VOL. 10, NO. 2, FEBRUARY 2015 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

www.arpnjournals.com

TASK SCHEDULING ALGORITHM BASED ON PARTICLE SWARM OPTIMIZATION (PSO) AND INVASIVE WEED OPTIMIZATION TO EXECUTE TASKS IN OVERLOADED SITUATION FOR PREEMPTIVE SYSTEM

Amir Hatami Hardoroudi[1] and Suriayati BT Chuprat[2]
[1]Faculty of Computing, University of Teknologi Malaysia (UTM), Johor, Malaysia
[2]Advanced Informatics School, Universiti Teknologi Malaysia (UTM), Kuala Lumpur, Malaysia
E-Mail: Amir.hatami@ieee.org,

ABSTRACT

So many studies have been done in order to execute all the tasks in real-time scheduler systems. However, different researcher are tried to tackle overload situation in real-time systems by using swarm algorithm. These studies have been categorized based on the various parameters which are important in real-time systems. As an instance, system cost, processor waiting time, number of tasks, balance use of system and etc. By increasing number of the task in task set, process time will be increased. In this situation, processor waiting time will be high when the number of the task increased and as result system cost is raising. To solve mentioned issue the authors proposed a task scheduler which is used PSO algorithm in order to cover deficiencies of previous studies in overloaded situation. This algorithm is suggested for preemptive tasks in uniprocessor in real-time systems. The result of the research has been shown PSO perform better while other common scheduling algorithm same as EDF and ACO are being over loaded. The authors by combine PSO and Invasive Weed Optimization (IWO) suggest a new algorithm that is called HPI algorithm which can perform better than PSO and schedule more tasks in overload situation.

Keywords: PSO. IWO, overloaded situation, real-time scheduling.

INTRODUCTION

Nowadays, real-time embedded systems are bring into being in many diverse application areas, as an instance automotive electronics, avionics, telecommunications, space systems, medical imaging, and consumer electronics. In all of these areas, there is rapid technological progress. (Davis and Burns 2011) An important issue in real time system is scheduling tasks in order to do most of the tasks as CPU has minimum idle time. There are some powerful algorithm in uniprocessor real time system that schedule tasks such as EDF and ACO But they are not optimal when task overloading happens in this situation ACO algorithm much better than EDF But it's scheduling time and missing task is not optimal. In this study has been effort to present and implement PSO scheduling algorithm in uniprocessor real time system to reduce scheduling time and missing tasks. Finally PSO scheduling results will be compared with EDF and ACO when task overloaded happens.

Particle swarm optimization (PSO) is a computational method that optimizes a problem by iteratively trying to improve a candidate solution with regard to a given measure of quality. PSO optimizes a problem by having a population of candidate solutions, here dubbed particles, and moving these particles around in the search-space according to simple mathematical formulae over the particle's position and velocity. Each particle's movement is influenced by its local best known position but, is also guided toward the best known positions in the search-space, which are updated as better positions are found by other particles. This is expected to move the swarm toward the best solutions (Short 2010).

Companies building embedded real-time systems are driven by a profit motive. To succeed, they aim to meet the needs and desires of their customers by providing systems that are more capable, more flexible, and more effective than those of their competitors, and by bringing these systems to market earlier. This desire for technological progress has resulted in a rapid increase in both software complexity and the processing demands placed on the underlying hardware. (Davis and Burns 2011).

To address demands for increasing processor performance, silicon vendors no longer concentrate wholly on the miniaturization needed to increase processor clock speeds, as this approach has led to problems with both high power consumption and excessive.

RELATED WORK

Introduction of swarm algorithm

Most crucial requirement for real-time and embedded system is effective tasks arrangement and scheduling. (Short 2010) "Tasks' scheduling has always been a central problem in the embedded real-time systems community. As in general the scheduling problem is NP-hard, researchers have been looking for efficient heuristics to solve the scheduling problem in polynomial time."(Andrei *et al*. 2010) "If these overheads - CPU time reserved for making task scheduling decisions - are not carefully managed, system performance can be negatively affected leading to reduced real-time response and increased power consumption. "(Short 2010) Earliest Deadline first is one of the famous and important scheduling strategies has been proved and known. "It is known that EDF is optimal for uniprocessor platforms for many cases, such as: non-preemptive synchronous tasks (i.e., all tasks have the same starting time and cannot be interrupted), and preemptive asynchronous tasks (i.e., the tasks may be interrupted and may have arbitrary starting time)." The Earliest Deadline First (EDF) algorithm is a dynamic priority-scheduling algorithm in which the absolute deadlines caused individual jobs priority. An EDF algorithm can generate a feasible schedule for a system of N independent, preemptive tasks as long as the total density of the system is less than one.

