JOURNAL OF KOREA MULTIMEDIA SOCIETY, VOL. 10, NO. 12, DECEMBER 2007(pp. 1741-1751)

Scheduling Algorithm to Minimize Total Error for Imprecise On-Line Tasks

Gi-Hyeon Song†

ABSTRACT

The imprecise computation technique ensures that all time-critical tasks produce their results before their deadlines by trading off the quality of the results for the computation time requirements of the tasks. In the imprecise computation, most scheduling problems of satisfying both 0/1 constraints and timing constraints, while the total error is minimized, are NP-complete when the optional tasks have arbitrary processing times. In the previous studies, the reasonable strategics of scheduling tasks with the 0/1 constraints on uniprocessors and multiprocessors for minimizing the total error are proposed. But, these algorithms are all off-line algorithms. Then, in the on-line scheduling, NORA(No Off-line tasks and on-line tasks Ready upon Arrival) algorithm can find a schedule with the minimum total error. In NORA algorithm, EDF(Earliest Deadline First) strategy is adopted in the scheduling of optional tasks. On the other hand, for the task system with 0/1 constraints, NORA algorithm may not suitable any more for minimizing total error of the imprecise tasks. Therefore, in this paper, an on-line algorithm is proposed to minimize total error for the imprecise real-time task system with 0/1 constraints. This algorithm is suitable for the imprecise on-line system with 0/1 constraints. Next, to evaluate performance of this algorithm, a series of experiments are done. As a consequence of the performance comparison, it has been concluded that IOSMTE(Imprecise On-line Scheduling to Minimize Total Error) algorithm proposed in this paper outperforms LOF(Longest Optional First) strategy and SOF(Shortest Optional First) strategy for the most cases.

Keywords: Performance Comparison, Selection Strategy, Imprecise On-Line Scheduling, Minimize Total Error, 0/1 Constraints, IOSMTE Algorithm

1. INTRODUCTION

The imprecise real-time system, proposed in [1], provides flexibility in scheduling time-critical tasks. Examples of its applications include image processing and tracking.

For some applications, execution of the optional parts is valuable only if they are executed completely before the deadline, and of no value if they are executed partially.

The systems with such imprecise tasks are called systems with 0/1 constraints.

Most scheduling problems of satisfying both 0/1 constraints and timing constraints, while the total error is minimized, are NP-complete when the optional tasks have arbitrary processing times [1]. By the total error, it means the sum of the processing times of all optional tasks that could not be scheduled.

In [1], Liu suggested a reasonable strategy of scheduling tasks with the 0/1 constraints on uniprocessors for minimizing the total error. This method schedules the first optional task with the longest processing time. This method is called as LOF(Longest Optional First) strategy. Song et al suggested a reasonable strategy of scheduling tasks with the 0/1 constraints on multiprocessors for minimizing the total error in [2]. The results

※ Corresponding Author : Gi-Hyeon Song, Address : (300-711) 77-3 Gayang 2-dong, Dong-gu, Daejeon Korea, TEL : +82-42-670-9292, FAX : +82-42-670-9290, E-mail : ghsong@hit.ac.kr

Receipt date : Apr. 13, 2007, Approval date : Aug. 27, 2007

† Associate Professor in MIS department at Daejeon Health Sciences College

of this paper show that the longest processing first selection strategy(LOF strategy) outperforms random or minimal laxity policy.

On the other hand, in the case of on-line scheduling[3-7], Shih and Liu proposed NORA(No Off-line tasks and on-line tasks Ready upon Arrival) algorithm which can find a schedule with the minimum total error for a task system consisting solely of on-line tasks that are ready upon arrival in [4]. But, for the task system with 0/1 constraints, it has not been known whether NORA algorithm can be optimal or not in the sense that it guarantees all mandatory tasks are completed by their deadlines and the total error is minimized. In NORA algorithm, the EDF(Earliest Deadline First) strategy[8] is adopted in the optional scheduling.

Furthermore, NORA algorithm is not designed for the task system with 0/1 constraints. So, NORA algorithm is not suitable for the task system with 0/1 constraints.

Hence, in NORA algorithm, the execution of some mandatory tasks with later deadlines may be postponed when trying to finish more optional tasks with earlier deadlines[4].

Therefore, in this paper, an on-line algorithm is proposed to overcome this shortage of NORA algorithm. Also, this algorithm is suitable for the imprecise on-line system with 0/1 constraints to minimize total error. Then, to evaluate the performance of this algorithm, a series of experiments are done. The aim of these experiments is to compare the total errors generated among the proposed algorithm and LOF(Longest Optional First) strategy and SOF(Shortest Optional First) strategy. The reason why LOF and SOF strategy are to be compared with the proposed algorithm is as follows.

