A Task Scheduling Algorithm Based On Hybrid PSO in Overloaded Situation

Ranjandeep Kaur Khera[1], Dr. V.K. Banga[2] and Dr. Neha Sethi [3]

[1] Assistant Professor, Khalsa College for Women, Amritsar, India.
[2] Principal, ACET, Amritsar, India.
[3] Assistant Professor, Hindu College, Amritsar, India.
[1]ranjan_khehra@yahoo.co.in,[2]vijaykumar.banga@gmail.com,[3]neha_tejpal@yahoo.com

Abstract: **Many recent studies have been carried to perform various tasks in real-time scheduling environment. However, several researchers are trying to address the heavily loaded situations in real-time environment systems using a swarming technique. Such studies are classified by the different parameters that play major role in real-time environment system. For example, it increases cost of the system, processor latency, number of tasks, optimum utilization of system, and so on. With the increase in the number of tasks in the set, processing time also increases. In this situation, processor latency is at peak as the number of tasks increases and system costs increase. So the above mentioned problem is handled by proposing a task scheduler that uses a PSO algorithm to remove the limitations of past studies in a heavily loaded situation. The proposed algorithm is recommended for preventive tasks in the single-processor in real-time environment systems. The experiments have shown that PSO perform better, as compared to other common planning algorithms like ACO and EDF. The PSO and Invasive Weed Optimization (IWO) are combined to propose a new technique called the HWO algorithm that performs much better than PSO even in overload situations.**

Keywords: **IWO, PSO, heavily loaded situation, real-time environment scheduling.**

I. INTRODUCTION

Embedded real-time environment systems have emerged today in various applications, like automatic electronics, telecommunication industry, aeronautical systems, and consumer based electronics. In above mentioned areas, tremendous technical growth has been noted. (Davis and Burns 2011) One major problem in the real-time environment system is scheduling jobs to achieve maximum work, as the CPU has a least idle time. However there are few more powerful algorithms in the single-processor real-time environment system that executes tasks using EDF and ACO. But, they do not optimize if the tasks are heavily loaded in a particular situation. This study attempted to introduce and execute a PSO task scheduling technique in a single-processor real-time environment system to minimize the scheduling time and missed tasks. The results of PSO are compared to ACO and EDF in a heavily loaded situation.

Particle Swarm Optimization (PSO) is a technique which optimizes a problem repetitively and tries to find out an optimal solution with respect to a particular quality measure. PSO find optimum solution by considering a set of solution candidates, which refer to as particles, and by moving these particles in the search space using position and velocity. Each particle's movement is affected by its most local position and is directed towards the best position searched by other particles. So, this has made the swarm the best solution (Short 2010).

Companies that work on real-time environment systems have motive of profit earning, So they want to fulfill the needs of their potential customers by facilitating them with powerful system. This technological revolution has let to rapid growth in complexity of software as well as processing requirements of the hardware. (Davis and Burns 2011).

In order to meet the requirement of increased performance of processor, silicon vendors are not fully focused on the miniaturization required to improve clock rates of processor. This technique leads to high power consumption and excessive power consumption issues.

II. RELATED WORK

The most important need of real-time environment system is efficient work scheduling. (Short 2010) "Task scheduling has always been a central problem in the embedded real-time system community. As the scheduling problem in general is NP hard, researchers are looking for skilled successors to solve the scheduling problem in polynomial time". (Andrei et al. 2010) "If these overheads CPU time reserved for task scheduling decisions are not managed carefully, then system performance can be negatively affected by real-time feedback and power consumption can be reduced". (Short 2010) The first is one of the important and familiar scheduling strategies, proven and known. "It is clear that EDF is optimal for a multivariable platform for many cases, such as non-preemptive synchronous tasks (i.e. all tasks have the same starting time and cannot be interrupted), and antecedent asynchronous tasks (i.e. the task may be interrupted and have same starting time)". The earliest time limit (EDF) algorithm is the one which is priority scheduling algorithm in which the deadline creates preference for particular jobs. The EDF algorithm with N independent and preemptive functions is responsible for generating an optimum schedule till the total density of the system remains less than one.

