Current Results on EDZL Scheduling for Multiprocessor Real-Time Systems

Hsin-Wen Wei¹, Yi-Hsiung Chao², Shun-Shii Lin², Kwei-Jay Lin³, Wei-Kuan Shih¹
¹Dept. of Computer Science, National Tsing Hua University, Hsinchu, Taiwan
²Dept. of Computer Science and Information Engineering, National Taiwan Normal University, Taipei, Taiwan
³Dept. of Electrical Engineering and Computer Science, University of California, Irvine, CA
{bertha, wshih}@rtlab.cs.nthu.edu.tw, {g92470231, linss}@csie.ntnu.edu.tw, klin@uci.edu

Abstract

Many optimal uniprocessor schedulers, such as Earliest Deadline First (EDF) and Rate Monotonic (RM), do not have a good schedulability bound on multiprocessor systems. In this paper, we study an on-line algorithm Earliest Deadline First until Zero laxity (EDZL) for multiprocessor systems. A set of tasks scheduled by EDZL is scheduled using EDF until a job experiences a zero laxity. To avoid the job from missing its deadline, the priority of the job is immediately promoted to the highest priority. We derive the schedulability bound of $3/2 + |u_{max} - 1/2|$ for two-processor systems, where $u_{max}$ is the maximum utilization of an individual task in the given task set. We also discuss the best known upper bound and lower bound on EDZL schedulability conditions.

1. Introduction

Recently, many multiprocessor (or multicore) systems become available. As the design technologies of microprocessor are advancing, many real-time embedded systems are exploring the possibility of using more concurrent computing nodes to handle higher workloads. However, the challenge of real-time scheduling for multiprocessor systems is that most of the well-known scheduling algorithms designed for single-processor cannot be directly applied to two or more processors without losing their optimality.

The scheduling algorithms for real-time tasks on multiprocessor systems have been studied by many researchers. Well-known scheduling algorithms for uniprocessor real-time systems, such as Earliest Deadline First Algorithm (EDF) and Rate Monotonic Algorithm (RM), have been shown to have very low schedulability conditions on multiprocessor systems [2] [3]. In multiprocessor systems, it has been shown that there exist task systems that cannot be scheduled by EDF on $m$ processors if the total utilization of the task set exceeds $m - (m - 1)u_{max}$ [1] [4]. Therefore, the bound of EDF may be very low when the value of $u_{max}$ is large. Least Laxity Algorithm (LLA) may obtain a higher utilization bound than EDF in multiprocessor system, but it suffers from a high context switching overhead.

In this paper, we study the EDZL algorithm [6], which is a hybrid algorithm of EDF and LLA. EDZL schedules tasks based on both their deadlines and laxity values to obtain a better scheduling bound than EDF and a lower context switching overhead than LLA. A set of tasks scheduled by EDZL is scheduled by their dynamic priorities defined by EDF until the laxity of a job becomes zero. To avoid the job from missing its deadline, the priority of the job is immediately promoted to the highest priority above any EDF priorities in the system. As a result, EDZL can obtain a higher scheduling bound than EDF with the priority promotion.

Some recent results on EDZL schedulability bounds are presented in this paper. We show that EDZL has a schedulability bound of $3/2 + |u_{max} - 1/2|$ for 2-processor systems. It has been shown in [8] that any task set can be scheduled by EDZL on $m$ processor systems if the total utilization is less than or equal to $(m+1)/2$. We also note that there exist task sets not schedulable by EDZL when $U(T) > m(1-1/e)$ [10], where $e$ is the Euler’s number. The above two bounds still have some gap that may be converged further. In addition, we show that EDZL is an optimal scheduling algorithm when all tasks have the same release time and deadline.
The rest of this paper is organized as follows. Section 2 shows the task model and some previous result on multiprocessor real-time scheduling. Section 3 presents the EDZL algorithm and its fundamental properties in multiprocessor processor systems. In Section 4, we discuss the schedulability of EDZL for 2 processors, and in Section 5, for \( m \geq 3 \) processors. In Section 6, we discuss the optimality of EDZL for task sets with some constraints. The conclusion of this paper is presented in Section 7.

2. Related Work

2.1 Task Model

In this paper, a real-time system with \( m \) processors platform is assumed to have \( n \) periodic tasks sorted by increasing period lengths. \( \tau_i \) denotes the task with the \( i \)-th smallest period. Each task \( \tau_i = (c_i, p_i) \) is defined by two parameters: execution time \( c_i \) and period \( p_i \) (or relative deadline). Each task has a utilization \( u_i = c_i / p_i \), and the total utilization of a task set \( T \) is defined as \( U(T) = \sum_{i=1}^{n} u_i \). A set of tasks is said to be schedulable using a scheduling algorithm if and only if there exists a valid schedule in which no deadlines of tasks will be missed.

