Adaptive Weighted Particle Swarm Optimization for Scheduling Independent Tasks

Vidya G
2nd Yr PG IT
PSG College of Technology
Coimbatore, India

Sarathambekai S
Asst. Professor of IT
PSG College of Technology
Coimbatore, India

Yamunadevi S P
2nd Yr PG IT
PSG College of Technology
Coimbatore, India

***Abstract*—Scheduling is the main difficulty in heterogeneous computing (HC) systems in achieving the high performance. In this paper, a meta-heuristic approach based on Particle Swarm Optimization (PSO) is adopted for solving task scheduling problem. PSO is a population-based algorithm to find the optimal solutions, but its performance is decreased when considering multi-optimization problem. In this paper, an Adaptive Weighted Particle Swarm Optimization is proposed for multi-objective optimization. AWPSO is an efficient and simple tool for multi-objective and multi-dimensional problem. AWPSO enhance the global search ability and to overcome the local optimum by introducing an acceleration factor. The goal is to minimize the makespan and flowtime. The experimental results showed that the performance of the proposed method is effective compared with other heuristic optimization technique namely PSO in finding the optimal solutions.**

Keywords—Scheduling, Adaptive Weighted Particle Swarm Optimization,accelaration factor, makespan, flowtime

I. Introduction

Real-world HC systems are complex combinations of hardware, software and network components. The problem of scheduling a set of dependent or independent tasks in a distributed computing system is a well-studied area. The most common goal of scheduling is to minimize the expected runtime of a task set. Optimal scheduling relates to mapping a set task to a set of resources to proficiently achieve the abilities of the systems. In this paper, the multiprocessor scheduling is considered. The main motivation for multiprocessor scheduling is the desire for increased speed in the execution of a workload. Parts of the workload, called tasks can be spread across several processors and thus be executed more quickly than on a single processor.

In this paper, the tasks are independent and non-pre-emptive and here the static allocation in heterogeneous systems is considered. The static allocation [1] can be applied to a large set of real-world applications that are able to be formulated in a manner which allows for deterministic execution. Some advantages of these techniques over dynamic ones, which determine the module assignment during runtime, are that, static techniques have no runtime overhead and they can be designed using very complex algorithmic mechanisms which fully utilize the known properties of a given application. The main objective of this paper is to minimize the makespan and flowtime. The tasks are assigned to processors and the completion time is called as makespan (the maximum total processing time is minimized). Flowtime is the sum of completion times of all the jobs. In this paper, Adaptive Weighted Particle Swarm Optimization (AWPSO) is presented helps to improve the performance of PSO in multi-objective optimization. The proposed method assigns the tasks to processors and avoids becoming trapped in local optimum and also leads to faster convergence towards the targeted solution.

The residue of this paper is structured as: section 2 analyses the algorithms that are applied to the task scheduling, section 3 formulates the problem, in section 4 PSO is briefly described, and section 5 illustrates the proposed method and section 6 reports the experimental results. Finally section 7 concludes the work.

II. Related Work

Optimal mapping of independent computational tasks to available machines in a distributed computing system is a NP-hard problem as stated earlier and as such, it is a subject to various heuristic and meta-heuristic algorithms. The heuristics applied to the task scheduling problem include sufferage [3], min-min, max-min [4], LJFR-SJFR [5], min-max [6], etc. The most popular of meta-heuristic algorithms are Genetic algorithm (GA) [7], simulated annealing (SA) [8], ant colony optimization (ACO) [9] and particle swarm optimization (PSO) [10]. The above referred heuristics and meta-heuristics aimed at minimizing a single criteria, the makespan of the schedule.