However EDF is an optimal scheduling algorithm but by increasing workload queue this algorithm not function well. It means that EDF cannot support overload situations. Swarm Intelligence (SI) is introduced by Beni and Jing Wang in 1989 which is employed in work on artificial intelligence. Based on SI, “system behavior in reaction to an alteration in the environment external to the system is neither random nor predictable from physical measurements of the environment”(Anon 1995). SI systems are typically made up of a population of simple agents or bird interacting locally with one another and with their environment.
In EDF scheduling algorithm:

- Total time of Task’s release and execution time must be less than task’s deadlines.
- Release time of tasks is the time that processors will be allocated to the task.
- Tasks do not suspend themselves
- Tasks have bounded execution time
- Tasks are independent
- Scheduling overhead negligible
- Tasks priorities are assigned according to their deadline; Shorter deadline means higher priority Real-Time Systems
- The ready task with the highest priority is executed.

One of the algorithms, which were suite for embedded real-time system, is proposed by Xian-bo. (Xian-bo 2010) One of the parameter for evaluating real-time systems performance is missing ratio of the system. Xian-bo suggested an algorithm that performs better in comparison with normal EDF algorithm. The suggested system-missing ratio is lesser than other systems in overload situation. This system by analysis the input and define a priority between tasks, categorized critical or important tasks from the less important tasks. According to Xian-bo output result, the suggested algorithm by increase and overloading the system workload most of the tasks from different priority have been scheduled.

Shah and Kotecha, which was using EDF and Ant Colony Optimization algorithm, have suggested an adaptive system. The hybrid algorithm is used EDF and ACO algorithm in order to overcome overload situation. “This algorithm is providing balance between exploration and exploitation along with robustness and simplicity of individual agent.” (Dorigo *et al*. 1999) (Ramos *et al*. 2002) Multi processor systems are in two category of homogeneous and heterogeneous (Apurva and Ketan 2009) and Ant colony optimization algorithm illustrated that it could perform well by using inherent parallelism feature. (Shah and Kotecha 2011) Ant Colony optimization algorithm is performed well in overload situation on homogenous multiprocessor real time system. Although ACO performing well for schedule the tasks in overload situation but it takes longer time for exploration and exploitation in comparison with EDF.

The suggested adaptive algorithm is using advantages of EDF algorithm to handle the scheduling when system is not overloaded and had higher priority than once the system overloaded it will switch the system to ACO by using a centralized scheduler which notify deadline and execute time of each and every tasks. In this model, the authors imagine system is not having any resource discord problem. In ACO algorithm task execution will depend on the pheromone value laid on each scheduled task and heuristic function. (Apurva and Ketan, 2009).

Pseudo-code of the ACO based scheduling algorithm is:

1. Construct different tour of different ant and produce task execution sequence
2. Analyze task execution sequence
3. Update value of pheromone
4. Decide probability of each task
5. According to task probability decide which task must be execute
6. Omit executed task
7. If remained tasks is more than 2 go to step-2

According to real-time systems, deadline meeting is the most important scheduler activity in order to manage the tasks. Ketan and Apurva Success Ratio and Effective CPU Utilization in the presented model were higher than previous studies. Success Ration is the number of tasks that were successfully scheduled divided by total number of task arrived and Effective CPU utilization is sum of value of task that scheduled successfully divided by total time of scheduling.

PARTICLE SWARM OPTIMIZATION SCHEDULING

As a powerful optimization algorithm, particle swarm optimization (PSO) is applied for uni-processor heterogonous and preemptive real-time system. "Particle Swarm Optimization (PSO) is a swarm-based intelligence algorithm (Rahmani *et al*. 2013; Karimi *et al*. 2013; Kıran *et al*. 2012) influenced by the social behavior of animals such as a flock of birds is finding a food source or a school of fish protecting them from a predator." (Technology *et al*. 2010) A particle in PSO is similar to a bird or fish flying through a search (problem) space. Each particle's location is affected by velocity which has both magnitude and direction. The position of each particle during the time is effect by its best position (pBest) and the position of the best particle in a problem space (gBest). Each particle has a fitness value that will be evaluated by problem specific.

