AMSE JOURNALS-AMSE IIETA publication-2017- Series: Advances D; Vol. 22; N°1; pp 16-30
Submitted Jan. 2017; Revised March 3, 2017, Accepted April 15, 2017

A Study on Multi-Processor Scheduling Algorithm

Y. Duan*, W. Chen

Dept of Basic Science Guangdong University of Science and Technology, Dongguan, Guangdong, China (dy_01155@126.com)

Abstract

This paper deals with task scheduling problem of multi-processors by using *T-LET* plane method, and proposes a new algorithm named *BLREF* based on flow scheduling model. And the optimality of this algorithm is also demonstrated. Finally, we verify the superiority of it through specific examples. This is a simple algorithm is designed for simulation scheduling project, improvement of simulation technology has important significance. BLREF algorithm can pass the execution time of appropriating makes full use of the processor, so as to realize the feasible scheduling, is a kind of optimal multiprocessor scheduling algorithm. In addition, the misappropriation of the execution time for the analysis has very important reference value for other algorithm, and the other is of great reference value for multiprocessor scheduling problem research.

Keywords
Scheduling algorithm, multi-processors, *T-LET* plane, event *C*, event *B*.

1. Introduction

Two main strategies for multi-processor task scheduling are global scheme and classification scheme [1]. In the global scheduling scheme, every new real time task is always executed in different processors, and all processors run the same scheduling algorithm. While, in the classification scheme, all emergences of a task are executed in the same processor, and tasks are assigned to different processors beforehand by a task assignment algorithm. Typical multi-processor scheduling algorithms have many drawback [2, 3]. For example, the minimum relaxation algorithm and the global EDF algorithm, etc. are not the optimal algorithms, since we can find task set whose cut-off time could not be fulfilled by both algorithms. Therefore, this paper proposes an optimal multi-processors scheduling algorithm named *BLREF*, Can pass the execution time of appropriating makes full use of the processor, in order to realize the feasible scheduling, and gives proofs on its optimality.

2. T-LET plane based BLREF scheduling algorithm

Based on the *L-C* plane introduced by Dertouzos [4], this paper brings in a new method for abstracting task execution behaviors, and it is called the Time and Local remaining Execution-Time plane, abbreviated as *T-LET* plane. In *T-LET* plane, the flow scheduling model in uniprocessor is imported, which is shown by the imaginary line in Fig. 1, indicating that all tasks are executed with a constant velocity at any time. In the *T-LET* plane, the *x*-axis represents time, and *y*-axis indicates the remaining execution-time of the task. Let r_i be the origin, then the imaginary line from point $(0,e_i)$ to $(d,0)$ represents the flow scheduling with the slope being $-u_i$. If T_i runs as shown in Fig.1, then its execution path can be expressed by the broken line from $(0,e_i)$ to $(d,0)$. In this line, the proportions for *x*- and *y*-axis are equal, and the slope is -1 when the task runs, and 0 when it stops.

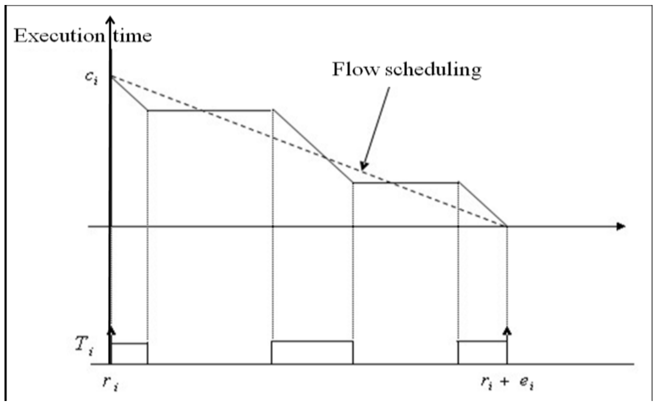

Fig.1 T-LET plane

Consider *N* tasks, their flow scheduling models could be constructed as shown in Fig.2. For all tasks, we can find an isosceles right triangle in every two successive scheduling events, and denote it as $T\text{−}LET^k$, where k progressively increases along time. The bottom edge of the isosceles right triangle represents time, and the vertical edge in the left side means a part of the remaining execution-time for the tasks which is called the current remaining execution-time, denoted as l_i . And it will be exhausted before the termination of each $T\text{−}LET^k$. In each $T\text{−}LET$ plane, slopes of flow scheduling for each task could be constructed overlapped.

