Asian Journal of Information Technology 15 (2): 365-371, 2016
ISSN: 1682-3915

Local Search Based Grid Scheduling Model for Independent Grid Tasks

[1]J. Shanthini and [2]S. Karthik
[1]Department of Information Technology,
[2]Department of Computer Science and Engineering, SNS College of Technology,
Coimbatore, India

Abstract: The grid computing is a flavor of distributed computing. It collects and coordinates resources across the administrative domain. The heterogeneous resources turns grid scheduling as challenging one as it has to deal resources with different resource behavior. So, the scheduler in grid environment has additional responsibilities than any other distributed environment scheduler. This research aims at minimizing the total tardiness of the schedule as it is directly proportionate to the cost of computation. A composite dispatching rule with heuristics to solve the grid scheduling problem is presented here. The first part of the study narrates about the composite dispatching rule I-ATC which is acombinationof Apparent Tardiness Cost (ATC) and Weighted Minimum Shortest Processing Time (WMSPT). I-ATC is an attempt to favor the early submitted jobs with along due date or with relatively low priority. These jobs may incur either early tardiness or lateness by the traditional dispatching rules. Later part of the study deals with the combination of I-ATC with Tabu search heuristics.

Key words: Distributed computing, grid scheduling, grid task scheduling, independent task scheduling, tardiness, early job submission

INTRODUCTION

Grid computing become a new era in computing technologies because of its ability to provide huge computing power and infrastructure less computing facility. It was coined in the middle of 1990's to enable the resource sharing. The idea of grid computing is conceived and established in by Foster and Kesselman (2003) and Shah *et al.* (2011), he defines a grid as" grid computing is coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organization". It enables on-demand resource sharing, a federation of diverse distributed resources and coupling scientific instruments with remote nodes. Grid combines and takes advantages of parallel and distributed computing technologies. These two existing systems are less synchronous and less accurate and incur more communication delay.

Grid computing provides resources with good synchronization and more accurate with less delay. The grid computing manages both parallel and distributed hardware, allows applications to aggregate and use distributed resources on demand. It has the following challenges while managing the distributed resources; they are autonomy, diversity, policy, manageability and controllability.

Literature review: The grid scheduler plays a vital role in optimizing the resources usage in grid netresearch along with the ensuring Quality Of Service (QoS) for the user. There are many researchers recorded their effort in minimizing the makespan, we here concentrated in minimizing the tardiness of the job. The makespan is the amount of time taken to complete the schedule or the time in which the last job is completed. The tardiness is another important aspect in job scheduling where lateness of each job is measured. Maheswaran *et al.* (1999) eleven heuristic algorithms used in distributed environments are compared, these algorithms include Minimum Completion Time (MCT), Minimum Execution Time (MET), K-Percent Best (KPB), Opportunistic Load Balancing (OLB), etc., concluded that min-min gives a relatively better performance. GA give better performance always for any heterogeneity applied, Min-min performs 12% of GA. Matsumura *et al.* (2007) R3Q algorithm is proposed which is an attempt at RR classification which combines the list scheduling with replica round robin.

The task scheduling algorithm disused in concentrates in parameter-sweep application in computational grid (Fujimoto and Hagihara, 2004). This study introduced a Total cycle consume Processor Cycle Consumer (TPCC) represents the total computing power consumed by a parameter-sweep application. A dynamic

Corresponding Author: J. Shanthini, Department of Information Technology, SNS College of Technology, Coimbatore, India

scheduling algorithm RR with TPCC is used. At the beginning of the dynamic scheduling by RR, every processor is assigned exactly, one task, respectively. If some task of the assigned tasks is completed then RR receives the result of the task and assigns one of yet unassigned tasks to the processor.

