TS-Preemption Threshold and Priority Optimization for the Process Scheduling in Integrated Modular Avionics

Qianlin Zhou, Hui Lu(✉), Honglei Qin, Jinhua Shi, and Rongrong Zhou

School of Electronic and Information Engineering,
Beihang University, Beijing 100191, China
mluhui@vip.163.com

Abstract. Avionics is confronted with transitioning from a federated avionics architecture to an Integrated Modular Avionics (IMA) architecture. IMA architectures utilize shared, configurable computing, communication, and I/O resources to increase system scalability. Therefore, resources scheduling becomes a critical issue for IMA. This paper focuses on the process scheduling. We use preemption threshold scheduling strategy to improve process scheduling performance, and propose a two-stage tabu algorithm to optimize the preemption threshold and the priority respectively. Firstly, we investigate a convergence criterion to stop iteration of level-i busy period which is used to calculate the worse-case response time. Secondly, we propose the difference analysis method based on weight to evaluate the optimal schedule. Finally, we propose TS-preemption threshold and priority optimization algorithm to obtain the near-optimal assignment of the priority and the preemption threshold. The experiment results of different sizes of process scheduling problems illustrate the validity and effectivity of the algorithm.

Keywords: Integrated Modular Avionics · Process scheduling · Two-stage tabu algorithm · Evaluation method · Convergence analysis

1 Introduction

As a new generation of avionics system, IMA attracts more attentions because of the significant advantages compared with the traditional avionics [1,2]. IMA introduces the concept of partition to support one or more avionics applications and allow them execute independently to improve the stability of the system [3]. IMA dynamically dispatch system resources to partition and process. Therefore, the partition scheduling problem and process scheduling problem are critical to the performance of IMA. The partition scheduling, which is different from region division in many-objective optimization [4], aims at dispatching the time window of applications. One application in a partition can have numbers of processes. The process scheduling focuses on the better way to ensure the reliability of applications. Therefore, a high performance algorithm to schedule the process plays a critical role for IMA.

C. He et al. (Eds.): BIC-TA 2017, CCIS 791, pp. 9–23, 2017.
https://doi.org/10.1007/978-981-10-7179-9_2

In fact, IMA is based on the ARINC 653 standard and inherits the characteristics of real-time operating systems. The design of the process scheduling can obtain experience from the real-time system. The process scheduling for real-time system includes fixed priority scheduling and non-fixed priority scheduling. The former has been widely used in varieties of real-time systems. It can be divided into three categories. They are fully preemptive [5,6], fully non-preemptive [5,7] and limited preemptive scheduling. There are many methods for the first two categories, like Rate-Monotonic (RM) algorithm [8,9] and Deadline-Monotonic (DM) algorithm [10]. As a kind of limited preemptive scheduling strategy, preemption threshold scheduling (PTS) [11] had been proved to have more advantages than fully preemptive and fully non-preemptive scheduling. It can improve the schedulability for a set of processes and achieve a reasonable use of the processor with reasonable preemption thresholds and priorities.

There are some discussions of PTS. Lehoczky [12] introduced the concept of the level-i busy period [13,14] to calculate the worse-case response time (WCRT) of the processes. Wang and Saksena [11] investigated some useful conclusions to guide the preemption threshold optimization, and proposed an algorithm to search a feasible assignment. Redell and Torngren [15] proposed a computational method of the exact worse-case response time for static priority scheduled tasks with offsets and jitter. Keskin et al. [16] presented a revised analysis for WCRT and extended PTS algorithm. Buttazzo et al. [17] further summarized the analysis method of WCRT.

However, there are some problems for the existing assignment algorithm. Firstly, only when $L(i)$, the length of the level-i busy period for process τ_i, is convergent can we calculate WCRT for process τ_i, Secondly, we cannot obtain the better solutions because of the lack of evaluation criteria for the performance of the assignments. Finally, the assignment obtained by the existing method [11] is not the global optimal solution, because the algorithm will terminate when a feasible assignment of the preemption threshold is obtained with given priorities.

In this paper, we use PTS to schedule processes in IMA. We focus on improving above problems and propose TS-preemption threshold and priority optimization algorithm (TPTPOA) to solve the assignment problem for preemption threshold and priority in PTS.

Firstly, we prove two conclusions of the convergence of $L(i)$ when the sum of the execution time and the blocking time meet specific conditions for process τ_i. In other conditions, we investigate a termination criterion for $L(i)$ to prevent the algorithm from getting stuck in an infinite loop.

Secondly, we propose an evaluation method to select the optimal preemption threshold with given priorities. A different weight is allocated to different process based on the difference between the deadline and WCRT. We calculate the fitness value of the arbitrary two assignments combined the weight and the quantification of the relative differences. Then we compare the better one with others and update the better one continually until all assignments are compared.

