JOURNAL OF KOREA MULTIMEDIA SOCIETY, VOL. 8, NO. 12, DECEMBER 2005(pp. 1589-1596)

An On-line Algorithm to Search Minimum Total Error for Imprecise Real-time Tasks with 0/1 Constraint

Gi-Hyeon Song†

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. On the other hand, in the case of on line scheduling, Shih and Liu proposed the 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. But, for the task system with 0/1 constraint, it has not been known whether the 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. So, this paper suggests an optimal algorithm to search minimum total error for the imprecise on-line real-time task system with 0/1 constraint. Furthermore, the proposed algorithm has the same complexity, $O(N \log N)$, as the NORA algorithm, where N is the number of tasks.

Keywords: on line algorithm; imprecise scheduling; real-time task; search minimum total error; 0/1 constraint;

1. INTRODUCTION

In a (hard) real-time system, every time-critical task must meet its timing constraint, which is typically specified in terms of its deadline. It is essential for every time-critical task to complete its execution and produce its result by its deadline. Otherwise, a timing fault occurs, and the result produced by the task is of little or no use. Unfortunately, many factors, such as variations in processing times of dynamic algorithms and congestion on the communication network, make meeting all timing constraints at all times difficult. An approach to minimize this difficulty is to trade off the quality of the results produced by the tasks with the amounts of processing time required to produce the results. Such a tradeoff can be realized by using the imprecise computation technique[2]. The imprecise real-time system, 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[5,6]. By the total error, it means the sum of the processing times of all optional tasks that could

※ 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 : Jun. 30, 2005, Approval date : Oct. 12, 2005

†Assistant Professor in MIS department at Daejeon Health Sciences College

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. [3] suggested a reasonable strategy of scheduling tasks with the 0/1 constraint on multiprocessors for minimizing the total error. The results of this paper show that the longest processing first selection strategy outperforms random or minimal laxity policies. On the other hand, in the case of on-line scheduling, Shih and Liu proposed the 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]. But, for the task system with 0/1 constraint, it has not been known whether the 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.

So, this paper suggests an optimal algorithm to search minimum total error for the imprecise on-line real-time task system with 0/1 constraint.

This paper is organized as follows : Section 2 presents the imprecise on-line real-time task system model. Section 3 describes the NORA algorithm under 0/1 constraint.

Next, section 4 provides an on-line search algorithm to minimize total error of the imprecise tasks with 0/1 constraint. Section 5 showes an numerical example.

Finally, section 6 concludes the paper.

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

Each task T_i in a basic imprecise on-line 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. THE NORA ALGORITHM UNDER 0/1 CONSTRAINT

In [6], Shih and Liu proposed the 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. The 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, the 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, the NORA algorithm has a good schedulability performance for all mandatory tasks, but for the optional tasks with 0/1 constraint, it is doubtful whether the NORA algorithm can produce minimum total error or not. In the NORA algorithm, some error is produced as a result of EDF (Earliest Deadline First) strategy as the scheduler of the 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, the EDF scheduling may not be optimal in the sense that the total error is minimized. Therefore, in this paper, an on-line algorithm for the imprecise tasks with 0/1 constraint to search minimum total error is proposed. In the next section, this algorithm is described.

4. AN ON-LINE SEARCH ALGORITHM TO MINIMIZE TOTAL ERROR OF THE IMPRECISE TASKS WITH 0/1 CONSTRAINT

In this section, the on-line scheduling problem of the imprecise computation model to minimize total error is considered. An on-line algorithm to minimize total error of the imprecise tasks with 0/1 constraint is suggested. The overview of this algorithm is depicted in Figure 1. Hereafter, this algorithm is called as IOS(Imprecise On-line Search) algorithm.

