0/1 제약조건을 갖는 부정확한 태스크들의 총오류를 최소화시키기 위한 개선된 온라인 알고리즘

(An Improved Online Algorithm to Minimize Total Error of the Imprecise Tasks with 0/1 Constraint)

송 기 현 †
(Gi-Hyeon Song)

요 약 부정확한 실시간시스템은 시간적으로 긴급한 태스크들을 융통성있게 스케쥴링해줄 수 있다. 총오류를 최소화시키면서 0/1제약조건과 시간적제약조건들을 모두 만족시키는 대부분의 스케쥴링문제들은 선택적태스크들이 임의의 수행시간을 갖고 있을 때 NP-complete이다. Liu는 단일처리기상에서 0/1제약조건을 갖는 태스크들을 총 오류가 최소화되도록 스케쥴링시킬 수 있는 합리적인 전략을 제시하였다. 또한, 송 등은 다중처리기상에서 0/1제약조건을 갖는 태스크들을 총 오류가 최소화되도록 스케쥴링시킬 수 있는 합리적인 전략을 제시하였다. 그러나, 이러한 알고리즘들은 모두 오프라인 알고리즘들이다. 그런데, 온라인 스케쥴링에 있어서, NORA알고리즘은 부정확한 온라인 태스크 시스템상에서 최소의 총 오류를 갖는 스케쥴을 찾을 수 있다. 이러한 NORA알고리즘에 있어서, EDF전략이 선택적스케쥴링에 적용되었다. 한편, 0/1제약조건을 갖는 태스크시스템에 있어서는, EDF스케쥴링이 총 오류가 최소화된다는 측면에서 최적이 아닐수도 있다. 더욱이, 선택적태스크들이 그들의 실행요구시간의 오름차순으로 스케쥴될 때, EDF전략이 적용된 NORA알고리즘이 최소의 총오류를 산출할 수 없을지도 모른다. 그러므로, 본 논문에서는, 0/1제약조건을 갖는 부정확한 태스크 시스템의 총오류를 최소화시키는 온라인 알고리즘이 제안되었다. 그리고나서, 제시된 알고리즘과 NORA 알고리즘 사이의 성능을 비교하기 위하여 여러 가지 실험들이 수행되었다. 두 알고리즘들 사이의 성능비교의 결과로서, 선택적태스크들이 그들의 실행요구시간들의 임의의 순서대로 스케쥴될 때는 제안된 알고리즘이 NORA알고리즘과 비슷한 총오류를 산출하지만 특별히 선택적태스크들이 그들의 실행요구시간들의 오름차순으로 스케쥴될 때는 제안된 알고리즘이 NORA알고리즘보다 더 적은 총오류를 산출할 수 있음이 밝혀졌다.

키워드 : 온라인 알고리즘, 총오류의 최소화, 부정확한 태스크, 0/1제약조건, 성능비교

Abstract The imprecise real-time system provides flexibility in scheduling time-critical tasks. Most scheduling problems of satisfying both 0/1 constraint and timing constraints, while the total error is minimized, are NP-complete when the optional tasks have arbitrary processing times. Liu suggested a reasonable strategy of scheduling tasks with the 0/1 constraint on uniprocessors for minimizing the total error. Song et al suggested a reasonable strategy of scheduling tasks with the 0/1 constraint on multiprocessors for minimizing the total error. But, these algorithms are all off-line algorithms. In the online scheduling, the NORA algorithm can find a schedule with the minimum total error for the imprecise online task system. In NORA algorithm, EDF strategy is adopted in the optional scheduling. On the other hand, for the task system with 0/1 constraint, EDF_Scheduling may not be optimal in the sense that the total error is minimized. Furthermore, when the optional tasks are scheduled in the ascending order of their required processing times, NORA algorithm which EDF strategy is adopted may not produce minimum total error. Therefore, in this paper, an online algorithm is proposed to minimize total error for the imprecise task system with 0/1 constraint. Then, to compare the performance between the proposed algorithm and NORA algorithm, a series of experiments are

† 정 회 원 : 대전보건대학 경영정보학과
ghsong@hit.ac.kr

논문접수 : 2002년 12월 27일
심사완료 : 2007년 7월 12일

performed. As a conseqence of the performance comparison between two algorithms, it has been concluded that the proposed algorithm can produce similar total error to NORA algorithm when the optional tasks are scheduled in the random order of their required processing times but, the proposed algorithm can produce less total error than NORA algorithm especially when the optional tasks are scheduled in the ascending order of their required processing times.

