ISSN: 0974-5823 Vol. 7 No. 5 May, 2022

International Journal of Mechanical Engineering

Heuristic Oriented Process Scheduling for Homogeneous Multiprocessor Environment

[1]Sunita Kushwaha, [2]Varsha Thakur

[1] Asst. Professor, MSIT Dept.

MATS University, Raipur, CG, India

[2] Asst. Professor, CS Dept.

Govt. NPG Science College, Raipur, CG, India

Abstract. Optimize the schedule length in multiprocessor environment is indispensable and fiddly task. In Consequence, scheduling has perpetually been a leading area of interest for researchers. Multiprocessor scheduling is an NP complete problem. Heuristic techniques in scheduling algorithm yield good results in less than polynomial time. In heuristic scheduling there are mainly three categories list scheduling, cluster based scheduling and Task Duplication Based (TDB) scheduling. Task duplication based and cluster based scheduling are applicable only for dependent task sets, whereas list scheduling algorithm is applicable for dependent as well as independent task set both. This paper, apply heuristic techniques to achieve optimum schedule length (Makespan) for multiprocessor system. Simulation study has been done for M numbers of homogeneous processor and N numbers of process/task. Where all N processes are randomly arrive on different time instances. For simulation some heuristic scheduling algorithms has been selected, and compression of performance of newly proposed list heuristic algorithm has been done. Simulation studies show that newly proposed algorithm performs better than some well known selected algorithms.

Keywords: Heuristic Scheduling, List Scheduling, TDB, Cluster Based Scheduling

1. Introduction

Scheduling is not a contemporary conception; it endures from long standing years in the distinct province such as military perspective, pyramids, traditional railways etc. Without scheduling, we cannot speculate such kind of intricate task. Scheduling is not only a technical concept; but a concept that is used in our routine life [1]. A prevalent and engrossing problem of the multiprocessor system is "how to schedule multiple processes on multiple processors to get optimal solution" which is known as scheduling [2]. A Scheduling problem may consist of three main components: 1) Process 2) Processor 3) Policy (Scheduling Approach) the problem of task scheduling is one of the most imperative and challenging issues in homogeneous computing environments. Finding an optimal solution for a scheduling problem is NP-complete [3]. Therefore, it is necessary to have heuristic to find a reasonably good schedule rather than evaluate all possible schedules. List scheduling is generally accepted as an attractive approach, since it pairs low complexity with good results [4]. Therefore, list heuristic scheduling is selected for investigation.

2. Related Work

Multiprocessor system scheduling can be classified into static and dynamic scheduling classes [3-5]. Further static scheduling algorithms are, classified into optimal and suboptimal scheduling algorithms. Suboptimal scheduling algorithms can also be classified as heuristic and approximate scheduling algorithms [3-10]. Heuristic scheduling algorithm follows a thumb rule to moves from one point in search space to another point in search space. Heuristic scheduling algorithms are more adaptive [14] method and are used to find near optimal solution [15-16]. Heuristic scheduling algorithms are classified into Task Duplication Based (TDB) scheduling, Cluster Based scheduling and List scheduling.

In TDB scheduling algorithms predecessor tasks (processes) are copied on all those processors where successor tasks of those predecessor tasks are present. This strategy reduces communication overhead, consequently the system in which data band width is low, TBD performs better. In cluster based scheduling, a set of tasks that need to communicate among themselves are grouped together to form a cluster. Each of these task clusters is then scheduled on the available processors [1, 8, 10-15]. Now entire cluster is scheduled instead of task. Therefore, there are two levels of scheduling one is inter cluster scheduling and another is intra cluster scheduling. In inter cluster scheduling entire cluster is schedule on one particular processor. In intra cluster scheduling all the processes within a cluster are scheduled in a particular manner on that processor only [1, 8, 10-15]. In a list scheduling an ordered list is generated on the basis of certain property. Higher priority task is scheduled earlier. In list scheduling DAG (Direct Acyclic Graph) is used to define the dependencies among the tasks. List scheduling is used to schedule both the dependent and the independent tasks [3, 10-15, 17-24, 26].

3. Problem Statement

Optimization of scheduling in multiprocessor environment is seemed a fiddly task, it is not easy due to obtain better solution in cost effective way. In this situation heuristic techniques are used to achieve simple and optimal or near optimal solution. Heuristic follows a thumb rule, where one or two attributes are selected in scheduling process and other attributes going to be ignored.

Let consider the N number of processes/tasks are arrived randomly, and scheduled on M number of homogeneous processors.

Suppose that T_s is set of tasks/ processes in which $\{T_1, T_2, T_{3,}\ldots..,T_N\}$ are n tasks/processes. Release time or arrival time of these tasks are $\{A_1,A_2,A_3,\ldots.,A_N\}$. Similarly the processing time are $\{\{P_1,P_2,P_3,\ldots..,P_N\}$. Also the completion time of a task/process are $\{C_1,C_2.C_3,\ldots..,C_N\}$

So that

$T_i(C_i) \geq T_i(A_i)+ T_i(P_i)$

And $T_i(C_i) \leq$ Makespan , Where $i =1,2,3,\ldots.,N$.

Where T_i (C_i) is completion time of ith task/process and T_i (A_i) as arrival/release time and T_i (P_i) is processing time of ith task/process. Makespan is overall completion time of task set T_s (scheduled on M number of homogeneous processors). Minimization of the makespan has been used to achieve the balancing the load over the M processor, which is an important objective in multiprocessor environments.

4. Performance Parameters

In these investigational studies it is seen that all the processors of multiprocessor system are not busy exactly for same time. Consequently the busy time of all processor of multiprocessor system is always less than or equal to their makespan. Hence, the idle or inactive time of all processor is calculated as a performance parameter under the name of "Average Idle Rate" (AIR). As the idle time minimize performance of system goes high. Thus, Average Idle Rate (AIR) defined as:

$$AIR = \frac{(\text{Cmax} \ * \ \text{M}) - \ \text{Total processing time}}{\text{Makespan} \ * \ \text{M}}$$

Where M is number of processors.

5. Proposed List Heuristic Algorithm:

Proposed List Heuristic Algorithm:

Step 1: Let M number of processors (homogenous) and N number of processes with Processing time Pi and random Arrival/Release time Ai, where i=1,2,3….,N.

Step2: Sort the set of processes according decreasing order of processing time. If two or more processes have equal processing time then process with minimum arrival time is arranged first and store in MLPT(Pi,Ai).

