A Review of Task Scheduling Problems

Haibing Cheng[1,2](✉), Lin Li[2], and Ling You[2]

[1] PLA Strategic Support Force Information Engineering University, Zhengzhou 450001, China
277345570@qq.com

[2] National Key Laboratory of Science and Technology on Blind Signal Processing, Chengdu 610041, China
hb19980101@stu.xjty.edu.cn

Abstract. With the development of computer application in recent years, the computational allocation of resources becomes more important. Many Scholars have done a lot of research on task scheduling problem. In this paper, we classify scheduling problems by classifying the nature of tasks, and make a survey with the existing schedulingalgorithms and the decision conditions of scheduling problems.

Keywords: task scheduling · algorithms · decision conditions

1 Introduction

Task scheduling problem is a study of how to allocate scarce resources to different tasks within a certain period of time. It is a decision-making process that aims to optimize one or more goals. In this section, we first describe the model of the task, and then classify the scheduling problems.

1.1 Task Scheduling

The relevant terms used in task scheduling are defined as follows. In this paper, a six-tuple $(s_i, c_i, t_i, d_i, p_i, h_i)$ is used to describe a single periodic task τ_i, which is shown in Fig. 1.

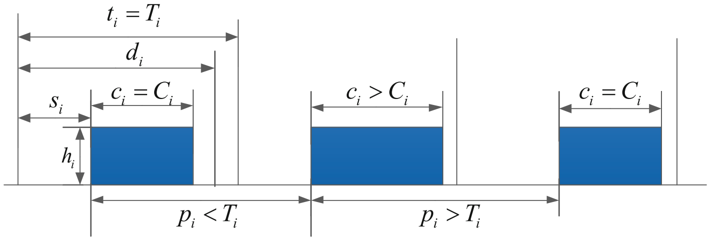

Fig. 1. Periodic task model diagram

J. Yen et al. (Eds.): ICBIS 2023, AHCS 14, pp. 1171–1179, 2023.
https://doi.org/10.2991/978-94-6463-198-2_121

Where s_i is the time interval between the request time and the execution start time of the task, it's generally thought that for tasks τ_i, $s_i = S_i$. c_i indicates the time required for a single execution of a task, $c_i = C_i$, and C_i indicates the minimum duration required for a single execution of a task. t_i indicates the time interval of two requests for the same task, this paper considers that the periodic task is satisfied $t_i = T_i$. d_i indicates the relative deadline of the task, the task must be completed within the deadline. p_i indicates the time interval between two adjacent execution moments of a task. h_i represents the power of the fourth task.

1.2 Scheduling Problem Classification

According to the description of periodic tasks above, scheduling problems are classified from different perspectives:

1. Preemptive scheduling and non-preemptive scheduling

When a task starts to be executed, if another task with a higher priority appears, you can divide scheduling problems into preemptive scheduling problems and non-preemptive scheduling problems based on whether the current task can be replaced. If a lower-priority task can be interrupted by a higher-priority task, it is called preemptive scheduling.

2. Periodic task scheduling and non-periodic tasks scheduling

Tasks are divided into periodic tasks and non-periodic tasks according to whether the time interval of two adjacent request moments of the same task is constant or not. The time interval between two requests for the same task $t_i = T_i$, T_i is a constant value, the task is called a periodic task. Otherwise, it is called non-periodic task.

3. Strict periodic tasks scheduling and non-strict periodic tasks scheduling

Periodic tasks are divided into strictly periodic tasks and non-strictly periodic tasks based on whether the time interval of two adjacent execution moments of the same task is constant. If $p_i = P_i$, where P_i is constant less than or equal to T_i, the task is called strictly periodic task. On the contrary, it is called non-strictly periodic task.

At present, great progress has been made in the research of scheduling problems in the academic circle. Next section summarizes the existing researches in the academic circle respectively in terms of the classification of scheduling problems (preemptive strict periodic task scheduling, non-preemptive strict periodic task scheduling, preemptive non-strict periodic task scheduling, non-preemptive non-strict periodic task scheduling, and non-periodic task scheduling).