A task generates a sequence of jobs. A ready job running on a processor is called the current job. During run time, a job of task \( \tau_i \) can be characterized by \( J_i = (r_i, e_i(t), t_i(t), d_i) \), where \( r_i \) and \( d_i \) are the ready time and deadline, \( e_i(t) \) and \( t_i(t) \) are the remaining execution time and the remaining time to deadline at time \( t \). When job \( J_i \) is just released at \( r_i \), \( e_i(t_i)=c_i \) and \( t_i(t_i)=p_i \). The laxity value of a job \( J_i \) at time \( t \) is defined by \( L_i(t)=t_i(t) - e_i(t) \). The laxity of a job provides a measure of urgency for this job. A negative laxity indicates that the job can never meet its deadline, while a zero laxity indicates that the job must be started immediately otherwise it will miss deadline.

2.2 Previous Result

For multiprocessor system, a variation of EDF algorithm called EDF-US[\( m/(2m-1) \)] has been studied in [9]. Using EDF-US[\( m/(2m-1) \)], all periodic tasks with a utilization higher than \( m/(2m-1) \) are assigned the highest priority in the system while the rest of the periodic tasks are assigned priorities according to the EDF scheduler. At run-time, jobs are scheduled according to their dynamic priorities whenever a processor is free or a new job arrives. It is shown that EDF-US[\( m/(2m-1) \)] can correctly schedule all jobs on \( m \) processors if the total utilization \( U(T) \) satisfies the condition: \( U(T) \leq m^2/(2m-1) \).

Goossens et al. [4] define priority-driven scheduling as that each job has a fixed priority; both EDF and RM can be classified as priority-driven policies. They show that a system of periodic tasks with \( m \) processors using EDF scheduling can meet all its deadlines if the total utilization \( U(T) \) satisfies the condition: \( U(T) \leq m - u_{\text{max}} \ast (m-1) \) where \( u_{\text{max}} \) is the largest utilization of any task in the system. In addition, it has been shown that no priority-driven scheduler can guarantee a correct scheduling of all periodic tasks on \( m \) processors if the total utilization \( U(T) > (m+1)/2 \) for large \( m \). These two results place a utilization upper bound of 0.5 on any priority-driven scheduler (including EDF).

[1] derives a schedulability test for preemptive deadline scheduling of periodic or sporadic real-time tasks on \( m \) processor system based on EDF. It also analyzes the class of algorithm EDF-US and concludes that EDF-US[\( 1/2 \)] is optimal, with a guaranteed schedulable utilization of \( (m+1)/2 \).

[7] explores the utilization bound of EDF with task partitioning policies. The basic periodic task model is extended to include low release jitter values, aperiodic tasks, non-preemptive models and etc. They prove that the utilization bound of EDF with any reasonable allocation algorithm is in the following interval:

\[
\left[ m-(m-1)\ast u_{\text{max}}, \left( \left\lfloor \frac{1}{u_{\text{max}}} \right\rfloor \ast m+1 \right) / \left( \left\lfloor \frac{1}{u_{\text{max}}} \right\rfloor + 1 \right) \right]
\]

All of the above studies suggest that \( u_{\text{max}} \) is an important factor on deriving schedulability bounds.

3. Earliest Deadline First until Zero Laxity (EDZL) Algorithm

3.1 EDZL Algorithm

The EDZL algorithm integrates EDF and LLA. EDZL uses EDF as long as there is no urgent job (that has a zero laxity) in the system. When the laxity of a job becomes zero, a current job with a positive laxity that has the latest deadline among all current jobs is preempted by this job. If the system is overloaded, i.e. no current job with a non-zero laxity exists, the scheduler will discard this job. When ties occur among current jobs, EDZL chooses the job with the smallest computation time. The EDZL algorithm is shown in Figure 1.

Here is an example (Figure 2) to illustrate the schedule using the EDZL algorithm. Given a set of periodic tasks \( \{\tau_1 = (2, 3), \tau_2 = (2, 3), \tau_3 = (2, 3)\} \). The task set is not schedulable on two processors using EDF (Figure 2(a)). But it is schedulable using EDZL (Figure 2(b)). At \( t=1 \), \( \tau_2 \) becomes a zero-laxity job and preempt \( \tau_3 \). \( \tau_2 \) resumes execution at \( t=2 \). All three jobs meet their deadlines since the priority of \( \tau_3 \) is promoted to a higher level than \( \tau_1 \) and \( \tau_2 \) at \( t=1 \).
3.2 Properties of EDZL Algorithm.