Step3: Sort the set of process according to increasing order of arrival time. If two or more processes have equal arrival time then processes with longest processing time is arranged first and store in MEST(Pi,Ai).

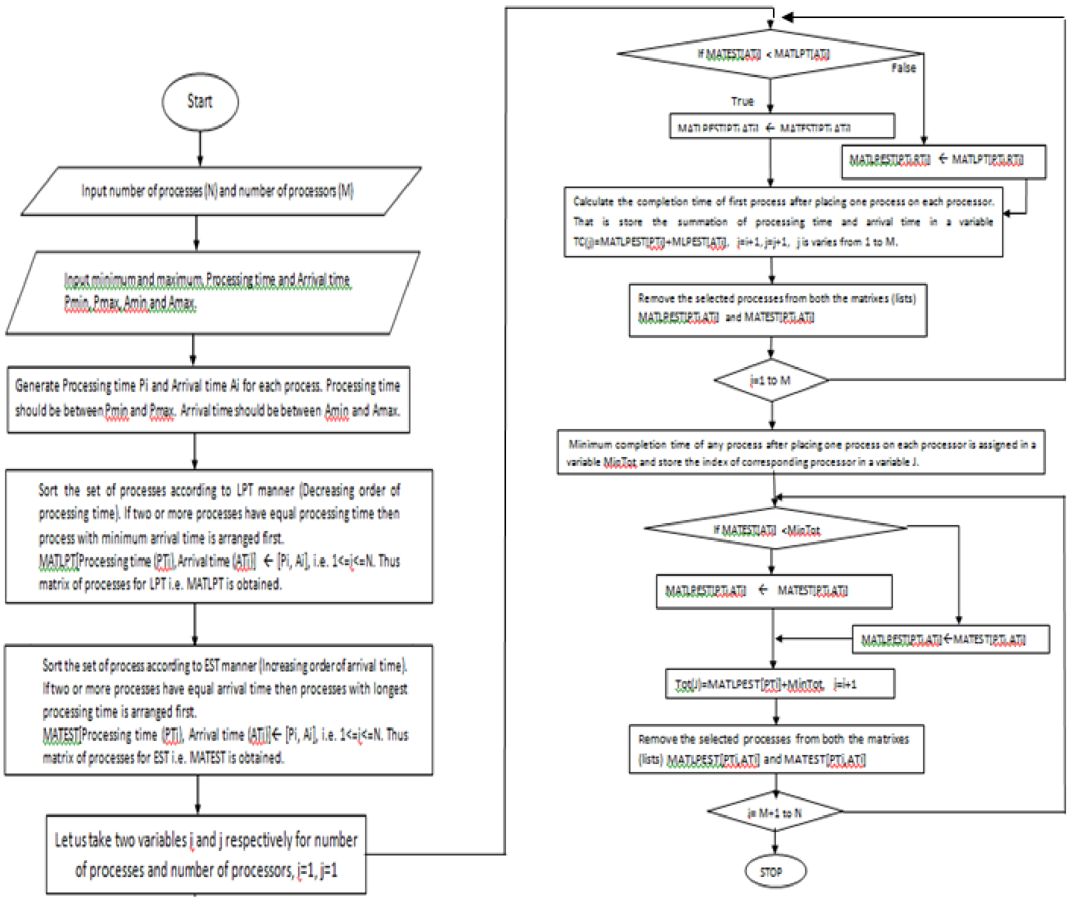
Step4:for I=1 to M , compare the arrival time of first process of both the matrix MLPT (Pi,Ai) and MEST(Pi,Ai). Select the process that have minimum (earlier) arrival form MLPT (Pi,Ai) or MEST(Pi,Ai) and store in the new matrix MNEW(Pi,Ai) as first process. And remove this process from both the matrix MLPT(Pi,Ai) and MEST(Pi,Ai).

Step5: for I = M+1 to N, Calculate Ai+Pi for named as PAT(processor available time) for each process after placing first process on all processors. And compare the arrival time of MLPT(Pi,Ai) first process with the minimum PAT. If Arrival Time of MLPT(Pi,Ai) process is higher than the minimum PAT then goto the step4 otherwise assign the remaining processes into the MNEW(Pi,Ai) according to LPT manner.

Step6: Schedule the process according to MNEW(Pi,Ai), and calculate PAT as Pi + min(PAT). To identify the availability of processor form the given set of processors. And schedule the next process from MNEW(Pi,Ai) on it.

Step7: End

Flow chart of Proposed List Heuristic Algorithm:


Start
Input number of processes (N) and number of processors (M)
Input minimum and maximum, Processing time and Arrival time Pmin, Pmax, Amin and Amax.
Generate Processing time Pi and Arrival time Ai for each process. Processing time should be between Pmin and Pmax. Arrival time should be between Amin and Amax.
Sort the set of processes according to LPT manner (Decreasing order of processing time). If two or more processes have equal processing time then process with minimum arrival time is arranged first. MATLPT[Processing time (PTi), Arrival time (ATi)] ← [Pi, Ai], i.e. 1<=i<=N. Thus matrix of processes for LPT i.e. MATLPT is obtained.
Sort the set of process according to EST manner (Increasing order of arrival time). If two or more processes have equal arrival time then processes with longest processing time is arranged first. MATEST[Processing time (PTi), Arrival time (ATi)] ← [Pi, Ai], i.e. 1<=i<=N. Thus matrix of processes for EST i.e. MATEST is obtained.
Let us take two variables i and j respectively for number of processes and number of processors, i=1, j=1
If MATEST[ATi] < MATLPT[ATi]
False
True
MATLPEST[PTi,ATi] ← MATEST[PTi,ATi]
MATLPEST[PTi,ATi] ← MATLPT[PTi,ATi]
Calculate the completion time of first process after placing one process on each processor. That is store the summation of processing time and arrival time in a variable TC(j)=MATLPEST[PTi]+MLPEST[ATi]. i=i+1, j=j+1, j is varies from 1 to M.
Remove the selected processes from both the matrixes (lists) MATLPEST[PTi,ATi] and MATEST[PTi,ATi]
j=1 to M
Minimum completion time of any process after placing one process on each processor is assigned in a variable MinTot and store the index of corresponding processor in a variable J.
If MATEST[ATi] <MinTot
MATLPEST[PTi,ATi] ← MATEST[PTi,ATi]
MATLPEST[PTi,ATi] ← MATEST[PTi,ATi]
Tot(J)=MATLPEST[PTi]+MinTot, i=i+1
Remove the selected processes from both the matrixes (lists) MATLPEST[PTi,ATi] and MATEST[PTi,ATi]
i= M+1 to N
STOP

