Scheduling Algorithm comparison between ACO and EDF-RM

Zhao Hongyu

A0224411X

+65 89420731

Department of Electrical Engineering

Scheduling Algorithm comparison between ACO and EDF-RM

Abstract

This report selects two well-known scheduling algorithms-ACO and joint EDF-RM, and compares and evaluates the performance of the two algorithms running on actual data. In experiment the decision-making time of ACO is relatively long, especially when the number of tasks is much larger than the number of CPUs; As for EDF-RM algorithm, the upper bound of RMS in the paper completely ignores the number of CPUs. The comparative experiment based on the same data set in this report shows that ECU XXX, SR XXX, and FR XXX. From the results above, the ACO algorithm is better than EDF-RM in XXX.

1 Introduction

Real Time System always executes tasks based on some time restrictions(e.g deadline inter-arrival period or tardiness). In a Hard RTS, missing deadline will cause serious outcomes; In a Soft RTS, missing part of deadlines will not affect the whole system. As for a firm RTS, completing tasks on time will obtain more rewards. Every scheduling are suitable for some kind of different conditions. EDF is based on deadline, it is not working well in overload condition. In contrast, RMS is not functioning well in under-load. And ACO-based algorithm is also related to deadline and it works well in under-load or slightly overload condition.

Recently, as the development of multiprocessor and Distributed system, load balancing among all the professors increase the whole performance with the two crucial methodologies-Task migration and duplication. Therefore, achieving a good task migration and duplication method result in the huge increment of the performance and efficiency.

2 Input&Parameters

Because both of the paper do not provide data, I randomly generate data according to the paper's definition. The task data structure and parameters of generating data details are shown in 2.1. Besides, there are some algorithms' internal parameters with default value, which have an important influence on algorithm performance, are also shown in 2.2.

2.1 How to generate data

Both of the algorithms share the very similar task structure, so I use abstract class in Java to represent the Task template, and each of algorithm extends the template with slightly additional parameters to better adapt their algorithm environment respectively.

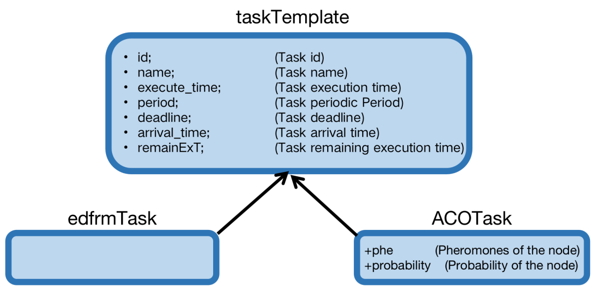

taskTemplate
id; (Task id)
name; (Task name)
execute_time; (Task execution time)
period; (Task periodic Period)
deadline; (Task deadline)
arrival_time; (Task arrival time)
remainExT; (Task remaining execution time)
edfrmTask
ACOTask
+phe (Pheromones of the node)
+probability (Probability of the node)

Figure 1: Task template

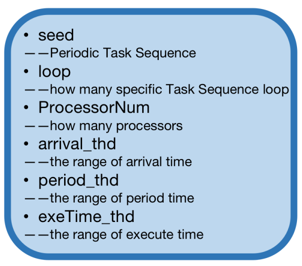

seed
——Periodic Task Sequence
loop
——how many specific Task Sequence loop
ProcessorNum
——how many processors
arrival_thd
——the range of arrival time
period_thd
——the range of period time
exeTime_thd
——the range of execute time

Figure 2: Generating data parameter

Task parameters are shown in Fig.1, according to the paper, we initialize deadline equals to period and remainEt equals to execute_time at the very beginning. And in order to randomly generate data, we need some additional parameters to definite the timing and the range threshold of every element in task template as shown in Fig.2 You can change the parameters in Fig.2 to generate different task set.

There are several task sequence and each of them will run several loops according to papers. In the experiments, if the task sequence number get bigger, accordingly both of the algorithm works not well because tasks have closed arrival time and there might be more crowed in the waiting queue. In contrast, if the loop number get bigger, both of two algorithms will work well because it won't be too crowed on any of processor as shown in Fig.3.