2 First Level Heading

2.1 Preemptive Strict Periodic Task Scheduling

At present, some mature results have been obtained for preemptive strict periodic task scheduling. In 1973, Liu and Layland made a detailed study on this problem [1], and proposed the static priority scheduling algorithm RM algorithm and the dynamic priority scheduling algorithm EDF algorithm. Among them, RM algorithm assigns its priority according to the task cycle. The shorter the cycle, the higher the priority of the task, and the priority of the task is fixed. The EDF algorithm determines the priority of a task by the distance between the task deadline and the current moment. The closer the distance is, the higher the priority will be. It is also proved that the two algorithms are respectively the optimal algorithms in the single-processor static priority scheduling algorithm and the dynamic priority scheduling algorithm (if the algorithm can not schedule the task set, there is no other algorithm can schedule). At the same time, the conditions of single-processor scheduling for the two algorithms are given.

A task set $(\tau_1, \cdots, \tau_m)$ is giving, and if the utilization rate of this task satisfies the condition of Eq. (1), then the task set can be scheduled using RM algorithm. When the number of tasks tends to infinity, it can be scheduled by RM algorithm when the utilization ratio of task set is less than 0.693.

$$U \leq m \cdot \left(2^{1/m} - 1\right) \tag{1}$$

For a task set composed of m tasks, the task set can be scheduled by EDF algorithm if and only if the utilization rate of the m tasks satisfies Eq. (2).

$$U \leq 1 \tag{2}$$

However, the schedulable judgment condition of RM algorithm in Eq. (2) is only a sufficient condition, some task sets that the utilization exceeds this limit can still be scheduled by RM algorithm [2]. Therefore, Lehoczky et al. conducted a more accurate study on the characteristics of static priority scheduling algorithm and proposed sufficient and necessary conditions for the schedulability determination of RM algorithm [3]. For a task set $(\tau_1, \cdots, \tau_m)$, Eq. (3) which is used in [3] to represent the accumulated demand of the previous i task in the task set for the processor within a time period $[0, t]$. Then, relevant parameters are defined as shown in Eq. (3):

$$\begin{gathered} W_i(t) = \sum_{j=1}^{i} C_j \cdot \lceil \tfrac{t}{T_j} \rceil \\ L_i(t) = \tfrac{W_i(t)}{t} \\ L_i = min_{\{0 \leq t \leq T_i\}} L_i(t) \\ L = max_{\{0 \leq t \leq T_i\}} L_i \end{gathered} \tag{3}$$

Based on the above definition, Lehoczky et al. gave the sufficient and necessary conditions for the schedulability of this task set as $L \leq 1$. RM algorithm requires the task's deadline to be equal to its period, and for those tasks whose deadline is not equal to the period, [4] proposes a static priority scheduling algorithm, deadtime-monotonic (DM) algorithm. Similar to RM algorithm, DM algorithm defines the priority of a task according to the size of the task's deadline. The task with the shortest deadline has the highest priority, and it is proved that DM algorithm is the best static priority scheduling algorithm for single processor when the deadline is less than the cycle.

A task set transformation method is proposed in [5]. By transforming the cycle and execution time of the original task, it is proved that if the transformed task set can be scheduled by a single processor, then the original task set can be scheduled by a single processor. The specific transformation method is shown in Eq. (4).

$$T_i' = T_i \cdot 2^{log\lfloor T_m/T_i \rfloor}$$
$$C_i' = C_i \cdot 2^{log\lfloor T_m/T_i \rfloor} \tag{4}$$

where $T_i\prime$ and $C_i\prime$ are the period and execution time of the task i in the changed task set, T_i and C_i are the period and execution time of the task i in the original task set, and T_m are the period of the task with the largest period among all the tasks in the original task set. In addition, it also analysed a special task set with increasing task cycles and the cycle ratio of adjacent tasks less than 2, and proved that sufficient conditions for this task set to be scheduled by a single processor were shown in Eq. (5) in [5].

