Semi-Fixed-Priority Scheduling: New Priority Assignment Policy for Practical Imprecise Computation

Hiroyuki Chishiro¹, Akira Takeda², Kenji Funaoaka² and Nobuyuki Yamasaki¹

¹School of Science for Open and Environment Systems
Keio University, Yokohama, Japan
{chishiro,yamasaki}@ny.ics.keio.ac.jp
²Toshiba Corporation, Japan
akira10.takeda@toshiba.co.jp, funaoaka@isl.rdc.toshiba.co.jp

Abstract

This paper proposes semi-fixed-priority scheduling to achieve both low-jitter and high schedulability. Semi-fixed-priority scheduling is for the extended imprecise computation model, which has a wind-up part as a second mandatory part and schedules the part of each extended imprecise task with fixed-priority. This paper also proposes a novel semi-fixed-priority scheduling algorithm based on Rate Monotonic (RM), called Rate Monotonic with Wind-up Part (RMWP). RMWP limits executable ranges of wind-up parts to minimize jitter. The schedulability analysis proves that one task set is feasible by RMWP if the task set is feasible by RM. Simulation results show that RMWP has both lower jitter and higher schedulability than RM.

1. Introduction

Real-time scheduling algorithms have used Worst Case Execution Time (WCET) to schedule real-time tasks. However, the analysis of the WCET is difficult on current real-time systems due to both hardware and software complexities. Moreover, Actual Case Execution Time (ACET) in real-time systems such as autonomous mobile robots [6, 8] tends to change from time to time, because their behaviors depend on their environments. In order to make use of the remaining time (WCET - ACET), the imprecise computation model [13] was presented.

The imprecise computation model is one of the techniques used to cope with such uncertainty. The crucial point is that the computation is split into two parts: mandatory part and optional part. A mandatory part affects the correctness of the result and an optional part only affects the quality of the result. By restricting the execution of the optional part to only after the completion of the mandatory part, real-time applications based on the imprecise computation model can provide the correct output with lower quality, by terminating the optional part. However, the imprecise tasks require the processings to output the results to their actuators. When the imprecise tasks terminate or complete their optional parts, the imprecise computation model cannot guarantee to complete them by their deadlines. In order to overcome the weakness of the imprecise computation model, we use the extended imprecise computation model [10, 12] with a second mandatory part, called wind-up part.

Extended imprecise tasks for autonomous mobile robots such as MPEG decoder, object detection and path search can be adapted to the extended imprecise computation model because they must guarantee to complete their wind-up parts by their deadlines. In real-time scheduling of extended imprecise tasks, Mandatory-First with Wind-up Part (M-FWP) [10, 12] has high-jitter of the shortest period task due to Earliest Deadline First (EDF) [14] based dynamic-priority scheduling [5]. Hence, M-FWP cannot be adapted to autonomous mobile robots because the jitter-sensitive task such as the motor control task with the shortest period in autonomous mobile robots requires the minimized jitter to achieve precise motions. Unfortunately, fixed-priority scheduling such as Rate Monotonic (RM) [14] with low-jitter of the shortest period task cannot be adapted to the extended imprecise computation model because one task may miss its deadline due to the overrun of the optional part.

This paper proposes semi-fixed-priority scheduling to achieve both low-jitter and high schedulability. Semi-fixed-priority scheduling is for the extended imprecise computation model and schedules the part of each extended imprecise task with fixed-priority. This paper also proposes...
a novel semi-fixed-priority scheduling algorithm based on RM, called Rate Monotonic with Wind-up Part (RMWP). RMWP limits executable ranges of wind-up parts to minimize jitter. The schedulability analysis proves that one task is feasible by RMWP if the task set is feasible by RM. The effectiveness of RMWP is evaluated in Section 5.

The contribution of this paper is to achieve real-time computation, called semi-fixed-priority scheduling. We believe that RMWP limits executable ranges of wind-up parts to minimize jitter. The schedulability analysis proves that one task set is feasible by RMWP if the task set is feasible by RM. The effectiveness of RMWP is evaluated in Section 5.

Finally we offer concluding remarks in Section 6.

2. System Model

Figure 1 shows the extended imprecise computation model [10, 12]. The extended imprecise computation model adds the wind-up part to the imprecise computation model [13]. The imprecise computation model assumes that the processing to terminate or complete the optional part is not required. However, motor control tasks in autonomous mobile robots require the processing to output the results to their actuators. They must guarantee to complete them by their deadlines so that the extended imprecise computation model has the wind-up part.