Thirdly, we propose the preemption threshold optimization algorithm with given priorities (PTOGP) to search the near-optimal assignments of the preemp-

tion threshold with given priorities by taking the assignment of the preemption threshold obtained by the existing algorithm [11] as initial minimum boundary. PTOGP uses Tabu search (TS) [18,19] or traversal search based on the size of the processes and obtains a better assignment of the preemption threshold.

Finally, we propose TPTPOA to search the global near-optimal assignments for the preemption threshold and the priority. Here, PTOGP is used as a function. As a result, the search space is decreased by one dimension. It is excellent for decreasing the computational complexity and space complexity of the scheduling algorithm.

The rest of the paper is organized as following. Section 2 gives the task model of the preemption threshold scheduling problem. Section 3 summarizes the background. In Sect. 4, we investigate a termination criterion for $L(i)$ and propose TPTPOA. Experiment results are presented in Sect. 5. Section 6 concludes the paper.

2 Optimization Model of the Preemption Threshold Scheduling Problem

On a single processor Λ, we assume a set of real-time processes $\Pi = \{\tau_1, \tau_2, \tau_3 \cdots \tau_n\}$. Each process τ_i consists of three elements C_i, D_i, T_i. They denote the execution time, the deadline and the period of process τ_i respectively. In addition, each process τ_i has a fixed priority P_i, and a threshold θ_i. There are some hypotheses.

- The switching time of the process is not considered. It means that the switch is instant whether it is caused by the preemption or the completion of a process. In addition, there is no blocking time caused by resource occupancy.
- The process will not stop unless it is preempted by other processes or it is finished.
- The priority $P_i \in [1, n]$ is unique for all processes, but the threshold $\theta_i \in [P_i, n]$ can be the same for different processes.
- The execution time is less than or equal to the deadline. At the same time, the deadline is less than or equal to the period of each process.
- The larger the value of P_i is, the higher priority the process has. Once process τ_i is being executed, it can only be preempted when the priority P_j of the new arrival process τ_j is larger than threshold θ_i.
- The necessary and sufficient condition for each process schedulability is that the WCRT of each process is not more than its deadline.

For a set of processes $\Pi = \{\tau_1, \tau_2, \tau_3 \cdots \tau_n\}$, the aim of the preemption threshold scheduling problem is to find whether there exists a feasible $\{P_i, \theta_i\}$ $(i = 1 \cdots n)$ that can make Π schedulable and give at least the near-optimal assignments if the feasible assignments are available.

3 Background

In this section, we introduce the method to calculate WCRT with a given assignment of the priority and the preemption threshold, and to obtain a feasible assignment of the preemption threshold when the priorities are given.

3.1 Analysis of the Worse-Case Response Time

The definition of the response time is shown as Fig. 1. Time 0 indicates the arrival time of a process. 'Start' and 'Finish' indicate the time when the process begins and finishes respectively. [0, Start] contains the blocking time of the lower priority processes with the higher preemption thresholds and the execution time of the higher priority processes arrived when the process is blocked. [Start, Finish] contains the execution time of the processes whose priorities are higher than the current process's preemption threshold and the current process's execution time.

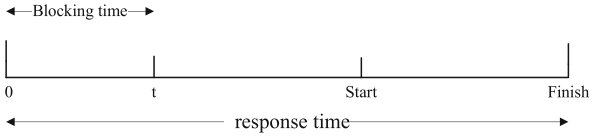

Fig. 1. The structure of the response time.

The definition of the blocking time of process τ_i is as following.

$$B(\tau_i) = max\{C_j | P_j < P_i \leq \theta_i\} \tag{1}$$

The level-i busy period is extended to calculate WCRT. It begins with the critical instant [8,11]. The process is as following [16,17].

$$\begin{cases} L^{(0)}(i) = B_i + Ci \\ L^{(s)}(i) = B_i + \sum_{\tau_j, P_j \geq P_i} \left\lceil \frac{L^{(s-1)}(i)}{T_j} \right\rceil C_j \end{cases} \tag{2}$$

We take the minimum value of $L(i)$ when $L^{(s)}(i) = L^{(s-1)}(i)$ as the length of the level-i busy period for process τ_i. Here, the number of the execution times in $L(i)$ for process τ_i can be calculated by the following formula.

$$M_i = \left\lceil \frac{L(i)}{T_i} \right\rceil \tag{3}$$

For all instances of process τ_i, the start time is as following [11,16,17].

$$\begin{cases} S^{(0)}(i,k) = B_i + \sum_{\tau_j, P_j \geq P_i} C_j \qquad (k = 1, 2 \cdots M_i) \\ S^{(s)}(i,k) = B_i + (k-1)C_j + \sum_{\tau_j, P_j \geq P_i} (\left\lfloor \frac{S^{(s-1)}(i,k)}{T_j} \right\rfloor C_j) \end{cases} \tag{4}$$

We take $S(i,k)$ when $S^{(s)}(i,k) = S^{(s-1)}(i,k)$ as the start time for the k-th instance of process τ_i.