Key words : Online Algorithm, Minimize Total Error, Imprecise Task, 0/1 Constraint, Performance Comparison

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 constraint.

Most scheduling problems of satisfying both 0/1 constraint 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 constraint 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 constraint on multiprocessors for minimizing the total error in [2]. The results 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 online scheduling, Shih and Liu proposed NORA algorithm which can find a schedule with the minimum total error for a task system consisting solely of online tasks that are ready upon arrival in [3]. But, for the task system with 0/1 constraint, it has not been known whether NORA algorithm can be optimal or not in the sense that the total error is minimized. In NORA algorithm, EDF(Earliest Deadline First) strategy[4] is adopted in the optional scheduling.

However, for the task system with 0/1 constraint, EDF strategy may not be optimal in the sense that the total error is minimized. Furthermore, when the optional tasks are scheduled in the ascending order of their required processing time, NORA algorithm which EDF strategy is adopted may not produce minimum total error.

Therefore, in this paper, an online algorithm is proposed to minimize total error for the imprecise task system with 0/1 constraint.

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

Section 4 presents an improved online scheduling algorithm for the imprecise real-time task system with 0/1 constraint to minimize total error. The results of simulation and analysis are described in section 5. And section 6 concludes this paper.

2. Imprecise Online Real-Time Task System Model

Each task T_i in a basic imprecise online real-time task system model consists of 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 finished
- 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 task T_i consists of two parts, a mandatory task M_i and an optional task O_i. m_i and o_i represent the execution time of M_i and O_i, respectively ($m_i + o_i = p_i$). 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; $TE = \sum_{k=1}^{n} e_k$

3. Related Works

There are many different imprecise task scheduling problems. These problems include minimization of total error, minimization of the maximum or average error, minimization of the number of discarded optional tasks, minimization of the number of tardy tasks and minimization of average response time.

In this paper, the problem of scheduling imprecise computations to meet timing constraints and 0/1 constraint is considered for minimizing total error. As expected, the general problem of scheduling to meet the 0/1 constraint 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 constraint 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]. When the optional tasks have arbitrary processing times, a good strategy for scheduling tasks with the 0/1 constraint 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].

But, these algorithms are all off-line algorithms. For the case of online scheduling[3,5,6] Shih and Liu proposed NORA algorithm which can find a schedule with the minimum total error for a task system consisting solely of online tasks that are ready upon arrival in [3]. 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 constraint, it is doubtful whether NORA algorithm can produce minimum total error or not. In NORA algorithm, some error is produced as a result of 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.

On the other hand, for the task system with 0/1 constraint, EDF scheduling may not be optimal in the sense that the total error is minimized. Furthermore, when the optional tasks are scheduled in the ascending order of their required processing time, NORA algorithm which EDF strategy is adopted may not produce the minimum total error.

Thus, [7] suggests an optimal algorithm to search minimum total error for the imprecise online real-time task system with 0/1 constraint. But, this algorithm may cause high complexity in the worst case.

Therefore, in this paper, an online algorithm is proposed to minimize total error for the imprecise task system with 0/1 constraint.

4. An Improved Imprecise Online Scheduling Algorithm

In this section, an improved online scheduling algorithm for the imprecise real-time tasks to minimize total error is described. The following Figure 1 showes this algorithm. Hereafter, we call this algorithm as IOS(Imprecise Online Scheduling) algorithm.

In this algorithm, at first, the procedure "Set-SystemParameter" is performed. In this procedure, all system parameters which are used in generating the imprecise online real-time task system are determined. These parameters include $ARo(\rho)$, $Amu(\mu)$, *Alamda*(λ), *FiDistribution*(*p*) and *NmberTasks*. The meaning of each parameter is explained in section 5. Next, from the "GenerateSystem" procedure, an imprecise online real-time task system can be generated randomly. The next "For loop" is performed whenever a task $T_i(1 \leq i \leq NumberTasks)$ arrived. Whenever an online task T_i is arrived, the "DetermineSchedulableTasks(T_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 on ascending order. Next, an online schedulability check function "CheckOnLineSchedulability()" is performed.