In this subsection, we describe some general properties of EDZL in multiprocessor systems.

**Definition 1.** A schedule of a task set $T$, which is produced using the EDZL algorithm is called an EDZL schedule.

EDZL is a work-conserving [5] scheduling algorithm in that it never leaves any processor idle intentionally if there are jobs waiting to be executed. Therefore, we can prove the following lemma.

**Lemma 1.** In an $m$-processor system using the EDZL scheduling algorithm, if a job $J$ misses its deadline at time $t$ and no jobs miss their deadlines before time $t$, the system must have $m+1$ jobs with zero-laxity simultaneously some time before $t$.

**Proof.** If the job $J$ misses its deadline at $t$, there must be a time $t' < t$ when $J$ has zero-laxity but cannot start execution immediately. According to EDZL, any job with zero-laxity will have the highest priority and can preempt any positive-laxity job. Since job $J$ cannot be executed at $t'$, there must have $m$ zero-laxity jobs in addition to $J$. Therefore, there are $m+1$ zero-laxity jobs at $t'$.

**Definition 2.** Priority promotion. Priority promotion is an action that promotes a job to the highest priority level when the laxity of the job becomes zero.

**Lemma 2.** An EDZL schedule is the same as an EDF schedule if and only if there is no priority promotion in the EDZL schedule.

**Proof.**

$\Rightarrow$ If the EDZL schedule is equal to an EDF schedule, then all tasks are scheduled by their deadlines, therefore, no priority promotion occurs in EDZL schedule.

$\Leftarrow$ We can show this by definition 2, since there is no priority promotion in the EDZL schedule, the priority of each task is based on its deadline. Therefore, the schedule is the same as the EDF schedule.

**Lemma 3.** For multiprocessor systems, if a task set $T$ can be scheduled by EDF algorithm, then $T$ can be scheduled by EDZL algorithm.

**Proof.** Let $S$ be an EDF schedule of task set $T$, and the task set is schedulable. Now, we switch the scheduling algorithm to EDZL. Since $T$ is schedulable by EDF, there is no priority promotion in schedule. Therefore, $T$ can be scheduled by EDZL algorithm.

4. Schedulability of EDZL on Two-Processor Systems

In this paper, we first explore the schedulability of EDZL on two-processor systems by considering task sets with different $\mu_{\text{max}}$ values where $\mu_{\text{max}}$ is the largest utilization of any task. In Section 4.1, we discuss the utilization bound of EDZL algorithm where $\mu_{\text{max}} < 1/2$ and in Section 4.2 the case that $\mu_{\text{max}} \geq 1/2$. We then summarize the utilization bound of EDZL for two-processor systems in Section 4.3.

4.1 $\mu_{\text{max}} < 1/2$

In this subsection, Lemma 4 provides a proof that EDZL has the same utilization bound as EDF if $\mu_{\text{max}}$ of the task set is less than 1/2.

**Lemma 4.** On two-processor systems, $T$ is a set of periodic tasks with $\mu_{\text{max}} < 1/2$, then $T$ is schedulable by EDZL if $U(T) \leq 2 - \mu_{\text{max}}$.

**Proof.** The reference papers [1] [4] show that $T$ can be scheduled by EDF, since the schedulability bound of EDF is $m*(1-\mu_{\text{max}})+\mu_{\text{max}}$, i.e. $2-\mu_{\text{max}}$. And by lemma 3, a task is schedulable by EDZL if the task set is schedulable by EDF. Therefore, if $U(T) \leq 2 - \mu_{\text{max}}$ and
$u_{\text{max}} < 1/2$, the task set $T$ can be scheduled by EDZL. □

In Figure 3, we give an example to show that the bound is tight. Given a task set with five tasks, $\{\tau_1=(2,5), \tau_2=(2,5), \tau_3=(1,5), \tau_4=(1.001,5), \tau_5=(802,2000)\}$. The $u_{\text{max}}=0.401$ and the total utilization of this example is 1.6012, which is only slightly larger than the bound in Lemma 4. We now show that this task set is not schedulable. As in the figure, the schedule is very regular. In every 5 units time, the remaining execution time of $\tau_5$ is decreased by 2. At $t=1996$, $\tau_5$ becomes zero-laxity, and has the highest priority and starts to execute. At the same time, $\tau_3$ is suspended until $t=1997$. At $t=1998$, $\tau_1$ starts to execute, but $\tau_1$ is preempted by $\tau_4$ at $t=1998.999$, since $\tau_4$ becomes zero-laxity. At $t=1999.999$, $\tau_3$ becomes zero-laxity but cannot be scheduled, since $\tau_4$ and $\tau_5$ also have zero-laxity. $\tau_5$ misses its deadline in this example.