$$\sum_{i=1}^{m} \frac{C_i}{T_i} \leq \sum_{i=1}^{m-1} \left[\frac{T_{i+1}}{T_i} \right] + 2\frac{T_1}{T_m} - m \tag{5}$$

where T_i and C_i are the cycle and execution time of the task i in the special task set, and T_{i+1}, T_i, T_m represents the cycle of the task $i+1$, i and the maximum cycle of the task in the task set, and m represents the number of tasks in the task set.

The above researches are all aimed at the scheduling decision conditions and scheduling algorithms on single processor. The current researches on task scheduling on multiprocessor are not very mature. The packing problem in [6] proves that the scheduling problem of assigning a task to a processor has been proved to be NP hard. Meanwhile, based on the prior knowledge in [1], a multi-processor scheduling algorithm with complexity $O(n \cdot log(n))$ is proposed. By transforming the task cycles in the task set, this algorithm can provide a higher utilization bound than the RM algorithm.

In [7], it studies the decision conditions of schedulability of multi-processor tasks, proving that when all tasks in the task set meet the deadline equal to the period, the sufficient and necessary condition of multi-processor schedulability is that the sum of the utilization rate of the task set is less than or equal to n the number of processors, as shown in Eq. (6).

$$U \leq \sum_{i=1}^{m} \frac{C_i}{T_i} \leq n \tag{6}$$

Meanwhile, for all task sets whose deadlines are smaller than their cycles, the scheduling condition satisfying Eq. (6) becomes a sufficient condition for multiprocessor schedulability.

2.2 Preemptive Non-strict Periodic Task Scheduling

In [1], Liu and Layland studied the scheduling problem of preemptive strict periodic tasks, and analyzed and proved that correlation algorithms and schedulable decision conditions were also applicable to preemptive non-strict periodic tasks. It studies the data transmission and communication technology of time deterministic network (TDN) in [8], and proposes a time deterministic network scheduling algorithm. Aiming at the preemptive non-strict periodic task scheduling problem, a non-strict periodic scheduling algorithm based on multi-matrix periodic joint scheduling is designed. The algorithm takes the receiving and sending time of network nodes as the optimization variable, and minimizes the sum of sending time points of the terminal system that triggers TT service at each time in the network as the optimization objective to establish an optimization problem. By solving the optimization problem, the time scheduling table of each network node in the time trigger network is obtained.

2.3 Non-preemptive Strict Periodic Task Scheduling

The execution conditions of tasks are limited in [9], and it is believed that different executions of the same task must be completed on the same processor. In this case, the sufficient and necessary conditions for the schedulability of two tasks by single processor are proved by analyzing the relationship between task cycle and execution time, as shown in Eq. (7).

$$\gcd\left(T_i, T_j\right) \geq C_i + C_j \tag{7}$$

where $\gcd\left(T_i, T_j\right)$ is the greatest common divisor of the period T_i of the task i and the period T_j of the task j, C_i and C_j represents the execution time of the task i and the task j respectively.

At the same time, it is proved that the non-preemptive strict periodic task scheduling problem is NP-complete. On the basis of the study in [9, 10] analysed that when two tasks were extended to m task, the sufficient and necessary conditions of the original uniprocessor schedulability became sufficient conditions, as shown in Eq. (8):

$$\sum_{i=1}^{m} C_i \leq gcd(\forall i, T_i) \tag{8}$$

It further analyzed and proved the decision conditions for the schedulability of two-task single processor in [11]. The sufficient and necessary conditions for the schedulability of two tasks $\tau_i = (c_i, t_i, s_i)$, $\tau_j = \left(c_j, t_j, s_j\right)$ in single processor are shown in Eq. (9).

$$c_i \leq \left(s_j - s_i\right) mod\left(g_{i,j}\right) \leq g_{i,j} - c_j \tag{9}$$

where c_i and c_j respectively represents the execution time of task τ_i and τ_j, s_i and s_j respectively represents the time interval between the request time and the execution time of task τ_i and τ_j, $g_{i,j}$ represents the greatest common divisor of the cycle of task τ_i and τ_j.