This fuction checks the schedulability of the schedulable tasks whenever each online task T_i is arrived. The explanation about this function is described precisely in [6].

In this function, even though only one task turned to be not schedulable, this algorithm is terminated abnormally. If the tasks are turned to be all schedulable, "MandatoryScheduling(T_i,*leng*)" procedure can be performed in a ready time interval of task T_i and T_{i+1}. In this, *leng* denotes the size of the ready time interval. MandatoryScheduling (T_i,*leng*) procedure schedules the schedulable mandatory tasks in the time interval with EDF strategy.

As a result of the procedure, *ML* list in step 2 is updated and *leng* value is decreased by the sum of scheduled mandatory processing times.

Next, OptionalScheduling(T_i,*leng*) procedure which is the main focus of the proposed algorithm can be performed. In this procedure, two lists *BTL* and *L* are introduced. The *BTL* list means a list of the burden tasks which incur maximum error on T_i arrival. The *L* list signify a list of the schedulable optional tasks on T_i arrival.

First, a spared ready time interval, *Length*, after MandatoryScheduling(T_i,*leng*) in step 15, and the sum of all processing times of the optional tasks in *L*, *HapCi*, are compared in step 25. Whenever *Length* is less than *HapCi*, EDF scheduling for the optional tasks in *L* is performed. In this EDF scheduling, each task in *L* is removed in turn. Whenever EDF scheduling except each task in *L* is performed, different total error may be produced. Hence, in each EDF scheduling except some optional task, the least total error can be produced. If any EDF scheduling except any optional task can produce the least total error, we say, this optional task is "burden task". So, in step 26, the burden task, *BT*, can be determined. Then, *HapCi* is decreased by the optional processing time of the burden task in step 27. Next, in step 28, the contents of the list *L* and *BTL* are adjusted. This process from step 25 to step 29 is repeated until *Length* is greater than or equal to *HapCi* from step 25 to step 29. Even though, the above condition is satisfied, all optional tasks in *L* may not be scheduled.

Therefore, the schedulability check using EDF strategy is performed in step 30. If only one task in *L* turned to be not schedulable, SelectBurdenTask(*L*, *NST*) procedure is performed to select a burden task and by removing this burden task, the least total error among the all possible EDF scheduling of *L* can be produced. This process which is described from step 31 to step 35 is repeated until all optional tasks in *L* are schedulable. Finally, when all optional tasks in *L* turned to be schedulable, the list *OL* in step 36 is adjusted, the least total error on T_i arrival becomes to be the sum of processing times of optional tasks in *BTL*.

As a result, the minimum total error can be determined as $MinErr_{Tid}$ in step 37 when an online task *Tid* is arrived.

On the other hand, in the proposed imprecise online scheduling algorithm which is depicted in Figure 1, the number of iterations that a "For loop" which is described from step 7 to step 19 is executed is bounded by $O(N)$, where N is the total number of tasks in an imprecise task system. Next, the number of schedulable tasks which "DetermineSchedulableTasks(T_i)" procedure determines on T_i arrival can be bounded by $\log N$ for the average case. The reason is explained in section 5.3.

Theorem 4.1.

The average number of schedulable tasks which DetermineSchedulableTasks(T_i) procedure of IOS algorithm determines on T_i arrival can be bounded by log$N$.

Proof.

The schedulable tasks on T_i arrival can be defined as those tasks of which deadlines are greater than T_i's ready time, r_i, and of which processing times of mandatory parts, m_is, are not finished.

But, in the online scheduling, the number of those schedulable tasks can not be predictable theoretically. Thus, the number of schedulable tasks on T_i arrival can not help investigating by simulation in section 5.3.

Eventually, the average number of schedulable tasks on T_i arrival turned to be log$N$ for the average case.

Theorem 4.2.