6. Performance Analysis

List heuristic scheduling techniques assign a priority to each task to be scheduled and then sort the list of tasks in decreasing order of that priority. As processors become available, the highest priority task in the task list is assigned to be processed and is removed from the list. If more than one task has the same priority, ties are broken using some method [10, 15] or typically by random method [16]. Some well known list heuristic scheduling algorithms are selected for analysis proposes namely: LPT (Longest Processing Time) are arranged the task in descending order of processing time. SPT (Shortest Processing Time) are arranged the tasks in increasing order of their processing times. In the ECT (Earliest Completion Time first), process with minimum completion time is scheduled first. Completion time is the summation of arrival time and processing time. In EST (Earliest Starting Time), processes are arranged in increasing order of their starting time. In EDD (Earliest Due Date) processes are arranged in increasing order of their due date. In EDF (Earliest Deadline First), process with earliest deadline is selected to schedule first. In WSPT (Weighted Shortest Processing Time) first of all the ratio of processing time and weight (P_i/W_i) is calculated. Then the process which has minimum P_i/W_i is scheduled first where P_i is processing time of i^{th} task and W_i is weight of i^{th} task [17-22] etc. Performance analysis of newly proposed heuristic list scheduling LPEST (Longest Processing Earliest Starting Time first) with selected heuristic scheduling algorithms has been done with the help of TORSCHE simulation tool based on MATLAB.

Table1: Environmental Setup

Parameters	Value
No of processors	1, 2, 3, 4, 5, 6
No of processes	10, 15, 20, 25
Scheduling algorithms	LPT, SPT, EST, ECT, WSPT, EDF, EDD, LPEST
Range of time instant for arrival	[0-4]
Range of processing time for each process	[5-10]

Makespan: In scheduling makespan is an important parameter which shows the maximum time taken to schedule a set of processes on given set of processors. For better performance makespan should be minimum [25-29].