Although EDF is an optimal scheduling algorithm, this algorithm does not work well by increasing the workload queue. This implies that EDF cannot handle heavily loaded situations. The concept of Swarm Intelligence (SI) was presented in 1989 by Benny and Jing Wang and was based on artificial intelligence. The Swarm Intelligence system consists of agents or bird populations that interact with each other and their environment.

EDF scheduling algorithms is as follows:

- Total time of the task which includes releasing and execution should be less than the task time limit.
- The releasing time of tasks is the time taken to allocate it to the processor.
- Work does not suspend itself.
- Execution time limit.
- Functions are independent.
- Scheduling overhead is minimal.
- Work priorities are assigned as per their deadlines. Early deadlines mean higher priority real-time environment systems.
- The highest priority finished work is executed.

Jian-Bo. (2010) proposed the algorithms, which was the suite for embedded real-time environment systems. Performance of the system is measured by the unavailable ratio of the system. Jian-Bo also suggested an algorithm which performs better than the normal EDF algorithm.The suggested system-missing ratio is lower compared to other systems in the overload condition. The system analyzes inputs and defines priority between tasks, classifying important or critical tasks by less important tasks. According to the Xian-Bo output result, the algorithm suggested by increasing the system workload and overloading determined most tasks by different priority.

Based on EDF and Ant Colony Optimization an adaptive algorithm system has been proposed by Shah and Kotecha. The overload situation has been addressed by the hybrid algorithm by using EDF and ACO algorithms."This algorithm provides a balance between exploration and exploitation as well as robustness and simplicity of the individual drug". (Dorigo et al., 1999) (Ramos et al., 2002) There exists two types of multiprocessor systems which has been given names as, homogeneous and heterogeneous (Apurva and Ketan 2009). The ant colony optimization algorithm has been shown to perform better using the inherent parallelism function (Shah and Kotecha 2011). Under homogenous multiprocessor the optimization algorithm for Ant Colony performs well in overload situations on real-time systems. Though it performs well in overcrowding situation, it takes more time for exploration and exploitation than EDF.

Proposed algorithm takes advantage of the EDF to schedule, when the system is not congested has a higher priority than when the system is overloaded. He uses a centralized scheduler to switch the system to ACO, which notifies the deadline and execution time for each appointment. In this model, the authors assume that the system has no problem with resource discordances. The execution of tasks in the ACO algorithm depends on the pheromone value set for each scheduled task and heuristic function. (Apurva and Ketan, 2009).

The Pseudo-code for the ACO based scheduling algorithm is:

1. Producing different ant's for different tours and producing a performance sequence.
2. Analyze the performance sequence.
3. Pheromone update price.
4. Determine the probability of each task.
5. Determine which task should be executed according to the work probability.
6. Performed OMIT.
7. If the work is more than 2 then go to step-2.

As per the real-time system, to manage tasks meeting deadlines are the most vital scheduler assignment activity. Ketan and Apoorva has proved and shown the significant improvement in effective CPU utilization, if compared with previous studies. Success ratio is nothing but ration of number of tasks that were successful to the total number of tasks scheduled.

A. Particle Swarm Optimization Scheduling

Particle Swarm Optimization (PSO) is the most authentic optimization algorithm and is generally implemented for uni-processor heterogeneous and preemptive real-time systems. "Particle swarm optimization (PSO) is a herd-based intelligence algorithm (Rahmani et al. 2013; Karimi et al. 2013; Kiran et al. 2012) influenced by the social behavior of animals such that a flock of birds is finding a food. Source or a school of fish that protects them from a predator (Technology et al. 2010). A particle in PSO behaves similar to a bird flying through space, here it is given as a search problem. The position of each particle is related and has the effect of velocity. The position of each particle over time is directly proportional to its best position (pBest) and the position of the best particle in a problem space (gBest). These both mutually affect each other. For a particular problem each particle has a fitness value.