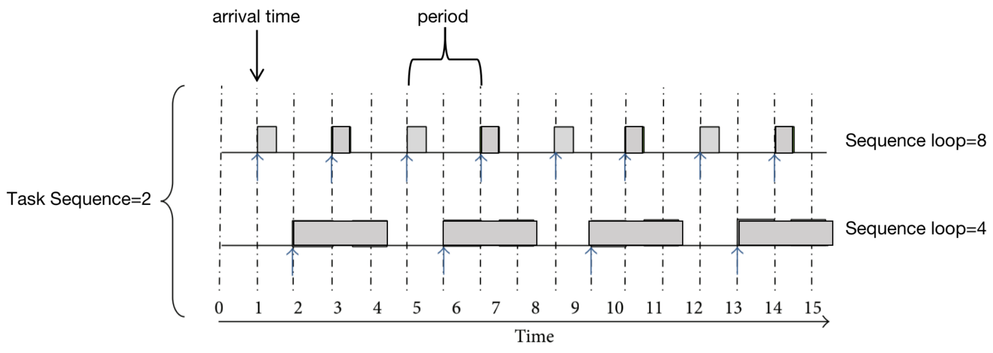


Figure 3: Task flow

2.2 How to change algorithm

There are also some other parameters, which will have an influence on algorithm performance if changed. And all of them have been given a default value in the papers. We can change these parameters to check model performance if needed.

- ρ , is a constant to evaporate pheromone, default value is 0.3
- K , is a constant to calculate probability, default value is 10
- α , is a constant to calculate probability, default value is 1
- β , is a constant to calculate probability, default value is 2
- C , is a constant to update pheromone, default value is 0.1
- B , is the upperbound task migration in EDF, default value is 0.81

3 Algorithm

3.1 ACO-based scheduling

3.1.1 Prepare/Modify job sequence

In each loop in the experiments, we need to calculate each task's probability based on their deadline, current time and pheromone. And sort task nodes according to their probability.

$$p_i(t) = \frac{(\tau_i(t))^\alpha * (\eta_i(t))^\beta}{\sum_{i \in R_1} (\tau_i(t))^\alpha * (\eta_i(t))^\beta} \tag{1}$$

η_i can be determined by

$$\eta_i = \frac{K}{D_i - t} \tag{2}$$

where t us current time, K is a constant, D_i is absolute deadline of ith node.

3.1.2 Choose the best node path

The number of ants should be the same with current number of tasks, so each ant could runs a completely different path. In each path, the ant need to count success tasks numbers or failure tasks numbers. After that, there will be only one ant walking the best path with the most success tasks number.

3.1.3 Update tasks' probability

In this part, all the nodes need to evaporate their pheromone to some extent, and laying reward for the best path's nodes step by step according to the sequence.

$$\tau_i = (1 - \rho) * \tau_i \tag{3}$$

τ means pheromone, ρ is attenuation coefficient.

$$\tau_i = \tau_i + \Delta\tau_i \tag{4}$$

where i $\in$ N1, N1 is the set of nodes at time t.

$$\Delta\tau_i = \frac{ph}{s} \tag{5}$$

$$ph = C * \frac{Number\ of\ Succeed\ Tasks}{Number\ of\ Missed\ Tasks} + 1 \tag{6}$$

3.1.4 Assign task to each processor

After update the best journal, sort all tasks by probability and execute all tasks in the sequence. Every time when a processor runs a task, it will record whether the task has down or dead(exceed deadline). In the experiments, if there is new tasks available or existing job has finished, it will restart all the procedure in a loop manner.

3.2 Joint EDF-RM scheduling

3.2.1 GlobalScheduler

At every time moment when their is new tasks available, it need calculate UB to decide upperbound of global queue in order to decide whether to drop the task or not.

$$UB = n * (2^{1/n} - 1) \tag{7}$$

n is the number of tasks, and UB is the upperbound.

$$tasku = \frac{execute\_time}{period} \tag{8}$$

3.2.2 Pselection & Taskmigration

The global scheduler randomly select one processor. If the processor is overload, it will choose one with the least utilization factor to run this task.

3.2.3 PQueue & Execution

Sort all tasks based on their deadline and each processor runs a task.

4 Simulation

4.1 Metrics

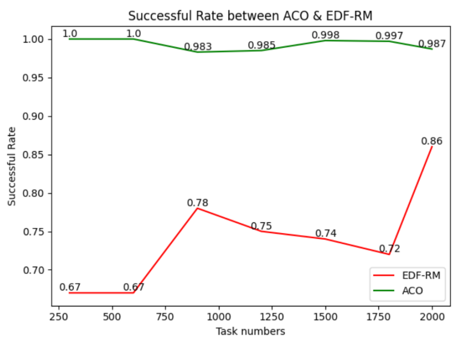
- SR is Successful Rate, which means Successfully scheduled tasks compared with Total number of tasks

$$SR = \frac{Successfully\ scheduled\ tasks}{Total\ number\ of\ tasks\ arrival} \tag{9}$$

- ECU is average cpu utilization, which means the average CPU utilization among all the processors.

$$EUC = \frac{Successfully\ scheduled\ tasks'\ computation\ time}{Total\ time\ of\ scheduling} \tag{10}$$

- Time consuming, means how much time the scheduler runs all the task, which is not the metrics from papers. But this metrics can help show the performance of the algorithms.