In the OptionalScheduling(*Tid,Length*) procedure, the number of iteration of two "Do While" loop can be bounded by log^2N respectively.

Proof.

The first "Do While" loop can be terminated when *Length* is greater than or equal to *HapCi*. If *HapCi* has zero value, i.e., the content of L is empty, this loop must be finished. So, the complexity of first "Do While" loop may be dependent on the number of elements in L. As the number of schedulable optional tasks in L on T_i arrival can be bounded by log$N$ in theorem 4.1 and the complexity of SelectBurdenTask(L, NST) in step 26 can be bounded by $O(\log N)$, the complexity of first "Do While" loop become to be $O(\log^2 N)$. The second "Do While" loop can be terminated when *EDFOK* value becomes to be "*True*". The number of iteration of second "Do While" loop may be dependent on the number of elements in L as *EDFOK* value of step 34 becomes to be "*True*" for the worst case of L's being empty.

Then, in the second "Do While" loop, the complexity of SelectBurdenTask(L, NST) and EDF_Schedulability(L) can be bounded by $O(\log N)$ respectively. Therefore, the complexity of second "Do While" loop becomes to be $O(\log^2 N)$.

Theorem 4.3.

The complexity of the proposed IOS algorithm is $O(N \log^2 N)$.

Proof.

In the proposed IOS algorithm which is depicted in Figure 1, the number of iterations that a "*For loop* " which is described from step 7 to step 19 is executed is bounded by $O(N)$, where N is the total number of tasks in an imprecise task system.

Next, by theorem 4.2, the complexity of Optional-Scheduling(*Tid,Length*) procedure which is main focus of the proposed IOS algorithm can be bounded by log^2N.

Therefore, the complexity of the proposed IOS algorithm becomes to be $O(N \log^2 N)$.

5. Simulation Study

In this chapter, the results of simulation are presented and analized. The aim of simulation is to compare performance of the proposed IOS algorithm and NORA algorithm. In order to compare the performance, 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) * *FiDistribution*, where *FiDistribution*(p) is fixed arbitrary from 0.2 to 0.9 for each experiment. 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 *FiDistribution*(p) becomes larger, the load of processor also becomes higher. If the generated task set is not schedulable by the CheckOnLineSchedulability() function of Figure 1, it is rejected and regenerated until all

```
1:  Sub Imprecise-OnLine-Algorithm()
2:    $ML$ = {a list of the mandatory tasks which have been scheduled}
3:    $OL$ = {a list of the optional tasks which have been scheduled}
4:    $ML = OL = \varnothing$
5:    Call SetSystemParameter                'Determine task system parameter
                                             ($ARo, Amu, Alamda, PiDistribution$).
6:    Call GenerateSystem                    'Generate task system.
7:    For $i=1$ To $NumberTasks$             'Whenever a task $T_i$ has arrived
8:      Call DetermineSchedulableTasks($T_i$)     'Determine the schedulable tasks in $[r_i, r_{i+1}]$.
9:      Call SortTaskByDeadline()                 'Sort the schedulable tasks by deadline.
10:     $OnLineCheck$ = CheckOnLineSchedulability()  'Check online schedulability of the tasks.
11:     If ($OnLineCheck = False$) Then           'If the online tasks are not schedulable
12:       Exit Sub                                'Terminate this algorithm.
13:     End If
14:     $leng = r_{i+1} - r_i$
15:     Call MandatoryScheduling($T_i, leng$)
16:     $ML = ML \cup \{M_k,\ k = 1,2,3,\ldots,n$ ; $M_k$ has been scheduled in the
                                    MandatoryScheduling($T_i, leng$)$\}$
17:     $leng = leng - \sum_{k=1}^{n} m_k,\ m_k \in ML$
18:     Call OptionalScheduling($T_i, leng$)
19:   Next $i$
20: End Sub
21: Sub OptionalScheduling($Tid, Length$)
22:   $BTL$ = {a list of the burden tasks on task $T_{Tid}$ arrival} = $\varnothing$
23:   $L$ = {a list of the schedulable optional tasks on task $T_{Tid}$ arrival}
24:   $HapOi = \sum_{i=1}^{NST} o_i,\ O_i \in L$; $NST$ denotes the number of elements in $L$
25:   Do While $Length < HapOi$
26:     $BT$ = SelectBurdenTask($L, NST$)
27:     $HapOi = HapOi - o_{BT}$
28:     $L = L - \{O_{BT}\},\ NST = NST - 1,\ BTL = BTL \cup \{O_{BT}\}$
29:   Loop
30:   $EDFOK$ = EDF_Schedulability($L$)
31:   Do While $EDFOK = False$
32:     $BT$ = SelectBurdenTask($L, NST$)
33:     $L = L - \{O_{BT}\},\ NST = NST - 1,\ BTL = BTL \cup \{O_{BT}\}$
34:     $EDFOK$ = EDF_Schedulability($L$)
35:   Loop
36:   $OL = OL \cup L$
37:   $MinErr_{Tid} = \sum_{j=1}^{NBT} o_j,\ O_j \in BTL$ ; $NBT$ denotes the number of elements in $BTL$
38: End Sub
```