In PSO, the population is the number of particles in a problem space. Initial particles are generated randomly. Meanwhile, fitness value has been evaluated by fitness function and store in particle generation. The pBest is equal to the best result according to fitness value that has been achieved by particle so far. Also gBest is the best particle among current population based on their fitness value. It was mentioned each particle has a velocity value in each generation that this position will be update as below in each generation:

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

$$x_i^{\alpha+1} = x_i^{\alpha} + x_i^{\alpha+1} \qquad (2)$$

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

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 for task scheduling:

```
a) Initial random swarm of particle (tasks) (In this article
   ACO output is used for initial swarm)
b) Calculate fitness value for each tasks or particle
c) If (current fitness< pBest) then
    pBest←current fitness
    Else, keep LAST PBEST
d) pBest ← Best pBest
e) Calculate velocity of each task
f) UPDATE tasks X or priority index according to
   velocity
g) Go to step 2 until finish irritation number
```

Figure-1 explains the PSO algorithm in details. As it can be observe it is same as PSO pseudo code. The algorithm is starting by initializing particle. The authors are assumed that problem dimension is equal to number of the tasks. After initialization, fitness of each swarm will be calculated and set with pBest. If current fitness bigger than the pBest, the previous pBest will be replaced by new fitness value. Otherwise algorithm finds pBest which is best swarm current position. gBest that is global best swarm position will be calculated in next step after finding pBest. After that it is time for calculate and assign new velocity to particles of swarms. In the next step system check that all the tasks have been scheduled or number of the iteration done. If none of the condition meet, algorithm will be goes to second step which is calculate the fitness or else algorithm will be finished.

Invasive weed optimization

Invasive weed optimization (IWO) has been introduced by Mehrabian and Lucas in 2006 which is a bioinspired statistical optimization algorithm. This algorithm somehow works same as weeds behavior to go and live in a colony and growth and reproduction. The way of reproduction, spatial dispersal, and competitive exclusion makes IWO became unique among other evolutionary algorithms. (Hajimirsadeghi and Lucas, 2009) In IWO initializing the population will be the first step this algorithm. It means that the initial population is randomly generated over the problem space. Then the population will be growing based on fitness of initial population. It means that IWO will reproduce best members in order to reduce worse member with lower fitness value. In the next step new produced members spread over the search space by normally distributed random numbers with mean equal to zero and an adaptive standard deviation. Standard deviation of each generation has been introduced in equation (4).

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

In this formula, iter max is represented maximum iteration number. Sigma iter is representing standard deviation at the current iteration. “n” represent nonlinear modulation index in this formula.

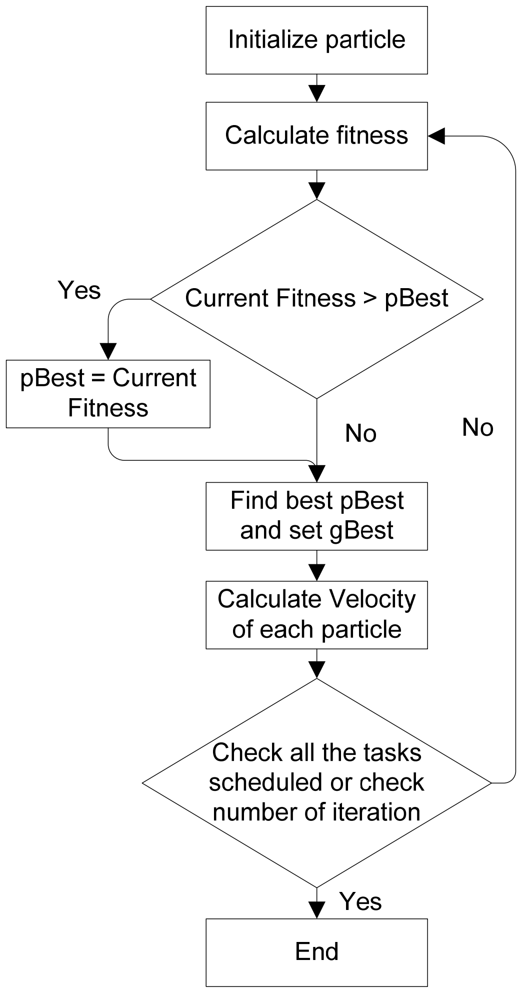
Initialize particle
Calculate fitness
Current Fitness > pBest
Yes
pBest = Current Fitness
No
No
Find best pBest and set gBest
Calculate Velocity of each particle
Check all the tasks scheduled or check number of iteration
Yes
End

Figure-1. PSO algorithm.