![Figure 3. Example for the tightness of EDZL utilization bound where $u_{\text{max}} < 1/2.$](image)

### 4.2 $u_{\text{max}} \geq 1/2$

To simplify our analysis on the EDZL algorithm, we introduce a transformation procedure, called job normalization. In the original EDZL schedules, the scheduler will not always schedule a specific task on a specific processor. Therefore, the schedule is hard to analyze. We propose in this subsection an algorithm to post-process a schedule $S$ of a task set $T$, which is scheduled under EDZL policy, i.e., an EDZL schedule, such that the execution interval of all jobs is maintained while the schedule is simplified. The normalization algorithm is shown in Figure 4.

After the job normalization, we may say that job $J_i$ (task $\tau_i$) is always scheduled on processor $MP_j$. Based on the same idea, we can always schedule a subset of task set $T$ with $k$ tasks on $k$ processors if $k \leq m$, here $m$ is the number of processors in the system.

Here is an example to show the Job Normalize algorithm. Suppose that there are four periodic tasks $\tau_1=(3,10)$, $\tau_2=(6,20)$, $\tau_3=(6,15)$, and $\tau_4=(18,30)$. The schedule of these tasks is shown in Figure 5(a). After applying the Job_Normalize algorithm to move all jobs of $\tau_4$ to $MP_1$, the new schedule is shown in Figure 5(b).

#### Algorithm Job_Normalize

**Input:** a job $J_i$ of task $\tau_i$ in $T$, a selected processor $MP_j$, and an EDZL schedule $S$.

**Output:** new EDZL schedule $S'$.

1. Let $S'=S$;
2. 
   foreach time unit $u_e$ of job $J_i$ scheduled in $S'$
   if $u_e$ is not scheduled on $MP_j$ then
      interchange $u_e$ with the time unit at $MP_j$ in $S'$;
   end if
3. end foreach
4. return $S'$;

![Figure 4. Job_Normalize Algorithm](image)

**Lemma 5.** On two-processor systems, a task set $T$ can be scheduled by EDZL, if $U(T) \leq 1+u_{\text{max}}$.

**Proof.** Given a task set $T$, and two processors denoted as $MP_1$, $MP_2$. Assume that there are $n$ tasks in $T$ and $\tau_{\text{max}}$ is the task with the maximum utilization $u_{\text{max}}$ among all tasks in $T$. Let $S$ be a schedule of a task set $T$, which is scheduled under EDZL policy, then $\tau_{\text{max}}$ can be scheduled on the first processor $MP_1$ in $S$ by the Job_Normalize algorithm. We need to check whether all tasks in $T$ can meet their deadlines. Let $T'=T-\{\tau_{\text{max}}\}$, i.e., $T'$ contains all tasks in $T$ except $\tau_{\text{max}}$. Since, $U(T) \leq 1+u_{\text{max}}$, $U(T') \leq 1$, it is obvious that $T'$ can be scheduled by EDZL on a single processor. Since $\tau_{\text{max}}$ is scheduled on the first processor $MP_1$ in $S$ by the Job_Normalize algorithm the computing capacity of $MP_2$ is reserved only for $T'$, therefore, all tasks in $T'$ would not miss their deadlines. Next, we check if the task $\tau_{\text{max}}$ would miss its deadline in the schedule $S$. Let’s examine each time unit in each period of $\tau_{\text{max}}$ in $S$. If there exists a task $\tau_i$ with deadline smaller than $\tau_{\text{max}}$, then $\tau_i$ would be scheduled before $\tau_{\text{max}}$ on $MP_1$. In this case, $\tau_{\text{max}}$ may enter into zero-laxity status. However, $\tau_{\text{max}}$ would be assigned highest priority and no task can preempt $\tau_{\text{max}}$ on $MP_1$ while $\tau_{\text{max}}$ is in zero-laxity.