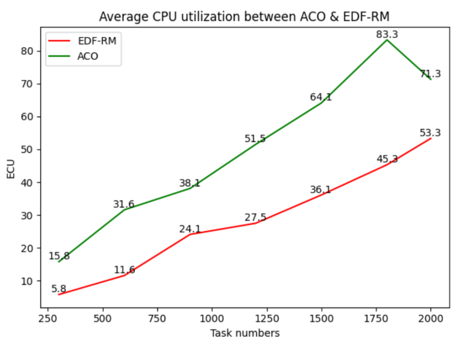


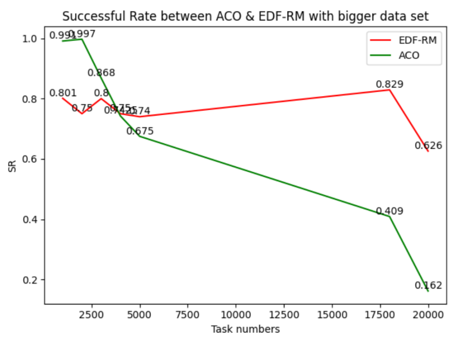
Figure 4: SR with smaller task set

Figure 5: ECU with smaller task set

4.2 Testing with smaller Task set

As shown in Fig.4 and Fig.5, with smaller task set suggested from papers(note: papers not provide data), we can see most of the time the ACO-based scheduling is better than joint EDF-RM in SR and ECU. And the test result is pretty similar with paper. As a result, with smaller task set, ACO-based scheduling algorithm works better than joint EDF-RM.

4.3 Testing with more Task set



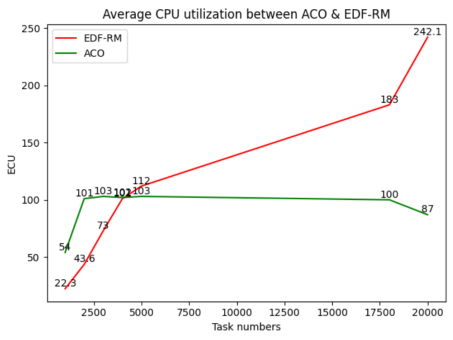


Figure 6: SR with more task set

Figure 7: ECU with more task set

However, when using more task set we can see that the performance of ACO-based scheduling drop quickly as the task set become larger, although the performance of joint EDF-RM is also slightly drop down, as shown in Fig.6 and Fig.7. Besides, the ECU of ACO has not changed or even decrease as task set become larger. In contrast, the ECU of joint EDF-RM increase. As a result, with more task set, joint EDF-RM works better than ACO-based algorithm.

5 Improvement

5.1 Improvement of joint EDF-RM

In the Eqa.7, we notice that the upper bound of global schedule do not consider the number of CPU, which means it does not make full use of CPU cores, wasting a lot computation power. Therefore it could be changed by times CPU cores.

$$UB = n * (2^{1/n} - 1) * N \tag{11}$$

N means the number of CPU cores. Besides, it is better to increase the upper bound migration threshold so that the improvement could be larger(equals to 1.0 in the experiment). As shown in Fig.8