Tseng *et al.* (2009) is an attempt to combine traditional machine scheduling algorithm Apparent Tardiness Cost with Setups (ATCS) along with MCT. She concluded that the new algorithm give better makespan and tardiness. This is a novel attempt to make use of machine scheduling algorithms in the grid environment. Pfund *et al.* (2008) ATCSR is proposed an algorithm which is an improvement on existing ATCS (Pinedo, 2008). They have incorporated ready time of the jobs, calculated using the uniform probability function with the range [dj-r-s×pj, dj], If [dj-r-s×pj] is <0, a range of [0, dj] is used for ready time generation. Tabu Search (TS) heuristics with hybrid neighborhood

Along with dynamic tenure is proposed by Goswami *et al.* (2011). Where it is applied for single machine total weighted tardiness problem. Eventually, such problems consist of independent tasks with priority, completion time and discrete processing time. The initial solutions were obtained from EDD, WSPT, R&M, R&M+ algorithms.

A local search bases scheduling approach by Goswami *et al.* (2011) is an attempt with simulating annealing which acts as an optimizer for dynamic, schedule base, space shared scheduling policies. It is proven that SA can possibly converge a best solution. On each iteration, a machine is chosen randomly and a ravenous job is removed randomly from that machine's schedule. If the optimizer finds a suitable gap, annealing criterion is computed. The decision is computed which is a function of the difference between the start time, slowdown and standard deviation. If new decision is greater than zero then annealing takes place. Etminani and Naghibzadeh (2007) min-min and max-min meta heuristics are analyzed and concluded that any of these algorithm can excel in a environment depending on the total length of the tasks yet to assign.

In an assumption if the execution environment has single large task and fewer small tasks, the max-min heuristic will execute the large task first and switch over to the smaller tasks concurrently with large task. On other hand, min-min executes all small task first and then the large task. This approach max-min results with better makespan, resource utilization and load balancing. There are many initiatives in using Tabu in the schedule optimization (Chen *et al.*, 2008; Beausoleil, 2011).

The Hybrid Scheduling algorithm (Shah *et al.*, 2011) titled dynamic multilevel Hybrid Scheduling algorithm using median (MHM) computes the time slice dynamically using the process time of all process available in the ready queue. The Hybrid Scheduling algorithm using square root (MHR) calculates the time slice dynamically by calculating the square root of average proceeding time of tasks in the ready queue.

Problem definition: An independent task scheduling was taken up here, adapted to a non preemptive scheduling property. There are m machine employed each equipped with n processing elements (cores) where n∈m. The execution environment is represented by triplet $\alpha|\beta|\gamma$ (Graham *et al.*, 1979):

$$U\left|r_i d_i\right|\sum W_i T_i \tag{1}$$

The first part of triplet denotes heterogeneous execution environment. In the part, the parameters that are important for making the decision is symbolized. The r_i and d_i, the represents arrival time and due date of task i. The objective function is being denoted in the last part which to minimize the weighted tardiness of task i.

MATERIALS AND METHODS

Framework and notation: In this research, we have dealt with independent and non pre-emptive grid task scheduling. Each Gridlet (Gi) submitted to the meta-scheduler has a processing time pi, arrival time ri, due date di and a weight, wi. The completion time of the job i is denoted by Ci and tardiness of job i is denoted by Ti. The completion time of the job i is denoted by Ci and tardiness of job I is denoted by Ti. The lateness of the job is formulated by:

$$L_i = C_i - d_i \tag{2}$$

The value remains negative for early completion and positive for late completion with respect to their due dates.

$$T_i = \max\left(C_i - d_i, 0\right) \tag{3}$$

$$L_i = \max\left(L_i, 0\right) = T_i \tag{4}$$

The make span is time of completion of the very last task in the job queue. It is models as max (C_1 ...

C_n) where C is completion time of tasks. The minimum makespan is an evident for good machine utilization.