![Figure 5. An example of job normalization](image)
therefore, $\tau_{\text{max}}$ would not miss its deadline. Since no tasks miss their deadlines in schedule $S$, the task set $T$ can be scheduled by EDZL, while $U(T) \leq 1 + u_{\text{max}}$. □

The utilization bound of EDZL on two-processor systems with $u_{\text{max}} \geq 1/2$ is tight. Suppose there are three tasks $\tau_1$, $\tau_2$, $\tau_3$. They are defined as follows.

$$
\tau_i = \begin{cases} 
(\alpha^* m^* u_{\text{max}}, \alpha^* m) & \text{if } i = 1, 2 \\
((\beta - 1)^* \alpha^* m^*(1 - u_{\text{max}}) + \alpha^* m, \beta^* \alpha^* m) & \text{if } i = 3
\end{cases}
$$

The parameters $\alpha$ and $\beta$ are reasonably large integers such that $\alpha$ and $\beta > m$. In this case, the total utilization is slightly larger than $3/2 + u_{\text{max}} - 1/2|$. Before $t = (\beta-1)^* \alpha^* m - (1 - u_{\text{max}}) \alpha^* m$, the laxity of $\tau_3$ is larger than 0. At time $t$, $\tau_1$ becomes zero-laxity and starts to execute by preempting $\tau_1$ or $\tau_2$. If $\tau_2$ is preempted by $\tau_1$, $\tau_2$ becomes zero-laxity at time $t' = \beta^* \alpha^* m^* u_{\text{max}}$. And then $\tau_1$ becomes zero-laxity at $t'' = \beta^* \alpha^* m^*(2^* u_{\text{max}} - 1)$. At this time, all three jobs have zero-laxity and cannot be scheduled on 2 processors. In Figure 6, we show an example with three tasks, $\tau_1 = (5.01, 10)$, $\tau_2 = (5.01, 10)$, $\tau_3 = (5005, 5010)$, which are constructed with $\alpha = 5$, $\beta = 501$, $u_{\text{max}} = 0.501$. The utilization of this example is 1.502, which is only slightly larger than the bound in Lemma 5. In this example, the laxity of $\tau_3$ is decreased by 4.99 every 10 time units. At $t = 4995.01$, $\tau_3$ becomes zero-laxity and has a higher priority than $\tau_1$ and starts to execute. At $t = 5004.99$, $\tau_2$ preempts $\tau_1$ and starts to execute, since $\tau_2$ becomes zero-laxity. At $t = 5009.98$, $\tau_1$ becomes zero-laxity but can’t be scheduled, since both $\tau_2$ and $\tau_3$ also have zero laxity. $\tau_1$ misses its deadline in this example.

![Figure 6. An example to show the tightness of EDZL utilization bound when $u_{\text{max}} \geq 1/2$.](image)

4.3 General Utilization Bound of EDZL on Two Processors

Using results from the above, we present a general EDZL utilization bound on two-processor systems.

**Theorem 1.** On two-processor systems, a task set $T$ can be successfully scheduled by EDZL if $U(T) \leq 3/2 + |u_{\text{max}} - 1/2|$. 

**Proof.** To show the utilization bound of EDZL on two-processor systems, we consider two cases.

Case 1: $u_{\text{max}} < 1/2$. From Lemma 4, $T$ can be scheduled by EDZL if the total utilization is $2 - u_{\text{max}}$ at most. Since $u_{\text{max}} < 1/2$, $2 - u_{\text{max}} = 3/2 + 1/2 - u_{\text{max}} = 3/2 + 1/2 - u_{\text{max}} = 3/2 + u_{\text{max}} = 3/2 + 1/2 - u_{\text{max}}$. □

Case 2: $u_{\text{max}} \geq 1/2$. From Lemma 5, $T$ can be scheduled by EDZL if the total utilization is $1 + u_{\text{max}}$ at most. Since $u_{\text{max}} \geq 1/2$, $1 + u_{\text{max}} = 3/2 + u_{\text{max}} - 1/2 = 3/2 + u_{\text{max}} - 1/2$.

Figure 7 shows the improvement of the EDZL bound compared to that of EDF as a function of $u_{\text{max}}$. The metric is defined by:

$$\text{Improvement} = \frac{U_{\text{EDZL}} - U_{\text{EDF}}}{U_{\text{EDF}}}$$

where $U_{\text{EDZL}}$ and $U_{\text{EDF}}$ are the total utilization bounds of EDZL and EDF, respectively.

![Figure 7. Improvement of EDZL over EDF](image)

5. EDZL on Three or More Processors

We now study the EDZL schedulability bound on $m$-processor systems where $m \geq 3$.

5.1. Schedule $m+1$ Tasks on $m$ Processors

In this subsection, we study the case that the task set has $m+1$ periodic tasks running on $m$ processors.