In the optional scheduling of the imprecise real-time task system with 0/1 constraints, the method to select the next task to be scheduled is simple and restrictive. These selection methods include LOF, SOF and so on. LOF strategy has been known as a reasonable strategy of the imprecise real-time task system with the 0/1 constraints to minimize total error regardless of the number of processors in the case of off-line scheduling[1,2].

Next, SOF strategy, reverse to LOF strategy, needs to be compared.

So, in this paper, intensive simulations are done to compare total errors generated among the strategics. The processing times are randomly generated for the mandatory and optional parts of tasks.

The rest of this paper is organized as follows; Section 2 provides an imprecise on-line real-time task system model. In section 3, the related works are described.

Section 4 presents an imprecise on-line real-time scheduling algorithm for minimizing total error. The results of simulation and analysis are described in section 5. And section 6 concludes this paper.

2. IMPRECISE ON-LINE REAL-TIME TASK SYSTEM MODEL

In this section, an imprecise task model is described. here are given a set of n tasks, $T=\{T_1, T_2, T_3, \cdots\cdots T_n\}$. The tasks are preemptable.

Let an imprecise task T_i be composed of the mandatory part M_i and the optional part O_i, and characterized by its arrival time r_i, deadline d_i, and computational requirement m_i and o_i or M_i and O_i, respectively. Let P_i be the sum of m_i and o_i ($p_i = m_i + o_i$).

Each task T_i is characterized by the following parameters.

- Ready time (r_i) : the time instant at which T_i becomes ready for execution
- Deadline (d_i) : the time instant by which T_i has to be completed
- Processing time (p_i) : the time required to

execute the entire T_i

- Processing time of mandatory part (m_i) : the time required to execute the mandatory part of task T_i
- Processing time of optional part (o_i) : the time required to execute the optional part of task T_i

A schedule determines the amount of service time to be given to each task T_i during the schedulable interval which is defined as an interval between the task's arrival time and its deadline. If a scheduling algorithm assigns x_i units of execution time for task T_i, the error e_i of task T_i becomes $p_i - x_i$.

Total error can be defined as follows assuming that there are n tasks; $TE = \sum_{k=1}^{n} e_k$.

One effective way to minimize the bad effects of timing faults is to leave less important tasks unfinished if necessary. In other words, rather than treating all tasks equally, the system views important tasks as mandatory and less important tasks as optional. It ensures that all mandatory tasks are scheduled and executed to completion before their deadlines. Optional tasks may be left unfinished during a transient overload when it is not feasible to complete all the tasks. The imprecise computation technique uses this basic strategy but carries it one step further. In a system that supports imprecise computations, every time-critical task is structured in such a way that it can be logically decomposed into two subtasks: a mandatory subtask and an optional subtask[1]. On the other hand, in the imprecise computation technique, we note that one would gain no benefit by completing a sieve in part, while incurring the cost in processing that part. Therefore, an optional subtask that is a sieve should be either executed to completion before its deadline or is not scheduled, that is, discarded, entirely. In this case, we say that the execution of the optional subtask satisfies the 0/1 constraints[1]. In the imprecise tasks satisfying 0/1 constraints, the processing time p_i of some task T_i can be defined as $m_i + o_i$ or m_i.

3. RELATED WORKS

There are many different imprecise scheduling problems. These problems include minimization of total error[2,4,7,9], minimization of the maximum[3, 10] or average error, minimization of the number of discarded optional tasks, minimization of the number of tardy tasks, minimization of the weighted error[3,11] and minimization of average response time.

In this paper, the problem of scheduling imprecise computations to meet timing constraints and 0/1 constraints[12] is considered for minimizing total error. As expected, the general problem of scheduling to meet the 0/1 constraints and timing constraints as well as to minimize the total error, is NP-complete when the optional tasks have arbitrary processing times [1]. When the processing times of all optional tasks are equal, the DFS (Depth-First-Search) algorithm is optimal for scheduling tasks with timing constraints and the 0/1 constraints to minimize total error [1]. When the tasks have identical ready times, a simpler algorithm, called the LDF(Latest Deadline First) algorithm can be used to find optimal schedules [1]. A good strategy for scheduling tasks with the 0/1 constraints to minimize total error is to try to schedule first the optional tasks with long processing times regardless of the number of processors[1,2].

In an earlier paper[13], Shih and Liu proposed an algorithm(called Algorithm F) that can find optimal schedules of imprecise, preemptable tasks with arbitrary ready times and deadlines on a uniprocessor system.