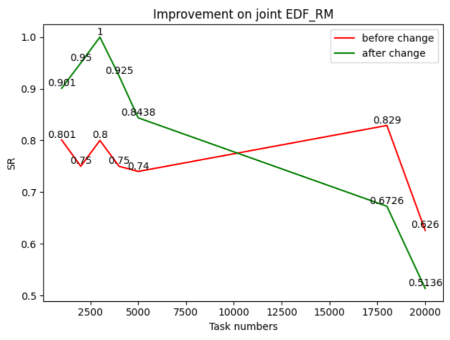


Figure 8: SR Improvement of joint EDF-RM

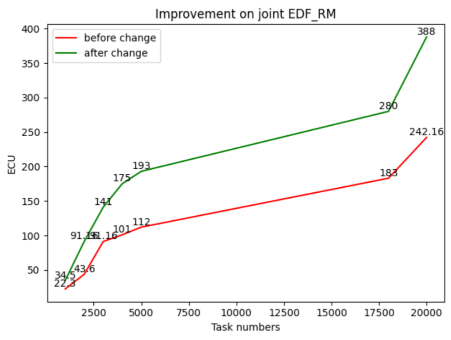


Figure 9: ECU Improvement of joint EDF-RM

5.2 Improvement of ACO-based scheduling

In the ACO-based scheduling, the biggest problem is that it is very time consuming compared to other algorithm, although it has a better performance, shown in Fig.10. In the paper, it said every turn, all the ants must start from all the different task nodes to start to search the best path, which wasting much of time and most of the searching is useless. We could lower the searching time by using the probability obtained from ahead step Eqa.1. We could just select several top probability nodes and start to run. Becase in the experiment, the best path is most likely coming from the top several probability nodes. And this method will not hurt the SR of ACO-based scheduling, as shown in Fig.11 and Fig.12

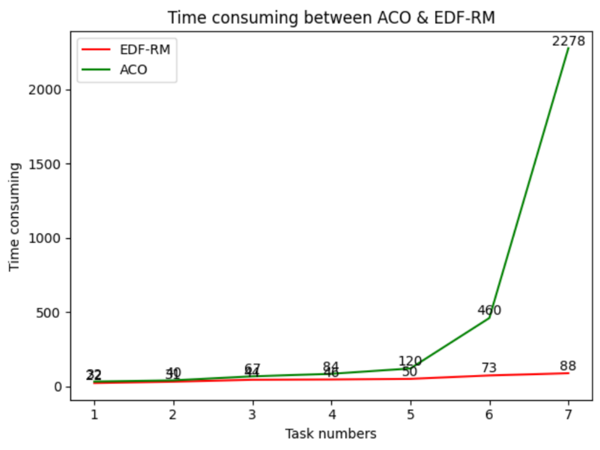


Figure 10: Time comsuming between ACO EDF-RM

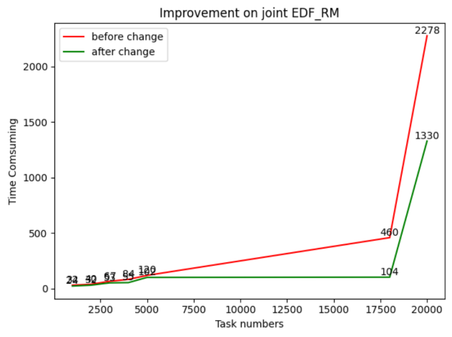


Figure 11: Time Consuming Improvement of ACO

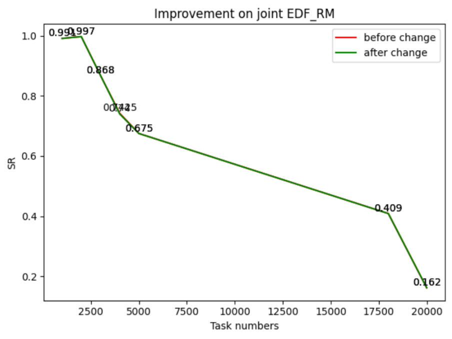


Figure 12: SR of ACO

6 Conclusions reference(s) used

In this report, I have discussed details of the eperiments of both the ACO-based scheduling [1]. and EDF-RM. [2]. Although the author has come up with two good ideas and has show some experiments data. Because the task date is not fixed, the result is also not fixed. And there are some big problems in both of the papers. And after do some improvement, both of the model has achieved a better performance.

References

[1] Shah, Apurva and Ketan Kotecha. ”ACO Based Dynamic Scheduling Algorithm for Real-Time Multiprocessor Systems.” IJGHPC 3.3 (2011):

20-30. Web. 27 Feb. 2021.

[2] Rashmi Sharma, Nitin, "Performance Evaluation of New Joint EDF-RM Scheduling Algorithm for Real Time Distributed System", Journal of Engineering, vol. 2014, Article ID 485361, 13 pages, 2014.

Code effort: all the code are written by myself so there is no github or any other reference code.

Shah, Apurva and Ketan Kotecha. ”ACO Based Dynamic Scheduling Algorithm for Real-Time Multiprocessor Systems.” IJGHPC 3.3 (2011): 20-30. Web. 27 Feb. 2021.

Rashmi Sharma, Nitin, ”Performance Evaluation of New Joint EDF-RM Scheduling Algorithm for Real Time Distributed System”, Journal of Engineering, vol. 2014, Article ID 485361, 13 pages, 2014.