The completion time for all instances of process τ_i is as following.

$$\begin{cases} F^{(0)}(i,k) = S(i,k) + C_i & (k = 1, 2 \cdots M_i) \\ F^{(s)}(i,k) = S(i,k) + C_i + \sum_{\tau_j, P_j > \theta_i} (\left\lceil \frac{F^{(s-1)}(i,k)}{T_j} \right\rceil - \left\lfloor \frac{S^{(s-1)}(i,k)}{T_j} \right\rfloor + 1) C_j \end{cases} \quad (5)$$

Similar to the start time, we take $F(i,k)$ when $F^{(s)}(i,k) = F^{(s-1)}(i,k)$ as the completion time for the k-th instance of process τ_i.

Therefore, we obtain WCRT for process τ_i using the following formula.

$$R_{imax} = max\{F(i,k) - (k-1)T_i\} \qquad (k = 1, 2 \cdots M_i) \quad (6)$$

3.2 The Minimum Boundary of the Preemption Threshold Assignment with Given Priorities

There are some lemmas [11] which indicate the search strategy for a feasible assignment of the preemption threshold.

Lemma 1. The worse-case response time of the processes with the lower priority will not change when processes with the higher priority change their preemption thresholds.

Lemma 2. For a set of schedulable processes, if one of the processes reduces its preemption threshold and is still schedulable, the set of the processes will also be schedulable.

Lemma 3. For a particular process, if we set its preemption threshold to n, it is still non-schedulable. It cannot be schedulable for arbitrary preemption threshold.

Based on the above lemmas, we can obtain the idea of searching the preemption thresholds minimum boundary. The assignment of the preemption threshold can start from the lowest priority process to the highest priority process. The preemption threshold should be increased from processs priority value to the maximum which denotes the number of processes until the process is schedulable. If it is still non-schedulable when the preemption threshold is the maximum, the minimum boundary of the preemption threshold is non-existent for the current assignment of the priority.

4 TS-Preemption Threshold and Priority Optimization Algorithm

In this section, we propose TPTPOA to search near-optimal assignments of the priority and the preemption threshold. The whole structure of the algorithm is shown as Fig. 2.

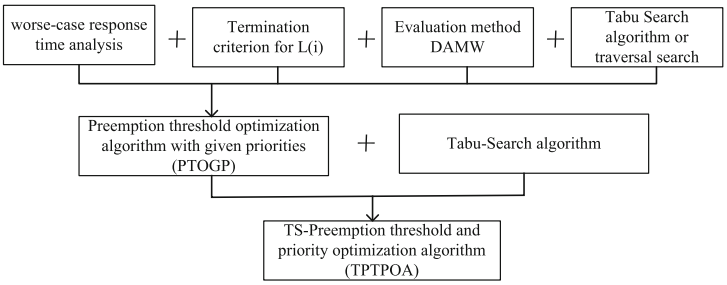

Fig. 2. The whole structure of TPTPOA.

4.1 Termination Criterion When $L(i)$ is Not Convergent

Assuming there are M processes which meet $P_j \geq P_i$ and using S_{wcet} to denote the sum of the M processes' execution time. We propose the following two theorems.

Theorem 1. If the sum of S_{wcet} and B_i is not more than the minimum period of the M processes (i.e. $S_{wcet} + B_i \leq min\{T_j \mid P_j \geq P_i\}$), $L(i)$ for process τ_i is convergent.

Proof. Assuming there are only two processes τ_s and τ_k with higher priority than process τ_i, and $T_s = 1.5T_i$, $T_k = 2T_i$. According to the computational method of $L(i)$, the three processes arrive concurrently at time 0 for their first time. $L^{(0)}(i)$ is equal to $B_i + C_i$, and $L^{(1)}(i)$ will be $B_i + C_s + C_k + C_i$. If $B_i + C_s + C_k + C_i$ is not more than T_i, the second period for all three processes are not available at the time $B_i + C_s + C_k + C_i$. Therefore, $L^{(2)}(i)$ will not increase. It means that $L^{(2)}(i) = L^{(1)}(i)$, and $L(i)$ is convergent. In addition, the number of the processes and the size of their periods have no influence on the conclusion.

Theorem 2. If S_{wcet} is more than the maximum period of the M processes (i.e. $S_{wcet} > max\{T_j \mid P_j \geq P_i\}$), $L(i)$ for process τ_i is not convergent.