Figure 1 Imprecise online scheduling algorithm

mandatory tasks in the imprecise online 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 constraint can be adopted.

In there, two different selection strategics are considered. The first one is EDF strategy and which strategy selects one task with the earliest deadline among the schedulable optional tasks. The famous NORA algorithm adopts this strategy. The second strategy is selecting and removing the burden task and so incurring the least total error among the schedulable optional tasks. In the proposed IOS algorithm in Figure 1, this strategy is adopted.

To compare performance between two selection strategics, the following metrice, described in section 5.2, is used.

5.2 Total Errors of Generated Task Sets

The proposed IOS algorithm can be compared with NORA algorithm on the aspect of total errors incurred from the generated task sets. To compare total errors derived from IOS algorithm and NORA algorithm, an experiment is performed. In this experiment, 100 task sets are generated and each task set is composed of 300 tasks. From each task set, total errors are incurred by IOS algorithm and NORA algorithm respectively.

The following Table 1 and Table 2 show the number of task sets producing less total error than that of another algorithm for ρ = 1~4, p = 0.2~ 0.9 when the processing times of optional parts is distributed on ascending order or random order respectively.

As we can see in Table 1, IOS algorithm has better performance than NORA algorithm except for the case of ρ = 3, p = 0.3 and ρ = 4, p = 0.8. Even in these cases, the number of task sets producing less total error are similar between two algorithms.

On the other hand, when the processing times of optional parts are distributed on random order, we can see in Table 2 that IOS algorithm can produce less total error than that of NORA algorithm when ARo(ρ) and PiDistribution(p) value become large.

Therefore, we can conclude that the proposed IOS algorithm has better performance than NORA algorithm when the processing times of optional parts are distributed on ascending order and even when the processing times of optional parts are distributed on random order, IOS algorithm has better performance than NORA algorithm as ARo (ρ) and PiDistribution(p) values are increase.

Table 1 A performance comparison between IOS and NORA algorithm (ascending order of o_i)

ARo	Algorithm	PiDistribution							
		0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
1	IOS	0	2	10	56	74	85	83	84
	NORA	0	0	10	31	25	12	12	7
	Equal	100	98	80	13	1	3	5	9
2	IOS	0	2	21	77	81	56	36	29
	NORA	0	2	16	13	2	5	1	1
	Equal	100	96	63	10	17	39	63	70
3	IOS	0	1	30	54	41	24	10	13
	NORA	0	2	12	4	4	1	0	3
	Equal	100	97	58	42	55	75	90	84
4	IOS	0	5	26	25	14	5	2	5
	NORA	0	2	11	5	4	3	4	2
	Equal	100	93	63	70	82	92	94	93

Table 2 A performance comparison between IOS and NORA algorithm (random order of o_i)

ARo	Algorithm	PiDistribution							
		0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
1	IOS	51	55	57	43	44	51	51	51
	NORA	44	39	37	54	48	41	40	41
	Equal	5	6	6	3	8	8	9	8
2	IOS	38	36	30	32	44	48	48	45
	NORA	46	46	57	54	48	37	30	16
	Equal	16	18	13	14	8	15	22	39
3	IOS	26	31	25	50	33	40	39	16
	NORA	45	45	39	39	17	19	5	0
	Equal	29	24	36	11	50	41	56	84
4	IOS	36	27	25	25	31	33	11	9
	NORA	28	36	32	13	6	3	4	0
	Equal	36	37	43	62	63	64	85	91

5.3 Average Number of Schedulable Tasks

As we can see in theorem 4.1 of chapter 4, the average number of schedulable tasks which DetermineSchedulableTasks(T_i) procedure of IOS algorithm determines on T_i arrival can be bounded by log$N$.