**Theorem 2.** Let $T$ be a set of $m+1$ tasks such that the ratio of the largest period and the smallest period is less than or equal to 2. Then $T$ is schedulable by EDZL algorithm on $m$ processors if the total utilization is less than or equal to $m - 2$.

**Proof.** Given a task set $T = \{\tau_1, \tau_2, \ldots, \tau_{m+1}\}$. Without loss of generality, we assume that $p_1 \leq p_2 \leq \ldots \leq p_{m+1}$. Therefore, $p_{m+1} \leq 2^* p_1$. According to lemma 1, if the set of tasks is not schedulable by EDZL, there exists a time instant, denoted by $t_d$ when all tasks must be in zero-laxity status at that time instant. If we schedule task $\tau_{i+1}$ on processor $MP_i$ for $1 \leq i \leq m$, by the
algorithm Job Normalize, then only jobs of task \( \tau_i \) can be migrated between different processors. Let’s consider the time interval \([t_{j-1} - p_{m+1}, t_j]\) and a task \( \tau_j \), here \( j \neq 1 \). \( \tau_i \) is exclusively scheduled on \( MP_{j,1} \). The only other task that can be scheduled on \( MP_{j,1} \) is \( \tau_i \). Since \( \tau_i \) is in zero-laxity status, it means that the amount of processing time of \( \tau_j \) scheduled on processor \( MP_{j,1} \) is at least \( p_j \) \((1-u_j)\) over the time interval \([t_{j}-p_{m+1}, t_j]\).

Therefore, to make task \( \tau_2 \) to \( \tau_{m+1} \) entering the zero laxity status simultaneously, the total processing time needed of \( \tau_1 \) is computed as follows:

\[
\sum_{j=2}^{m+1} p_j \cdot (1-u_j)
\]

Because \( p_1 \leq \ldots \leq p_{m+1} \),

\[
\sum_{j=2}^{m+1} p_j \cdot (1-u_j) \geq \sum_{j=2}^{m+1} p_1 \cdot (1-u_j)
\]

But the maximum processing time for \( \tau_1 \) in the time interval \( [t_{j-1} - p_{m+1}, t_j] \) is at most \( p_1 \cdot (m - \sum_{j=2}^{m+1} u_j) \).

Therefore, the set of tasks is schedulable if

\[
2 \cdot p_1 \leq p_1 \cdot (m - \sum_{j=2}^{m+1} u_j)
\]

\[
2 \leq m - \sum_{j=2}^{m+1} u_j
\]

Since the total utilization of all tasks is \( u_1 + u_2 + \ldots + u_{m+1} = u_1 + (u_2 + u_3 + \ldots + u_{m+1}) \)

\[
2 \leq m - U(T) + u_1
\]

Thus, the set of tasks is schedulable if total utilization \( U(T) \leq m - 2 + u_1 \). In other words, the utilization bound is \( m - 2 \) when \( u_1 \to 1/\infty \).

**Corollary 1.** For a set of \( m+1 \) tasks scheduled on \( m \) processors, if the ratio of the largest period and the smallest period is less than or equal to \( k \), where \( k > 1 \) is a constant number. The set of tasks is schedulable if the total utilization of this task set is less than or equal to \( m-k \).

**Proof.** As proved in Theorem 2, the set of tasks is schedulable if

\[
k \cdot p_1 \leq p_1 \cdot (m - \sum_{j=2}^{m+1} u_j)
\]

\[
k \leq m - U(T) + u_1
\]

Therefore, the set of tasks is schedulable if total utilization \( U(T) \leq m - k + u_1 \).

We have an example to show that the upper bound of total utilization \( U(T) \) in such a system is smaller than or equal to \( m - 1/2 \), if the periods of tasks are not constrained. Given a task set \( T \), where \( T=\{\tau_i\}, 1 \leq i \leq m+1 \). Let \( u_1 = 1, 4 \leq j \leq m+1, \) and assume that \( \tau_i=(5.01, 10), \tau_j=(5.01, 10), \tau_j=(2505, 5010) \), which are same with the tasks as we show in Figure 6. The total utilization of task set \( T \) is \( m-1/2+0.002 \), which is slightly higher than \( m-1/2 \).

The above result is valid only for systems with \( m+1 \) periodic tasks running on \( m \) processors. Unfortunately, the result is not applicable in the general case, which we discuss below.

### 5.2. General Bound on \( m \) Processors

For the general case, it has been shown that a set of periodic tasks can be scheduled by EDZL algorithm on \( m \) processor system, if the total utilization is less than or equal to \((m+1)/2\) [8]. However, authors in [8] suggest that the general bound of EDZL is higher than \((m+1)/2\). In this subsection, we show the study from [10] that constructs unschedulable task sets to derive an upper bound of the achievable utilization for EDZL.