In this algorithm, input is a task set TS with N preemptive imprecise tasks running on a single processor and output is the minimum total error; $MTE = Min\{\alpha \mid TS_\alpha \text{ is schedulable}\}$, and a feasible schedule which minimizes total error. In this algorithm, "For loop" is performed whenever an on-line task T_i $(1 \le i \le N)$ is arrived. Whenever an on-line task T_i is arrived, the "DetermineSchedulableTasks(i)" procedure determines schedulable tasks in time interval $[r_i, r_{i+1}]$, then the "SortTaskByDeadline" procedure sorts the schedulable tasks by deadlines on ascending order. Next, the on-line schedulability check function "CheckOnLineSchedulability()" is performed. This fuction checks schedulability of the tasks on T_i arrival based on EDF strategy. Even though only one task turned to be not schedulable, this algorithm is terminated abnormally. If the tasks are all schedulable, the next procedure "MandatoryScheduling" can be performed in the time interval. $[r_i, r_{i+1}]$. The "MandatoryScheduling" procedure schedules the mandatory parts of all schedulable tasks with the EDF(Earliest Deadline First) strategy in the time interval $[r_i, r_{i+1}]$. If any processing time is available after executing the mandatory parts of all schedulable tasks then the "OptionalScheduling" executes the optional parts

```
1 : Sub Imprecise_Online_Search_Algorithm()
2 :   Dim i, N As Integer
3 :   Dim OnLineCheck As Boolean

4 :   For i = 1 To N                          ' Whenever a Task T_i has arrived
5 :     Call DetermineSchedulableTasks (i)    ' Determine Schedulable Tasks in [r_i, r_i+1].
6 :     Call SortTaskByDeadline ()            ' Sort the Schedulable Tasks By Deadline.
7 :     OnLineCheck = CheckOnLineSchedulability()  ' Check On-Line Schedulability.
8 :     If (OnLineCheck = False) Then         ' If the Tasks are not Schedulable
9 :       Exit Sub                            ' Terminate This Algorithm.
10 :    End If
11 :    Call MandatoryScheduling (i, r_i, r_i+1)   ' Perform Mandatory Scheduling in [r_i, r_i+1].
12 :    Call OptionalScheduling (i, leng)     ' Perform Optional Scheduling as leng.
13 :  Next i
14 : End Sub
```

Fig. 1. The overview of the IOS algorithm.

of the schedulable tasks in order to minimize total error. In the optional scheduling, 0/1 constraint should be applied. The following Fig. 2 showes the OptionalScheduling (i, $leng$) procedure.

Whenever an on-line task T_i is arrived, the above OptionalScheduling (i, $leng$) procedure is performed. In this procedure, the minimum total error of the schedulable tasks in the time interval $[r_i, r_{i+1}]$ is obtained using binary search method. In this procedure, at first, all possible combinations generated from the optional parts of the schedulable tasks in the $[r_i, r_{i+1}]$ are enumerated and all

```
1 : Sub OptionalScheduling (Tid, Length)
2 :   Dim NC, NSE, SP, j, low, high, mid, MTE As Integer
3 :   Dim Check As Boolean

' Enumerate all possible combinations of the optional parts within Length and compute an error from
each combination, where NC, NST, NSE and SP denote Number of Combinations, Number of
Schedulable Tasks, Number of Sorted Errors and Stack Pointer, respectively.
4 :   NC = 2^NST ; NSE = 0 ; SP = 0
5 :   For j = 0 To NC - 1
6 :     (b_n, b_{n-1}, ..., b_3, b_2, b_1)_2 = (j)_10
7 :     err_j = Σ_{k=1}^{n} o_k , for all b_k = 0
8 :     IF (err_j > Length) Then
9 :       GoTo Step 5
10 :    End IF
11 :    NSE = NSE + 1
12 :    ER_NSE = err_j
13 :  Next j

' Sort the errors which are derived from the combinations.
14 :    Call SortofErrors()

' Search the minimum total error of the schedulable tasks on task T_Tid arrival.
15 :    low = 1 ; high = NSE
16 :    While low ≤ high Do
17 :      mid = ⌊(low + high) / 2⌋
18 :      Check = SchedulabilityofTS (mid)
19 :      If (Check = True) Then
20 :        SP = SP + 1 ; STACK_SP = ER_mid ; high = mid - 1
21 :      Else
22 :        low = mid + 1
23 :      End If
24 :    End While

' Obtain the minimum total error on task T_Tid arrival.
25 :      MTE_Tid = STACK_SP
26 : End Sub
```

Fig. 2. The OptionalScheduling (i, $leng$) procedure.

errors are computed from the combinations. If an error err_j generated from any combination is greater than the length of the scheduling interval $[r_i, r_{i+1}]$ denoted by *Length* in step 8, this error err_j is cancelled. Next, in step 14, all errors obtained from step 4 to step 13 are sorted.

Finally, the minimum total error can be searched using binary search method from step 15 to step 24. In step 18, the schedulability of the given task set with a total error ER_{mid} is decided. Whenever the task set with the total error ER_{mid} can be fea sibly scheduled, ER_{mid} value is inserted into the top of stack.

Therefore, as a conseqence of this procedure, the minimum total error on the task T_{fid} arrival can be obtained from the top of stack as depicted in step 25.

Theorem.

The IOS algorithm is optimal in finding a schedule to minimize total error for an imprecise task system with 0/1 constraint.

Proof.

The optimality of the IOS algorithm requires that the following two conditions are satisfied :

1. All mandatory tasks are completed before their deadlines.
2. The total error is minimized.

The satisfaction of condition 1 for any task system is obvious whenever the *OnLineCheck* in Fig. 1 has "*True*" value.

