Hybrid Real-Time Task Scheduling Algorithm in Overload Situation for Multiprocessor System

A. Hatami¹,², S. Chuprat², H. Md Sarkan² and N. Firdaus Mohd Azmi²
¹Faculty of Computing, University of Teknologi Malaysia, 81310 Johor Bahru, Johor, Malaysia.
²Advanced Informatics School, Universiti Teknologi Malaysia, Kuala Lumpur, Malaysia.
Amir.hatami.h@live.com

Abstract—Real-time systems are reactive systems which should meet major constraints in scheduling tasks like time limitation and resources allocation for scheduling the task effectively when the system is overloaded condition. Failure of system in scheduling tasks when system is overloaded can result in catastrophic impacts. The goal of this research is to propose a task scheduling algorithm that able to perform better than traditional Earliest Deadline First (EDF) and minimize the overall completion time when the system is overloaded condition. The proposed scheduling algorithm is built based on three new improved scheduling algorithms namely: (1) Hybrid Particle Swarm Optimization (PSO) and Hybrid Invasive Weed Optimization (HPIO), (2) Enhanced Initial Swarm (EIS), and (3) Hybrid EDF, EIS and HPIO Optimization (HEDFPIO). The author proves that more successful tasks is scheduled by using HPIO in multiprocessor system in over loaded situation among PSO and ACO. The author uses EIS algorithm in order to improve local search in HPIO and have fair load balance among processors. Finally the author presents a new hybrid algorithm that combines HPIO, EIS and EDF which is called HEDFPIO. It is observed that we could achieve higher successful ratio in task scheduling and with shorter calculation time in overloaded situation.

Index Terms—Enhanced Initial Swarm; Hybrid; Invasive Weed Optimization; Particle Swarm Optimization Overload.

I. INTRODUCTION

A real-time scheduling system contains scheduler, clock and processor. Tasks are assigned to the processors and it will be executed in a specific time and specified deadline by the characteristic of the scheduling algorithm. There are many scheduling techniques and the interest is to find the most optimum algorithms. In this paper, it is presented the hybrid algorithms which uses best part of the selected optimal algorithms and finally analyses the performance of the newly introduced hybrid algorithms.

One of these optimal algorithms is Particle Swarm Optimization (PSO) which is based on swarm intelligence. This algorithm simulates the behavior of individual particle in a group to optimize the survival of species. One of the most advantages of PSO is its robustness in controlling parameters and its high computational efficiency [1].

Invasive Weed Optimization (IWO) is a stochastic algorithm that simulates the behavior of weeds. Also this algorithm is presented by Mehrabian and Lucas [2], IWO has shown successful results in many fields and solved many problems such as optimization and tuning of a robust controller [2].

In this paper, it is shows that HPIO achieves better results by increasing the number of successful scheduled task and decreasing calculation time. The author could achieve better result in comparison with other algorithms that will be reviewed in this paper such as Earliest Deadline First (EDF), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO) and Invasive Weed Optimization (IWO) in overloaded situation. Also for this study the author implements and tests all algorithms with both uniprocessor and multiprocessor system. The author uses a method to have fair load balance among the processor to have better CPU utilization and improve local search in PSO and IWO. As illustrated in graphs of experiment shows that new presented algorithms have better performance by increasing successful tasks with improved calculation time.

The author considers using homogenous processors to compare the performance of the algorithms with previous research. By using homogenous processors, rate of all the tasks will be same in identical processor. Also, there is no constraint on requested time since the tasks model is based on sporadic model.

II. RELATED WORKS

This research is categorized in three main parts. The first part author studies the current suggested solution by ACO, PSO and IWO in uniprocessor. In second part, the author checks the best fair load balance algorithm to combine the output with multiprocessor systems and enhance initial swarm to avoid HPIO getting trapped in local search. And in the final part, author checks the feasibility of improved task scheduling by Earliest Deadline First algorithm.

Shah and Kotecha [3, 5], and Shah et al. [4] have used ACO and EDF algorithm and introduced a Hybrid algorithm that performs very well in comparison with normal EDF algorithm. The suggested adaptive framework is using EDF algorithm in “under load” situation and when system is “overloaded”, it switches to the ACO algorithm for scheduling the tasks. “When a system is assigned to schedule an amount of task which is more than the available system resource can handle is called overloaded situation”.

Therefore, execution of tasks will depend on the pheromone value laid on each scheduled task and heuristic function. The Adaptive ACO framework schedules tasks in lesser execution time when compared to normal ACO and EDF in overload situation. The weakness of adaptive framework is observed when the number of the tasks is increased ACO algorithm, adaptive framework requires more time to calculate which does not make good candidate for real-time task scheduling systems.