We assume that the system has one processor and a task set \( \tau \) consisted of \( n \) tasks. Task \( \tau_i \) is represented as the following tuple \((T_i, D_i, OD_i, m_i, o_i, w_i)\): where \( T_i \) is the period, \( D_i \) is the deadline, \( OD_i \) is the optional deadline, \( m_i \) is the WCET of the mandatory part, \( o_i \) is the Required Execution Time (RET) of the optional part and \( w_i \) is the WCET of the wind-up part. The RET of each optional part tends to be underestimated or overestimated from time to time because autonomous mobile robots run in uncertain environments. The relative deadlines \( D_i \) of each task \( \tau_i \) is equal to its \( T_i \). The \( j^{th} \) instance of \( \tau_i \) is called job \( \tau_{i,j} \). The utilization of each periodic task is defined as \( U_i = (m_i + w_i)/T_i \). The reason why \( U_i \) does not include \( o_i \) is because the optional part of \( \tau_i \) is a non-real-time part so that completing it is no relevant to scheduling the task set successfully. Hence, the utilization of the system within \( n \) tasks can be defined as \( U = \sum_{i=1}^{n} U_i \). All tasks are ordered by priority and \( T_1 \leq T_2 \leq \ldots \leq T_n \) are ordered such that.

In addition, we define the following symbols as follows.

- \( \Gamma_s \): the group of successfully scheduled tasks
- \( o_{i,j} \): the actual case RET of \( \tau_{i,j} \)
- \( r_{i,j} \): the release time of \( \tau_{i,j} \)
- \( s_{i,j} \): the start time of \( \tau_{i,j} \)
- \( f_{i,j} \): the finishing time of \( \tau_{i,j} \)
- \( R_i(t) \): the remaining execution time of \( \tau_i \) at time \( t \)

We define jitter as Relative Finishing Jitter (RFJ) [4]. RFJ is the maximum deviation of the finishing time of two consecutive jobs:

\[
RFJ_i = \max_j |(f_{i,j+1} - r_{i,j+1}) - (f_{i,j} - r_{i,j})|.
\]

We describe RFJ in Figure 2. In this case, the RFJ of \( \tau_1 \) is the maximum of \((f_{1,2} - r_{1,2}) - (f_{1,1} - r_{1,1})\) and \((f_{1,3} - r_{1,3}) - (f_{1,2} - r_{1,2})\). Reducing jitter means reducing RFJ.

An optional deadline is a time when an optional part is terminated and a wind-up part is released. Each wind-up part is ready to be executed after each optional deadline and can be completed if each mandatory part is completed by each optional deadline. Figure 3 shows the optional deadline of each task. Solid up arrow, solid down arrow and dotted down arrow represent release time, deadline and optional deadline respectively. Task \( \tau_1 \) completes its mandatory part by \( OD_1 \) and executes its optional part until \( OD_1 \). After \( OD_1 \), then \( \tau_1 \) executes its wind-up part. In contrast, task \( \tau_2 \) does not complete its mandatory part by \( OD_2 \). When \( \tau_2 \) completes its mandatory part, \( \tau_2 \) starts to execute its wind-up part, not to execute its optional part.

3. Related Work

Scheduling with Liu and Layland’s model [14], called general scheduling, such as RM and EDF do not consider
ACET, which is usually less than WCET because WCET tends to be overestimated [7]. These algorithms cannot use the remaining time so that the imprecise computation model is an effective technique in uncertain environments.

In the imprecise computation model, Mandatory-First with Earliest Deadline (M-FED) [3] is based on EDF and has high-jitter of the shortest period task. Optimization with Least-Utilization [2] requiring known the WCET of each optional part cannot be adapted to autonomous mobile robots requiring that the WCET of each optional part is unknown.

Mandatory-First with Wind-up Part (M-FWP) [10, 12] adapts M-FED to the extended imprecise computation model. However, M-FWP as well as M-FED has high-jitter of the shortest period task. The jitter-sensitive task such as the motor control task with the shortest period in autonomous mobile robots requires the minimized jitter to achieve precise motions. Unfortunately, fixed-priority scheduling such as RM with low-jitter of the shortest period task cannot be adapted to the extended imprecise computation model because one task may miss its deadline due to the overrun of the optional part.