Whenever the *OptionalScheduling(i, leng)* procedure in Fig. 1 is performed, the minimum total error can be searched using binary search method. Therefore, condition 2 is proved.

5. NUMERICAL EXAMPLE

Consider a task set TS shown in Table 1, with five tasks $T_1 \sim T_5$. The proposed IOS algorithm works as follows ;

At time 1, tasks T_1, T_2, T_3 and T_4 are arrived. These tasks are all schedulable at time 1. Then, the 4 schedulable tasks are sorted by deadline on ascending order resulting as $T_1 - T_2 - T_3 - T_4$. Next, the "CheckOnLineSchedulability()" function is performed. This function checks the schedulability of the tasks on T_1 arrival based on EDF strategy. The result of this function is depicted in Figure 3. As a result of this function, "OnLineCheck" value becomes "*True*". As the value becomes "*True*", the low-level scheduling composed of "Mandatory-Scheduling" and "OptionalScheduling" procedure can be performed in a time interval [1, 18]. The result of "MandatoryScheduling" in [1, 18] is the same as Fig. 3. As a next step of the "MandatoryScheduling" procedure, "OptionalScheduling (1~4, 9)" is performed in a time interval [9, 18]. In this procedure, the minimum total error of the schedulable tasks T_1, T_2, T_3 and T_4 in the time interval [9, 18] can be searched. In the procedure, at first, all possible combinations generated from O_1, O_2, O_3 and O_4 are enumerated and all possible errors of the combinations are computed. If an error err_j generated from any combination is greater than the length of the interval [9, 18], equals 9, this error err_j is cancelled. Next, all these errors are sorted. Table 2 showes the result. Now, the minimum total error of the schedulable tasks T_1, T_2, T_3 and T_4 in the time interval [9, 18] can be searched. In the "OptionalScheduling (1~4, 9) procedure of Figure 2, initially, *low* is set to be 1 and *high* is *NSE* (Number of Sorted Errors); 14. A feasible schedule for TS_7 exists as shown in Figure 4, and hence *high* becomes equal to 6. Then, from the feasible schedule TS_7, an error $ER_7 = 5$ is derived and this error ER_7 is inserted into top of stack. In the

second iteration, *mid* becomes equal to 3, and TS_3 is also schedulable as shown in Figure 5, and hence *high* becomes equal to 2. Next, from TS_3, $ER_3 = 3$ is derived and this error is inserted into the top of stack. In the third iteration, *mid* becomes equal to 1, and TS_1 is not schedulable and hence *low* becomes equal to 2. In the fourth iteration, *mid* becomes equal to 2, and TS_2 is not schedulable and hence *low* becomes equal to 3. Then, *low* and *high* become 3, 2, respectively. As *low* is greater than *high*, the while loop is terminated and the minimum total error on $T_1 \sim T_4$ arrival becomes equal to $ER_3 = 3$ which is on the top of stack. Next, at time 18, a task T_5 is arrived, and the same processes are repeated. Figure 6 represents the result of the IOS algorithm when T_5 is arrived. Therefore, the minimum total error of the sample task set in Table 1 becomes equal to o_1 plus o_5, i.e., 8.

On the other hand, Figure 7 showes the result of the NORA algorithm scheduling the same task set as adopted in the IOS algorithm. In this case, the total error of the NORA algorithm becomes equal to o_4 plus o_5, i.e., 9. Finally, from this example, it is concluded that the proposed IOS algorithm can produce less total error than the NORA algo-

Table 1. A sample task set.

	r_i	d_i	m_i	o_i	p_i
T_1	1	13	2	3	5
T_2	1	14	1	2	3
T_3	1	15	2	1	3
T_4	1	16	3	4	7
T_5	18	20	1	5	6

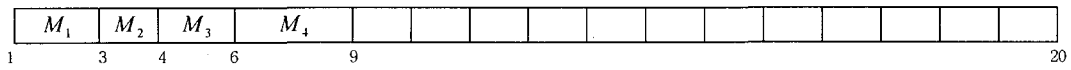

Fig. 3. The result of "CheckOnLineSchedulability()" function on T_1 arrival.

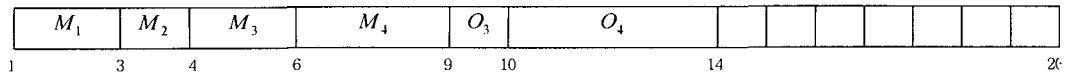

Fig. 4. A feasible schedule for task set TS_7 by EDF.