In PSO population is referred as the total number of particles in a problem space. Initial particles are generated randomly. Meanwhile, fitness function has been evaluated and store in particle generation. pBest is nothing but the the best result value for the fitness, which has been attained by the particle so far. Furthermore gBest is the best particle from the population, as per its fitness value. It is known fact that each particle has a velocity value in each generation that this position will update in each generation as below:

$$v_i^{\alpha+1} = \omega v_i^{\alpha} + c_1 R_1 \times (pBest_i - x_i^{\alpha}) + c_2 R_2 \times (gBest - x_i^{\alpha}) \quad (1)$$

$$x_i^{\omega+1} = x_i^{\omega} + x_i^{\omega+1} \quad (2)$$

$$\omega = \omega_{max} - \frac{\omega_{max} - \omega_{min}}{T} \times t \quad (3)$$

Where

$v_i^{\alpha+1}$	velocity of particle i at iteration α
$v_i^{\alpha+1}$	velocity of particle i at iteration $\alpha + 1$
ω	inertia weight
c_1, c_2	acceleration coefficients; j = 1, 2
R_1, R_2	random number between 0 and 1; i = 1, 2
x_i^{α}	current position of particle i at iteration α
$x_i^{\alpha+1}$	position of the particle i at iteration $\alpha + 1$
$pBest_i$	best position of particle i
$gBest$	position of best particle in a population

PSO pseudo code is as follows:

a) Random swarm of particle.
b) Calculation of the fitness value for each task.
c) If (current fitness < pBest)
   pBest.
   Else
   pBest = current fitness
   End
d) Set gBest.
e) Calculation of velocity of each task.
f) As per velocity priority index is updated.
g) Go to step 2 till all the task are scheduled or iteration number is exhausted.

Figure-1 describes in detail the PSO algorithm. It is same as PSO pseudo code. The algorithm starts by initializing particle. The authors have taken an assumption that number of problem dimension is equal to number of the tasks. Initially, fitness of each swarm will be calculated and set with pBest. Then it is observed that if current fitness is bigger than the pBest, the previous pBest shall be replaced by new fitness value. In case of the opposite algorithm finds pBest which is best swarm current position. Next step takes the calculation of gBest after finding pBest. The next step followed is to calculate and assign new velocity to particles of swarms. In the last and the final step system check that all the tasks have been scheduled or number of the iteration done. If none of the condition meets, algorithm will be go to second step which will calculate the fitness or else algorithm will be finished.

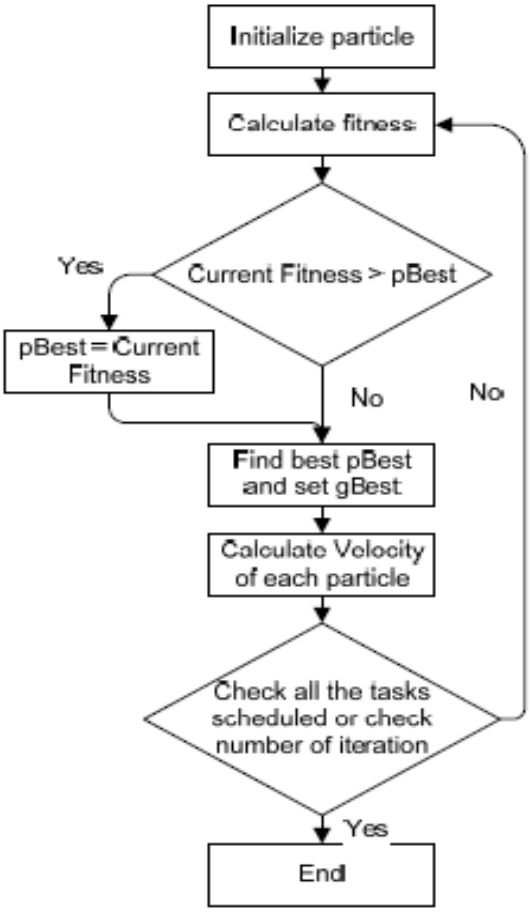

Figure-1. PSO algorithm

B. Invasive Weed Optimization

Introduced in 2006 by Mehrabian and Lucas Invasive Weed Optimization (IWO) is a bioinspired statistical optimization algorithm. This algorithm behaves as a weed to live in the colony and growth and reproduction. The particular way of reproduction, spatial dispersion, and competitive exclusion makes IWO unique among other evolutionary algorithms. (Hazimirsdegi and Lucas, 2009) This algorithm will be the first step to introduce population into IWO. This means that the initial population at the site of the problem arises irregularly. Then based on the fitness, initial populations starts getting increased. This will lead to IWO reproducing the best members to reduce the worse member with lower fitness value. In the next step the newly produced members normally spread to the search space by random numbers equal to zero and an adaptive standard deviation. The standard deviation of each generation is introduced in equation (4).