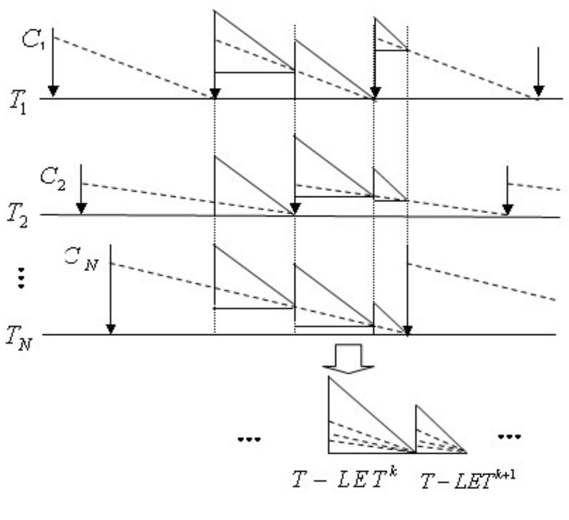

Fig.2 T-LET plane for multi-task

Since the whole *T-LET* plane is the repetition in time, a good scheduling algorithm for a single *T-LET* plane would also work for all repeated ones.

Fig.3 gives details on the *k*-th *T-LET* plane in which the status of each task is represented by a sign. The horizontal axis of the position of the sign is the current time, and the vertical axis denotes the current remaining execution-time of the task. Here, the current remaining execution-time means that it must be exhausted before time t^k_f, and it is not the cut-off time of the task. After the decision of scheduling time, move the sign of each task on the *T-LET* plane. Though, the ideal path is the imaginary line in Fig.3, the sign of the task could move towards only two directions. If a task is selected and then executed, the sign of it will move down along the diagonal line, as T_N does. In other situations, it will horizontally move right, as T_1

does. Consider *M* processors, there will be at most *M* signs moving along the diagonal line at the same time.

The scheduling target on the *k*-th *T-LET* plane is moving signs of all tasks to the vertex at the rightmost on the *T-LET* plane. If the current remaining execution-time at time t^k_f is 0, then we call it successful arrival, or current feasibility. If all task signs on each T-LET plane are current feasible, then these tasks may be schedulable on all continuous *T-LET* planes in time by moving their ideal paths.

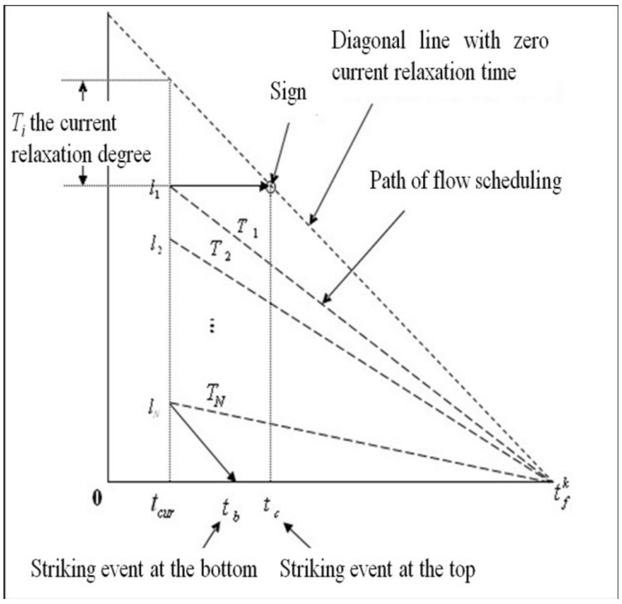
Diagonal line with zero current relaxation time
Sign
T_i the current relaxation degree
Path of flow scheduling
l_1
l_2
T_1
T_2
l_N
T_N
0
t_{cur}
t_b
t_c
t_f^k
Striking event at the bottom
Striking event at the top

Fig.3 The k-th T-LET plane

For convenience, the current relaxation time of a task is defined as 0. The hypotenuse of the *T-LET* plane has an important implication, that is, if a task sign strikes a certain edge, it means that the task has no current relaxation time at all. And if this task is not selected immediately, then it can't satisfy current feasible scheduling target. We call the hypotenuse of the *T-LET* plane the non-current relaxation diagonal line. There are two moments when scheduling should be re-determined, and they are the moment when the current execution-time of a task is exhausted, denoted as event *B*, and when the current relaxation time is 0, denoted as event *C*.

To distinguish these two events from traditional scheduling ones, such as release initiation of tasks, we treat event *B* and *C* as sub-events. In order to satisfy current feasibility, first select *M* tasks with biggest current remaining execution-time, which is called Bigger Local Remaining Execution Time First (*BLREF*) strategy. Besides, it is not allowed to select tasks with 0 current remaining execution time, and signs for them are inactivated while those for tasks with current remaining execution time bigger than 0 is activated. At time t^k_f, the events for the next task are released immediately, and ***T−LET***$^{k+1}$ for the next T-LET plane starts and *BLREF* keeps survivable. Thus, *BLREF* scheduling strategy is applied to each event.

The fundamental feature of *BLREF* algorithm is its optimality in scheduling- the total usage ratio of the current task is no more than the capacity of all processors of the system. This algorithm can satisfy the cut-off time of all tasks.

3. Necessary and sufficient conditions of "schedulability" in BLREF algorithm

Definition 1. When events *C* and *B* occur, denoted as t_j, where $0<j<f$, call *BLREF* algorithm re-scheduling tasks. Here, define the current usage ratio of task T_i at time t_j as:

$$r_{i,j} = l_{i,j} / (t_f - t_j) \qquad (1)$$

where $l_{i,j}$ is the current remaining execution time of task T_i at time t_j .