Proof. Assuming there are only two processes τ_s and τ_k with higher priority than process τ_i, and $T_s = 1.5T_i$, $T_k = 2T_i$. If $C_s + C_k + C_i$ is more than $2T_i$, $B_i + C_s + C_k + C_i$ must be more than $2T_i$. Therefore, $L^{(1)}(i)$ which is equal to $B_i + C_s + C_k + C_i$ is later than the second arrival of the process τ_k with the maximum period. Therefore, the increment from $L^{(1)}(i)$ to $L^{(2)}(i)$ will be not less than $C_s + C_k + C_i$. It means that the interval of $[L^{(2)}(i), L^{(3)}(i)]$ contains at least one new arrival for all three processes. The increment from $L^{(3)}(i)$ to $L^{(4)}(i)$ will be not less than $C_s + C_k + C_i$. As a result, the iteration will be endless and $L(i)$ will be not convergent.

If the conditions of the two theorems are not satisfied, termination criterion is effective to prevent the algorithm from endless loop when $L(i)$ is not convergent.

Termination Criterion. If the difference of $L(i)$ between two consecutive iterations is more than the maximum period of all processes whose priorities are not less than P_i, the algorithm will terminate. It is as following.

$$L^{(s)}(i) - L^{(s-1)}(i) > max\{T_j | P_j \geq P_i\} \tag{7}$$

Assuming $L(i)$ is convergent, $L^{(s+1)}(i)$ is equal to $L^{(s)}(i)$ and $L^{(s)}(i)$ is equal to $L^{(s-1)}(i)$. Therefore,

$$\sum_{\tau_j, P_j \geq P_i} \left\lceil \frac{L^{(s-1)}(i)}{T_j} \right\rceil C_j = \sum_{\tau_j, P_j \geq P_i} \left\lceil \frac{L^{(s)}(i)}{T_j} \right\rceil C_j \tag{8}$$

For different processes, the execution time is different. Therefore,

$$\begin{cases} \left\lceil \frac{L^{(s)}(i)}{T_{j_i}} \right\rceil = \left\lceil \frac{L^{(s-1)}(i)}{T_{j_i}} \right\rceil \\ \quad \vdots \quad \vdots \quad \vdots \\ \left\lceil \frac{L^{(s)}(i)}{T_{j_M}} \right\rceil = \left\lceil \frac{L^{(s-1)}(i)}{T_{j_M}} \right\rceil \end{cases} \Rightarrow \begin{cases} n_1 T_{j_M} < L^{(s-1)}(i) \leq L^{(s)}(i) \leq (n_1+1) T_{j_M} \\ \quad \vdots \quad \vdots \quad \vdots \quad \vdots \\ n_M T_{j_M} < L^{(s-1)}(i) \leq L^{(s)}(i) \leq (n_M+1) T_{j_M} \end{cases} \tag{9}$$

We find that $L^{(s-1)}(i)$ and $L^{(s)}(i)$ are in the same period for all processes. It means that $L^{(s)}(i)$ appears before the next arrival of all M processes.

We choose the maximum period of all processes whose priorities are not less than the current process's priority as the threshold of the increment of two adjacent iterations. If the increment is more than the maximum period, the algorithm will terminate. It means that the set of processes is non-schedulable with the current priority. It is worth mentioning that the termination criterion is only a sufficient condition.

4.2 Evaluation for Assignment of the Preemption Threshold

For a set of processes, there may exist many feasible assignments of the preemption threshold with given priorities. Therefore, rational evaluation method is essential for selecting the best assignment and optimizing the design of algorithm. We propose DAMW to solve this problem.

The difference between the deadline and WCRT can be used to judge the schedulability of a process. We define a variable V_{mi} to represent the difference and evaluate which preemption threshold has a better performance for process τ_i. The bigger value of V_{mi} means the better fault tolerance and the better performance. In addition, we also define F_{mi} to denote the superiority of V^*_{mi} which is the *i-th* element in the sorted V_m in ascending order compared with $V^*_{ni} (n \neq m)$. F_m is the sum of F_{mi} and denotes the fitness value of V_m. For a large number of feasible assignments, we choose two of them randomly and obtain the better one by using DAMW, and compare the better one with all other assignments. The detailed procedure of DAMW for two assignments is as following.

1. Obtain two sets $V_1 = \{V_{11}, V_{12}, \cdots V_{1n}\}$ and $V_2 = \{V_{21}, V_{22}, \cdots V_{2n}\}$ by using $V_{mi} = D_i - WCRT_i$ $(m = 1, 2;\ i = 1, 2 \cdots n)$
2. Sort elements in ascending order for each group V_m $(m = 1, 2)$
3. Assign weight as $(n - i + 1)$ for each sorted element V^*_{mi} $(m = 1, 2;\ i = 1, 2 \cdots n)$
4. Calculate F_m $(m = 1, 2)$ for each group respectively.

For each set V^*_{1i} and V^*_{2i} $(i = 1, 2 \cdots n)$, we compare the size of the two values. There are three possibilities.