But, these algorithms are all off-line algorithms. For the case of imprecise on-line scheduling to minimize total error[4,6,7], Shih and Liu proposed

NORA algorithm which can find a schedule with the minimum total error for a task system consisting solely of on-line tasks that are ready upon arrival in [4]. NORA algorithm is optimal in the sense that it guarantees all mandatory tasks are completed by their deadlines and the total error is minimized. Especially, NORA algorithm maintains a reservation list for all mandatory tasks that have arrived but are not yet completed and uses it as a guide in deciding where to schedule optional tasks and how much time to assign to them. So, NORA algorithm has a good schedulability performance for all mandatory tasks, but for the optional tasks with 0/1 constraints, it is doubtful whether NORA algorithm can produce minimum total error or not. In NORA algorithm, some error is produced as a result of the EDF(Earliest Deadline First) scheduling as the scheduler of NORA algorithm maintains a prioritized task queue in which tasks are ordered on the EDF basis. However, when trying to finish more optional tasks with earlier deadlines, the execution of some mandatory tasks with later deadlines may be postponed. This is the main cause of suboptimality of NORA algorithm[4].

Furthermore, NORA algorithm is not designed for the task system with 0/1 constraints. So, NORA algorithm is not suitable for the task system with 0/1 constraints.

Next, [11] suggests an optimal algorithm to search minimum total error for the imprecise on-line real-time task system with 0/1 constraints. But, this algorithm may cause high complexity in the worst case.

As a recent study, an imprecise on-line scheduling algorithm is proposed to minimize total error when the optional tasks with 0/1 constraints are scheduled in the ascending order of their processing requirement time[9]. This algorithm is designed to improve the defect of the previous NORA algorithm. But this algorithm is restricted to task scheduling order.

Therefore, in this paper, an on-line algorithm is proposed to minimize total error for the imprecise real-time task system with 0/1 constraints. This algorithm is suitable for the imprecise on-line system with 0/1 constraints.

Also, this algorithm can overcome the shortage of NORA algorithm and do not cause high complexity in the worst case as the algorithm in [7]. Also, this algorithm is not restricted to task scheduling order as the algorithm in [9].

4. MPRECISE ON-LINE TASK SCHEDULING ALGORITHM

In this section, an on-line scheduling algorithm for the imprecise real-time tasks with 0/1 constraints to minimize total error is described. The following Fig. 1 shows the overview of this algorithm. Initially, two optional task sets SOL (Scheduled Optional task List) and OL (Optional task List) are setted as empty and all optional tasks of the imprecise system respectively. Next, FT value is initialized as 0. The FT value denotes the schedule finishing time.

In this algorithm, at first, the procedure "SetSystemParameter" is performed. In this procedure, all system parameters which are used in generating the imprecise on-line real-time task system are determined. These parameters include *ARo* (ρ), *Amu*(μ), *Alamda*(λ), *PiDistribution*(*p*) and *Number Tasks*. The meaning of each parameter is explained in section 5. Next, from the "GenerateSystem" procedure, an imprecise on-line real-time task system can be generated randomly. The next "For loop" is performed whenever a task $T_i(1 \leq i \leq Number\ Tasks)$ arrived. Whenever an on-line task T_i is arrived, the "DetermineSchedulableTasks (i)" procedure determines the schedulable tasks in a time interval $[r_i, r_{i+1}]$, then the "SortTaskByDeadline" procedure sorts the schedulable tasks by deadlines in ascending order.