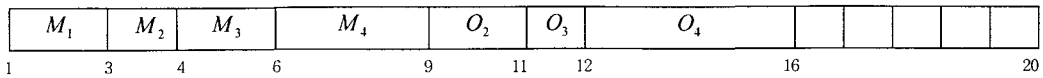

Fig. 5. A scheduled task set TS_3 by EDF.

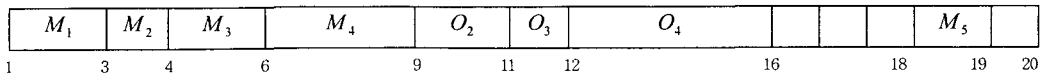

Fig. 6. The result of the IOS algorithm when T_5 is arrived.

Table 2. All sorted combinations generated from o_1, o_2, o_3 and o_4.

Combination number	$o_1(3)$	$o_2(2)$	$o_3(1)$	$o_4(4)$	err_j
1	1	1	0	1	1
2	1	0	1	1	2
3	0	1	1	1	3
4	1	0	0	1	3
5	0	1	0	1	4
6	1	1	1	0	4
7	0	0	1	1	5
8	1	1	0	0	5
9	0	0	0	1	6
10	1	0	1	0	6
11	0	1	1	0	7
12	1	0	0	0	7
13	0	1	0	0	8
14	0	0	1	0	9

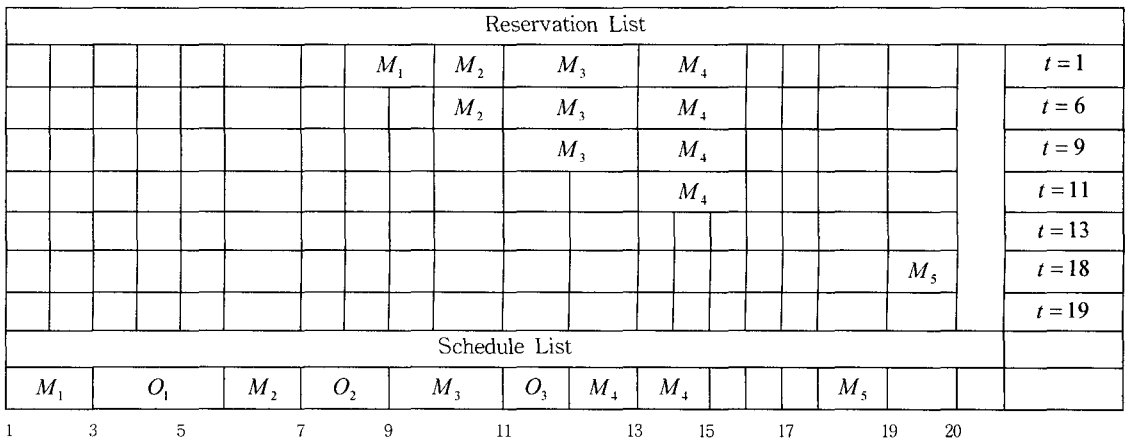

Fig. 7. The reservation list and the schedule list of the NORA algorithm.

rithm when the optional tasks have 0/1 constraint.

6. CONCLUSION

In this paper, the on-line scheduling problem of the imprecise real-time tasks with 0/1 constraint to minimize total error is considered. In this field, Shih and Liu proposed the famous NORA algorithm which can find a schedule with the minimum total error for an on-line task system. But, for the task system with 0/1 constraint, it has not been known whether the NORA algorithm can be optimal or not. So, this paper suggests an optimal algorithm to search minimum total error for the imprecise on-line real-time task system with 0/1 constraint. Furthermore, the proposed algorithm has the same complexity, $O(N \log N)$, as the NORA algorithm, where N is the number of tasks.

7. REFERENCES

[1] J.W.S. Liu, K.J. Lin, W.K. Shih, A.C. Yu, J.Y. Chung, and W. Zhao, "Algorithm for Scheduling Imprecise Computations," *Computer*, Vol. 25, No. 3, 1991.

[2] J.Y. Chung, J.W.S. Liu, and K.J. Lin,

"Scheduling Periodic Jobs that Allow Imprecise Results," *IEEE transactions on Computers*, Vol. 19, No. 9, pp. 1156-1173, 1990.

[3] 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, 1997.

[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] 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.

[6] W.K. Shih, J.W.S. Liu, and J.Y.Chung, "Fast Algorithms for Scheduling Tasks with Ready Times and Deadlines to Minimize Total Error," *Proceedings of the 10th IEEE Real-Time Systems Symposium*, 1989.

Gi-Hyeon Song

He 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. He is an assistant professor in MIS department at Daejeon Health Sciences College since 1990. His research interests include real-time scheduling and web database.