Different criteria can be used for evaluating the efficacy of scheduling algorithms. Few attempts have been made to optimize multiple criteria. [6] Investigates the efficacy of five popular heuristics for minimizing makespan and flowtime on HC environments with various characteristics of both machines and tasks. Page and Naugton [7] used a genetic algorithm method for scheduling HC systems, in this method the scheduling strategy operates in a dynamically changing computing resource environment and adapts to variable communication costs and variable ability of processing resources. G.Subashini and M.C.Bhuvanesawari [12] used NSPSO for scheduling the tasks in heterogeneous systems. In HPSO [14] used the combination of PSO and SA for task scheduling problem with dynamically varying inertia weight. Gen and Cheng propose adaptive weight approach genetic algorithm (AWA GA) [13], it used information of the current population to adjust weight of different objectives. [11] Used adaptive weighted sum method for multi-objective optimization problems.

III. Problem Definition

This paper considers the task scheduling problem with the following scenario. Let the set of independent tasks T= {T_1, T_2,…,T_n} are scheduled on set of m processors P={P1,P2,…Pm} on the HC system. To model the problem estimation or prediction of the computational load of each task, the computing capacity of each resource, and an estimation of the prior load of each one of the resources is required. This is the Estimated Time to Compute (ETC) matrix. An ETC matrix is an n x m matrix, which n is the number of tasks and m is the number of machines. In ETC matrix one row comprises the estimated execution time for a given task on each machine and one column compromises the estimated execution time of a given machine for each task. To formulate the objective where, $C_{i,j}$ (i €{1,2,…,m}, j € {1,2,…,n}) is the execution time for executing j^{th} task in i^{th} machine and W_i (i €{1,2,…,m}) is the preceding workload of machine M_i. The time required for machine M_i to complete the jobs is shown in the equation (1) [10].

$$\sum C_{ij} + W_i \tag{1}$$

The makespan (2) [10] and flowtime (3) [12] is minimized and it can be estimated as

$$\text{Makespan} = \max \{\sum C_{ij} + W_i \}, i € \{1, 2,.., m\} \tag{2}$$

$$\text{Flowtime} = \sum_{i=1}^{m} \sum C_{ij} \tag{3}$$

By minimizing the makespan the set of tasks can be executed promptly and by minimizing the flowtime exploit the computing environment in effectively. The goal of the scheduler is to minimize the makespan and flowtime.

IV. Particle Swarm Optimization

Particle Swarm Optimization [10] is a population based stochastic optimization technique developed by Dr.Eberhert and Dr.Kennedy in 1995. PSO simulates the behaviors of bird flocking. That is a group of birds are randomly searching for food in an area. But al birds know how far the food.

So the best strategy is to follow the bird which is nearest to the food. PSO learned from the scenario and used it to solve the optimization problems. In PSO, each single solution is a bird (particle) in the search space. All of particles have fitness values which are evaluated by the fitness function to be optimized, and have velocities which direct the flying of the particles. The particles fly through the problem space by following the current optimum particles. PSO is initialized with a group of random particles (solutions) and then searches for optima by updating generations. For each iteration, all the particles are updated by following two best values. The first one is the best solution (fitness) it has achieved and this value is called personal best position or pbest. Another best value is tracked by the particle swarm optimizer, obtained so far by any particle in the population. This best value is a global best and called as neighborhood best position or gbest. When a particle takes part of the population as its topological neighbors, the best value is a local best and is called pbest.

After finding the two best values, the particle update its velocity V_{ij} using equation (4) and position using equation (5) [12].

$$V_{ij} = W * V_{ij} + [c1 * rand1 (pbest_{ij} – particle_{ij}) + c2 * rand2 (gbest_{ij} – particle_{ij})] \tag{4}$$

$$particle_{ij} = particle_{ij} + V_{ij} \tag{5}$$

where c1 and c2 are the cognitive coefficient, $particle_{ij}$ is the current particle and rand1 and rand2 are random real numbers drawn from U (0, 1). The c1 shows how much the particle trusts its own past experience, it is called cognitive parameter, and c2 shows how much it trusts the swarm, it is called the social parameter. The inertia weight w controls the momentum of the particle a large inertia weight pressures towards global exploration while a smaller inertia weight pressures towards global exploration fine tuning the current search area. The inertia weight is introduced in equation (6) to balance the global and local search abilities. The large inertia weight facilitates global search while the small inertia weight facilitates local search.