$$\sigma_{iter} = \frac{(iter_{max} - iter)^n}{(iter_{max})^n}(\sigma_{initial} - \sigma_{final}) + \sigma_{final} \quad (4)$$

In this formula, intermix is represented as the maximum number of iterations. Sigma item is representing the standard deviation in the current iteration. The "n" in this formula represents the nonlinear modulation index.

"Seeds produced with their parents are considered a possible solution for the next generation. Finally, a competitive exclusion is conducted in the algorithm, namely, that the population reaches its maximum after several iterations. For this, seeds and their parents are kept together and people with better fitness survive and become reproducible." (Hazimirsdegi and Lucas, 2009)

C. Hybrid PSO with IWO (HWO)

As illustrated in Figure 2, the authors combine the IWO algorithm with PSO to achieve better results in a shorter period of time. In this hybrid algorithm, most of the steps are similar to the PSO algorithm. Therefore, in the first phase, all particles are at initial stage. Then the fitness of all the particles in the swarm will be calculated. After calculating the fitness value, pBest will be calculated. Top fitness will be selected to expand. In this research the authors selected a number of top fitness swarms. After adding the velocity of new nodes to each node will be calculated based on equation 1. Fitness will be recalculated in the final phase and swarms who have low fitness will be removed from the swarms list. The number of swarms that will be removed from the list is equal to the number of swarms that were added to the list at the initial stage.

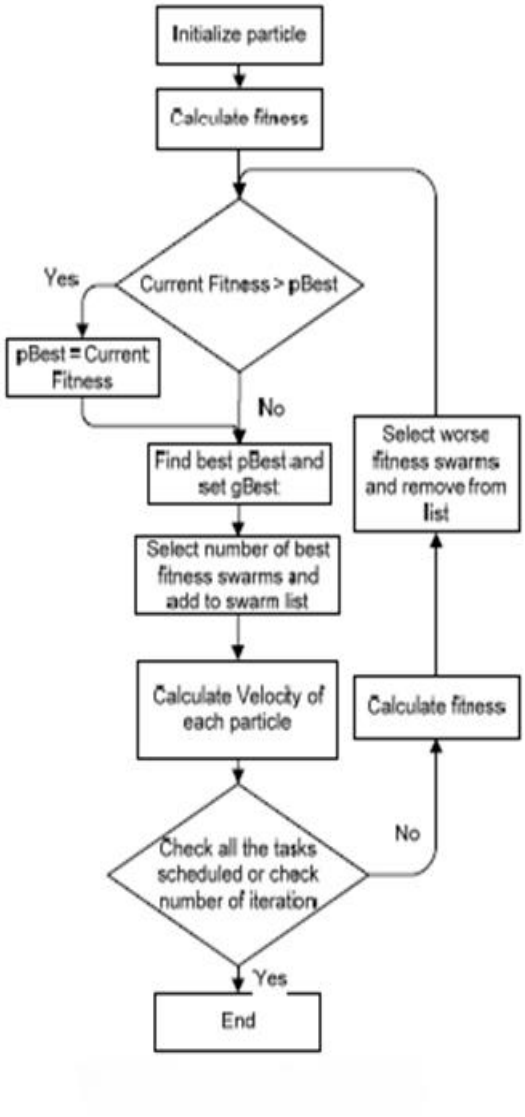

Figure-2. Hybrid PSO / IWO (HWO)

III. RESULT AND DISCUSSIONS

In this paper, the authors implement the algorithms in MATLAB 2013a tool and have been tested in DELL Notebook Computer with 8GB Ram and 2.4 GHz Intel Core i5 Processor.