Suppose there are 3 processors in a real-time system. Let \( T \) be a task set with two subsets \( T = T_2 \cup T_1 \). The subset \( T_2 = \{ z_1=(660, 1200), z_2=(56400, 84000) \} \) and \( T_1 = \{ \tau_1, \ldots, \tau_{121} \} \), where \( \tau_1 = \tau_2 = \ldots = \tau_{121} = 1 \). The total utilization of \( T_1 \) is \( U(T_1)=121/120 \approx 1.008 \) and \( U(T) \approx 2.2297 \). Figure 8 shows the schedule of task set \( T \).

![Figure 8. An example of a task set not schedulable by EDZL.](image)

In this example, the schedule pattern from time 0 to 120 is repeated until time 960. At time 1020, task \( z_1 \) is completed, both \( MP_1 \) and \( MP_2 \) are idle during [1020, 1080) and [1120, 1200). In the mean time, task \( z_2 \) has executed 800 time units in [0, 1200). At time 82720, the laxity of task \( z_2 \) becomes zero. It is promoted to the highest priority and executed on \( MP_3 \) without preemption till its deadline of time 84000. At time 82800, \( z_1 \) and all jobs in \( T \) become ready again. Since there are only two processors available at time 82800, the computing capacity reserved for \( z_1 \) is not enough and the laxity of \( z_1 \) becomes zero at time 83820. Similarly, \( z_1 \) is executed.
on processor MP2 without preemption until time 84000, and all jobs in $T_1$ released at time 83880 must be executed on MP1. As a result, one job in $T_1$ will miss deadline at time 84000, since $U(T_1)>1$. Therefore, EDZL fails to schedule this task set.

Similar to this example, we can construct a special task set that follows the rules given as below. There are three parameters: $(m, a, p_k)$, where $m$ is the number of processors, $1 \leq a \leq m-1$, and $p_k$ is an arbitrary positive integer. For example, the task set given in Figure 8 is defined with $m=3$, $a=1$, and $p_k=120$. Let $T$ be a special task set such that $T = T_2 \cup T_3$, $T_2 = \{z_i, z_2, \ldots, z_m\}$ and $T_3 = \{\tau_{i1}, \ldots, \tau_{ip_{k+1}}\}$. In $T$, each task $\tau_i = (1, p_k)$, $1 \leq j \leq pk^*$, the total utilization of $T_1$ is $a+1/p_k$. In $T_2$, each task $z_i$ has a utilization $u_i = i/(a+i)+f/p_i$, where $f_i = pk^*a/(a+1)$, $f_i = p_{a^2}\sum a/(a+i)$, $2 \leq i \leq m-a-2$, $f_{a^2}=p_{m-a^2}a/(m-1)$, and $f_{a^2}=p_{m-a^2}a/m$. Note that, $p_k$ is a factor of the period length $p_i$ of the task $z_i$, i.e., $p_k \mid p_i$ and the period length $p_i$ of $z_i$ is also a factor of the period length $p_{i^2}$ of $z_{i+1}$, $1 \leq i \leq m-a-1$, i.e., $p_k \mid p_{i^2} \ldots p_{m-a^2}$. From these rules, the utilization of $T$ can be simply derived as $U(T) = \min(a + \sum_i i/(a+i) + e \mid 1 \leq a \leq m-1$, where $e = 1/p_k + \sum_{i=a}^{m-1} i/(p_i)$.

Since, the value of $U(T)$ is varied with the value of $a$, we need to find the minimum value of $U(T)$ to derive upper bound. In the above equation, $e$ would approach 0 and $U(T) = \min(a + \sum_{i=a}^{m-1} i/(a+i) \mid 1 \leq a \leq m-1$, if $p_k$ is increased.

Moreover, the equation of $U(T)$ can be transformed to $m-a(H_m-H_a)$, where $H_i = 1/1 + 1/2 + \ldots + 1/i$ is the $i^{th}$ harmonic number. For large $m$, $H_m = \ln m + \gamma$, where $\gamma = 0.5772$. If $m$ approaches infinity, $U(T)$ has the minimum value when $a \rightarrow m/e$, $e$ is the Euler’s number. Therefore, there exists a set of tasks which is not schedulable by EDZL when $U(T) > m-a(H_m-H_a) = m-m/(\ln m + \ln (m/e))/e = m(1-1/e) \approx 0.6321m$.