4. Semi-Fixed-Priority Scheduling

Semi-fixed-priority scheduling fixes the priority of each part in the extended imprecise task and changes the priority of each extended imprecise task only in the two cases: when the extended imprecise task completes its mandatory part and begins its optional part and when the extended imprecise task terminates or completes its optional part and begins its wind-up part. Semi-fixed-priority scheduling splits one extended imprecise task into two general tasks. The two general tasks have same periods and same or different release times, cannot be executed simultaneously and are scheduled by fixed-priority in Figure 4. Task $\tau_i^m$ and $\tau_i^w$ are the mandatory part and the wind-up part of $\tau_i$. The release times of the first jobs of $\tau_i^m$ and $\tau_i^w$ are 0 and $OD_i$ respectively. When there is no task which is ready to execute its mandatory or wind-up part, the optional part of each task is executed.

Figure 5 shows the difference between general scheduling such as RM and EDF with Liu and Layland’s model [14] and semi-fixed-priority scheduling. In general scheduling, when $\tau_i$ is released at 0, $R_i(t)$ is set to $m_i + w_i$ and monotonically decreasing until $R_i(t)$ becomes 0 at $m_i + w_i$. In semi-fixed-priority scheduling, when $\tau_i$ is released at 0, $R_i(t)$ is set to $m_i$ and monotonically decreasing until $R_i(t)$ becomes 0 at $m_i$. When $R_i(t)$ is 0 at $m_i$, then $\tau_i$ sleeps until $OD_i$. When $\tau_i$ is released at $OD_i$, then $R_i(t)$ is set to $w_i$ and monotonically decreasing until $R_i(t)$ becomes 0 at $OD_i + w_i$. If $\tau_i$ does not complete its mandatory part by $OD_i$, then $R_i(t)$ is set to $w_i$ at the time when $\tau_i$ completes its mandatory part. In both schedulings, $\tau_i$ completes its wind-up part by $D_i$.

4.1 RMWP Algorithm

RMWP is one of semi-fixed-priority scheduling algorithms with the extended imprecise computation model to achieve both low-jitter and high schedulability. As shown
1. When $\tau_i$ becomes ready, set $R_i(t)$ to $m_i$, dequeue $\tau_i$ from SQ and enqueue $\tau_i$ to RTQ. If $\tau_i$ has the highest priority in RTQ, preempt the current task.

2. When $\tau_i$ completes its mandatory part:
   (a) If $OD_i$ expired, set $R_i(t)$ to $w_i$.
   (b) Otherwise set $R_i(t)$ to $\alpha_i$, dequeue $\tau_i$ from RTQ and enqueue $\tau_i$ to NRTQ. If there are one or multiple tasks in RTQ or NRTQ which have higher priority than $\tau_i$, preempt $\tau_i$.

3. When $\tau_i$ completes its optional part, dequeue $\tau_i$ from NRTQ and enqueue $\tau_i$ to SQ.

4. When $OD_i$ expires:
   (a) If $\tau_i$ is in RTQ and does not complete its mandatory part, do nothing.
   (b) If $\tau_i$ is in NRTQ, terminate and dequeue $\tau_i$ from NRTQ, set $R_i(t)$ to $w_i$ and enqueue $\tau_i$ to RTQ. If $\tau_i$ has the highest priority in RTQ, preempt the current task.
   (c) If $\tau_i$ is in SQ, dequeue $\tau_i$ from SQ, set $R_i(t)$ to $w_i$ and enqueue $\tau_i$ to RTQ.

5. When $\tau_i$ completes its wind-up part, dequeue $\tau_i$ from RTQ and enqueue $\tau_i$ to SQ.

6. When there are one or multiple tasks in RTQ, perform RM in RTQ.

7. When there is no task in RTQ and there are one or multiples tasks in NRTQ, perform RM in NRTQ.

**Figure 7. RMWP algorithm**

In Figure 6, RMWP manages three task queues: Real-Time Queue (RTQ), Non-Real-Time Queue (NRTQ) and Sleep Queue (SQ). RTQ holds tasks which are ready to execute their mandatory or wind-up parts in RM order. One task is not ready to execute its mandatory and wind-up parts simultaneously. NRTQ holds tasks which are ready to execute their optional parts in RM order. Every task in RTQ has higher priority than that in NRTQ. SQ holds tasks which complete their optional parts by their optional deadlines or wind-up parts by their deadlines. Figure 7 shows RMWP algorithm. RMWP executes seven scheduling events when their conditions are met. In order to execute these events, each task has the optional deadline. We next describe how to calculate the optional deadline and analyze the schedulability of RMWP with general and harmonic task sets.