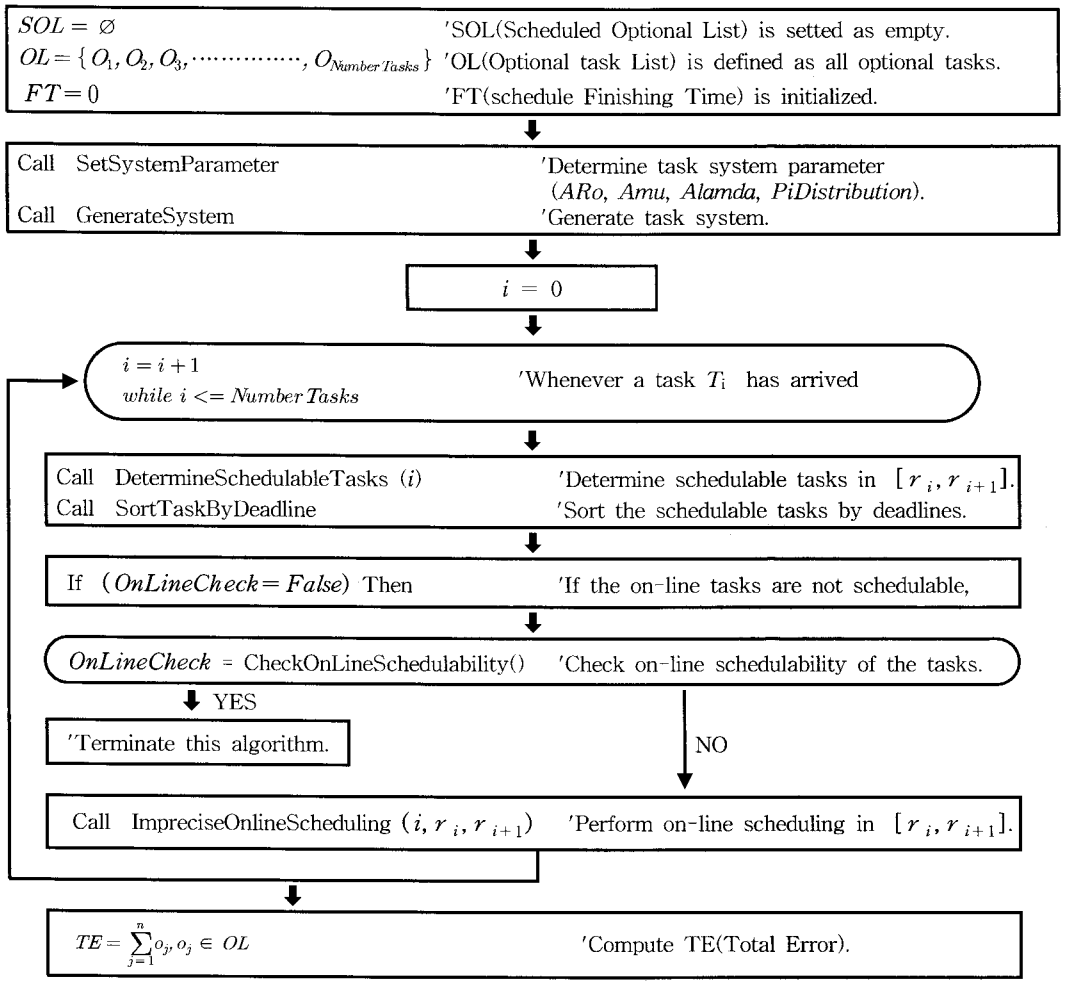


Fig. 1. Overview of the proposed algorithm.

Next, the on-line schedulability check function "CheckOnLineSchedulability()" is performed. This fuction checks the schedulability of the schedulable tasks on T_i arrival[11]. Even though only one task turned to be not schedulable, this algorithm is terminated abnormally. If the tasks are turned to be all schedulable, the final procedure "ImpreciseOnLineScheduling" can be performed in the time interval $[r_i, r_{i+1}]$.

The following Fig. 2 represents "ImpreciseOnlineScheduling" procedure. Hereafter, "ImpreciseOnlineScheduling" procedure is called as IOSMTE algorithm. The proposed IOSMTE algorithm works as follows; In the time interval $[r_i, r_{i+1}]$ on an task T_i's arrival, it schedules the mandatory parts of all schedulable tasks with the earliest deadline first. If any processing time is available after executing the mandatory parts of all tasks that can be scheduled then the algorithm executes the optional parts of the schedulable tasks in order to minimize total error. This scheduling order in some time interval $[r_i, r_{i+1}]$ is the main focus of the proposed IOSMTE algorithm. Therefore, in the optional scheduling of IOSMTE algorithm, the execution of some mandatory tasks with later deadlines may not be postponed due to the optional tasks with earlier deadlines.

In this optional scheduling, 0/1 constraints can

```
Sub ImpreciseOnlineScheduling (Tid, CurTime, Deadline)
Dim i, j, k, Task, Length, NbtPrevStep As Integer

================================== Mandatory Scheduling ==================================
For j = 1 To NbtPrevStep   'Initialize mandatory service time of all schedulable tasks in
[CurTime, Deadline].
  Task = ListTask(j)
  T Task.RunningYi = 0
Next j

Length = Deadline - CurTime        'Calculate a schedule length of the interval [CurTime, Deadline].

i = 1
While (i ≤ NbtPrevStep)            'For all schedulable tasks in [CurTime, Deadline]
  Task = ListTask(i)               'Select a task in ascending order of deadline.
  If (Length > m Task) Then        'If the schedule length is greater than m Task
    T Task.RunningYi = m Task      'Allocate m Task to T Task.
    Length = Length - m Task       'The schedule length is decreased by m Task.

  Else                             'If the schedule length is not greater than m Task
    T Task.RunningYi = Length      'Allocate the schedule length to T Task.
    Length = 0                     'The schedule length becomes zero.
  End If

  FT = Max2(FT, r Task)            'Set the schedule finishing time.
  FT = FT + T Task.RunningYi       'Increase the schedule finishing time by T Task's
                                                allocated mandatory service time.
    i = i + 1                      'Increase task index i by one.
Wend

Call ReComputeMi   'Decrease each mandatory computational requirement of all scheduled mandatory tasks
                   in the interval [CurTime, Deadline] by its allocated mandatory service time.

================================== Optional Scheduling ==================================
                                   'Schedule the optional tasks by the order of fast deadlines.
k = 1
while (k ≤ NbtPrevStep)            'For all schedulable tasks in [CurTime, Deadline]
  Task = ListTask(k)               'Select a task in ascending order of deadline.
  If (Length > o Task) Then        'If a remaining schedule length is greater than o Task
    SOL = SOL ∪ {o Task}           'Include o Task to SOL.
    Length = Length - o Task       'The remaining schedule length is decreased by o Task.
  End If
  k = k + 1                        'Increase task index k by one.
wend
End Sub
```