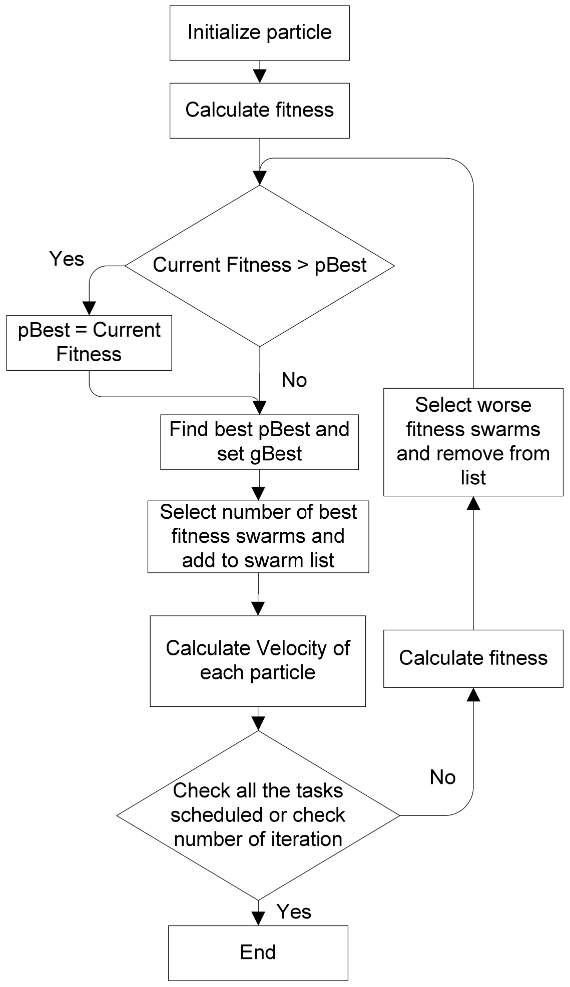
Initialize particle
Calculate fitness
Current Fitness > pBest
Yes
pBest = Current Fitness
No
Find best pBest and set gBest
Select number of best fitness swarms and add to swarm list
Calculate Velocity of each particle
Check all the tasks scheduled or check number of iteration
No
Calculate fitness
Select worse fitness swarms and remove from list
Yes
End

Figure-2. Hybrid PSO/IWO (HPI).

"The produced seeds, accompanied by their parents are considered as the potential solutions for the next generation. Finally, a competitive exclusion is conducted in the algorithm, i.e., after a number of iterations the population reaches its maximum, and an elimination mechanism should be employed. To this end, the seeds and their parents are ranked together and those with better fitness survive and become reproductive." (Hajimirsadeghi and Lucas, 2009)

Hybrid PSO with IWO (HPI)

As it is illustrated in Figure-2, the authors combined IWO algorithm with PSO in order to get better result in shorter time. In this hybrid algorithm, most of the steps are same as PSO algorithm. Therefore, in first step, all the particles are initialized. Then fitness of all the particles in swarms will be calculated. After calculating fitness value, p best and Gbest will be calculated. Top fitness will be selected for expanding. In this research the authors selected a quartet number of top fitness swarms. After adding new nodes velocity of each node will be calculated based on equation 1. In final stage fitness will be recalculated and those swarms which having less fitness will be remove from the swarm list. Number of swarm which will be remove from list is equal to the number of the swarm was added to the list in initial stage.

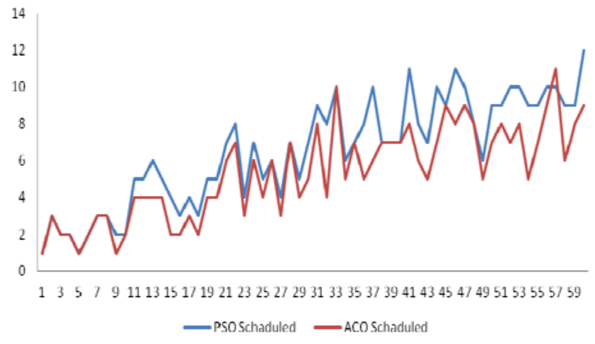

Figure-3. Comparison of scheduled tasks by ACO and PSO.

RESULT AND DISCUSSIONS

In this paper, the authors implement all algorithms in C# and have been tested in Intel Core i5 M580 2.67 GHz with 4GB installed memory (RAM). The processor speed allows testing higher number of particle in each swarm.