Theorem 1. When all task signs on the *k*-1st T-LET plane move to the vertex on the rightmost, the initial current usage ration $r_{i,0} = u_i$ for all tasks T_i on the *k* th T-LET plane.

Proof: If all task signs arrive at time t_f^{k-1} and $l_i = 0$, then they will restart from the ideal flow scheduling line. The slope of the flow scheduling path for task T_i is u_i , then $r_{i,0} = l_{i,0} / t_f = u_i$. End of the proof.

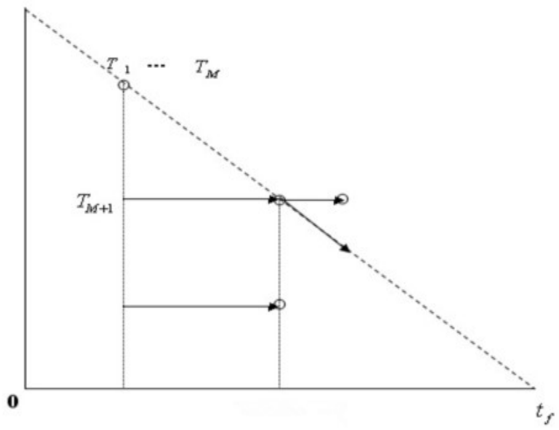

Fig.4 The critical time

We define the notion -critical time to describe necessary and sufficient conditions of non-current feasible of task signs.

Definition 2: When more than *M* task signs sequentially strike the non-current relaxation diagonal line, the critical time is the time when the first sub-event occurs, shown in Fig.4. On the right of the critical time, only *M* task signs are selected on the non-current relaxation diagonal line. And those not selected will be removed from the triangle, followed by the fact that they can't move to the vertex on the rightmost on the *T-LET* plane.

Theorem 2: All task signs on the *T-LET* plane are not current feasible if and only if at least one critical time occurs.

Proof: Necessity: If one critical time appears, then task signs not selected will not in the *T-LET* plane. This is because slopes of task sign paths are 0 and -1 only, and signs not in *T-LET* plane are impossible to reach the rightmost vertex.

Sufficiency: Suppose task signs are not current feasible, then there must be a critical time. If not, the number of task signs on the non-current relaxation diagonal line will never exceed *M*. Thus, these task signs will not be selected by *BLREF* algorithm until time t_f. And this is contradictory to our hypothesis, so theorem is End of the proof.

Definition 3: The total current usage ratio of the j th sub-event is defined as S_j where

$$S_j = \sum_{i=1}^{N} r_{i,j} . \tag{2}$$

Corollary 1: At the critical time of the j th sub-event, we have $S_j>M$.

Proof: According to the $r_{i,j}$ definition of 1,For tasks on the non-current relaxation diagonal line at critical time of the j th sub-event, the current remaining execution time $l_{i,j}$ is $t_f - t_j$ since the *T-LET* plane is a isosceles right triangle. Therefore, we have

$$S_j = \sum_{i=1}^{N} r_{i,j} = \sum_{i=1}^{M} \frac{t_f - t_j}{t_f - t_j} + \sum_{i=M+1}^{N} \frac{l_i}{t_f - t_j} > M, \quad (l_i > 0) . \tag{3}$$

End of the proof.

4. Event *C*

When a not selected task sign strikes the non-current relaxation diagonal line, then event *C* occurs. And the selected task signs will not strikes that line. Event *C* means there is no current relaxation time for the task which must be chosen immediately. In Fig.5, event *C* happens at time t_c , and task T_{M+1} strikes the non-current relaxation diagonal line.

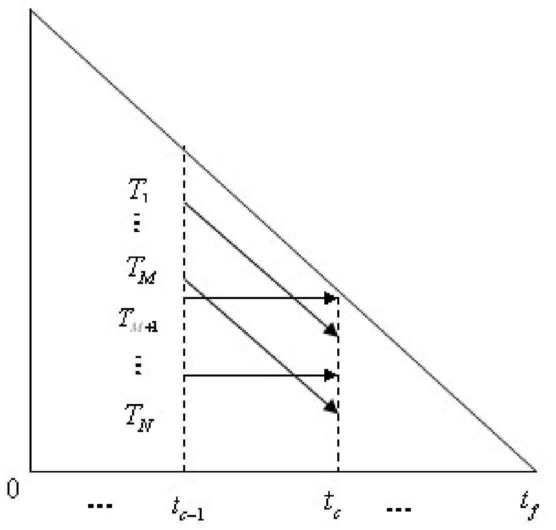

Fig.5. Event C

In Fig. 5, give a lower subscript i than task sign with higher current usage ratio, namely, $\forall j, r_{i,i} \geq r_{i+1,i}$, where in *1≤i<N* .In this case, *BLREF* will select *M* tasks from T_1 to T_M, and their signs will move along the diagonal line.