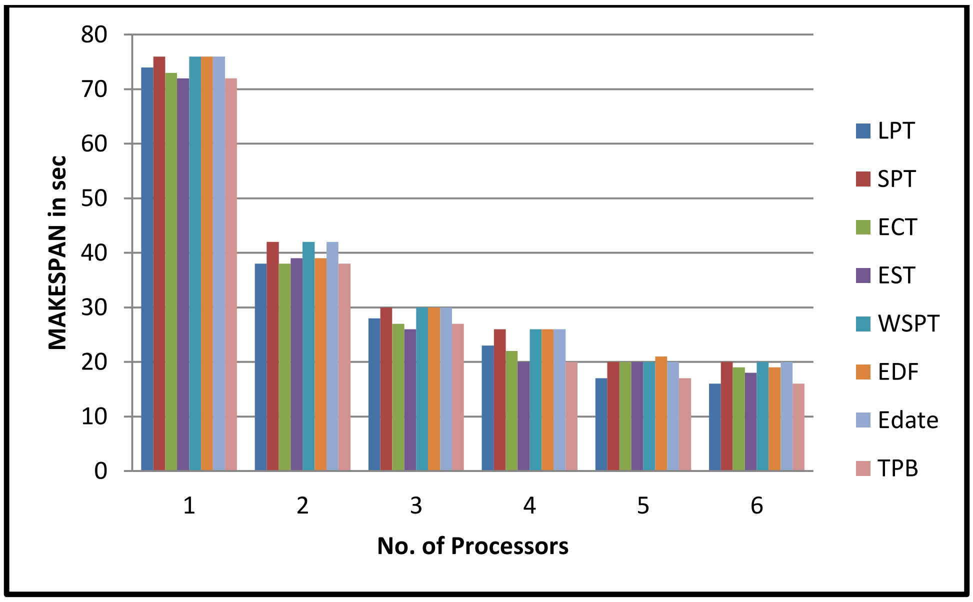


Figure 1: Makespan for 10 processes

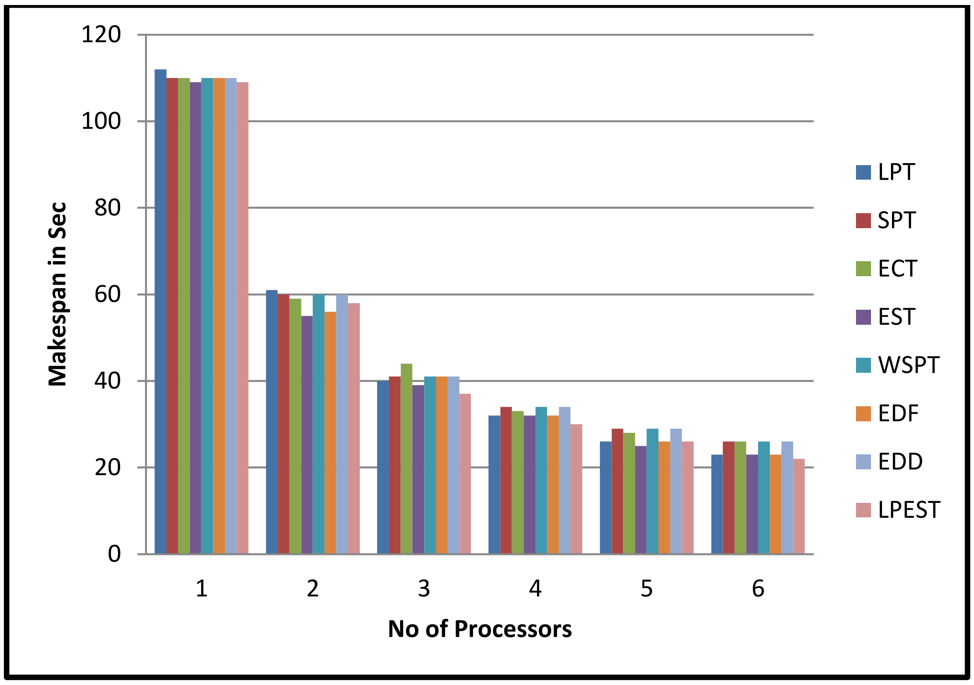


Figure 2: Makespan for 15 processes

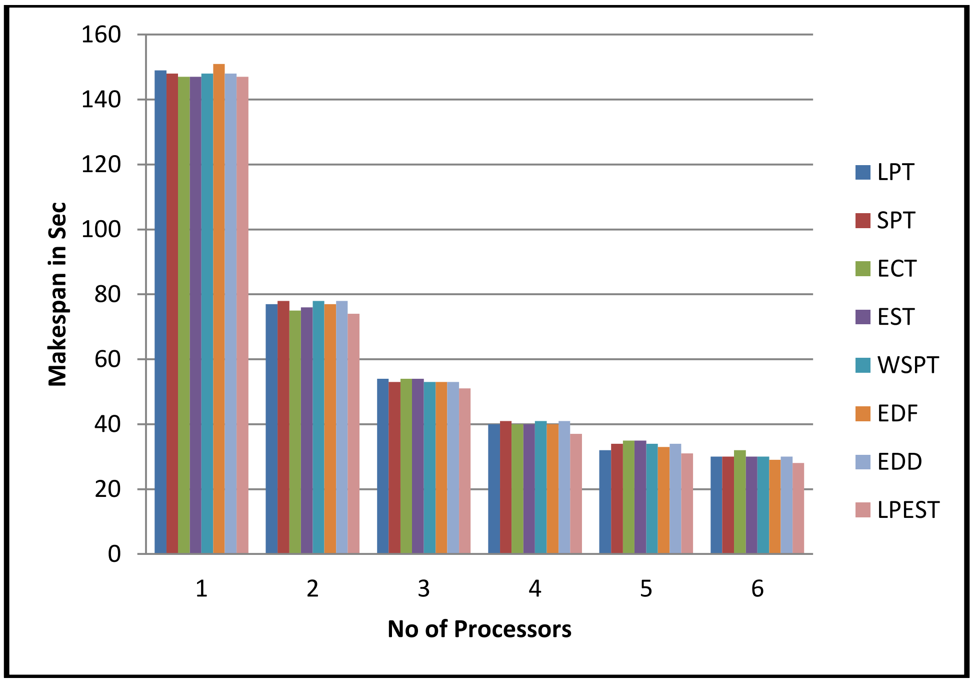


Figure 3: Makespan for 20 processes

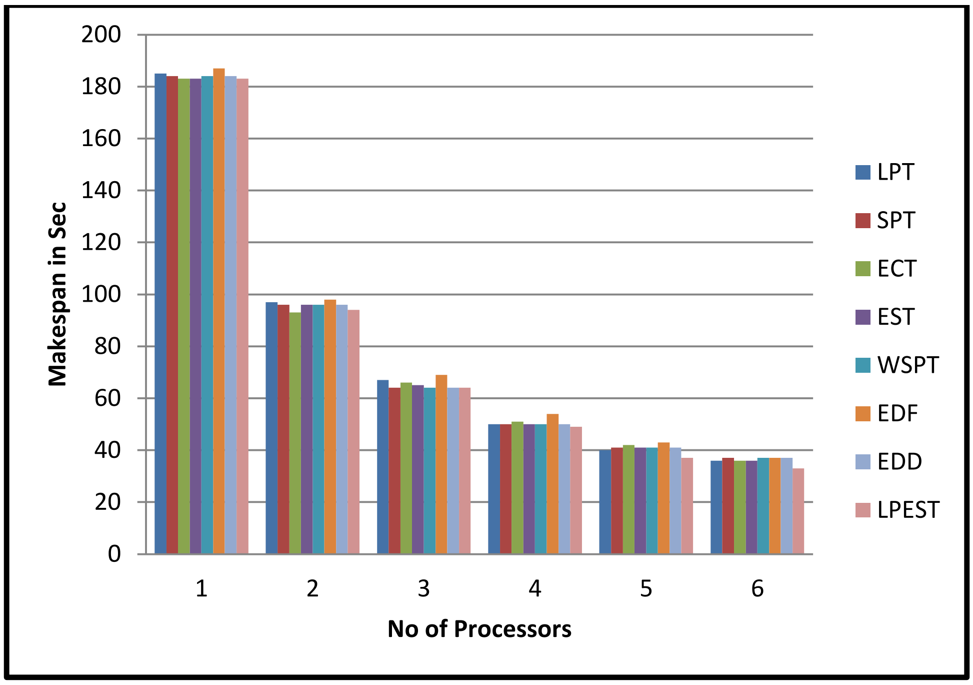


Figure 4: Makespan for 25 processes

Speedup: Speedup is the ratio of parallel time of uniprocessor system P(1) and multiprocessor system P(N). For better performance speedup should be high.