- If $V^*_{1i} > V^*_{2i}$, $(n-i+1)$ * $(V^*_{1i} - V^*_{2i})$ / V^*_{1i} will be added to F_1, and F_2 remains unchanged
- If $V^*_{1i} < V^*_{2i}$, $(n-i+1)$ * $(V^*_{2i} - V^*_{1i})$ / V^*_{2i} will be added to F_2, and F_1 remains unchanged
- If $V^*_{1i} = V^*_{2i}$, neither F_1 not F_2 changes its value.

Compare F_1 and F_2, and the larger one corresponds to the better assignment of the preemption threshold.

4.3 The Preemption Threshold Optimization Algorithm with Given Priorities

We adopt different strategies based on the size of the processes. The traversing method is used for small scale of processes. For the large scale of processes, we propose TS-preemption threshold optimization algorithm based on TS. Here, we take 6 processes as the boundary value. The detailed processes are listed as below.

1. Use the algorithm introduced in Sect. 3 to calculate the minimum boundary of the preemption threshold with given priorities. If the minimum boundary is not exist, there is no feasible assignment of the preemption threshold with the given priorities.
2. Use the minimum boundary to initialize the current assignment and the optimal assignment, and initialize the Tabu list and put the minimum boundary into it.
3. Search all neighborhoods of the current assignment and obtain the best one with DAMW.
4. Compare the optimal assignment in the neighborhood with the assignments in the Tabu list. If it is already in the list, set it to zero and return to 3.
5. Use DAMW to compare the optimal assignment in the neighborhood with the optimal assignment. If the former one is better than the later one, use the optimal assignment in the neighborhood to update the optimal assignment.
6. Use the optimal assignment in the neighborhood to update the current assignment and put it into the Tabu list. Return to 3.
7. Terminate iteration and output the optimal preemption threshold.

4.4 TS-Preemption Threshold and Priority Optimization Algorithm

Based on the algorithm in Sect. 4.3, at least the near-optimal assignments with given priorities can be obtained. If we set up a corresponding relationship between the preemption thresholds and the given priorities, the problem will reduce one dimension. However, the solution space of the priority optimization is $O(n!)$. We propose an effective algorithm based on TS to solve this problem. The process is similar to the Sect. 4.3 and the differences between them are listed as below.

- EDF [20] has been proved to be the most effective algorithm for the preemptive scheduling problem. Although the preemption threshold scheduling is not fully preemptive and its priority is fixed, the assignment of the priority based on EDF is still a good solution and can be used as the initial assignment of the priority.
- The neighborhood solutions of the priority are obtained by exchanging the priority with each other in one step, while the neighborhood solutions of the preemption threshold only have one process with different threshold compared with the current assignment. The demonstration for it is shown as Table 1.

Table 1. The differences of the neighborhood for priority and preemption threshold

Process	Current assignment	Neighborhood of priority			Neighborhood of threshold		
		1st	2nd	3rd	1st	2nd	3rd
τ_1	1	2	3	1	2	3	1
τ_2	2	1	2	3	2	2	3
τ_3	3	3	1	2	3	3	3

4.5 The Flowchart of the TPTPOA

The flowchart is shown as Fig. 3. Switching the sequence is carried out four times. The purpose of the first two times is to guarantee that the processes' order is from the lowest priority to the highest priority, which can bring great convenience for the optimization of the preemption threshold. The third time is to ensure that the optimal neighborhood assignment of the preemption threshold corresponds to the right sequence of the processes when compared with the Tabu list. The fourth execution is to guarantee that the optimal assignments are in the original processes sequence.

5 Experiments and Results

The following experiments are implemented in Matlab R2014a, which runs in a computer with Inter(R) Core(TM) i5-4590T CPU and 4 GB RAM.