Fig. 2. 'ImpreciseOnlineScheduling' procedure.

be applied. Thus, in this paper, three different heuristic selection strategics are considered. These strategics include LOF, SOF and EDF strategy. Longest Optional First(LOF) strategy selects one task with the longest optional first among the schedulable optional tasks in the time interval.

Shortest Optional First(SOF) strategy selects one task with the shortest optional first.

Finally, Earliest Deadline First(EDF) strategy selects one optional task according to the order of fast deadline. The proposed IOSMTE algorithm adopts this strategy. The reason why LOF strategy is considered is that it has been known as a reasonable strategy for minimizing total error in the scheduling of imprecise off-line tasks with 0/1 constraints regardless of the number of processors[1,2]. Next, SOF strategy is considered to compare the performance of minimizing total error with LOF strategy.

Finally, EDF strategy is considered as it is adopted in the optional scheduling of NORA algorithm[4] which has been proved optimal in finding a schedule with the minimum total error for the imprecise on-line tasks without 0/1 constraints. The proposed IOSMTE algorithm adopts EDF strategy in the optional scheduling of imprecise on-line real-time tasks with 0/1 constraints. A major difference between IOSMTE algorithm and NORA algorithm is the existence of the 0/1 constraints. Another difference is that IOSMTE algorithm executes the optional parts of the schedulable tasks after executes all mandatory parts of the tasks.

On the other hand, in the overview of the algorithm which is depicted in Fig. 1, the number of iterations that the "For loop" is executed is bounded by $O(N)$, where N is total number of tasks in the imprecise task system. Next, the number of iterations that each procedure or function in the "For loop" is executed is bounded by $O(\log N)$ because the number of schedulable tasks which the "DetermineSchedulableTasks (i)" procedure determines at some task T_i arrival can be bounded by $\log N$. This value is denoted by "*NbtPrevStep*" in Fig. 2. Hence, the comflexity of "ImpreciseOnlineScheduling" in Fig. 2 is bounded by $\log N$. Eventually, the complexity of the proposed algorithm in Fig. 1 becomes $O(N\log N)$.

5. SIMULATION STUDY

In this section, the results of simulation are presented and analized. The aim of simulation is to compare performances among the selection strategics of optional parts. In order to compare the performances a series of experiments are performed.

5.1 Task Set Generation

For each experiment, a task set with three hundred tasks, modeled as an M/M/Infinity queuing system, in which the distribution characteristic of task arrival time is Poisson; the service time is exponentially distributed is generated. The processing time of mandatory part of each task is taken uniformly from zero to (its deadline - its ready time) * *PiDistribution*, where *PiDistribution*(*p*) is fixed arbitrary from 0.2 to 0.9 for each experiment. In this experiment, the deadline of each task is assumed by the sum of its arrival time(r_i) and processing time(p_i). The arrival rate over the service rate, (defined as *ARo*(ρ)) is the average number of tasks which can be scheduled in some time interval of the system, where *ARo*(ρ) is fixed arbitrary from 1 to 4 for each experiment. As *ARo* (ρ) or *PiDistribution*(*p*) becomes larger, the load of processor also becomes higher. If a generated task set is not schedulable by the CheckOnLineSchedulability() function of Fig. 1, it is rejected and regenerated until all mandatory tasks in the generated imprecise on-line task system are guaranteed. After the mandatory scheduling for the generated task set, the optional scheduling is performed. When the optional scheduling is performed, the 0/1 constraints can be adopted. In there, three different selection strategics are considered. LOF(Longest Optional First) strategy selects one task with the longest optional part among the schedulable optional tasks. This strategy is called as LOF_Scheduling. The second strategy selects one task with the shortest optional

part among the schedulable optional tasks. This strategy is called as SOF_Scheduling. The third strategy is the optional scheduling strategy of IOSMTE algorithm. This strategy is called as IOSMTE_Scheduling. To compare efficiencies among the selection strategics, the following metrices are used.

