessary feasibility condition [28].
3) If it passes the necessary feasibility condition, we include the task for evaluation. Then, we generate a new task set by adding a new task into the old task set and return to Step 2). Otherwise, we discard the task and return to Step 1).

We create 10,000 task sets for each task utilization model (bimodal or exponential model with a given parameter value chosen among 0.1, 0.3, 0.5, 0.7 and 0.9), in total 100,000 task sets for a given $m$. 