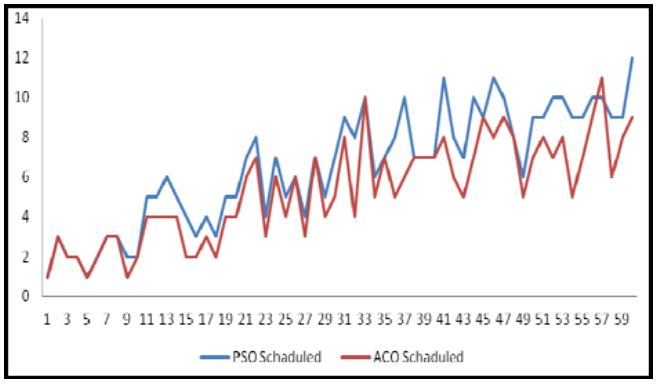

Figure-3. Comparison of scheduled tasks by ACO and PSO.

In this experiment, the swarm size was taken equal to 30 swarms. The maximum iteration will be 100 and the amplitude of the problem will be the same as the number of particles in the swarm. C1 and C2 are constant values assuming it to be 2. The value of upper bound W is 1.0 and that of lower bound is 0.0. The value of P is between 0.2 and 0.4 in the ACO algorithm. Parameter of alpha and beta is 1.0. The parameter K is chosen as 5 and the suggested range is 5 to 10.

The sample data has been tested for working sets of 7, 15, 25, 50, 70 and 100. As can be seen, when the number of tasks in a work set is not overloaded, most algorithms perform well and throughout tasks. The possible scenario is tested by the corresponding algorithm.

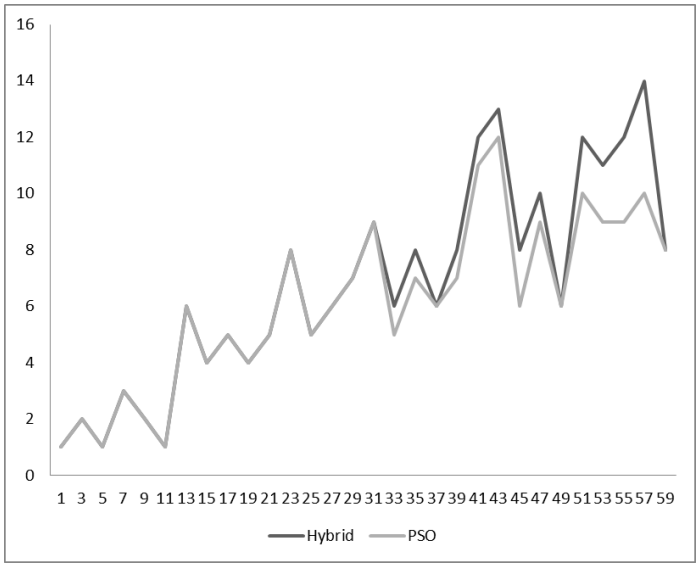

Figure-4. Comparison of scheduled tasks by PSO and Hybrid (PSO/IWO).

Figure 4 shows comparison between the PSO scheduling algorithm and the hybrid (PSO / IWO) algorithm. In the Figure-4, Task set ID is depicted on the horizontal line and number of successful scheduled tasks on the vertical line. It is assumed that each task set consists of ten tasks.

So it can be concluded PSO and Hybrid (PSO / IWO) produces the same results up to task id 29, but after that the hybrid algorithm out performs the PSO algorithm.

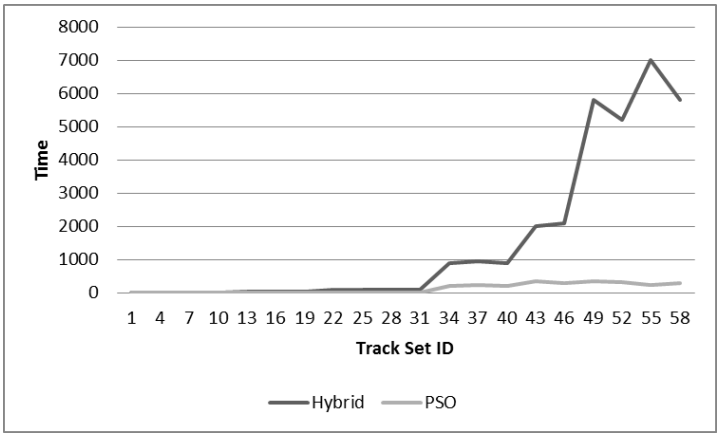

Figure-5. Calculation time for PSO and Hybrid algorithm.