### 4.2 RMWP with General Task Sets

An optional deadline is a time when an optional part is terminated and a wind-up part is released. Each wind-up part is ready to be executed after each optional deadline and can be completed if each mandatory part is completed by the optional deadline. Each optional deadline is set to the time as late as possible to expand the executable range of each optional part. The wind-up part of each task must not miss the deadline if the system is idle or executes lower priority tasks between the time when the mandatory part is completed and the wind-up part is released. In order to calculate the optional deadline, we first estimate the worst case interference time $I^i_k (i < k)$ which is the upper bound time when $\tau_k$ is interfered by $\tau_i$.

**Theorem 1** (Worst Case Interference Time by Higher Priority Tasks). The worst case interference time $I^i_k (i < k)$ which is the upper bound time when $\tau_k$ is interfered by $\tau_i$ is

\[
I^i_k = \left\lfloor \frac{T_k}{T_i} \right\rfloor (m_i + w_i).
\]

**Proof.** If the optional deadline of each task is equal to 0 in Figure 8, the interference time of task $\tau_k$ interfered by $\tau_i$ is equal to equation 1. Moreover, there is no case that the interference time of task $\tau_k$ interfered by $\tau_i$ is more than equation 1.

By theorem 1, we next calculate the optional deadline with general task sets.

**Theorem 2** (Optional Deadline with General Task Sets). The optional deadline $OD_k$ of task $\tau_k$ is

\[
OD_k = D_k - w_k - \sum_{i=0}^{k-1} I^i_k.
\]

**Proof.** It is clear that task $\tau_k$ completes its wind-up part by its deadline if $\tau_k$ completes its mandatory part by its optional deadline by theorem 1.

RMWP can calculate optional deadlines by theorem 2. In contrast, in M-FWP, analyzing that what job maximizes the worst case interference time $I^i_k$ is too complex due to dynamic-priority scheduling. Therefore, in order to calculate $OD_k$ easily, RMWP is a semi-fixed-priority scheduling algorithm and only considers tasks, priorities of which are higher than the priority of $\tau_k$. RMWP can delay the release time of the wind-up part until the time when the wind-up part does not miss the deadline if the mandatory part is

![Figure 8. Case of worst case interference time](image)
completed by the optional deadline. We next analyze that RMWP is at least as effective as RM.

**Theorem 3** (RMWP is at least as effective as RM). One task set is feasible by RMWP if the task set is feasible by RM.

*Proof.* This proof is shown by contraposition. We show that if one task set is not feasible by RMWP, the task set is not feasible by RM. By theorem 2, it is clear that \( T_i \) completes its wind-up part by its deadline if \( T_i \) completes its mandatory part by its optional deadline. Task \( T_i \) misses its deadline only if \( T_i \) executes its mandatory part after its optional deadline \( OD_i \). In this case, \( T_i \) executes its mandatory and wind-up parts continuously without executing its optional part. In RM, task \( T_i \) also misses its deadline because of executing its mandatory and wind-up parts continuously. Hence, this theorem holds. 

**Theorem 4** (Least Upper Bound of RMWP with General Task Sets). For a set of \( n \) tasks with semi-fixed-priority assignment, the least upper bound of RMWP with general task sets is \( U_{lab} = n(2^{1/n} - 1) \).

*Proof.* RMWP is at least as effective as RM by theorem 3 and generates the same schedule as RM in the case of worst case interference time by theorem 1. Therefore, the least upper bound of RMWP is the same as that of RM [14].

Figure 9 shows an example of schedule generated by RMWP and RM. The following task set \( \Gamma = \{ T_1 = (10, 10, 7, 3, 1, 3); T_2 = (15, 15, 1, 3, 1, 2) \} \) is scheduled by RMWP and RM in Figure 9(a) and 9(b) respectively. Each optional deadline is calculated by theorem 2. This example shows that there is at least one task set which is feasible by RMWP and RM. The following task set is non-feasible by RMWP. By theorem 4, it is clear that if one task set is not feasible by RMWP, the task set is not feasible by RMWP.