5.2 The Guarantee Ratio of Optional Tasks

The total errors of generated task set can be denoted as percentage. The percentage of total errors generated from LOF_Scheduling, SOF_Scheduling, and IOSMTE_Scheduling can be computed as follows respectively;

$$LOF\_ERR\_RATIO = (LOF\_ERR/TOT\_ERR)\times 100 \quad (1)$$

$$SOF\_ERR\_RATIO = (SOF\_ERR/TOT\_ERR)\times 100 \quad (2)$$

$$IOSMTE\_ERR = (IOSMTE\_ERR/TOT\_ERR)\times 100 \quad (3)$$

In this, *LOF_ERR*, *SOF_ERR* and *IOSMTE_ERR* denote the total error generated from LOF_Scheduling, SOF_Scheduling, and IOSMTE_Scheduling respectively.

TOT_ERR means the sum of processing times of all optional parts in a generated task set and this value can be represented as follows;

$$TOT\_ERR = \sum_{k=1}^{n} o_k,\ n = \text{the total number of tasks in a generated task set} \quad (4)$$

LOF_ERR_RATIO, *SOF_ERR_RATIO* and *IOSMTE_ERR_RATIC* denote the percentage of total error generated from LOF_Scheduling, SOF_Scheduling, and IOSMTE_Scheduling respectively. Then, the guarantee ratios of optional tasks in the LOF_Scheduling, SOF_Scheduling, and IOSMTE_Scheduling can be computed as follows;

$$G\_LOF = 100 - LOF\_ERR\_RATIO \quad (5)$$

$$G\_SOF = 100 - SOF\_ERR\_RATIO \quad (6)$$

$$G\_IOSMTE = 100 - IOSMTE\_ERR\_RATIO \quad (7)$$

In this, *G_LOF*, *G_SOF* and *G_IOSMTE* denote the guarantee ratios of optional tasks in the LOF_Scheduling, SOF_Scheduling, and IOSMTE_Scheduling respectively.

The first metric is the guarantee ratio of optional tasks. Fig. 3 and Fig. 4 depict the distributions for guarantee ratios of the optional tasks for $\rho = 2$, $p = 0.3$ and $\rho = 3$, $p = 0.9$ respectively. The x-axis represents the index of each task set which consists of 300 tasks and the y-axis its corresponding guarantee ratio.

From the Fig.s, it can be easily noted that IOSMTE_Scheduling outperforms LOF_Scheduling or SOF_Scheduling. But, for the special case of $\rho = 1$, the distributions of guarantee ratios generated from IOSMTE_Scheduling and LOF_Scheduling are similar as shown in Fig. 5.

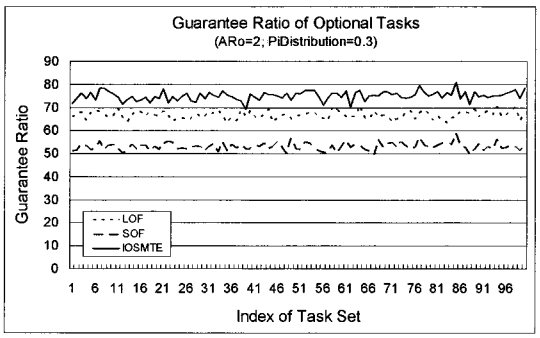


Fig. 3. Guarantee ratio of optional tasks. (ρ=2, p=0.3)

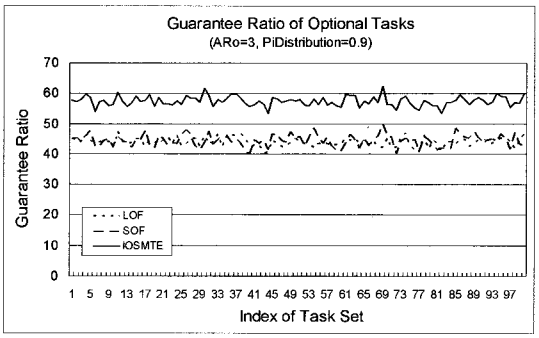


Fig. 4. Guarantee ratio of optional tasks. (ρ=3, p=0.9)

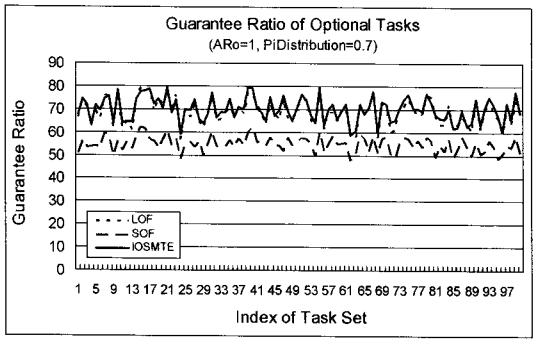


Fig. 5. Guarantee ratio of optional tasks. (ρ=1, p=0.7)