Lemma 1: Event *C* happens at time t_c, and $r_{i,c-1} \geq r_{i+1,c-1}$ where $1 \leq i < N < 1$, then we have $1 - r_{M+1,c-1} \leq r_{M,c-1}$.

Proof: If the sub-event at time t_c is event *C*, then the time when task $\boldsymbol{T}_{M+1}$ strikes the non-current relaxation diagonal line must be early than that when task T_M strikes the bottom of the *T-LET* plane. The time when task $\boldsymbol{T}_{M+1}$ strikes *NLLD* is $t_{c-1} + (t_f - t_{c-1} - l_{M+1,c-1})$, among them $f > c-1$. Contrarily, the time when task T_M strikes the bottom of the *T-LET* plane is $t_{c-1} + l_{M,c-1}$, then we obtain

$$t_{c-1} + (t_f - t_{c-1} - l_{M+1,c-1}) \leq t_{c-1} + l_{M,c-1} \tag{4}$$

$$\frac{t_f - t_{c-1} - l_{M+1,c-1}}{t_f - t_{c-1}} \leq \frac{l_{M,c-1}}{t_f - t_{c-1}}, \quad (t_f - t_{c-1} > 0) \tag{5}$$

Namely,

$$1 - r_{M+1,c-1} \leq r_{M,c-1} \tag{6}$$

End of the proof.

Corollary 2: Event *C* happens at time t_c and $r_{i,c-1} \geq r_{i+1,c-1}$, where 1≤*i*<*N,* then

$$s_{c-1} > M(1 - r_{M+1,c-1}) \tag{7}$$

Proof:

$$S_{c-1} = \sum_{i=1}^{M} r_{i,c-1} + \sum_{i=M+1}^{N} r_{i,c-1} > \sum_{i=1}^{M} r_{i,c-1} \geq M^{*} r_{M,c-1} \tag{8}$$

By Lemma 1, we have

$$M^{*} r_{M,c-1} \geq M^{*}(1 - r_{M+1,c-1}) \tag{9}$$

$$S_{c-1} > M(1 - r_{M+1,c-1}) \tag{10}$$

End of the proof.

Theorem 3: Event C occurs at time t_c and $S_{c-1} \leq M$, then for $\forall c, 0 < c \leq f$ and $r_{i,c-1} \geq r_{i+1,c-1}$, where 1≤*i*<*N*, we have

$S_c \leq M$.

Proof: Let $t_c - t_{c-1} = t_f - t_{c-1} - l_{M+1,c-1} = \delta$. The total current remaining execution time at time t_{c-1} is $\sum_{i=1}^{N} l_{i,c-1} = (t_f - t_{c-1}) S_c$, and decreases by $M^{*}\delta$ when M task signs moving along the diagonal line. Therefore, we have

$$(t_f - t_c) S_c = (t_f - t_{c-1}) S_{c-1} - \delta M \tag{11}$$

$$l_{M+1,c-1} = t_f - t_c \tag{12}$$

$$l_{M+1,c-1} * S_c = (t_f - t_{c-1}) S_{c-1} - (t_f - t_{c-1} - l_{M+1,c-1}) M \tag{13}$$

Thus, we obtain

$$S_c = \frac{1}{r_{M+1,c-1}} S_{c-1} + (1 - \frac{1}{r_{M+1,c-1}})M \tag{14}$$

It can be seen that S_c is a linear function regarding S_{c-1} whose function graph is shown if Fig.6. From **Corollary** 2, when event *C* happens at time t_c, we have $S_{c-1} > M(1-r_{M+1,c-1})$. Therefore, when $S_{c-1} \le M$, $S_c \le M$ holds. End of the proof.

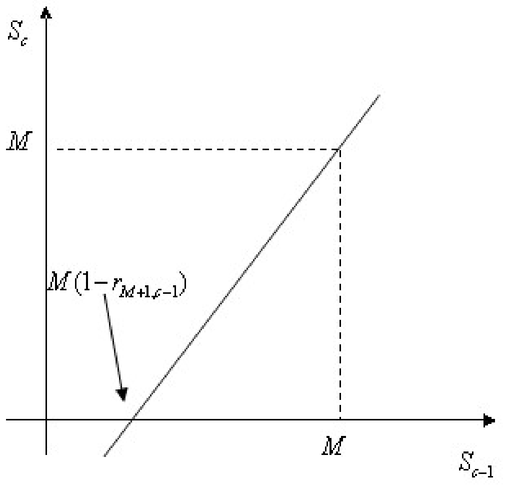

Fig. 6 Linear function of event *C*

5. Event *B*

When a selected task sign strikes the bottom of the *T-LET* plane, then event *B* occurs. And those signs that are not selected will not strike the bottom. Event *B* means that there is no current remaining time for the tasks, so it is a better choice to assign the processor to run another task, shown in Fig.7. In this figure, event *B* happens at time t_b , and task T_{M+1} strikes the bottom at the same time. Similar to analyzing event *C*, give a lower subscript *i* to the task sign with higher usage ratio, namely, $\forall j, r_{i,j} \ge r_{i+1,j}$, where $1 \le i < N$.