Karimi [6] used particle swarm optimization for task scheduling in Grid computing. PSO is considered as a
population based stochastic optimization method that simulates the behaviors of bird flocking [7]. In this method, the best result will be calculated based on “follow the bird which is nearest to the food”.

Karimi [6] used PSO to have better result in task scheduling by repeating same method until better result is found. In PSO model all the possible scenario which is helping to solve the problem is considered as a bird or particle. Each particle has a fitness value which is calculated by fitness function. In PSO model it is needed to define a problem space and all particle fly through the problem space. Each particle has a velocity which will be recalculated in iteration. Velocity will be calculated to follow the current optimum particle.

PSO algorithm has set of random particles that are created and then an optimal particle will be selected in each iteration. Two parameters play an important role in PSO algorithm which is called pBest and gBest. pBest or Personal best is considered as best fitness which has achieved and gBest or neighborhood best position which is tracked by the particle swarm optimizer, pBest and gBest are those which are obtained so far by any particle in the population [8].

Karimi’s [6] project design is in grid computing, the challenge for the author was assigning tasks to the resource and the problem arises when the system is overloaded. Therefore, Maryam used PSO algorithm to reduce execution time and utilize maximum resource. PSO is performing fast enough; she was looking for an algorithm to have fair destitute in Grid system. She used few algorithms which were benchmarked in many researches. These algorithms are Opportunistic Load Balancing (OLB), Min-min, Max-min and Discrete Particle Swarm Optimization (DPSO).

The outcome of the mentioned research was HDPSO which was combination of Min-Min and DPSO which could achieve better results when compared to other algorithms like OLB, Max-min and DPSO. Max-min heuristic is efficient only when most of the jobs arriving to the grid system are shortest [8].

Ghalenoei et al. [9] introduced a novel swarm base optimization algorithm which is inspired from Invasive Weed Optimization. (IWO) to do task scheduling of unmanned aerial vehicles (UAVs). The authors compared the result of IWO with Genetic Algorithms (GA) that is based on the simulation of result. In this experiment, IWO obtains better performance in comparison with GA.

According to Mehrabain and Lucas [2], IWO has three main parts. These parts are initialization, reproduction and spatial dispersal. In the first step, sample population is created based on initial seeds randomly. In second part, each individual seed is growing and it is allowed to reproduce new seeds and linearly depending on their own. In third part, the generated seeds are being randomly scattered with a normal distribution over the search space. The meaning of distribution is equal to the location of parent plant, but standard deviation (SD), $\sigma$, will be reduced from a specified initial value, $\sigma_{\text{initial}}$, to the final value, $\sigma_{\text{final}}$, according to Equation (1).

$$\sigma_{\text{iter}} = \frac{(\text{iter}_{\text{max}} - \text{iter})^n}{(\text{iter}_{\text{max}})^n} (\sigma_{\text{initial}} - \sigma_{\text{final}}) + \sigma_{\text{final}}$$  (1)

where $\sigma_{\text{iter}}$ is the standard deviation at the present step, and $\sigma_{\text{initial}}, \sigma_{\text{final}},$ iter $(_{\text{max}})$ (maximum number of iterations), and $n$ (modulation index) are other parameters. This nonlinear modification has shown satisfactory performance in many simulations [2]. This assumption means the seeds will be randomly distributed such that they lie close to the parent plant [10].

In next step, each weed allows to produce seeds and spreads them as mentioned in previous steps. Then all the seeds and their parents are ranked based on their fitness function. After that those seeds which are having lesser fitness are eliminated from the list. This method is based on “survival of the fittest” idea [11] (a common concept in evolutionary algorithms) gives a chance to plants with lower fitness to reproduce, and if their off springs have good fitness, they can survive in their offspring’s existence [2].

Finally, if maximum number of iteration has been reached then the result is considered as best fitness and nearest to optimal result.

Table 1: IWO parameters

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Quantity</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N_0$</td>
<td>Number of initial population</td>
<td>10</td>
</tr>
<tr>
<td>iter$_{\text{max}}$</td>
<td>Maximum number of iterations</td>
<td>400</td>
</tr>
<tr>
<td>dim</td>
<td>Problem dimension</td>
<td>18</td>
</tr>
<tr>
<td>$P_{\text{max}}$</td>
<td>Maximum number of plant</td>
<td>40</td>
</tr>
<tr>
<td>$S_{\text{max}}$</td>
<td>Maximum number of seeds</td>
<td>3</td>
</tr>
<tr>
<td>$S_{\text{min}}$</td>
<td>Minimum number of seeds</td>
<td>1</td>
</tr>
<tr>
<td>$n$</td>
<td>Nonlinear modulation index</td>
<td>3</td>
</tr>
<tr>
<td>$\sigma_{\text{init}}$</td>
<td>Initial value of standard deviation</td>
<td>1</td>
</tr>
<tr>
<td>$\sigma_{\text{final}}$</td>
<td>Final value of standard deviation</td>
<td>0.008</td>
</tr>
</tbody>
</table>

Ghalenoei et al. [9] follows all the mentioned steps to design his framework but he used different spatial dispersal module. His module designed to random selection of solution from a neighboring hypercube in the discrete space of solutions around the plant with a normal distribution. The sample of his pseudo code is provided for your reference. Please refer to Table 1 for parameters which have been used and Figure 1 regarding IWO pseudo code.