5.3 The Distributions of Average Guarantee Ratios of Optional Tasks

The second metric is the distributions of average guarantee ratios of optional tasks depend on *ARo* (ρ) and *PiDistribution*(*p*) values. To compare the average guarantee ratios depend on *ARo*(ρ) and *PiDistribution*(*p*) values among the three scheduling strategics, 100 task sets in which a task set consists of 300 imprecise tasks are generated. In the generation of tasks, *ARo*(ρ) and *PiDistribution*(*p*) values are considered, where *ARo*(ρ) is fixed arbitrary from 1 to 4 and *PiDistribution*(*p*) is fixed arbitrary from 0.2 to 0.9. For each combination of *ARo*(ρ) and *PiDistribution*(*p*) values, where 32(4×8) combinations are created, the average guarantee ratios of LOF_Scheduling, SOF_Scheduling and IOSMTE_Scheduling can be computed as follows;

$$AVG\_G\_LOF = TOT\_G\_LOF/100 \quad (8)$$

$$AVG\_G\_SOF = TOT\_G\_SOF/100 \quad (9)$$

$$AVG\_G\_IOSMTE = TOT\_G\_IOSMTE/100 \quad (10)$$

In this, *TOT_G_LOF*, *TOT_G_SOF* and *TOT_G_IOSMTE* mean the sum of guarantee ratios of the generated 100 task sets in the LOF_Scheduling, SOF_Scheduling and IOSMTE_Scheduling respectively. *AVG_G_LOF*, *AVG_G_SOF* and *AVG_G_IOSMTE* mean the average guarantee ratios of the generated 100 task sets in the LOF_Scheduling, SOF_Scheduling and IOSMTE_Scheduling respectively. The following Table 1 represents the average guarantee ratios depend on *ARo*(ρ) and *PiDistribution*(*p*) values in the LOF_Scheduling, SOF_Scheduling and IOSMTE_Scheduling.

In table 1, the numbers in bold-faced cells denote the largest average guarantee ratios among the three scheduling strategics. As we can see easily, the IOSMTE_Scheduling strategy outperforms the LOF_Scheduling and SOF_Scheduling strategics for the most cases. For the special case of *ARo*(ρ)

Table 1. The average guarantee ratios depend on ρ and p values

ARo(ρ)	Scheduling Strategy	*PiDistribution*(*p*)							
		0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
1	LOF	67.46	**69.02**	68.24	68.17	68.62	68.75	68.12	68.37
	SOF	52.05	53.23	53.28	53.45	54.21	54.70	54.39	55.05
	IOSMTE	**67.51**	69.00	**68.31**	**68.74**	**69.55**	**69.94**	**69.29**	**69.96**
2	LOF	67.24	66.75	66.47	64.71	62.12	58.66	54.31	49.82
	SOF	52.81	53.10	52.75	52.48	51.61	50.39	48.23	46.66
	IOSMTE	**75.55**	**75.13**	**75.17**	**74.06**	**72.34**	**70.01**	**66.07**	**61.35**
3	LOF	63.89	63.06	61.53	58.81	54.94	50.69	46.96	43.91
	SOF	55.02	54.62	54.52	53.64	51.90	49.83	47.04	44.35
	IOSMTE	**78.57**	**78.26**	**77.39**	**75.73**	**72.51**	**67.96**	**62.42**	**57.45**
4	LOF	62.23	61.32	59.12	55.30	51.94	49.12	46.74	43.89
	SOF	56.42	56.05	55.08	54.22	51.93	49.25	46.89	44.51
	IOSMTE	**81.45**	**80.87**	**79.55**	**76.96**	**72.23**	**67.17**	**62.49**	**57.42**

= 1, where the average number of tasks which can be scheduled in some time interval is 1, the differences of average guarantee ratios between IOSMTE_Scheduling strategy and LOF_Scheduling strategy are very small.

6. CONCLUSION

In the imprecise scheduling of real-time tasks, it has been studied that LOF(Longest Optional First) strategy has a good performance in the scheduling of imprecise tasks with 0/1 constraints for minimizing total error[1,2].

On the other hand, in the on-line scheduling, NORA algorithm can find a schedule with the minimum total error for the imprecise task system[4]. In NORA algorithm, EDF(Earliest Deadline First) strategy is adopted in the scheduling of optional tasks.