**Theorem 3.** [10] The utilization bound of EDZL is no greater than $m(1-1/e)$.

Details on the construction of unschedulable task sets and the proof of theorem 3 are omitted here. They can be found in [10]. Figure 9 shows the upper bound of $m(1-1/e)$ and the lower bound of $(m+1)/2$, for $m=1\sim100$. There is still a gap between the unschedulable condition (at top) and the schedulable condition (at bottom). We will continue to work on closing the gap to find the exact schedulability condition for EDZL.

**6. The EDZL\_GCD Algorithm**

In this study, we are also interested in the following question: what is the constraint for a set of tasks such that EDZL will achieve the optimal bound $m$ for the $m$ processor system? In this section, we show that if all tasks have the same release time and deadline, the schedulable utilization bound for such a simple system is $m$. We also introduce the EDZL\_GCD algorithm to help develop such a system.

The tasks in $m$ processor system are assumed to have the same release time and deadline. The system will achieve the full schedulability in this case. Using this property, we can design an optimal scheduling algorithm EDZL\_GCD.

**Theorem 4.** If a set of jobs have the same release time and deadline and the total utilization is less than or equal to $m$, it is EDZL feasible, on $m$-processor system.

**Proof.** Without loss of generality, we may assume that all jobs are released at time $t$ and have to be completed at $t'$. Two cases need to be considered in the EDZL schedule:

Case 1: At least one unit of idle time exists in the time interval $[t, t')$. If there exists one unit of idle time, it means the number of jobs that need to be completed during the time interval $[t, t')$ is less than $m$. Therefore, by lemma 1, no job will miss its deadline. Since, there must have $m+1$ jobs entering the zero-laxity status before any job missing its deadline.

Case 2: No idle time exists in the time interval $[t, t')$. In this case, processors are full utilized and the total utilization is not greater than $m$, thus no jobs will miss its deadline.

By Theorem 4, we transform all tasks in a system to tasks with the same periods by computing the GCD of all original periods and shorten the periods of tasks to this GCD. Although the periods of tasks are changed, the utilizations are maintained by assigning their processing time properly. Then, we have a simple system. This system can be scheduled as long as the
total utilization is less than or equal to $m$. We show the EDZL_GCD algorithm in Figure 10.

**Algorithm EDZL_GCD**

1. Calculate $g$ which is the GCD of all periods of tasks;
2. foreach task $\tau_i = (c_i, d_i)$
3. Translate $\tau_i$ into $\tau'_i = (g*c_i/d_i, g)$;
4. end foreach
5. Use EDZL to schedule the new task set \{ $\tau'_i$ \};

**Figure 10. EDZL_GCD Algorithm**

We now show an example (Figure 11(a)) to show that, if the periods are the same for all tasks, the utilization bound of EDZL may reach $m$. Given a task set \{ $\tau_1 = (9.9, 10),$ $\tau_2 = (9.9, 10),$ $\tau_3 = (0.2, 10)$ \}. At time $t=9.8$, $\tau_3$ becomes zero-laxity and $\tau_2$ becomes zero-laxity at $t = 9.9$; at the same time $t=9.9$, $\tau_1$ finishes its processing. Therefore, all tasks can be completed by their deadlines.

However, if the periods are not the same for all tasks, the utilization bound will not reach $m$. Given a task set \{ $\tau_1 = (9.9, 10),$ $\tau_2 = (9.9, 10),$ $\tau_3 = (0.22, 11)$ \}. At $t=30.1$, $\tau_3$ becomes zero-laxity, and $\tau_1$ becomes zero-laxity at $t = 30.2$. At $t=32.98$, $\tau_1$ becomes zero-laxity. Therefore, the task set cannot be scheduled by EDZL on two-processor system as shown in Figure 11(b).

**Figure 11. An example to show (a) the optimal case and (b) the sub-optimal case.**

7. Conclusion

In this paper, we study the EDZL algorithm for multiprocessor systems. EDZL integrates EDF and LLA, and has low context switching overheads and low deadline miss ratios. The flexibility of priority promotion allows system resources to be better utilized for meeting deadlines.

We have presented that the EDZL utilization bound for two-processor systems is $3/2 + |u_{max}-1/2|$, and the upper bound of $m(1-1/e)$ while $m$ approaches infinity. We believe the lower bound of $(m+1)/2$ and upper bound $m(1-1/e)$ still have room to converge. The general bound may also be improved by considering the $u_{max}$ factor. A simple algorithm EDZL_GCD is also included as an extension of EDZL. EDZL_GCD is an optimal algorithm if task periods are multiples of each other.

**References**