In this research, swarm size assumed as 30 swarm. Maximum iteration will be 100 and problem dimension will same as number of particle in swarm. C1 and C2 are constant value which assume as 2. W upper bound is 1.0 and w lower bound is 0.0.

In ACO algorithm value of P is between 0.2 and 0.4. Parameter of alpha and beta are 1.0. Parameter of K selected as 5 and the suggested range is 5 to 10.

As it is illustrated in Figure-3 the authors have been created sixty sample data for testing three algorithms of ACO, PSO. The horizontal line shows the task set ID and the vertical line shows number of the successful scheduled tasks. In this chart every task set is included ten tasks.

This sample data is tested for task set of 7, 15, 25, 50, 70 and 100. As it can be observed, when the number of tasks in the task sets are not overloaded, most of the algorithm are function well and all the possible scenario has been tested by respective algorithm.

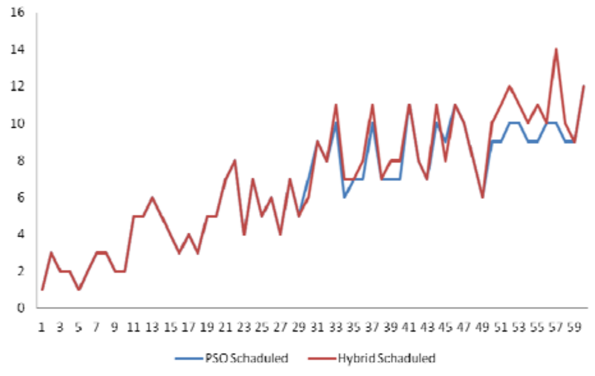

Figure-4. Comparison of scheduled tasks by PSO and Hybrid (PSO/IWO).

But by increase number of the tasks in task set it is achieve that PSO is able to test more possible scenario and achieve better result.

Based on the output, ACO and PSO algorithm are working almost similar for task set of seven but after that it proves that PSO achieve better result after task set 10 in comparison with result of ACO algorithm.

Figure-4 represents comparison of PSO scheduling algorithm with Hybrid (PSO/IWO) algorithm. As it is shown in Figure-4, the horizontal line represent task set ID which are as same as number of task set in Figure-3. The vertical line represent illustrate number of the successful scheduled tasks. It is assumed that each task set contains ten tasks. The sample data is tested for task set of 7, 15, 25, 50, 70 and 100.

As it can be observed, PSO and Hybrid (PSO/IWO) could achieve almost same result until task set ID 29 but after that Hybrid algorithm could achieve better result than in comparison with the PSO algorithm. Therefore, it is obvious that suggested Hybrid algorithm which is combination of PSO and IWO has better movements in each iteration by using good exploration and diversity that suggested with IWO algorithm.

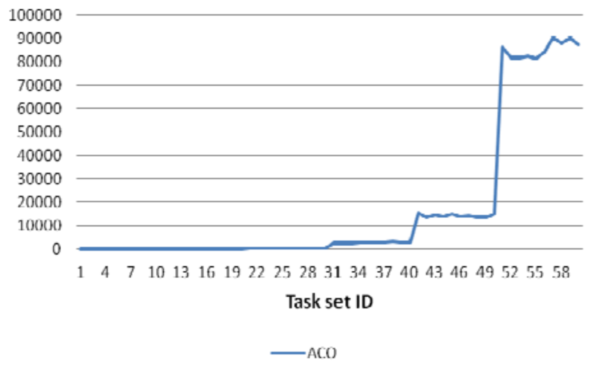

Figure-5. Calculation time for ACO algorithm.

It is obvious that Hybrid algorithm has better result in comparison with ACO algorithm. Also Hybrid algorithm able to achieve the PSO result with less iteration therefore it is time consuming. It is highlighted that Hybrid algorithm which is based on PSO and IWO has less complexity in comparison of ACO algorithm. As result, computation time of Hybrid algorithm is much faster than ACO.

Figure-5 illustrates calculation time for ACO algorithm. The horizontal line represents the task set ID and vertical line shows calculation time. As it can be observe by raising the number of the tasks, calculation time increase dramatically.