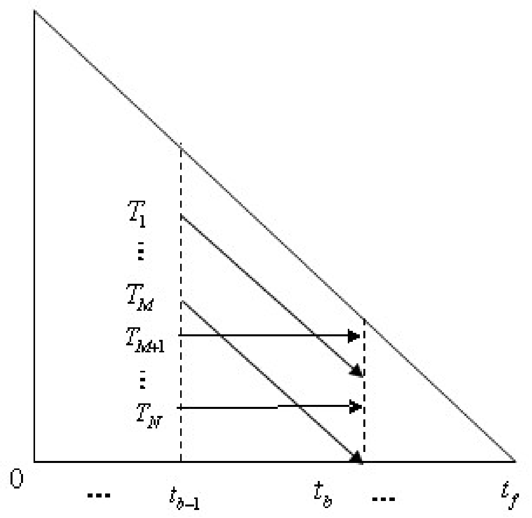

Fig. 7 Event B

Lemma 2. Event *B* happens at time t_b and $r_{i\,b-1} \geq r_{i+1,b-1}$, where $1 \leq i < N$, then we have

$$1 - r_{M+1,b-1} \geq r_{M,b-1}$$

Proof: If the sub-event at time t_b is event *B*, then the time when task T_M strikes the bottom of the *T-LET* plane (at time $t_{b-1} + l_{M,b-1}$) is earlier than that when task T_{M+1} strikes the non-current relaxation diagonal line (at time $t_{b-1} + (t_f - t_{b-1} - l_{M+1,b-1})$, $t_f - t_{b-1} > 0$). Then,

$$t_{b-1} + l_{M,b-1} < t_{b-1} + (t_f - t_{b-1} - l_{M+1,b-1}) \qquad (15)$$

$$\frac{l_{M,b-1}}{t_f - t_{b-1}} < \frac{t_f - t_{b-1} - l_{M+1,b-1}}{t_f - t_{b-1}} \qquad (16)$$

$$1 - r_{M+1,b-1} \geq r_{M,b-1} \qquad (17)$$

End of the proof.

Corollary 3. Event *B* happens at time t_b and $r_{i,b-1} \geq r_{i+1,b-1}$, where $1 \leq i < N$, then

$$S_{b-1} > M^* r_{M,b-1}.$$

Proof: $$S_{b-1} = \sum_{i=1}^{M} r_{i,b-1} + \sum_{i=M+1}^{M} r_{i,b-1} > \sum_{i=1}^{M} r_{i,b-1} \geq M^* r_{M,b-1} \qquad (18)$$

End of the proof.

Theorem 4. Event *B* happens at time t_b and $S_{b-1} \leq M$. In addition, $r_{i,b-1} \geq r_{i+1,b-1}$, where $1 \leq i < N$, then $S_b \leq M$.

Proof: Let $l_{M,b-1} = t_b - t_{b-1}$. The total current remaining execution time at t_{b-1} is

$\sum_{i=1}^{M} l_{i,b-1} = (t_f - t_{b-1})S_b$, and M task signs decrease by $M^* l_{M,b-1}$ when they are moving along the diagonal line. Therefore, we have

$$(t_f - t_{b-1})S_b = (t_f - t_{b-1})S_{b-1} - M^* l_{M,b-1} \qquad (19)$$

$$t_f - t_b = t_f - t_{b-1} - l_{M,b-1} \quad (20)$$

So,

$$(t_f - t_{b-1} - l_{M,b-1})S_b = (t_f - t_{b-1})S_{b-1} - M^* l_{M,b-1} \qquad (21)$$

Thus, we get

$$S_b = \frac{1}{1 - r_{M,b-1}} S_{b-1} + \frac{r_{M,b-1}}{1 - r_{M,b-1}} M \qquad (22)$$

The function graph for this linear function is shown in Fig.8. From Corollary 3, when event *B* occurs at time t_b, we have $S_{b-1} > M^* r_{M,b-1}$. Hence, $S_b \le M$ when $S_{b-1} \le M$. End of the proof.

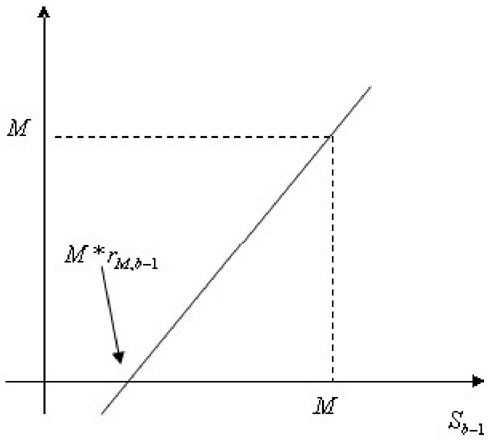

Fig.8 Linear function for event B

6. Proof on the current feasibility of BLREF algorithm on the T-LET plane

Aforementioned is under the condition of *N*>*M*. When *N* ≤*M*, the task signs on the T-LET plane are always current feasible in *BLREF* algorithm.