Figure 10. Pseudo code of RTA-OODH

**RTA – OODH(\( \Gamma \))**

```
while (\( \tau_k \in \Gamma \)) {
    \( A_k = D_k - w_k - \sum_{i=0}^{k-1} I_i \);
    \( I = 0 \);
    \( OD = I + A_k \);
    \( I = \sum_{i=0}^{k-1} \left( \left\lfloor \frac{OD}{T_i} \right\rfloor m_i + \left\lfloor \frac{OD - OD_k}{T_i} \right\rfloor w_i \right) \);
} while (\( I + A_k > OD \));
\( OD_k = OD \);
```

RMWP and is not feasible by RM. Moreover, in RMWP, job \( \tau_{1,1} \) and \( \tau_{1,2} \) executes its optional part in [14, 15] and [26, 27] respectively.

### 4.3 RMWP with Harmonic Task Sets

The optimal deadline by theorem 2 is pessimistic with general task sets. In order to improve the executable range of each optional part, we extend Response Time Analysis (RTA) [1] for Optimal Optional Deadline with Harmonic task sets (RTA-OODH). The optimal optional deadline \( OD_k \) of task \( \tau_k \) is defined as the time when the assignable time of \( \tau_k \) in \([OD_k, D_k]\) is equal to \( w_k \). That is to say, the optimal optional deadline is set to the time when the optional part of \( \tau_k \) is not terminated or discarded, though there is time to execute its optional part, if the ACET of \( \tau_k \) is always equal to its WCET. Also, we show that the optimal optional deadline by RTA-OODH is more than or equal to that by theorem 2. We first calculate the assignable time \( A_k \) of task \( \tau_k \) except \( w_k \) by theorem 1.

**Theorem 5** (Assignable Time with Harmonic Task Sets). The assignable time \( A_k \) of task \( \tau_k \) except \( w_k \)

\[
A_k = D_k - w_k - \sum_{i=0}^{k-1} I_i k.
\]

*Proof.* The hyperperiod in \( \tau_k \) and higher priority tasks than \( \tau_k \) is equal to \( T_k(D_k) \) with harmonic task sets. Moreover, the worst case interference time of each job is constant. Hence, the assignable time \( A_k \) of task \( \tau_k \) except \( w_k \) is equal to equation 3.

We next calculate the worst case interference time \( I_k \) of task \( \tau_k \) with harmonic task sets in \([0, OD_k]\).
Theorem 6 (Worst Case Interference Time in $[0, OD_k]$). The worst case interference time $I_k$ of $\tau_k$ in $[0, OD_k]$ is

$$I_k = \sum_{i=0}^{k-1} \left( \left\lfloor \frac{OD_k}{T_i} \right\rfloor m_i + \left\lceil \frac{OD_k - OD_i}{T_i} \right\rceil w_i \right).$$

Proof. We consider that one extended imprecise task $\tau_i$ is split into two general tasks $\tau_i^m$ and $\tau_i^w$ in Figure 4. The release time of each first job in $\tau_i^m$ and $\tau_i^w$ is equal to 0 and $OD_i$ respectively and the period of each task is equal to $T_i$. Hence, in $[0, OD_k)$, $\tau_k$ is interfered by $\tau_i^m$ in $\left\lfloor \frac{OD_k}{T_i} \right\rfloor$ times and by $\tau_i^w$ in $\left\lceil \frac{OD_k - OD_i}{T_i} \right\rceil$ times.

By theorem 5 and 6, we present RTA-OODH.

The optimal optional deadline $OD_k$ of task $\tau_k$ with harmonic task sets is

$$OD_k = A_k + I_k.$$  \hfill (4)

Proof. In equation 4, the optimal deadline $OD_k$ and the assignable time $A_k$ are the response time and the worst case execution time in RTA [1] respectively. The optional deadline by equation 4 is a time when the optional part of task $\tau_k$ is not terminated or discarded, though there is time to execute its optional part, if the ACET of $\tau_k$ is always its WCET. Moreover, the assignable time of each job is constant with harmonic task sets. Hence, the assignable time $A_k$ of $\tau_k$ except $w_k$ in $[OD_k, D_k)$ is equal to $w_k$ so that the optional deadline by equation 4 is optimal.