The reason is as follows;

In the online scheduling, the number of schedulable tasks on some task arrival can not be predictable theoretically. Therefore, in this section, an experiment is performed to investigate the number of schedulable tasks on T_i arrival.

In this experiment, each task set consisting of 300 tasks is generated as ARo(ρ) and PiDistribution(p) value. Whenever an online task T_i of some task set characterized by ARo(ρ) and PiDistribution(p) value is arrived, the number of schedulable tasks are determined by DetermineSchedulableTasks (T_i) procedure of IOS algorithm. Then, we can determine the average number of schedulable tasks by dividing the total number of schedulable tasks on each task arrival in a task set arrival by the number of tasks in a task set.

Each cell in Table 3 denotes the average number of schedulable tasks in a task set as ρ and p value. In Table 3, we can see that the average number of schedulable tasks are become large as ρ and p value increase. Next, the average number of every cells in Table 3 becomes to be 8, this value approximates $\ln 300 \simeq \log 300$; 300 denotes the number of tasks in a task set.

Finally, we can conclude that the average number of schedulable tasks on T_i arrival can be bounded by $\log N$.

Table 3 Average number of schedulable tasks in a task set as ρ and p value

ARo	PiDistribution							
	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
1	1	1	1	1	1	1	1	2
2	1	1	2	2	2	2	5	4
3	1	1	1	3	8	9	14	20
4	1	2	2	5	19	42	46	57

6. Conclusion

The general problems of scheduling to meet the 0/1 constraint and timing constraints, as well as to minimize the total error, are NP-complete when the optional tasks have arbitrary processing times.

Liu suggested a reasonable strategy of scheduling tasks with the 0/1 constraint on uniprocessors for minimizing the total error. Song et al suggested a reasonable strategy of scheduling tasks with the 0/1 constraint on multiprocessors for minimizing the total error. But, these algorithms are all off-line algorithms.

In the online scheduling, NORA algorithm can find a schedule with the minimum total error for the imprecise online task system. In NORA algorithm, EDF strategy is adopted in the optional scheduling.

On the other hand, for the task system with 0/1 constraint, EDF scheduling may not be optimal in the sense that the total error is minimized. Furthermore, when the optional tasks are scheduled in ascending order of their required processing times, NORA algorithm which EDF strategy is adopted may not produce minimum total error.

Therefore, in this paper, an online algorithm is proposed to minimize total error for the imprecise task system with 0/1 constraint. To compare the performance between two algorithms, a series of experiments are performed.

As a conseqence of the performance comparison between two algorithms, it has been concluded that the proposed algorithm can produce similar total error to NORA algorithm when the optional tasks are scheduled in the random order of their required processing times but, the proposed algorithm can produce less total error than NORA algorithm especially when the optional tasks are scheduled in ascending order of their required processing times.

References

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

[2] K. H. Song and K. H. Choi, et al, "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, p1~p6, 1997.

[3] Wei-Kuan Shih and Jane W. S. Liu, "Online Scheduling of Imprecise Computations to Minimize Error," SIAM J. COMPUT, Vol. 25, No. 5, p1105~p1121, October 1996.

[4] J. Hong, X. Tan, 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, p1736~p1744, December 1989.

[5] C. H. Lee, W. Ryu, K. H. Song, et al, "Online Scheduling Algorithms for Reducing the Largest Weighted Error Incurred by Imprecise Tasks," Proceedings of Fifth International Conference on Real-Time Computing Systems and Applications, p137~p144, 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, p1167~p1176, 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, p1589~p1596, 2005.

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.