The introduction of the inertia weight also eliminates the requirement of setting the maximum velocity. In PSO with fixed inertia, the inertia value is fixed to a constant value of 0.8 [14] during the whole run of the algorithm.

$$W = 0.8; \tag{6}$$

V. Adaptive Weighted Particle Swarm Optimization

The standard PSO is selected as a parent algorithm for AWPSO. The basic problem in PSO, it is started with a large inertia weight but is decreased over time due to this performance of PSO is declined. So, in order to obtain an optimized solution, the AWPSO uses the acceleration factor to improve the performance of PSO for multi-objective optimization. The acceleration term will increases as the number of iterations increases, which will help to improve the global search ability and also to skip from the local optimum. For AWPSO, the velocity is updated using the equation (7) and position using the equation (5) [12].

$$V_{ij} = W * V_{ij} + \alpha [c1 * rand1 (pbest_{ij} – particle_{ij}) + c2 * rand2 (gbest_{ij} – particle_{ij})] \tag{7}$$

For AWPSO algorithm, the inertia weight W is shown in the equation (8) [15].

$$W= W_0 + r (1-W_0) \tag{8}$$

where W_0 is the initial positive constant in the interval [0, 1] and r is random number obtained. From a uniform random distribution function in the interval [0, 1]. The suggest range for W_0 is [0, 0.5], which make the weight w randomly varying between 0 and 1.

Acceleration factor (9) [15] is formulated as

$$\alpha = \alpha_0 + \frac{t}{T} \qquad (9)$$

where, t is the current generation, T denotes the number of generations and range for α_0 is [0.5, 1].
Pseudo code of AWPSO is as follows,

```
Start
Create a swarm with P particles
For each particle initialize the position and velocity randomly;
   Repeat
     For each particle P=1…N do
        Calculate the fitness value of each particle;
        Update pbest and gbest for each particle;
        Update velocity using equation (7) and update
         position using equation (5);
    End
 Until stopping condition is true;
End
```

A. *Particle Generation*

In the initial step, the independent tasks are given as inputs for generation of particles. Each particle represents one solution to the problem and a set of particles is referred to as population. The first generated particle is termed as initial population (Swarm). The example for particle generation is described as follows

If task is given as 3 then the particles are generated as 123,231,321,132,213,312.

B. *Processor Allocation*

The processors are allocated to the task, randomly. Once the task is assigned to the processor and the same task will not be further assigned to any other processors. The example for processor allocation is described as follows

From the example of particle generation, consider the generated particle123 is allocated to 2 processors. In the table below particle1 is allocated to processor1, particle2 is allocated to processor2 and particle3 is allocated to processor1 in this manner for all the generated particles the processors are allocated.

TABLE I. PROCESSOR ALLOCATION TABLE

Generated Particles	Processors
Particle1	Processor 1
Particle2	Processor2
Particle3	Processor1

C. *Fitness Calculation*

The fitness function measures to what extent the particle solution S satisfies the objective of the optimization problem. The fitness function F(x) is a sum of two objectives, the makespan and flowtime as given by the equation (2) and (3). The approach called adaptive weighted sum approach [11] is used to weight the sum of the objectives. Each particle evaluation is given by the equation (10) [11].

$$J^{m}_{Total} = \alpha_{m-1}\, J^{m-1}_{Total} + (1-\alpha_{m-1})J_m\,, \quad m\geq 2 \qquad (10)$$

where m is the number of objectives, α_i is the ith weighting factor in the interval of (0, 1) and J is the objective function.

D. *Particle Updation*

At first the particle's velocity is updated using the equation (7) and then it is used for updating the particle's position using the equation (5). Velocity defines how quickly a particle changes location and position represents a possible solution point in the problem space. In PSO, every particle flutters through the solution space by equations (7) and (5) in each iteration. The loop is terminated when given iterations are met. When the termination criteria are met, the best solution and its corresponding objective function values as the optimized value are returned.