The assignable time $A_k$ of task $\tau_k$ except $w_k$ by equation 3 is equal to the optional deadline by equation 2 with harmonic task sets. It is clear that the optimal optional deadline by theorem 7 is more than or equal to the optional deadline by theorem 2. Figure 10 shows the pseudo code of RTA-OODH. RTA-OODH calculates the optimal optional deadline by the iteration, which is similar with RTA [1]. We next analyze the least upper bound $U_{lub}$ of RMWP with harmonic task sets by theorem 7.

Theorem 8 (Least Upper Bound of RMWP with Harmonic Task Sets). The least upper bound of RMWP with harmonic task sets is $U_{lub} = 1$.

Proof. By theorem 7, it is clear that task $\tau_k$ does not miss its deadline if $\tau_k$ completes its mandatory part by its optional deadline. Moreover, RMWP is at least as effective as RM by theorem 3 and generates the same schedule as RM in the case of worst case interference time by theorem 1. Therefore, the least upper bound of RMWP is equal to that of RM. That is to say, the least upper bound of RMWP with harmonic task sets is $U_{lub} = 1$.

Figure 11 shows an example of schedule by RMWP using RTA-OODH with the following task set $\Gamma = \{\tau_1 = (5, 5, 4, 1, 0, 1), \tau_2 = (10, 10, 8, 2, 0, 1), \tau_3 = (20, 20, 14, 2, 2, 2)\}$. Every $OD_i$ is calculated by theorem 7. For example, we calculate $OD_3$ of $\tau_3$ and the process of $OD_3$ through time is shown in the bottom part of Figure 11. The optional deadline $OD_3$ by theorem 2 is 4. In contrast, the optimal optional deadline $OD_3$ by theorem 7 is 14. Therefore, by the optimal optional deadline, job $\tau_{3,1}$ can execute its optional part in $[7, 8)$ and $[13, 14)$.

5. Simulation Studies

This section studies the effectiveness of RMWP using five performance metrics. These metrics describe details in Section 5.1. The simulation uses 1,000 task sets and compares RMWP with both RM and M-FWP. In autonomous mobile robots, there are tasks which have various periods. Therefore, the period $T_i$ of each task $\tau_i$ is selected within $[100, 200, 300, ..., 3000]$ with general task sets and $[100, 200, 400, 800, 1600, 3200]$ with harmonic task sets. Each $U_i$ is selected from $[0.02, 0.03, 0.04, ..., 0.25]$ and splits $U_i$ into two utilizations which are assigned to $m_i$ and $w_i$ respectively. The CPU utilization $U$ is selected from $[0.3, 0.35, 0.4, ..., 1.0]$. The simulation length of the $k^{th}$ task set is $H_k$ which is the hyperperiod of the $k^{th}$ task set.

The CPU utilization of $\alpha_{i,j}$ is within the range of $[\alpha_i - 0.05, \alpha_i + 0.05]$, where $\alpha_i$ is selected within $[0.1, 0.2, 0.3]$, except $w_k$ in $[OD_k, D_k)$ is equal to $w_k$ so that the optimal deadline by equation 4 is optimal.

The assignable time $A_k$ of task $\tau_k$ except $w_k$ by equation 3 is equal to the optional deadline by equation 2 with harmonic task sets. It is clear that the optimal optional deadline by theorem 7 is more than or equal to the optional deadline by theorem 2. Figure 10 shows the pseudo code of RTA-OODH. RTA-OODH calculates the optimal optional deadline by the iteration, which is similar with RTA [1]. We next analyze the least upper bound $U_{lub}$ of RMWP with harmonic task sets by theorem 7.

Theorem 8 (Least Upper Bound of RMWP with Harmonic Task Sets). The least upper bound of RMWP with harmonic task sets is $U_{lub} = 1$.

Proof. By theorem 7, it is clear that task $\tau_k$ does not miss its deadline if $\tau_k$ completes its mandatory part by its optional deadline. Moreover, RMWP is at least as effective as RM by theorem 3 and generates the same schedule as RM in the case of worst case interference time by theorem 1. Therefore, the least upper bound of RMWP is equal to that of RM. That is to say, the least upper bound of RMWP with harmonic task sets is $U_{lub} = 1$. 