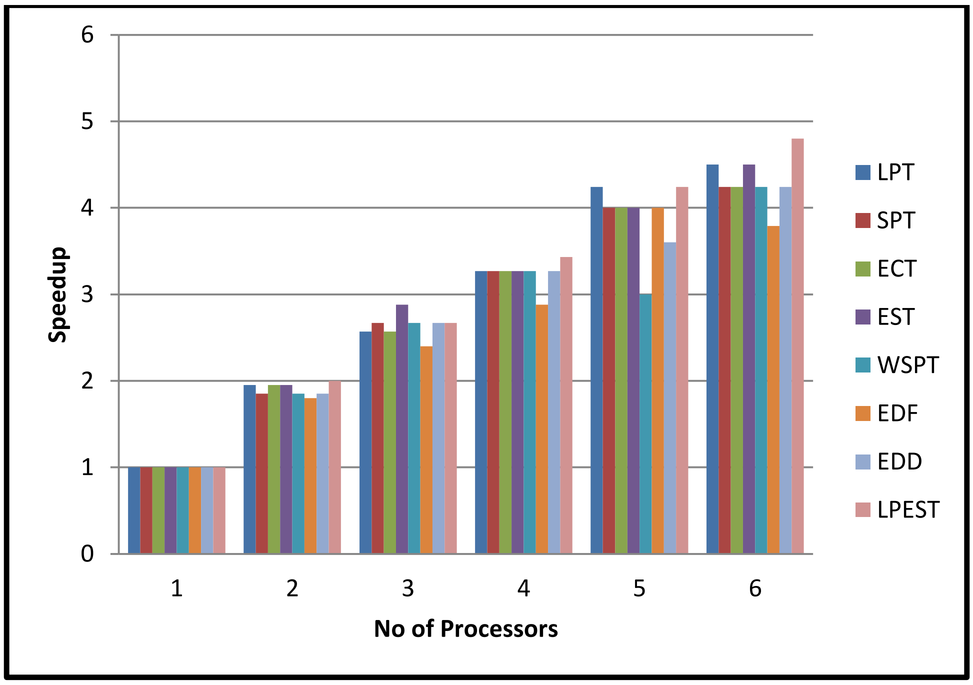


Figure 5: Speedup for 10 processes

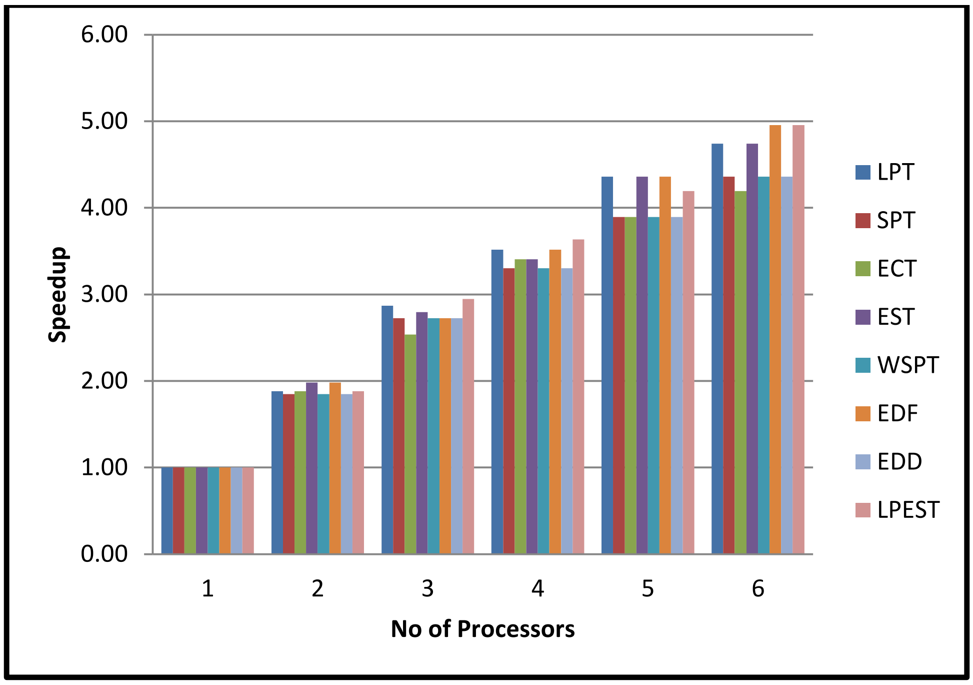


Figure 6: Speedup for 15 processes

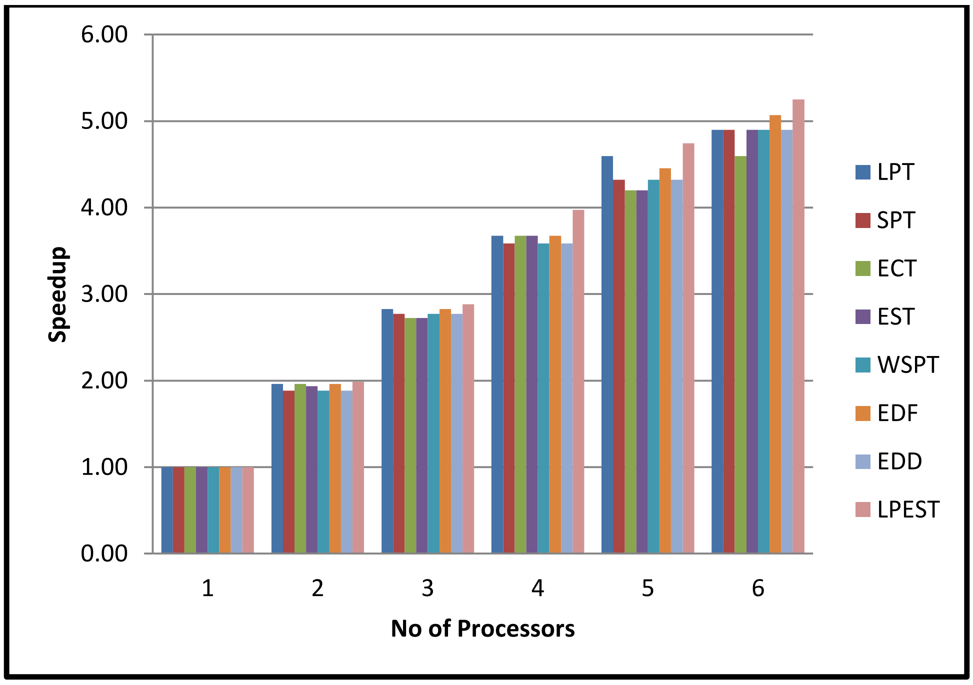


Figure7: Speedup for 20 processes

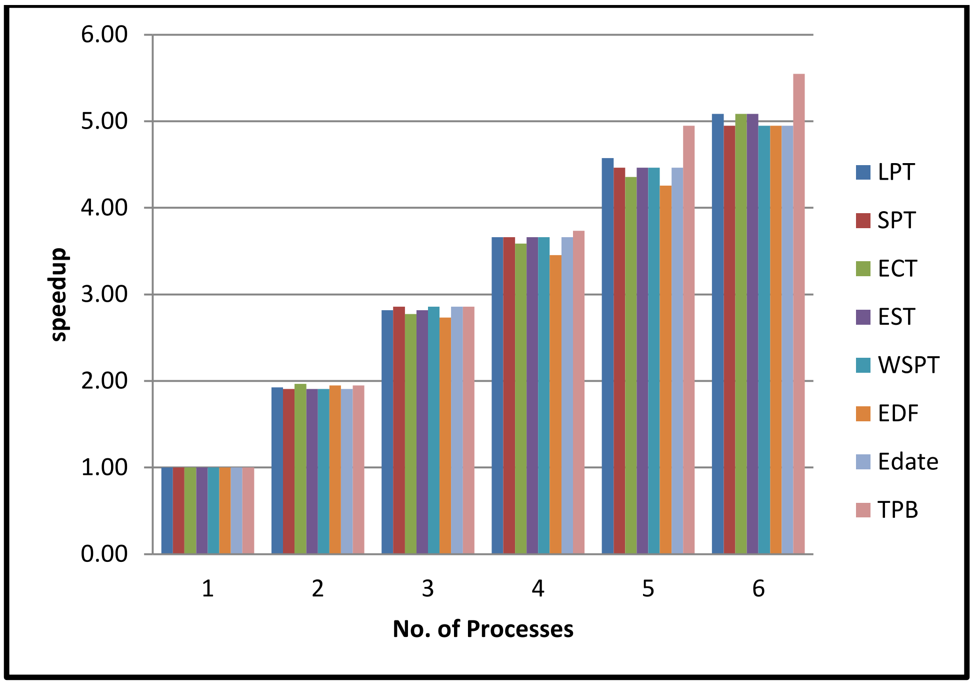


Figure8: Speedup for 25 processes

Average Idle Rate (AIR) is a newly proposed performance parameter. Average Idle Rate is the idle rate of multiprocessor system. Average Idle Rate defined as – ratio of difference between Total busy time of processors and Total processing time and Total busy time of processors. For better performance AIR should be low.