Theorem 5. When *N* ≤*M*, the task signs on the *T-LET* plane are always current feasible in *BLREF* .

Proof : First, assume that task signs are not current feasible in *BLREF* when *N* ≤*M*. Thus, there must exist a critical time in the *T-LET* plane from Theorem 2. But, the critical time indicates there must be at least one task sign that is not selected. And this is contradictory to the hypothesis stating that all task signs are selectable. End of the proof.

When the number of tasks is smaller than that of processors, *BLREF* could select and execute all tasks until their current remaining execution time becomes to 0. Event *C* is impossible to occur when *$N \le M$* [5, 6], since all task signs are selectable and moving along the diagonal line.

Now , we discuss the current feasibility in the case of *$N>M$.*

Theorem 6. When *$N>M$* and if *$S_0 \le M$*, then task signs in *BLREF* is always current feasible.
Proof: Use inductive method to prove it based on Theorem 3 and Theorem 4 which show that if $S_{j-1} \le M$ then we have $S_{j-1} \le M$, where *j* is the time at which sub-event happens. From Corollary 3, for each *j* , $S_j<M$ holds, then there exists no critical time in the *T-LET* plane. And from Theorem 2 we know that task signs are current feasible if there isn't critical time. End of the proof.

Given *N (N>M)* task signs in the *T-LET* plane, and *$S_0 \le M$,* there exists no critical time if events *B* and *C* occur from Theorem 6. Once event *B* happens, then the number of inactivated task signs will decrease until *M* signs are remained. After that, all task signs are selectable from Theorem 5, followed by the occurrence of sequential event *B*. Finally, they will reach the vertex at the rightmost of the *T-LET* plane.

Consider periodical tasks, if the total task usage ratio $\sum_{i=1}^{N} u_i < M$, then task signs in the first continuous *T-LET* plane are current feasible due to Theorem 5 and Theorem 6. For the second continuous *T-LET* plane *$S_0<M$,* BLREF guarantees the current feasibility of each *T-LET* plane by the induction of Theorem 1, and this will make all tasks meet their cut-off time. Therefore, we get the necessary and sufficient conditions of schedulability of tasks for multi-processor.

Theorem 7. Let $S = \sum_{i=1}^{N} u_i$, then the task set is schedulable in *BLREF* algorithm if and only if *$S \le M$.*

Proof: It is obvious of the sufficiency. Proof of the necessity: from the introduction of *T-LET* plane and *BLREF* algorithm , we can know that as long as the $T-LET^1$ plane is current feasible, then the whole task set is current feasible, too. So, we just need to prove the $T-LET^1$ plane is current feasible under the precondition *$S \le M$.* Here, we use reduction to absurdity method to prove it. If $T-LET^1$ is not current feasible, then there must be a critical time *t* such that $S_t>M$. But , *M* processors are always busy before time *t* from *BLREF* algorithm. Thus, we get *$S>M$* which is contradictory to the hypothesis. End of the proof.

7. Comparison and analysis of algorithms

In this section, two task sets are introduced, and execution results of several typical multi-processor scheduling algorithms and our proposed algorithm are also presented.

The first task set is listed in Table 1, where

$$S_1 = \sum_{i=1}^{5} u_i = \frac{1}{3} + \frac{1}{4} + \frac{2}{5} + \frac{5}{6} + \frac{6}{6} = \frac{169}{60} < 3 .$$

From the conclusion in the previous section, we know that this task set is satiable using *BLREF* algorithm.

Task	Cut-off time	Execution time	Relaxation degree
T_1	3	1	2
T_2	4	1	3
T_3	5	2	3
T_4	6	5	1
T_5	6	6	0

Tab. 1 First task set

The executive process of the minimum relaxation algorithm for this task set is shown in a Gantt diagram as Fig. 9.

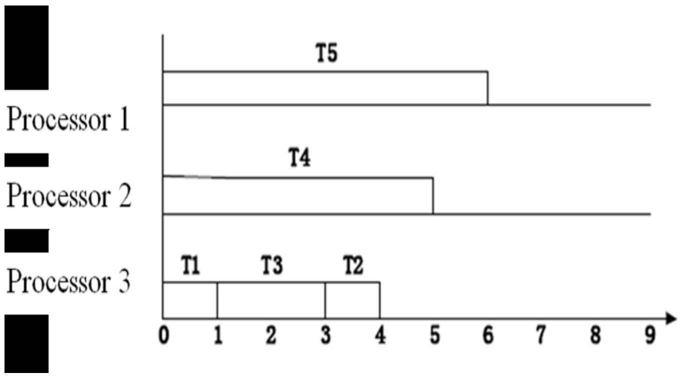

Fig.9. Gantt diagram of the minimum relaxation algorithm

From Fig.9 we can find that the minimum relaxation algorithm can meet the cut-off time of all tasks in the set. The Gantt diagram of the global *EDF* algorithm is shown in Fig. 10.