It analyzed the sufficient and necessary conditions for the scheduling of the $m - 1$ task when task m has been scheduled by a single processor in [12]. A multi-processor scheduling algorithm is proposed which provides an upper bound on the number of processors required for multi-processor scheduling. [13] proposes the concept of maximum scaling factor and presents a heuristic algorithm to determine schedulability and provide an efficient allocation method in multiprocessors.

A TSS algorithm is proposed in [14], which can transform any task set into a Harmonic task set that is convenient for scheduling. Harmonic task set is defined as any two tasks τ_i and τ_j, satisfied $T_i/T_j = a \vee T_j/T_i = a, a \in \mathrm{N}$. The specific steps of periodic transformation are as follows:

1. Arrange the tasks in the task set in a way that the cycle does not increase;
2. For each task, perform periodic transformation of the task set as shown in Eq. (10);
3. Select a task set with the smallest sum utilization rate of the task sets in all transformation periods as the transformed Harmonic task set.

$$T_j' = \begin{cases} \frac{T_{j+1}'}{T_{j+1}'/T_j}, j < i \\ T_j, j = i \\ T_{j-1}' \cdot \left\lfloor \frac{T_j}{T_{j-1}'} \right\rfloor, j > i \end{cases} \tag{10}$$

where $T_{j-1}\prime$,$T_j\prime$,$T_{j+1}\prime$ respectively represents the period of the task $j-1, j, j+1$ in the task set after the cycle change, and T_j, T_i represents the period of the task j, i in the original task set.

Based on the above conditions, a sufficient condition for the schedulability of the original task set is given, the sum of Harmonic task sets' utilizations is less than or equal to 1.

It has done some research on EDF algorithm in [15], it analyzed and proved the sufficient conditions for the algorithm to be capable of multi-processor scheduling. If the task set $\tau = (\tau_1, \cdots, \tau_m)$ meets the conditions shown in Eq. (11), it can be scheduled by EDF algorithm.

$$\begin{gathered} V_{sum}(\tau) \leq m - (m-1) \cdot V_{max}(\tau) \\ V(\tau_i, \tau) = \frac{c(\tau_i)}{max(0, t(\tau_i) - c_{max}(\tau))} \\ V_{sum}(\tau) = \sum_{\tau_i \in \tau} V(\tau_i, \tau) \\ V_{max}(\tau) = max_{\tau_i \in \tau} V(\tau_i, \tau) \end{gathered} \tag{11}$$

where $c(\tau_i)$, $t(\tau_i)$ represents the execution time and cycle of the task τ_i, and $c_{max}(\tau)$ represents the execution time of the task τ with the longest execution time in the task set.

2.4 Non-preemptive Non-strict Periodic Task Scheduling

As for the non-preemptive non-strict periodic task scheduling problem, [16] has done some studies on this problem, proving whether the non-preemptive non-strict periodic task can be scheduled as NP-complete by one processor, and giving some sufficient and necessary conditions to determine whether the task set is schedulable. At the same time, some related scheduling algorithms, such as pattern pruning method and fast solution method, are given. However, the number of tasks considered in the task scheduling problem is generally small, and no good scheduling algorithm has been proposed for the task set with a large number of tasks.

Pattern pruning method reduces the search space by using pruning method, which can significantly improve the efficiency of traversal algorithm and the speed of obtaining the optimal solution. However, when the number of tasks is large, the time complexity of the algorithm is too high, and the feasible scheme cannot be obtained within the tolerable time.