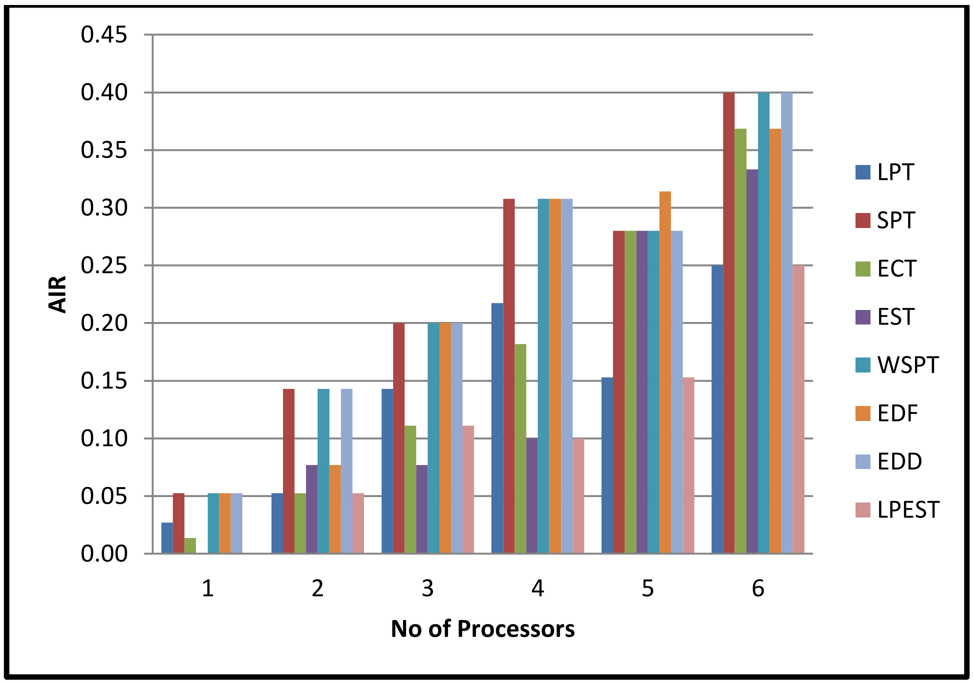


Figure 9: AIR for 10 processes

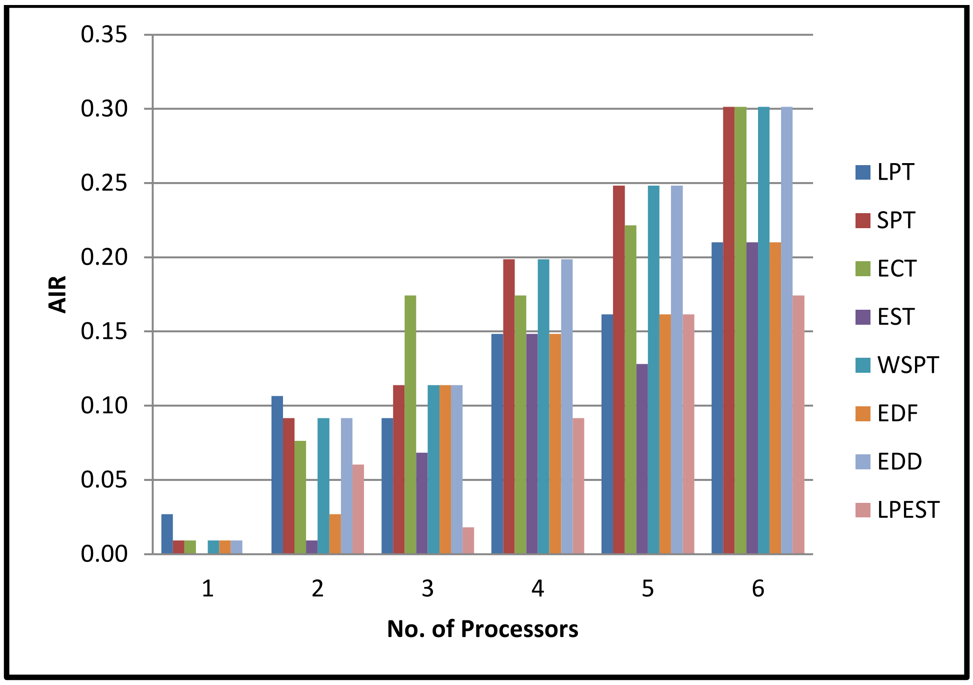


Figure 10: AIR for 15 processes

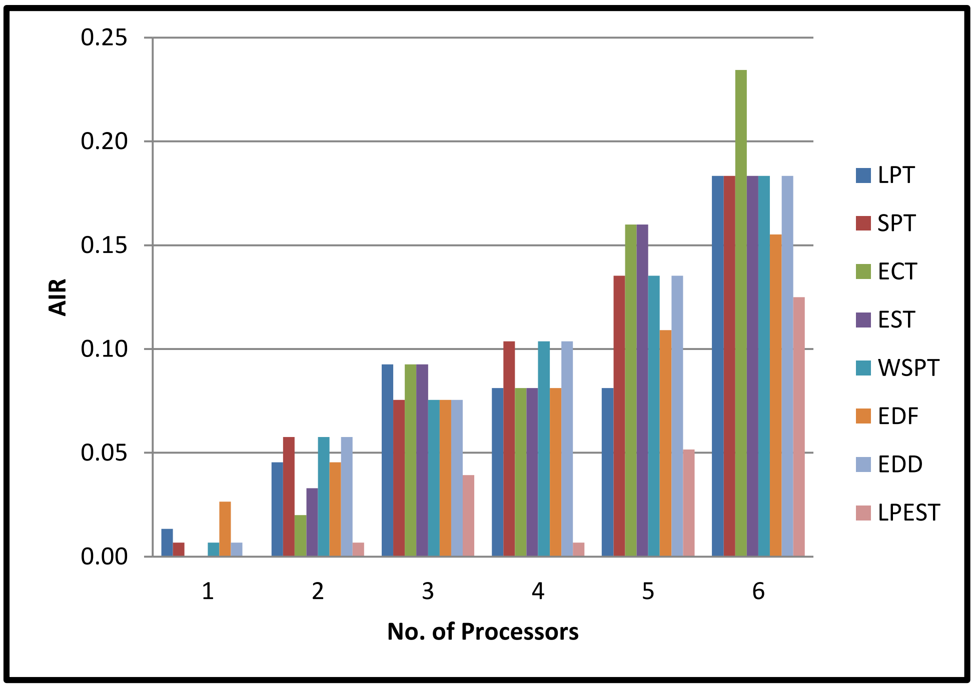


Figure 11: AIR for 20 processes

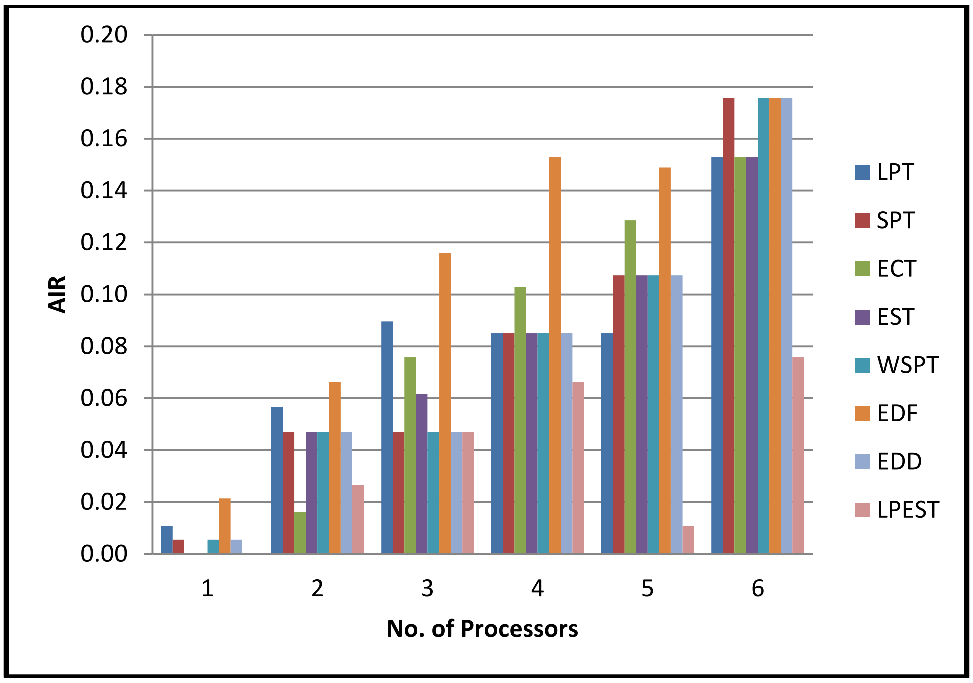


Figure 12: AIR for 25 processes

Performance metrics in terms of Makespan has been obtained and their relative values have been compared from some selected scheduling algorithms. Figure1, Figure 2, Figure3, and Figure 4. The values obtained by experiments are shown in the form of graphs, from graphs (figure 1 to figure 4) it is shows that the makespan of proposed scheduling algorithm (LPEST) has least in most of the time as compare to other list heuristic scheduling algorithms.

Similarly, Speedup has been obtained and their relative values have been compared with selected scheduling algorithms. Related values of speedup for selected scheduling algorithms and newly proposed algorithm are shows in Figure 5, figure 6 figure 7 and figure 8, These figures shows that proposed scheduling algorithm (LPEST) has higher Speedup than other list heuristic scheduling algorithms. Performance metrics in terms of AIR has been also obtained and their relative values have been compared and revealed in. Figure 9, figure 10, figure 11 and figure 12. In these figures AIR has least values for proposed scheduling algorithm (LPEST) it means in this algorithm Average idle time of multiprocessor system has reduced and performance of algorithm should be increases.

After the extensive studies some interesting observation are found i.e. when number of processes and number of processors are equal then all scheduling algorithms will take same time. That means, proposed algorithm is not better in this case than other algorithms. This algorithm is