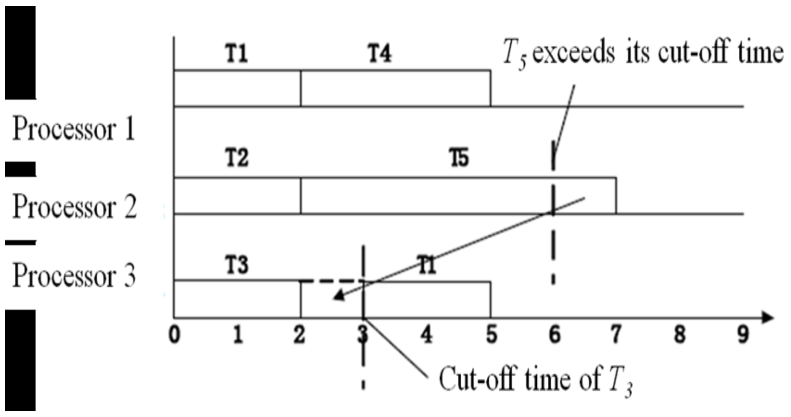

Fig.10. Gantt diagram of the global EDF algorithm

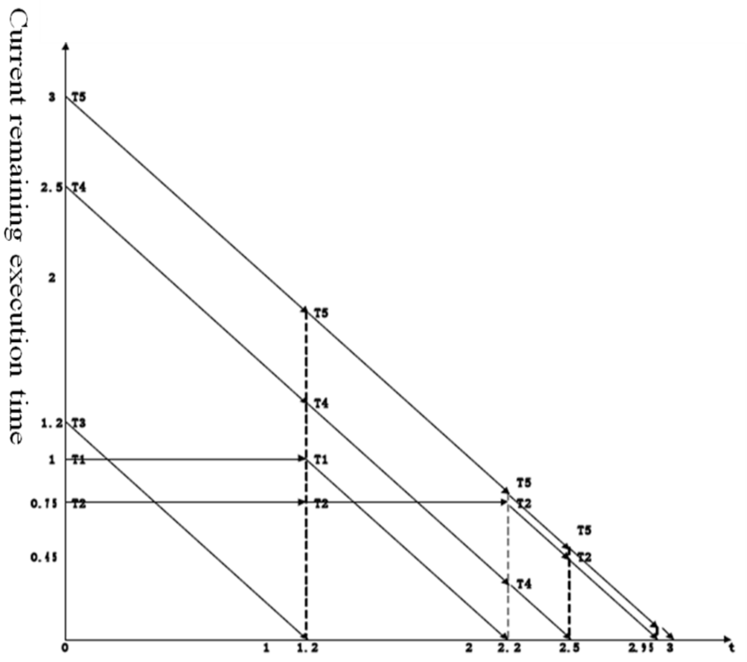

Fig. 11 Flow chart of BLREF algorithm

From Fig.10 we can see that the global *EDF* algorithm can't meet the cut-off time of all tasks, shown by the imaginary lines. This task set is schedulable in case that the period of time that exceeds the cut-off time of task T5 could be shifted onto processor 3 for execution, and task T3 is delayed one unit time.

Using *BLREF* algorithm to execute this task set, and the process is shown by the *T-LET* plane as Fig 11.

From this figure we find *BLREF* algorithm can meet the cut-off time of all tasks in the set. The minimum relaxation algorithm is able to satisfy all tasks in the first task set, but it is not the best algorithm. However, the second task set is not met by the minimum relaxation algorithm, shown in Table 2.

Tab 2. Second task set

Task	Cut-off time	Execution time	Relaxation degree
T_1	3	2	1
T_2	3	2	1
T_3	4	3	1
T_4	6	5	1

Since $S_2 = \sum_{i=1}^{4} u_i = \frac{2}{3} + \frac{2}{3} + \frac{5}{6} = \frac{35}{12} < 3$, we know that this task set is satiable by using *BLREF* algorithm from the conclusion in the previous section.

The execution process of the minimum relaxation algorithm shown by Gantt diagram is available in Fig. 12.

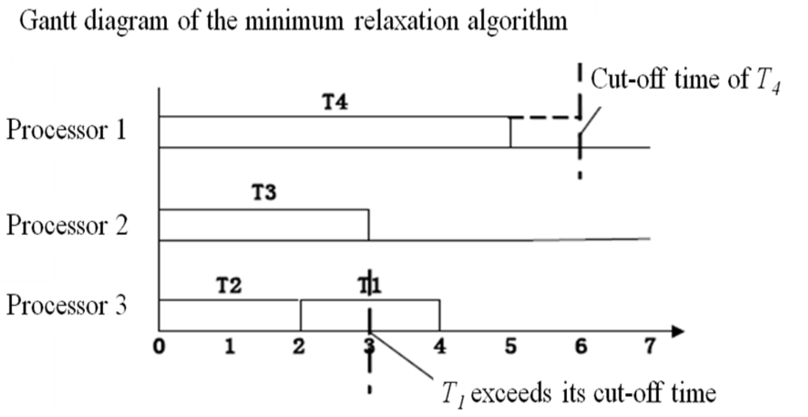

Fig. 12 Gantt diagram of the minimum relaxation algorithm on the second task set

From Fig.12 we can find that the minimum relaxation algorithm can't meet the cut-off time of all tasks in the second set. The execution process of *BLREF* algorithm expressed by the *T-LET* plane is shown in Fig.13.