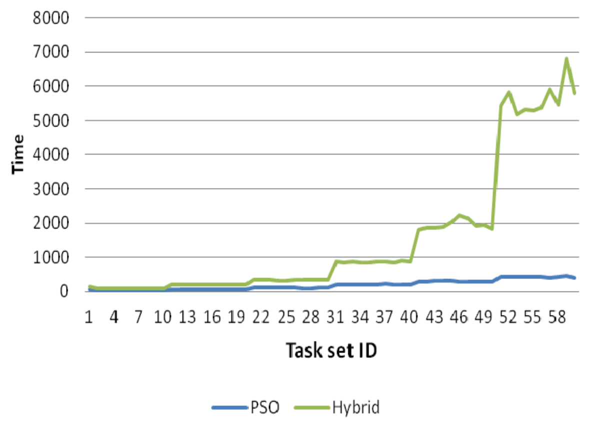

Figure-6. Calculation time for PSO and Hybrid algorithm.

Figure-6 is representing the total calculation time for scheduling sample dataset by using PSO and Hybrid scheduling algorithm. In this graph, horizontal line represents the task set ID and the vertical line illustrate total calculation time which each algorithm required to finish the scheduling respective.

Based on the authors result, ACO is performed faster than PSO and HPI algorithm for all the tasks set less than 15 tasks. However, after that for remaining pending task set calculation time for ACO scheduling algorithm increased dramatically. Hybrid calculation time is a bit longer than normal PSO due to using invasive weed optimization for achieve better result. By analysis Figures 3, 4, 5 and 6 it is obvious that suggested Hybrid algorithm by using more time able to achieve better result in overload situation and it could be a suitable scheduling system for real-time systems. This suggestion is regarding that in real-time systems it is not possible scheduler takes long time for scheduling all the tasks since it could have unexpected result for the system. In addition, number of task in task set is important because sometime system face with overload situation and input will be more than normal situation. In this research the authors assume that system is Preemptive.

CONCLUSIONS

Scheduling may seem straight forward and is easy enough to understand, but once more tasks are added and task overloading happens it becomes more difficult to complete a schedule. As is clear from the results ACO and PSO has most optimal scheduling result than EDF and also PSO in comparison with ACO has better result and The evidence suggests that PSO scheduling task is time consuming and also misses less task in comparison with ACO.

New method for uniprocessor real-time system has been presented that is combination of IWO and PSO that called HPI (Hybrid PSO and IWO) and it will be tried to show that HPI acts better than PSO in real time task scheduling.

As feature work the authors are working on new way of optimizing HPI algorithm in order to decrease calculation time by using other optimization algorithm.

REFERENCES

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. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5591845 [Accessed September 23, 2012].

Anon. 1995. HRT-HOODTM: A Structured Design Method for Hard Real-Time Ada Systems. , 3.

Apurva S. and Ketan K. 2009. Adaptive Scheduling Algorithm for Real-Time Multiprocessor Systems., (March). pp 6-7.

Davis R.I. and Burns A. 2011. A survey of hard real-time scheduling for multiprocessor systems. ACM Computing Surveys, 43(4): 1-44. Available at: http://dl.acm.org/citation.cfm?doid=1978802.1978814 [Accessed October 19, 2012].

Dorigo M., Caro G. Di and Gambardella L.M. 1999. Ant Algorithms for Discrete Optimization. pp. 1-36.

Hajimirsadeghi H. and Lucas C. 2009. A Hybrid Iwo/Pso Algorithm For Fast And Global Optimization. (978-1-4244-3861-7/09). pp. 1964-1971.

Karimi M., Motameni H. and Branch S. 2013. Tasks Scheduling in Computational Grid using a Hybrid Discrete Particle Swarm Optimization Corresponding Author: Karimi1064@yahoo.com. 6(2): 29-38.

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, 219(4): 1515-1521. Available at: http://linkinghub.elsevier.com/retrieve/pii/S0096300312007813 [Accessed October 22, 2014].

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. 123: 163-170. Available at: http://linkinghub.elsevier.com/retrieve/pii/S0167610513002249 [Accessed October 16, 2014].

Ramos V., Muge F. and Pina P. 2002. Self-Organized Data and Image Retrieval as a Consequence of Inter-Dynamic Synergistic Relationships in Artificial Ant Colonies Δ. p. 87.

Shah A. and Kotecha K. 2011. ACO Based Dynamic Scheduling Algorithm for Real-Time Multiprocessor Systems. International Journal of Grid and High Performance Computing. 3(3): 20-30. Available at: http://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/jghpc.2011070102 [Accessed November 1, 2012].

Short M. 2010. Improved Task Management Techniques for Enforcing EDF Scheduling on Recurring Tasks. 2010 16th IEEE Real-Time and Embedded Technology and Applications Symposium. pp. 56-65. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5465964 [Accessed September 23, 2012].

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

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. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5487132.