a combination of LPT and EST. Hence it utilizes the feature of both the algorithms and gives better results than LPT and EST. If sequence of incoming processes is already in decreasing order in term of processing time then proposed algorithm and LPT will give same result. However, effect of the proposed algorithm is obvious only when number of processes is quite larger than the number of processors.

7. Conclusion

Heuristic scheduling is widely used in multiprocessor system for find optimal scheduling.. This work surveyed the different classes of heuristic scheduling algorithms and proposed a new one. On the basis of this survey, it is found that the list heuristic scheduling algorithm suitable for both dependent and independent process sets. Several list heuristic scheduling algorithms are implemented and compared. In most of the cases, it is difficult to compare different scheduling algorithms since each has different assumptions (random arrival time, zero arrival time, dependent task set, independent task set), system type (homogeneous, heterogeneous) etc.

Performance of proposed list heuristic scheduling algorithm has been evaluated by changing number of processors and processes under some assumption. For that, some performance parameters such as makespan, speedup and AIR have been obtained by simulation. Simulation result shows that performance of proposed scheduling algorithm in most of the time better than some well known selected list heuristic scheduling algorithms such as LPT, SPT, EST, ECT, EDF, EDD and WSPT. Proposed algorithm-LPEST is more suitable in case of set of independent process where processes arrived at random time.

Reference

[1] C. Mihaila, “Evolutionary Computation in Scheduling”, Ph.D. dissertation”, Babes-Bolyai University, Cluj-Napoca, Romania, 2011.