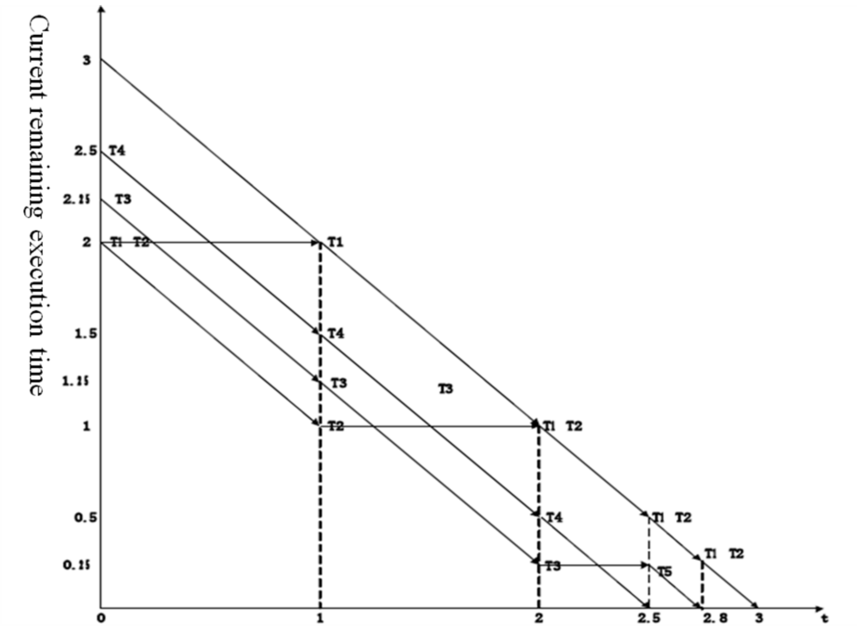

Fig. 13 The flow chart of BLREF algorithm on the second task set

For the global *EDF* algorithm, this task set may be met if properly shifting the execution order of tasks, that is, postponing tasks whose cut-off time is not reached and executing tasks that are close to the deadline in advance.

8. Conclusion

This paper proposes a *BLREF* algorithm which can make full use of processors by diverting the execution time, and it realizes the feasibility of scheduling. *BLREF* is an optimal multi-processor scheduling algorithm. Besides, the diversion of execution time has significant reference value for analyzing other algorithms. The quality of an algorithm largely depends on the degree of diversion, the more the higher efficiency will be [7,8]. Compare to other algorithms, *BLREF* has more task switching. Conversely, the global *EDF* is a very effective algorithm in terms of reducing the number of preemptions, though it is not an optimal algorithm [9,10]. The study in this paper is very indicative when considering other multi-processor scheduling problems.

References

[1] J. Carpenter, S. Funk. A categorization of real-time multiprocessor scheduling problems and algorithms [A]. Handbook on Scheduling Algorithms, Methods and Models[C]. 2004. Chapman and Hall/CRC

[2] K Ramamritham, J.A. Stankovic. Scheduling Algorithms and Operating Systems Support for Real-time Systems. Proceeding of the IEEE, Vol 83(1):55-67, 1994.

[3] Gao Li'e, A.L. Tong , F.T. Kang, W.D. Liu, N.N. Zhao, Application and Research of Dynamic Scheduling Algorithm for Multiprocessor[J], Computer Engineering and Applications, Vol. 34,196－199, 2005.

[4] L.M. Dertouzos. Multiprocessor On-Line Scheduling of Hard-Real-Time Tasks [J] IEEE Transactions on Software Engineering, vol 15(12):1497-1506,1989.

[5] P. Holman, J.H. Anderson. Adapting Pair Scheduling for Symmetric Multiprocessors [J]. Journal of Embedded Computing, Vol.1 Number 4, 543-564, 2005.

[6] C. Feng, J. Liang, Solve the more general travelling salesman problem [J]. AMSE Journals –2014- Series: Modelling D. Vol 35, Issue 1, 9-23, 2014.

[7] S. Lui . Real Time Scheduling Theory: A Historical Perspective [J]. Real-Time Systems, 2004, 28:101-155.Kluwer Academic Publishers. Manufactured in The Netherlands.

[8] C. Feng，J. Liang. The solution of the more general traveling salesman [J]. AMSE Journals –2014-Series: Advances A, Vol 51, Issue 1：27-40, 2014

[9] Y. Duan., Optimization design of the single processor scheduling algorithm in real-time system research [J]. Journal of operational research, Vol 17, No.1, 27-34, 2013.

[10] Y. Duan., Y. Xiang, Comparative study of different genetic operator combination to solve TSP problem [J]. Science and technology. Vol 28, No.5, 27-31, 2012.