Therefore when NORA algorithm tries to finish more optional tasks with earlier deadlines, the execution of some mandatory tasks with later deadlines may be postponed. This is the main cause of suboptimality in NORA algorithm. Furthermore, NORA algorithm is not designed for the task system with 0/1 constraints. So, NORA algorithm may not suitable for the task system with 0/1 constraints any more.

Thus, to improve the shortage of NORA algorithm, in this paper, an on-line algorithm for minimizing total error is proposed. This algorithm is suitable for the imprecise on-line system with 0/1 constraints. Then, to evaluate the performance of this algorithm, a series of experiments are done. The aim of these experiments is to compare the guarantee ratios of optional tasks among the proposed algorithm and LOF(Longest Optional First) strategy and SOF(Shortest Optional First) strategy.

As a consequence of the performance comparison among three selection strategics of optional parts, it has been concluded that the proposed IOSMTE algorithm outperforms LOF and SOF strategics for the most cases.

REFERENCES

[1] André M. van Tilborg, and Gary M. Koob, *Foundations of Real-Time Computing Scheduling and Resource Management*, Kluwer Academic Publishers, Boston/Dordrecht/London, 1991.

[2] K. H. Song, K. H. Choi, S. K. Park, D. K. Choi, and K. O. Yun, "A Heuristic Scheduling Algorithm for Reducing the Total Error of an Imprecise Multiprocessor System with 0/1 Constraint," *Journal of Electrical Engineering and Information Science*, Vol.2, No.6, pp. 1-6, 1997.

[3] C. H. Lee, W. Ryu, K. H. Song, K. H. Choi, G. H. Jung, and S. K. Park, "On-line Scheduling Algorithms for Reducing the Largest Weighted Error Incurred by Imprecise Tasks," *Proceedings of Fifth International Conference on Real-Time Computing Systems and Applications*, pp. 137-144, 1998.

[4] Wei-Kuan Shih and Jane W. S. Liu, "On-Line Scheduling of Imprecise Computations to Minimize Error," *SIAM J. COMPUT*, Vol.25, No.5, pp. 1105-1121, 1996.

[5] Sanjoy K. Baruah and Mary Ellen Hickey, "Competitive On-Line Scheduling of Imprecise Computations," *IEEE Transactions on Computers*, Vol.47, No.9, pp. 1027-1032, 1998.

[6] Gi-Hyeon Song, "Online Schedulability Check Algorithm for Imprecise Real-time Tasks," *Journal of the Korea Computer Industry Education Society*, Vol.3, No.9, pp. 1167-1176, 2002.

[7] Gi-Hyeon Song, "An On-line Algorithm to Search Minimum Total Error for Imprecise Real-time Tasks with 0/1 Constraint," *Journal of Korea Multimedia Society*, Vol.8, No. 12, pp. 1589-1596, 2005.

[8] J. Hong, X. Tan, and D. Towsley, "A Performance Analysis of Minimum Laxity and Earliest Deadline Scheduling in a Real-Time System," *IEEE Transactions on Computers*, Vol.38, No.12, pp. 1736-1744, 1989.

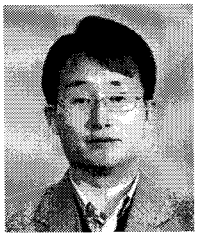
[9] Gi-Hyeon Song, "An Efficient Algorithm to Minimize Total Error of the Imprecise Real Time Tasks with 0/1 Constraint," *Journal of the Korea Computer Industry Education Society*, Vol.7, No.4, pp. 309-319, 2006.

[10] Wei-Kuan Shih and Jane W. S. Liu, "Algorithms for Scheduling Imprecise Computations with Timing Constraints to Minimize Maximum Error," *IEEE Transactions on Computers*, Vol.44, No.3, pp. 466-471, 1995.

[11] Wei-Kuan Shih, Che-Rung Lee, and Ching-Hui Tang, "A Fast Algorithm for Scheduling Imprecise Computations with Timing Constraints to Minimize Weighted Error," *IEEE*, pp. 305-310, 2000.

[12] Kun-Ming Yu, "Algorithms for Imprecise Tasks With 0/1-Constraints," *Journal of Information Science and Engineering*, Vol.17, pp. 73-83, 2001.

[13] W. K. Shih, J. W. S. Liu, and J. Y. Chung, "Algorithms for scheduling imprecise computations with timing constraints," *SIAM J. COMPUT*, Vol.20, pp. 537-552, 1991.



Gi-Hyeon Song

Received the B.S. and M.S. degrees in computer science and statistics from Chungnam University in 1985 and 1987 respectively and his Ph.D. from Ajou University in 1999. He is an associate professor in MIS department at Daejeon Health Sciences College since 1990. His research interests include real-time scheduling and web database.