[2] A. S. Tanenbaum, “Modern Operating Systems”, 3rd Edition, PHI, 2012, pp 23, 536, 560, 565, 581.

[3] T. Casavant and J. G. kuhl, “A Taxonomy of Scheduling in General- Purpose Distributed Computing Systems”, IEEE Trans. on Software Engineering, Vol. 14, No. 2, 1988, pp. 141-154.

[4] T. Hagias and J. Janacek, “Static vs. Dynamic List-Scheduling Performance Comparison”, ActaPolytechnica, Vol.3, No. 6, 2003, pp 16-21.

[5] Chapin, Steven J. and Weismann Jon B, “Distributed and Multiprocessor Scheduling”, Electrical Engineering and Computer Science, Headbook, 2002, paper 40.

[6] Silberschatz, Galvin and Gagne, “Operating System Concepts”, 6rd Edition, John Wiley & Sons, 2003, pp 12, 169.

[7] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. Rinnooy Kan, “Optimization and approximation in deterministic sequencing and scheduling theory: a survey”, Ann. Discrete Math, 1979, pp 287-326.

[8] K. Hwang and F. A. Briggs, “Computer Architecture and Parallel Processing”, International edition 1985, Tata McGraw Hill, 2012, pp 25,459, 460, 468, 590, 591, 592, 596,598.

[9] K. Hwang and N. Jotwani, “Advanced Computer Architecture”, 2nd edition, Tata McGraw Hill, 2011, pp 17, 18, 22, 30, 95.

[10] I. Ahmad, Y. K. Kwok and M.Y. Wu, “Analysis, Evaluation, and Comparison of Algorithms for Scheduling Task Graphs on Parallel Processors”, Parallel Architectures, Algorithms and Networks Symposium, Beijing, China, 1996.

[11] L. Zhou and S. Shi-Xin, “A Self-Adaptive Genetic Algorithm for Tasks Scheduling in Multiprocessor System”, IEEE International Conference on Communications, Circuits and Systems, Guilin, China, 25-28 Jun 2006, pp 2098-2101.

[12] M. I. Daoud and N. Kharma, “An Efficient Genetic Algorithm for Task Scheduling in Heterogeneous Distributed Computing Systems”, Congress on Evolutionary Computation, Vancouver, BC, Canada, 2006, pp 3258-3265.

[13] R. Kaur and R. Kaur, “Multiprocessor Scheduling Using Task Duplication Based Scheduling Algorithms: A Review Paper”, IJAIEM, Vol. 2, No. 4, 2013, pp 311-317.

[14] X. Tang, K. Li, G. Liao and R. Li, “ List scheduling with duplication for heterogeneous computing systems”, Journal of Parallel and Distributed Computing, Vol. 70, 2012, pp 323-329.

[15] G.Q. Liu, K.L. Poh and M. Xie, “Iterative list scheduling for heterogeneous computing”, Journal of Parallel and Distributed Computing, Vol. 65, 2005, pp 654-665.

[16] S.R. Vijayalakshmi and G. Padmavathi, “Multiprocessor Scheduling For Tasks With Priority Using GA”, International Journal Of Computer Science And Information Security, Vol. 6, No. 3,2009, pp 1-8.

[17] M. Maheswaran, S. Ali, H. J. Siegel, D. Hensgen and R. F. Freund, Dynamic Mapping of a Class of Independent Tasks onto Heterogeneous Computing Systems”, Journal of Parallel and Distributed Computing, Vol.59, 1999, pp 107-131.

[18] T. D. Braun, H. J. Siegel, N. Beck, L. L. Boloni, M. Maheswaran, A. I. Reuther, J. P. Robertson and M. D. Theys, “A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems”, Journal of Parallel and Distributed Computing, Vol. 61, 2001, pp 810-837.

[19] M. S. Garshasbi and M. Effatparvar, “Tasks Scheduling on Parallel Heterogeneous Multi-Processor Systems using Genetic Algorithm”, International Journal of Computer Applications, Vol. 6, No.9, 2013, pp 23-27.

[20] TORSCHE Scheduling Toolbox for Matlab User’s Guide (Release 0.4.0), Department of Control Engineering, Czech Technical University in Prague, 2007, pp 1-153.

[21] Gunter Schmidt, “Scheduling with limited machine availability”, Invited Review, Elsevier, European Journal of Operational Research, Vol. 121,2000, pp 1-15.

[22] Y. Maa, C. Chu and C. Zuo, “A survey of scheduling with deterministic machine availability constraints”, Elsevier, Computers & Industrial Engineering, Vol. 58, 2010, pp 199-211.

[23] Arabinda Pradhan, Sukant Kishoro Bisoy and Amardeep das, "A survey on PSO based meta-heuristic scheduling mechanism in cloud computing environment", Journal of King Saud University - Computer and Information Sciences, vol 33, issue 1, January 2021

[24] Essam H.Housseina, Ahmed G.Gadb, Yaser M.Wazerya and Ponnuthurai Nagaratnam Suganthanc, “Task Scheduling in Cloud Computing based on Meta-heuristics: Review, Taxonomy, Open Challenges, and Future Trends”, Swarm and Evolutionary Computation, Volume 62, April 2021.

[25] Reza NoorianTalouki, Mirsaeid Hosseini Shirvani, Homayun Motameni, “A heuristic-based task scheduling algorithm for scientific workflows in heterogeneous cloud computing platforms”, Journal of King Saud University - Computer and Information Sciences, 2021, 1-12.

[26] Wei Hu; Yu Gan; Xiangyu Lv; Yonghao Wang; Yuan Wen, “A Improved List Heuristic Scheduling Algorithm for Heterogeneous Computing Systems “, 2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC), 11-14 Oct. 2020.

[27] Sawsan Alshattnawi, Raneen Khraisat and Hala Majdalawi, “Meta-heuristic Algorithms for Task and Workflow Scheduling in Cloud Computing Environment: An Overview". Acta Scientific Computer Sciences, vol 3,Issue.3 2021, 02-10.

[28] Alhaidari F, Balharith T, Eyman AY (2019) Comparative analysis for task scheduling algorithms on cloud computing. In: 2019 International Conference on Computer and Information Sciences , (ICCIS). IEEE, 2019, pp. 1-6

[29] Kaur A, Kaur B, Singh D, “Meta-heuristic based framework for workflow load balancing in cloud environment”, Int J Inf Technol vol 11 Issue 1, 2019, 119–125