Figure-5 depicts the total computation time that schedules a sample dataset with the use of PSO and hybrid scheduling algorithm. In the above graph the task set ID is represented using the horizontal line and the total computation time is depicted on the vertical line.

The hybrid computation time is slightly more than normal PSO due the use of invasive weed optimization to yield better results. It is clear from the analysis that the hybrid algorithm has been suggested to get better results in heavily loaded situations and is suitable real-time environment systems.

IV. CONCLUSIONS

Scheduling jobs are simple, but once we increase the number of jobs it becomes bit difficult to complete a particular schedule. It is evident from the comparative analysis that Hybrid PSO out performs the PSO scheduling algorithm in case of heavily loaded situations.

In future, the authors are working to find out a method to optimize the HWO algorithm to reduce computation time with the help of other optimization algorithms.

REFERENCES

[1]. Andrei S. et al. (2010), *Optimal Scheduling of Urgent Preemptive Tasks*, IEEE 16th International Conference on Embedded and Real-Time Computing Systems and Applications, pp: 377-386.

[2]. Anon. (1995), *HRT-HOODTM: A Structured Design Method for Hard Real-Time Ada Systems*, pp: 3.

[3]. Apurva S. and Ketan K. (2009), *Adaptive Scheduling Algorithm for Real-Time Multiprocessor Systems*, pp: 6-7.

[4]. Davis R.I. and Burns A. (2011), *A survey of hard real-time scheduling for multiprocessor systems*, ACM Computing Surveys, pp: 1-44.

[5]. Dorigo M., Caro G. Di and Gambardella L.M. (1999), *Ant Algorithms for Discrete Optimization*. pp: 1-36.

[6]. Hajimirsadeghi H. and Lucas C. (2009). *A Hybrid Iwo/Pso Algorithm For Fast And Global Optimization*, pp: 1964-1971.

[7]. Karimi M., Motameni H. and Branch S. (2013), *Tasks Scheduling in Computational Grid using a Hybrid Discrete Particle Swarm Optimization*, pp: 29-38.

[8]. Kıran M.S., Gündüz M. and Baykan Ö.K. (2012*). A novel hybrid algorithm based on particle swarm and ant colony optimization for finding the global minimum*, Applied Mathematics and Computation, pp: 1515-1521.

[9]. Rahmani R. et al. (2013), *Hybrid technique of ant colony and particle swarm optimization for short term wind energy forecasting*, Journal of Wind Engineering and Industrial Aerodynamics, pp: 163-170.

[10]. Ramos V., Muge F. and Pina P. (2002), *Self-Organized Data and Image Retrieval as a Consequence of InterDynamic Synergistic Relationships in Artificial Ant Colonies*, pp: 87.

[11]. Shah A. and Kotecha K. (2011), *ACO Based Dynamic Scheduling Algorithm for Real-Time Multiprocessor Systems, International Journal of Grid and High Performance Computing*, pp: 20-30.

[12]. Short M. (2010), *Improved Task Management Techniques for Enforcing EDF Scheduling on Recurring Tasks*, 16th IEEE Real-Time and Embedded Technology and Applications Symposium, pp: 56-65.

[13]. Technology I.Haghnazar R. and Rahmani A.M. (2010), *Prune PSO: A new task scheduling algorithm in multiprocessors systems*, pp: 161-165.

[14]. Xian-bo H. (2010). *An improved EDF scheduling algorithm based on fuzzy inference being suitable for embedded soft real-time systems in the uncertain environments*, 2nd International Conference on Advanced Computer Control, pp: 588-592.