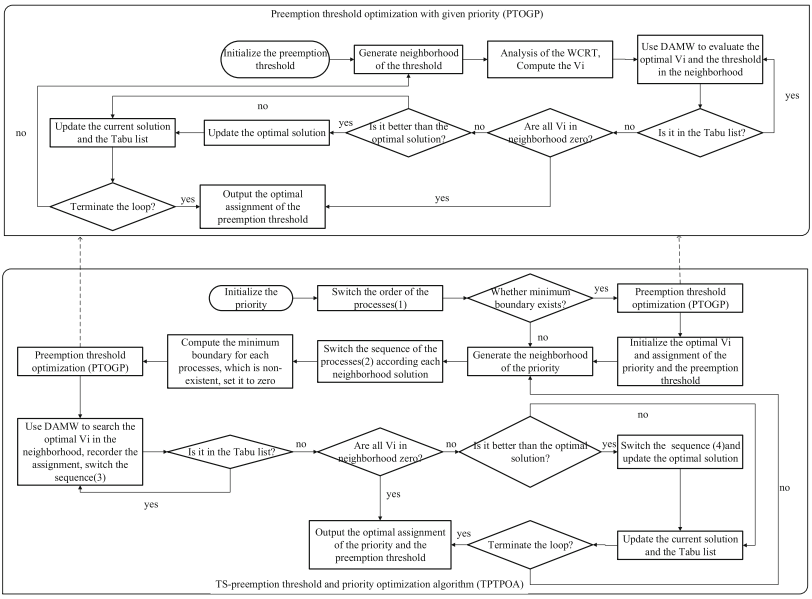

Fig. 3. The flowchart of TPTPOA

5.1 The Performance Analysis of TPTPOA

We conduct five experiments to test the validity of TPTPOA. Four, five, six, seven, eight processes are used respectively. In the algorithm, we use six as the threshold to decide which one will be used between TS and traversal search when optimizing the preemption threshold. However, we use both of them in the experiments respectively. We define TPTPOA when TS is used to optimize the preemption threshold as $TPTPOA_1$, and TPTPOA when traversal search is used to optimize the preemption threshold as $TPTPOA_2$. $Priority/Threshold_1$ and $Priority/Threshold_2$ denotes the priorities and the preemption thresholds obtained by $TPTPOA_1$ and $TPTPOA_2$ respectively. $WCRT_1$ and $WCRT_2$ denotes the worse-case response time for each process corresponding to the assignments searched by $TPTPOA_1$ and $TPTPOA_2$ respectively.

As shown in Tables 2, 3 and 5, the assignments of the priority and the preemption threshold obtained by $TPTPOA_1$ and $TPTPOA_2$ are the same. In Table 4, the assignments obtained by $TPTPOA_1$ is a part of the assignments obtained by $TPTPOA_2$. For more processes, $TPTPOA_1$ may not obtain the optimal solutions, but it can also obtain the near-optimal assignments at least. Therefore, TPTPOA with TS used to optimize the preemption threshold is an effective algorithm to optimize the assignments for the priority and the preemption threshold.

For the results in Tables 2, 3, 4, 5 and 6, the running time for $TPTPOA_1$ is 0.518199 s, 0.281927 s, 2.227995 s, 4.914453 s, 6.639471 s, respectively. The running time for $TPTPOA_2$ is 0.201472 s, 0.141070 s, 6.551539 s, 317.807542 s, none,

Table 2. The assignments of the threshold and the priority with four processes

Process	1	2	3	4
Deadline	140	180	160	90
WCET	18	40	35	30
Period	140	180	160	120
$Priority/Threshold_1$	3/3	1/4	2/4	4/4
	3/4	1/4	2/4	4/4
	3/3	2/4	1/4	4/4
	3/4	2/4	1/4	4/4
$Priority/Threshold_2$	3/3	1/4	2/4	4/4
	3/4	1/4	2/4	4/4
	3/3	2/4	1/4	4/4
	3/4	2/4	1/4	4/4
$WCRT_1$	88	123	123	70
$WCRT_2$	88	123	123	70

Table 3. The assignments of the threshold and the priority with five processes

Process	1	2	3	4	5
Deadline	120	160	180	140	100
WCET	25	35	40	30	20
Period	140	180	200	160	120
$Priority/Threshold_1$	4/4	2/5	1/5	3/3	5/5
	4/5	2/5	1/5	3/3	5/5
	4/4	2/5	1/5	3/4	5/5
	...	...	...	...	...
$Priority/Threshold_2$	4/4	2/5	1/5	3/3	5/5
	4/5	2/5	1/5	3/3	5/5
	4/4	2/5	1/5	3/4	5/5
	...	...	...	...	...
$WCRT_1$	85	150	150	115	60
$WCRT_2$	85	150	150	115	60

respectively. None denotes we cannot obtain the results in a short and tolerable time. When the number of the processes is less than six, $TPTPOA_2$ have less computational cost than $TPTPOA_1$, and it also obtains all the assignments with the same optimal WCRT. However, the computational cost will increase sharply if the number of the processes is not less than six. It cannot obtain the effective assignments in a short time with only eight processes in Table 6. Compared with $TPTPOA_2$, the computational cost of $TPTPOA_1$ increases slowly, and it will

Table 4. The assignments of the threshold and the priority with six processes

Process	1	2	3	4	5	6
Deadline	120	160	180	140	150	200
WCET	25	28	24	21	22	30
Period	140	180	200	160	160	240
$Priority/Threshold_1$	6/6	3/3	2/6	5/5	4/4	1/6
	6/6	3/4	2/6	5/5	4/4	1/6
	...	...	...	...	...	...
$Priority/Threshold_2$	6/6	3/3	2/6	5/6	4/4	1/6
	6/6	3/3	2/6	5/5	4/4	1/6
	6/6	3/3	2/6	5/6	4/5	1/6
	...	...	...	...	...	...
$WCRT_1$	55	126	150	76	98	150
$WCRT_2$	55	126	150	76	98	150