Figure 11 shows an example of schedule by RMWP using RTA-OODH with the following task set $\Gamma = \{\tau_1 = (5, 5, 4, 1, 0, 1), \tau_2 = (10, 10, 8, 2, 0, 1), \tau_3 = (20, 20, 14, 2, 2, 2)\}$. Every $OD_i$ is calculated by theorem 7. For example, we calculate $OD_3$ of $\tau_3$ and the process of $OD_3$ through time is shown in the bottom part of Figure 11. The optional deadline $OD_3$ by theorem 2 is 4. In contrast, the optimal optional deadline $OD_3$ by theorem 7 is 14. Therefore, by the optimal optional deadline, job $\tau_{3,1}$ can execute its optional part in $[7, 8)$ and $[13, 14)$.
represented such as RMWP-10, RMWP-20 and RMWP-30, computed at every task release, because autonomous mobile robots run in uncertain environments so that each $o_{i,j}$ is fluctuated. If the CPU utilization of $o_{i,j}$ is always equal to 0, the result is represented as RMWP. Also, we consider the tasks, the ACETs of which tend to fluctuate from time to time in autonomous mobile robots so that we evaluate three cases where ACET/WCET is uniformly varies in the range of $[0.5, 1.0]$, $[0.75, 1.0]$ and 1.0.

5.1 Performance Metrics

We use five performance metrics to evaluate the effectiveness of RMWP from various perspectives. The performance metrics are defined as the following equations.

$$\text{Success Ratio} = \frac{\text{# of successfully scheduled task sets}}{\text{# of scheduled task sets}}$$

$$\text{Reward Ratio} = \frac{\sum_{i,j} \frac{R_{i,j}}{T_{i,j}}}{\text{# of tasks in successfully scheduled task sets}}$$

$$\text{Switch Ratio} = \frac{\text{# of successfully scheduled task sets}}{\text{# of context switches}}$$

$$\text{RFJ Ratio} = \frac{\sum_{i} \sum_{j} R_{i,j}}{\text{# of tasks in successfully scheduled task sets}}$$

$$\text{SPJ Ratio} = \frac{\sum_{i} \sum_{j} R_{i,j}}{\text{# of successfully scheduled task sets}}$$

In the above, task $\tau_i$ is in the group of $\Gamma_x$. If the success ratio of the CPU utilization is 0, the results of the CPU utilization evaluated by other performance metrics are omitted. Moreover, the results of RMWP-10, RMWP-20 and RMWP-30 in success ratio and RFJ ratio are the same as RMWP and those of M-FWP-10, M-FWP-20 and M-FWP-30 in success ratio are the same as M-FWP so that they are omitted. The reason why we evaluate not only RFJ ratio but also SPJ ratio is because the jitter-sensitive task such as the motor control task with the shortest period in autonomous mobile robots is important to achieve precise motions. In the evaluation with harmonic task sets, we do not show the result of the success ratio because it is clear that the least upper bounds of RMWP, M-FWP and RM are equal to 1 by theorem 8 and in [9] and [14] respectively.

5.2 Simulation Results with General Task Sets

Figure 12, 13, 14, 15 and 16 show simulation results with general task sets. In Figure 12(a), 12(b) and 12(c), the success ratio of M-FWP is always 1. In contrast, both success ratios of RMWP and RM drop when the CPU utilization is higher than 0.8. The success ratio of RMWP is always higher than or equal to that of RM by theorem 3. The range of ACET/WCET is wider and wider, the success ratios of RMWP and RM are improved.

In Figure 13(a), 13(b) and 13(c), M-FWP-10, M-FWP-20 and M-FWP-30 outperform RMWP-10, RMWP-20 and RMWP-30 respectively because RMWP sets each optional deadline statically and M-FWP calculates the assignable time of each optional part dynamically.

In Figure 14(a), 14(b) and 14(c), the switch ratio of RMWP is higher than that of RM because RMWP splits one extended imprecise task into two general tasks. In RMWP-10, RMWP-20 and RMWP-30, the RET of each optional part is more and more, the switch ratio drops because of executing each part continuously more frequently. The switch ratio of RMWP is higher than that of M-FWP because when the optional part of each task is completed by each optional deadline, the task sleeps until the optional deadline in RMWP. In contrast, in M-FWP, when the optional part of each task is completed, the wind-up part of each task is executed immediately.

In Figure 15(a), 15(b) and 15(c), RMWP outperforms RM and M-FWP by limiting executable ranges of wind-up parts. The RFJ ratio of M-FWP-10, M-FWP-20 and M-FWP-30 are dramatically the different results to the RFJ ratio of M-FWP because they calculate the assignable time of each optional part when each mandatory part is completed. If the RET of each optional part is more than 0, the optional part is ready to be executed. Otherwise the wind-up part is ready to be executed immediately. Therefore, the RFJ ratios of M-FWP-10, M-FWP-20 and M-FWP-30 are dramatically higher than that of M-FWP. On the other hand, RMWP can minimize jitter regardless of the assignable time of each optional part by each optional deadline and is lower jitter than both RM and M-FWP.