1. Generate random population of $N_0$ individuals from the set of feasible solutions
2. $i := 1$
3. do
   a. Compute maximum and minimum fitness in the colony
   b. For each individual $w \in W$
      i. Compute the number of seeds for $w$, corresponding to its fitness
      ii. Randomly select the seeds from the feasible solutions around the parent plant ($w$) in a neighborhood with normal distribution
      iii. Add the generated seeds to the solution set, $W$
   c. If $|W| > N > P_{\text{max}}$
      i. Sort the population $N$ in descending order of their fitness
      ii. Truncate population of weeds with smaller fitness until $N = P_{\text{max}}$
   d. $i := i + 1$
4. Repeat 3 until the maximum number of iterations

Figure 1: Pseudo-code of HPIO

Ghalenoei et al. [9] compared his results with various algorithms such as ACO, PSO and IWO. But IWO could achieve better result.

III. HYBRID PSO WITH IWO (HPI)

As it was mentioned before, the objective of this research was to improve minimum time cost and the author plans to achieve it by combining PSO and IWO and use their strengths to introduce a new algorithm. PSO could schedule tasks very
fast and accurate [6, 12] and IWO could have better result in comparison with PSO [9] but IWO algorithm requires more time to reproduce and eliminate seeds with lower fitness. In this research the author proves that, by using new algorithm we can schedule more tasks when compared to PSO and take lesser time in comparison with IWO.

In our experiment, we create HPI algorithm based on Figure 3. The main difference in this algorithm with IWO is the way particle is created and the particles which are eliminated are with less fitness. We created more initial particle in the initial sample and then truncate them in each iteration. Therefore, search space has become bigger to find the best order of the task scheduling. This makes the algorithm work faster than the IWO as it is not required to generate sample particle again. Removing sample in each iteration is based on the truncate value.

Truncate process will be continuous until swarm size becomes greater than truncate value. By creating bigger initial population, we could achieve optimal result in comparison with previous results of PSO and IWO in shorter time as shown in Figure 2. Optimal result means we can have more successful tasks scheduled in comparison with other algorithms. In Figure 2, point “Z” in graph is consider as an optimal result.

Meanwhile, Figure 3 shows the pseudo code of HPIO. In Line 1 and 2, swarm sample will be created and store in swarm list then in line 3 and 4 fitness value of each swarm will be calculated. From line 5 to 16, pBest and gBest will be calculated in line 16, new voracity will be calculated and replace in system according to equation2 and after that in line 17 new locations will be calculated based on equation3 and apply in line 18. Then fitness is calculated accordingly in line 19. In line 20 to 22. The parameter used in HPIO algorithm is mentioned in Table 2 which is based on the trial and error experiment and best parameter selected for Table 2.

$$\text{vel}_{t+1} = \omega \text{vel}_t + c_1 R_1 \ast (p\text{Best}_t - x_{i,t}) + c_2 R_2 \ast (g\text{Best}_t - x_{i,t}) \quad (2)$$

$$\text{Loc}_{t+1} = \text{Loc}_{t} + \text{Loc}_{t+1} \quad (3)$$

$$\omega = \omega_{max} - \frac{\omega_{max} - \omega_{min} \times t}{T} \quad (4)$$

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Swarm Size</td>
<td>60</td>
</tr>
<tr>
<td>Number of initial population</td>
<td>same as number of tasks</td>
</tr>
<tr>
<td>Maximum Number of iteration</td>
<td>10</td>
</tr>
<tr>
<td>Problem Dimension</td>
<td>2</td>
</tr>
<tr>
<td>Self-recognition coefficient (C1)</td>
<td>2</td>
</tr>
<tr>
<td>Social coefficient (C2)</td>
<td>2</td>
</tr>
<tr>
<td>W_UPPERBOUND</td>
<td>1.0</td>
</tr>
<tr>
<td>W_LOWERBOUND</td>
<td>0.0</td>
</tr>
<tr>
<td>Truncate population</td>
<td>2</td>
</tr>
</tbody>
</table>

Figure 3: Pseudo code of HPIO

Below pseudo code illustrates the EIS algorithm that we used to have fair load balance in multiprocessor system and improve the local search in HPIO algorithm. The author expects to have better load balance after using EIS algorithm in the output result. As part of experiment we are using EIS in multiprocessor and compared the output with other algorithm such as ACO and PSO. Please refer to Figure 4 for more information.