Table 5. The assignments of the threshold and the priority with seven processes

Process	1	2	3	4	5	6	7
Deadline	140	170	160	160	170	150	180
WCET	20	23	21	25	26	25	20
Period	150	180	175	180	190	180	195
$Priority/Threshold_1$	7/7	2/7	5/5	4/4	3/3	6/6	1/7
	7/7	2/7	5/6	4/4	3/3	6/6	1/7
	7/7	2/7	5/7	4/4	3/3	6/6	1/7
$Priority/Threshold_2$	7/7	2/7	5/5	4/4	3/3	6/6	1/7
	7/7	2/7	5/6	4/4	3/3	6/6	1/7
	7/7	2/7	5/7	4/4	3/3	6/6	1/7
$WCRT_1$	43	160	89	114	140	68	160
$WCRT_2$	43	160	89	114	140	68	160

significantly lower than that of $TPTPOA_2$ when the number of the processes is not less than six. Therefore, we select six as the threshold to decide which strategy will be used to optimize the preemption threshold in TPTPOA. In addition, though the running time increases slowly for TPTPOA when the number of the processes is not less than six, it may also take much time when the number of the processes is very large. We can reduce the number of the neighborhood solutions involved in updating the optimal solution in each iteration to control the increasing time cost effectively.

Table 6. The assignments of the threshold and the priority with eight processes

Process	1	2	3	4	5	6	7	8
Deadline	140	170	160	160	170	150	180	250
WCET	20	23	21	21	24	23	20	18
Period	150	180	175	180	190	180	195	280
$Priority/Threshold_1$	8/8	2/8	5/5	6/6	4/4	7/7	3/8	1/7
	8/8	2/8	5/5	6/6	4/4	7/7	3/8	1/8
	8/8	2/8	5/6	6/6	4/4	7/7	3/8	1/8
	8/8	2/8	5/6	6/6	4/4	7/7	3/8	1/7
	8/8	2/8	5/7	6/6	4/4	7/7	3/8	1/7
	8/8	2/8	5/7	6/6	4/4	7/7	3/8	1/8
	8/8	2/8	5/8	6/6	4/4	7/7	3/8	1/8
	8/8	2/8	5/8	6/6	4/4	7/7	3/8	1/7
	8/8	2/8	5/8	6/7	4/4	7/7	3/8	1/7
$WCRT_1$	43	170	108	87	132	66	152	190

5.2 Summary

We have following conclusions based on the experiment results.

Firstly, TPTPOA can obtain the assignments for both preemption threshold and priority, but the existing algorithm can only have the preemption threshold. As shown in Table 5, TPTPOA has 18 assignments of the priority and the preemption threshold with the same WCRT. We can select arbitrary one as the final assignment.

Secondly, the initial assignment of the processes' priority based on the main idea of EDF is not necessarily the optimal solution. It can be seen from Table 6. Process 2 have the less deadline than process 7, while the priority of process 2 is also less than that of process 7 in the optimal assignments. However, the assignment of the processes' priority based on EDF is still an effective initial assignment and it even could be the optimal assignment of the priority occasionally.

Finally, the better assignment does not necessarily have the larger preemption threshold. If the execution time of the process with the lowest priority is the largest one, every increase by one of its preemption threshold will make the number of the affected processes increase one. For each affected process, its blocking time will have a significant increase. It means that the performance of the new assignment becomes worse and the processes could even be non-schedulable. There is an example with the given priorities in Table 7. For process 1, all feasible preemption thresholds are 2, 3 and 4. However, the optimal preemption threshold is 2 instead of 4.

Table 7. The assignments of threshold with given priorities

Process	1	2	3	4
Deadline	200	140	100	90
WCET	40	35	30	25
Period	250	160	140	120
Minimum boundary	2	4	3	4
All feasible thresholds	2	4	3	4
	2	4	4	4
	3	4	3	4
	3	4	4	4
	4	4	3	4
	4	4	4	4
Optimal threshold	2	4	3	4
	2	4	4	4

6 Conclusion

This paper focuses on the process scheduling problem for IMA. Compared with the fully preemptive and fully non-preemptive scheduling, PTS is a more comprehensive scheduling strategy. Aiming at the problem of assigning preemption threshold and priority, we propose TPTPOA. The experiments show that algorithm has more extensive global searching ability than the existing algorithm. At least near-optimal solutions can be obtained within the acceptable time. Though the running time will increase when the number of the processes is too large, the effective assignments can still be obtained by decreasing the number of the neighborhood solutions in each iteration. Therefore, TPTPOA can be used in IMA for processes' scheduling and other real-time systems meeting the hypotheses to improve their schedulability.

The future work focuses on applying the algorithm to the model considering the sequential relationship of the process, and improving the efficiency of the algorithm.

Acknowledgments. This research is supported by the National Natural Science Foundation of China under Grant No. 61671041.