In Figure 16(a), 16(b) and 16(c), the SPJ ratios of M-FWP-10, M-FWP-20 and M-FWP-30 are also dramatically higher than M-FWP as the results in Figure 15. Interestingly, the SPJ ratio of M-FWP is lower than that of RM in Figure 16(b) and 16(c). In M-FWP, if the RET of each optional part is always 0, M-FWP generates the same schedule as EDF. RM dominates EDF in the jitter of the highest priority task [5], which is true only if ACET of each task is always equal to its WCET in Figure 16(a). If ACET of each task is less than or equal to its WCET in Figure 16(b) and 16(c), EDF dominates RM.

5.3 Simulation Results with Harmonic Task Sets

Figure 17, 18, 19 and 20 show simulation results with harmonic task sets. In Figure 17(a), 17(b) and 17(c), RMWP-10 slightly outperforms M-FWP-10 by theorem 7. Moreover, the reward ratios of RMWP-20 and RMWP-30 are the same as...
those of M-FWP-20 and M-FWP-30 respectively. Therefore, RMWP outperforms M-FWP with harmonic tasks sets though M-FWP outperforms RMWP with general task sets.

In Figure 18(a), 18(b) and 18(c), RMWP, M-FWP and RM are the ascending order of the switch ratio as well as the results in Figure 14.

In Figure 19(a), the RFJ ratios of RMWP and M-FWP and RM are the lowest. In Figure 19(b) and 19(c), the RFJ ratio of RMWP is the lowest and both the RFJ ratios of M-FWP and RM are higher than RMWP. Unfortunately, the RFJ ratios of M-FWP-10, M-FWP-20 and M-FWP-30 are dramatically higher than other algorithms.

In Figure 20(a), 20(b) and 20(c), it is interesting that the SPJ ratios of RMWP and M-FWP are the lowest in evaluated algorithms. M-FWP, which generates the same schedule as EDF only if the RET of each optional part is always 0, also dominates RM.

### 5.4 Discussion

Considering the simulation results, we discuss which algorithm is well suited to autonomous mobile robots. RM does not consider the case that the ACET of each task is less than its WCET. However, in autonomous mobile robots, the ACET of each task is usually less than its WCET so that RM is not well suited to autonomous mobile robots. In RMWP and M-FWP, we discuss the trade-offs between reward ratios and jitters to realize autonomous mobile robots. M-FWP is suited to applications requiring high schedulability with non-jitter sensitive tasks because M-FWP has higher success ratio than RMWP. In contrast, RMWP is suited to autonomous mobile robots having the jitter-sensitive task such as the motor control task with the shortest period in uncertain environments because not only is the jitter of RMWP not affected by the execution time of each optional part, but also RMWP has lower jitter than M-FWP. More-
over, the reward ratio of RMWP outperforms that of M-FWP by theorem 7 with harmonic task sets and the reward ratio of M-FWP outperforms that of RMWP with general task sets. Our goal is to achieve real-time scheduling with low-jitter and high schedulability for practical imprecise computation so that RMWP is well suited to autonomous mobile robots.

6. Concluding Remarks

This paper proposed semi-fixed-priority scheduling to achieve both low-jitter and high schedulability. Semi-fixed-priority scheduling is for the extended imprecise computation model and schedules the part of each extended imprecise task by fixed-priority. This paper also proposed a novel semi-fixed-priority scheduling algorithm, called RMWP. The advantage of RMWP is that RMWP is at least as effective as RM and has higher success ratio than RM and lower jitter than both RM and M-FWP. The disadvantage of RMWP is that RMWP has lower success ratio than M-FWP and higher switch ratio than M-FWP and RM. However, the jitter of RMWP is not affected by the execution time of each optional part. Therefore, RMWP is well suited to autonomous mobile robots. In future work, we will implement RMWP to RT-Frontier [11], a real-time operating system supporting the extended imprecise computation model.

References


Figure 18. Switch ratio with harmonic task sets

Figure 19. RFJ ratio with harmonic task sets

Figure 20. SPJ ratio with harmonic task sets