The computational complexity of the fast solution method is much lower than that of the model pruning method, and the feasible solution can be obtained in a faster time. However, the algorithm mainly scales the value of T the task and determines whether it is schedulable by the sufficient condition of schedulable decision of single processor. So there are some cases where there is a feasible solution, but the algorithm can't get the feasible solution.

3 Non-periodic Task Scheduling

At present, the division of non-periodic tasks in academic circles can be roughly divided into two categories. One is the sporadic tasks with minimum lower bound between task request times, that is $t_i \geq T_i$, and t_i is not equal to a certain constant. The other is that the task is executed only once. For the single-execution non-periodic task scheduling, it mainly analyzes the manufacturing period, total completion time and other objectives. At present, a complete theoretical system has been developed, and this issue has been studied and analyzed in detail in [17]. For single-execution tasks, both preemptive scheduling and non-preemptive scheduling problems have been proven to be NP hard [18].

As for the scheduling problem of sporadic tasks, T.P.Baker et al. proposed sufficient conditions for multi-processor scheduling of sporadic task sets through comparative analysis of sporadic tasks and strict periodic tasks [19][20].

For sporadic task set $\tau = (\tau_1, \cdots, \tau_m)$, when the task set satisfies Eq. (12), it can be scheduled by EDF algorithm on n processor.

$$\sum_{i=1}^{m} \frac{C_i}{T_i} \leq n(1-\lambda) + \lambda \tag{12}$$

where $\lambda = \max\{C_i/T_i | i = 1, \cdots, m\}$.

Based on the research of T.P.Beaker et al., [21] proposed the sufficient condition that sporadic task sets could be scheduled by n processor, as shown in Eq. (13).

$$\sum_{i=1}^{m} \frac{C_i}{T_i} \leq \frac{n^2}{2n-1} \tag{13}$$

As for the static priority scheduling algorithm RM and dynamic priority scheduling algorithm EDF proposed in the previous paper, which have been proved to be the optimal of single processor, [22] has proved that the algorithm is no longer optimal in the multi-processing scheduling problem.

[23] proposes a new task set $\tau = (\tau_1, \cdots, \tau_m)$, which is composed of strict periodic tasks and sporadic tasks. By analyzing relevant parameter information of the task set, it

is proved that sufficient conditions for the task set to be scheduled on n processing are shown in Eq. (14).

$$\begin{gathered}\lambda_{tot} \leq n(1-\lambda_{max}) + \lambda_{max} \\ \lambda_k = \frac{C_k}{D_k}\end{gathered} \tag{14}$$

where C_k and D_k respectively indicate the execution time and deadline of the task τ_k, $\lambda_{tot} = \sum_{\tau_k \in \tau} \lambda_k$, $\lambda_{max} = max_{\tau_k \in \tau}(\lambda_k)$.

4 Conclusion

This paper describes the properties of the task and build the model of the task. on the basic of the difference on the properties of task scheduling problems, we divided the task scheduling problems into five parts (pre-emptive strict periodic task scheduling, non-pre-emptive strict periodic task scheduling, pre-emptive non-strict periodic task scheduling, non-pre-emptive non-strict periodic task scheduling, and non-periodic task scheduling), and elaborate the research of those scheduling problem in details. In future, we will design the scheduling algorithms to solve those scheduling problems.