References

1. Miller, S.P., Cofer, D.D., Sha, L., Meseguer, J., Al-Nayeem, A.: Implementing logical synchrony in integrated modular avionics. In: 2009 IEEE/AIAA 28th Digital Avionics Systems Conference, Orlando, FL, pp. 1.A.3-1–1.A.3-12 (2009)
2. Ju, H., Wang, S., Zhao, T.: A modeling method of IMA dynamic reconfiguration based on AADL. In: 2015 First International Conference on Reliability Systems Engineering (ICRSE), Beijing, pp. 1–5 (2015)

3. Aeronautical Radio, Inc.: ARINC653 P1-2, Avionics application software standard interface part 1-required services, pp. 2–45 (2006)
4. Pan, L.Q., He, C., Tian, Y., Su, Y.S., Zhang, X.Y.: A region division based diversity main-taining approach for many-objective optimization. Integr. Comput. Aided Eng. **24**(3), 1–18 (2017)
5. George, L., Rivierre, N., Spuri, M.: Preemptive and non-preemptive real-time uniprocessor scheduling. Technical report N2966, INRIA, pp. 1–55 (1996)
6. Baruah, S.K., Rosier, L.E., Howell, R.R.: Algorithms and complexity concerning the preemptive scheduling of periodic, real-time tasks on one processor. Real Time Syst. **2**, 301–324 (1990)
7. Marouf, M., Sorel, Y.: Scheduling non-preemptive hard real-time tasks with strict periods. In: ETFA 2011, Toulouse, pp. 1–8 (2011)
8. Liu, C.L., Layland, J.W.: Scheduling algorithms for multiprogramming in a hard real-time environment. J. Assoc. Comput. Mach. **20**, 40–61 (1973)
9. Lehoczky, J., Sha, L., Ding, Y.: The rate monotonic scheduling algorithm: exact characterization and average case behavior. In: Proceedings of Real-Time Systems Symposium, Santa Monica, CA, pp. 166–171 (1989)
10. Bertossi, A.A., Fusiello, A., Mancini, L.V.: Fault-tolerant deadline-monotonic algorithm for scheduling hard-real-time tasks. In: Proceedings of the 11th International Parallel Processing Symposium, Geneva, pp. 133–138 (1997)
11. Wang, Y., Saksena, M.: Scheduling fixed-priority tasks with preemption threshold. In: Sixth International Conference on Real-Time Computing Systems and Applications, RTCSA 1999, Hong Kong, pp. 328–335 (1999)
12. Lehoczky, J.P.: Fixed priority scheduling of periodic task sets with arbitrary deadlines. In: Proceedings of the 11th Real-Time Systems Symposium, Lake Buena Vista, FL, pp. 201–209 (1990)
13. Bril, R.J., Lukkien, J.J., Verhaegh, W.F.J.: Worst-case response time analysis of real-time tasks under fixed-priority scheduling with deferred preemption revisited. In: 19th Euromicro Conference on Real-Time Systems (ECRTS 2007), Pisa, pp. 269–279 (2007)
14. Joseph, M., Pandya, P.: Finding response times in a real-time system. Comput. J. **29**, 390–395 (1986)
15. Redell, O., Torngren, M.: Calculating exact worst case response times for static priority scheduled tasks with offsets and jitter. In: Proceedings of the Eighth IEEE Real-Time and Embedded Technology and Applications Symposium, pp. 164–172 (2002)
16. Keskin, U., Bril, R.J., Lukkien, J.J.: Exact response-time analysis for fixed-priority preemption-threshold scheduling. In: 2010 IEEE 15th Conference on Emerging Technologies and Factory Automation (ETFA 2010), Bilbao, pp. 1–4 (2010)
17. Buttazzo, G.C., Bertogna, M., Yao, G.: Limited preemptive scheduling for real-time systems. A survey. IEEE Trans. Industr. Inf. **9**(1), 3–15 (2013)
18. He, L., Yabo, L., Hong, L.: Schedule optimization for NC resource sharing based-on greed algorithm and Tabu search. In: 2009 Second International Conference on Intelligent Computation Technology and Automation, Changsha, Hunan, pp. 282–285 (2009)
19. Darmawan, I., Kuspriyanto, Priyana, Y., Joseph, M.I.: Grid computing process improvement through computing resource scheduling using genetic algorithm and Tabu Search integration. In: 2012 7th International Conference on Telecommunication Systems, Services, and Applications (TSSA), Bali, pp. 330–334 (2012)
20. Lee, J., Shin, K.G.: Preempt a job or not in EDF scheduling of uniprocessor systems. IEEE Trans. Comput. **63**(5), 1197–1206 (2014)

http://www.springer.com/978-981-10-7178-2

Bio-inspired Computing: Theories and Applications
12th International Conference, BIC-TA 2017, Harbin, China, December 1–3, 2017, Proceedings
He, C.; Mo, H.; Pan, L.; Zhao, Y. (Eds.)
2017, XV, 638 p. 290 illus., Softcover
ISBN: 978-981-10-7178-2