VI. EXPERIMENTAL RESULTS

An effective scheduling algorithm has been developed to schedule the tasks onto processors in a distributed computing system. In order, to evaluate the performance of the proposed method, the approach was compared with PSO for task assignment problem in multiprocessor systems. The goal of the scheduler in these methods is to minimize the makespan and flowtime. These methods are implemented using Java.

For the proposed method the following ranges of parameter values were tested: c1 and c2 = [1, 2], w_0 = [0, 1], r = [0, 1] and rand1 and rand2 = [0, 1]. Based on the experimental results the proposed AWPSO algorithm performs best under the following settings: r= 0.5, c1= c2= 1.2, w_0 = 0.5, w1=w2=0.5, rand1 = rand2=0.2, w= 0.8 (fixed Inertia weight for PSO) and α_0 = 0.5.

In this section, both the AWPSO and PSO algorithms are applied and the results are plotted. The plots are placed for the same number of tasks but for different number of iterations, population sizes and processors.

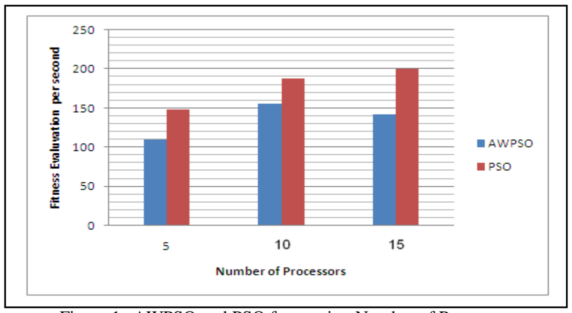

Figure 1. AWPSO and PSO for varying Number of Processors

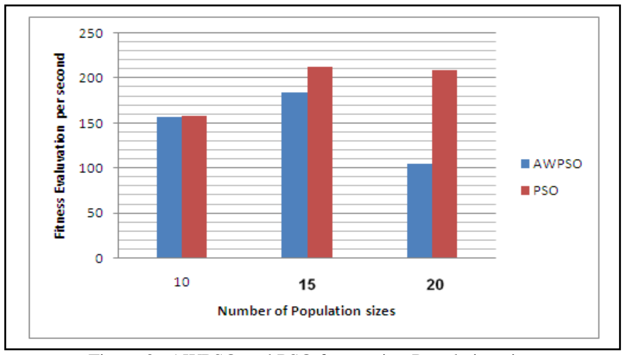

Figure 2. AWPSO and PSO for varying Population sizes

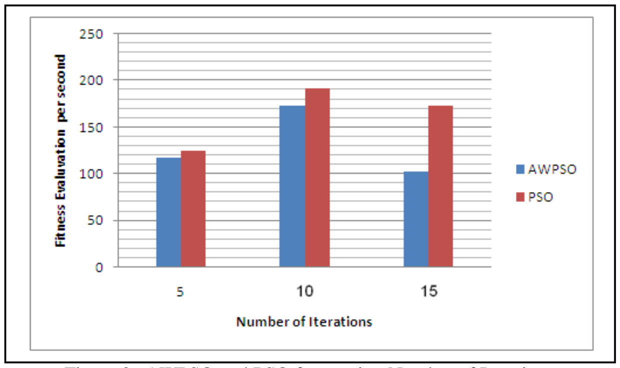

Figure 3. AWPSO and PSO for varying Number of Iterations

From the figures above i.e., Figure 1, Figure 2 and Figure 3 showed that the proposed algorithm Adaptive Weighted Particle Swarm Optimization outperforms other methodology namely PSO. Thus the task scheduling is better in AWPSO than PSO.

VII. CONCLUSION

In distributed computing systems, the assignment of tasks to set of processors is important for effective utilization of resources. In this paper, the proposed algorithm AWPSO finds optimal solutions by minimizing the makespan and flowtime simultanously. The Experimental results proved that the AWPSO algorithm shows better results compared to the PSO algorithm with fixed Inertia weight. However, further work could be carried out with AWPSO algorithm for scheduling the jobs with precedence constraints or in dynamic environments.