EIS algorithms has been customized in order to receive list of the tasks as input and then based on number of processor rearrange them to have almost same task work load among all the processors. The EIS works better by queuing shorter execution time of tasks for scheduling and by having lower complexity it enhances the initial particles and improve the result.
In equation 5, “U” shows the total utilization of the task sets and $C_i$ represents execution time of task $\tau_i$, and $T_i$ will be period of task $\tau_i$. Those task sets have a condition to be scheduled by the EDF and will be sent to EDF algorithm and remaining task will proceed to schedule by HPIO.

$$U = \sum_{i=1}^{n} \frac{C_i}{T_i}$$

VI. RESULT AND DISCUSSION

A. Compare ACO and HPIO in uniprocessor

The author performs task generation for testing purpose according to previous studies. He selects a set of random tasks which contains 7, 14, 20, 25, 35, 50, 75, 100, 150, 200, 300, 350, 500, 750, 1000, 1500 and 2000 task sets to create similar condition for previous algorithms and proposed algorithm.

In part A, the author performs task scheduling for all the task sets and based on the Figure 6 it shows number of successful task in uniprocessor by using HPIO, PSO and ACO. As it can be observed in most of the cases HPIO could schedule more tasks in comparison with ACO and PSO algorithms. Based on Figure 6, in task set 150, 300 and 350 ACO could schedule more task than HPIO and PSO. Table 3, shows the data related to timing each algorithm requires to finish the calculation. ACO performs better that other algorithms but in terms of calculation time, ACO takes more time than PSO and HPIO which is not acceptable in real-time scheduling systems. HPIO could schedule 525 tasks while ACO scheduled 485 tasks and PSO scheduled 461 tasks. HPIO improved the result to 8% in comparison with ACO and 14% improvement in comparison with PSO.

B. Enhanced HPIO by EIS Algorithm in Multiprocessor

Figure 7 shows total successful tasks using three algorithms.
of HPIO-EIS and PSO-EIS and ACO-EIS in uniprocessor, dual, triple and quad processor. As illustrated HPIO could schedule more tasks in comparison with PSO and ACO algorithm in all processors. In this experiment for uniprocessor we did not use EIS algorithm but for dual, triple and quad processor EIS is combined with HPIO, PSO and ACO. Based on result it is observed that by increase in number of processors and using EIS we could achieve more successful tasks. Total number of tasks as input is 7076. Based on Figure 7, HPIO-EIS improved the result by almost 6% when compared to PSO-EIS and ACO-EIS in dual processors. HPIO-EIS improve the result by 12% when compared to PSO-EIS and HPIO-EIS improved by 10% when compared to ACO in triple processor. This improvement for HPIO-EIS in quad core processor is 4% when compared to PSO-EIS and 7% when compared to ACO-EIS.

Figure 7: Total successful task by using EIS in different processor

Figure 8 shows details related to successful ratio in dual processor. Figure 9 and Figure 10 shows the data related to triple processor and quad processor. As it is shown in Figure 7, HPIO performs better when compared to other algorithms; the same expected result is presented in Figure 8, 9 and 10 also. This enhancement in result is due to usage EIS to arrange the inputs of the processor and effect on algorithms and have a divergent result in the local search.

Figure 8: Successful ratio in dual processor

VII. HYBRID EDF, EIS AND HPIO

In this part, we explain the result related to Hybrid EDF, EIS and HPIO. Figure 11 shows how the algorithm switch between EDF and HPIO when load is increasing. As it can be observed until task 35 the system could handle all the tasks by using EDF algorithm which is performed so fast but after that when load is increased to system then it goes to overloaded situation and from EDF slowly it switches to the HPIO algorithm.

Figure 9: Successful ratio in triple processor

Figure 10: Successful ratio in quad processor

Figure 9 shows the successful ratio of scheduled tasks in triple core processor. In this test also HPIO with EIS could achieve higher result in comparison with PSO and ACO algorithm. As you might observe ACO perform well if the tasksets are big but it is highlighted that ACO require long time to process the data in comparison with PSO or HPIO. Since time is an important factor in this research therefore ACO cannot be a good candidate.
In conclusion, as it is observed HPIO can achieve better results in comparison with ACO and PSO for number of successful scheduled tasks. By using EIS algorithm with HPIO, the author could improve the initial population and therefore, better result achieved in multiprocessor. The author combined EDF algorithm with EIS and HPIO for improving the calculation time in multiprocessor. HEDFPIO could be performing better and faster in comparison with HPIO. The author conducts many research to achieve the result and all the algorithms to be implemented in C#.

ACKNOWLEDGMENT

This research was partly funded by Research University Grant of Universiti Teknologi Malaysia (Vote: 07462).

REFERENCES