References

1. C. L. Liu, J. W. Layland, Scheduling algorithms for multiprogramming in a hard-real-time environment, J. Comp 22 (2002). doi:https://doi.org/10.1007/s001800200116.
2. M. N. Huang, Real-time Scheduling Algorithm and Experimental Research of Industrial Ethernet Cycle Information, Dalian, China, 2008.
3. J. Lehoczky, L. Sha, Y. Ding, Rate monotonic scheduling algorithm: exact characterization and average case behavior, in: Proceedings of the real-time systems symposium, Santa Monica, USA, 1989, pp. 166–171. doi:https://doi.org/10.1109/REAL.1989.63567.
4. Y. T. Leung, J. Whitehead, On the complexity of fixed-priority scheduling of periodic, real-time tasks, Performance Evaluation 2 (1982) 237-250. doi:https://doi.org/10.1016/0166-5316(82)90024-4.
5. S. Lauzac, R. Melhem, D. Mosse, An efficient RMS admission control algorithm and its application to multiprocessor scheduling, in: Proceedings of the first merged international and symposium on parallel and distributed, Florida, USA, 1998, pp. 511–518. doi:https://doi.org/10.1109/IPPS.1998.669964.
6. D. Muller, Accelerated simply periodic task sets for RM scheduling, in: Proceedings of embedded real-time software and system, Toulouse, France, 2010.
7. E. G. Coffman, Introduction to deterministic scheduling theory, in: Proceedings of computer and job-shop scheduling theory, Wiley, NY, 1976, pp. 1–50.
8. Y. X. Song, Design of time deterministic network scheduling algorithm, Xian, China, 2020.
9. J. Korst, E. Aarts, J. K. Lenstra, J. Wessels, Periodic multiprocessor scheduling, volume 505 of Data Mining Case Studies, Berlin, Germany, 1990.
10. O. Kermia, Y. Sorel, Schedulability analysis for non-preemptive tasks under strict periodicity constraints, in: Proceedings of 14^{th} IEEE International conference on embedded and real-time computing systems and applications, Kaohsiung, Taiwan, 2008, pp. 25–32. doi:https://doi.org/10.1109/RTCSA.2008.44.

11. J. Chen, C. Du, F. Xie, Schedulability analysis of non-preemptive strictly periodic tasks in multicore real-time systems, J. RT. Sys 52 (2016).
12. J. Chen, C. Du, F. Xie, Allocation and scheduling of strictly periodic tasks in multicore real-time systems, in: Proceedings of IEEE International conference on embedded and real-time computing systems and applications, Daegu, Korea, 2016, pp.130–138. doi:https://doi.org/10.1109/RCTSA.2016.38.
13. J. Chen, C. Du, P. Han, Scheduling independent partitions in integrated modular avionics systems, J. Plos. One 11 (2016).
14. M. Fan, Q. S. Han, S. Liu, S. L. Ren, Enhanced fixed-priority real-time scheduling on multi-core platforms by exploiting task period relationship, J. Systems and Software 99 (2014). doi:https://doi.org/10.1016/j.jss2014.09.010.
15. S. K. Baruah, The non-preemptive scheduling of periodic tasks upon multiprocessors, J. Real-Time Systems 32 (2006).
16. Z. X. Li, A special scheduling method for non-preemptive periodic tasks, J. Comp. Eng. App 54 (2018).
17. M. Pinedo, Scheduling: theory, algorithms and systems, Beijing-THU, China, 2005.
18. K. Jeffay, D. F. Stanat, C. U. Martel, On non-preemptive scheduling of period and sporadic tasks, in: Proceedings of Real-Time Systems Symposium, Chicago, USA, 1991, pp. 129–139. doi:https://doi.org/10.1109/REAL.1991.160366.
19. T. P. Baker, An analysis of fixed priority schedulability on a multiprocessor, J. RTS 32 (2006). doi:https://doi.org/10.1007/S11241-005-4686-1.
20. T. P. Baker, Multiprocessor EDF and deadline monotonic schedulability analysis, in: Proceedings of 24th IEEE International Real-Time Systems Symposium, Cancun, Mexico, 2003, pp. 120–129.
21. M. Bertogna, M. Cirinei, G. Linpari, Schedulability analysis of global scheduling algorithms on multiprocessor platforms, J. IEEE. Trans. PDS 20 (2008). doi:https://doi.org/10.1109/TPDS.2008.129.
22. S. K. Dhall, C. L. Liu, On a real-time scheduling problem, J. Oper. Res 26 (1978). doi:https://doi.org/10.1287/opre.26.1.127.
23. L. Y. Yang, X. W. Lu, Approximation algorithms for some position-dependent scheduling problems, J. Else. BV 3 (2021).