REFERENCES

[1] Prasana Sugavanama, H.J.Seigel, Anthony A.Maciejewski, Mohana Oltikar, Ashih Mehta, Ron Pichel, Aaron Horiuchi, Vladimir Shestak, Mohammad Al-Otaibi, MahirAydin, Panho Lee, Kumara Guru, Micheal Raskey, Alam Pippin," Robust static allocation of resources for independent tasks under makespan and dollar cost constraints", in Journal of Parallel Distributed Computing,vol.676,pp.400-416,2007.

[2] A.Abraham, H.Liu, W.Zhang, T.G. Chang,"Scheduling Jobs on Computational Grids Using Fuzzy Partilce Swarm Algorithm", in Springer- Verlag Berlin Heidelberg ,pp.500-507,2006.

[3] M.Macheswaran, S.Ali, H.J.Seigel,D.Hensen and R.F.Freund, "Dynamic mapping of a class of independent tasks onto heterogeneous computing systems", in Journal of Parallel Distributed Computing, vol.59,pp.107-131,1999.

[4] R.F.Freud, M.Gherrity, S.Ambrosius, M.Campbell, M.Halderman, D.Hensgen, E.Keith, T.Kidd, M.Kussow, J.D.Lima, F.Mirabile, L.Moore, B.Rust and H,J.Seigel, "Scheduling resources in multi-user,heterogeneous, computing environments with SmartNet", in 7th IEEE Herterogeneous Computing Workshop, pp.184-199, 1998.

[5] A.Abraham, R. Buyya and B.Nath, "Nature's heuristics for scheduling jobs on computational grids", in 8th IEEE International Conference on Advanced Computing and Communications, pp.45-52,2000.

[6] H.Izakian, A.Abraham and V.Snasel, " Comparision of heuristics for scheduling independent tasks on hereogeneous distributed environments", in International Joint Conference on Computational Sciences and Optimization, pp. 8-12,2009.

[7] J.Page and J.Naughton, " Framework for task scheduling in heterogeneous distributed computing using genetic algorithms", in Artificial Intelligence Review,vol.24,pp.415-429,2005.

[8] A.Yarkhan and J.Dongarra, "Experiments with scheduling using simulated annealing in a grid", in 3rd International Workshop on Grid Computing, MD, USA, November18,pp.232-242,2002.

[9] G.Ritchie and J.Levine, "A fast, effective local search for scheduling independent jobs in heterogeneous computing environments", Technical 17 Non Dominated Particle Swarm Optimization report, Centre for Intelligent System and their Applications, School of Informatics, University of Edinburgh,2003,

[10] Hesam Izakian, Ajith Abraham and Vaclav Snasel, " Metaheuristic Based Scheduling Meta-Tasks in Distributed Heterogeneous Computing Systems", in Proc. Sensors, ISSN 1424-8220, vol.9, pp.5339-5350, 2009.

[11] II Yong Kim and Olivier de Week," Adaptive Weighted Sum Method for Multiobjective Optimization", in Proc. AIAA, 2004.

[12] G.Subashini and M.C.Bhuvaneswari, " Non Dominated Particle Swarm Optimizatio for Scheduling Independent Tasks on Heterogeneus Distributed Environments", in ICSRS Publication, vol.3, ISSN 2074-8523, 2011.

[13] R.Cheng and M.Gen, " An Adaptive superplane approach for multiple object optimization problems", in Technical report, Asikaga Institute of Technology, 1998.

[14] S.N.Sivanandam and P.Visalakshi, "Dynamic Tsk Scheduling with Load Balancing using Hybrid Particle Swarm Optimization", in ICSRS, vol.2, ISSN 1998-6262, 2009.

[15] C.Agees kumar and N.Kesavan Nair, "Multi-objective PID Controller based on Adaptive Weighted PSO with application to steam temperature control in Boiler", in International Journal of Engineering Science and Technology, vol.2(7),pp.3179-3184,2010.