The proposed algorithm

I-ATC: The composite dispatching rule Improved ATC (I-ATC) is an attempt to take the advantage of both the algorithm ATC (Foster *et al*., 2001) and WMSPT (Vepsalainen and Morton, 1987). I-ATC algorithm attempts to use the arrival time of the job i to decide the ranking index of i. I-ATC makes a profit out of look ahead parameter to decide the index. We have used the range factor R and due date tightness factor τ to calculate the value of look ahead parameter α. The look ahead parameter α acts as a due date scaling parameter:

$$R = \frac{dmax - dmin}{Cmax} 4.5 + R, \text{ for } R \le 0.5$$
$$\alpha = 6 - 2R, \text{for } R > 0.5 \quad (5)$$

$$I(t) = \left(\frac{\max(d_i - p_i - r_i, p_i)}{\alpha P_{avg}} \right) \quad (6)$$

The I(t) denotes ranking index at time t which is calculated using Eq. 2 and due date tightness is selected using Eq. 1. Initially, the jobs are collected in the schedule queue G. The composite dispatching rule I-ATC presented in Alogrithm 1 arranges the gridlets in the non-incrementing index and allows the gridlets to execute considering arrival time, process time, due date tightness. The due date factor is not changed with the previous position of ATC. Though, the weights of the gridlets are not considered, it does not affect the objective function much. Each gridlet submitted by a grid user is collected in the queue of meta-scheduler. The index values of each grid job are calculated and tasks are arranged in a non-decreasing order with respect to index. The scheduler searches for as suitable resource to execute the gridlet i. If the resource ri is suitable for executing job i, the scheduler checks whether the amount of processing power available is suitable enough to execute the job. If so, the job is dispatched to the corresponding resource queue for execution. The local resource management elements monitor the progress of job and report to the grid level monitors.

Local search based scheduling model: Tabu search is a heuristic method which has a propensity to optimize the problem iteratively to uphold the solution in hand. The Tabu search is a computational method that optimizes a problem iteratively trying to improve a candidate solution. Tabu makes fewer or no assumptions about the problem on hand and searches very large solution space. It uses a combinatorial optimization in which the optimal solution is over a discrete space.

Tabu search is applied to the schedule which is already sequenced with I-ATC. Tabu search based heuristic is tabulated and presented in Alogrithm 1. The incoming jobs are placed in the queue of the meta-scheduler and I-ATC algorithm is applied there. This I-ATC index serves as initial solution S0 to the Tabu search heuristics. The neighborhood lists are created. Swap-based moves are employed, the adjacent jobs are swapped to get the neighborhood structure after evaluating the moves. The Tabu tenure is used to disallow the repeated moves happening in the neighborhood structure. The intensification is applied to explore the search space more thoroughly to get the optimal solution.

Neighbourhood list: The neighborhoods are generated using pair wise swap based moves. The swap move interchanges the location two jobs x and y. On each iteration, two adjacent jobs are swapped and the moves are evaluated on the basis of tardiness function. In each neighborhood N(x) there are n(n-1)/2 swap moves where x denotes current permutation solution. If the tardiness after the job swap is less than the older one and the moves are not in Tabu list that neighborhood is considered as local optima.

Tabu tenure: The Tabu list consists of move that yields the good neighborhood structure. Tabu tenure defines the number of moves to be available in the Tabu list. However, this Tabu list protects the search from revisiting the previous space again and again.

Intensification: Intensification allows the search space to be explored more thoroughly to get an optimal solution. Here, we define intensification through iteration of the search. The new variable called sint is initialized, on every on non-improved iteration this sint in incremented by one after a threshold level, the jobs which involved in non-improvement are again sequenced using I-ATC and replaced in the queue.

Alogrithm 1; local search based model algorithm:

```
//Initialization
        Let S be the schedule
Resource R = {R1, R2....Rn}
Resource Schedule RS= {RS1, RS2....RSn}
G = global queue of unmapped jobs
Job i= {i1, i2, i3, ... , in}
S0 = Initial schedule
Sbest = best solution
ASP = aspiration criteria;
// New Incoming Jobs (I-ATC Index)
For each job i = 0 to n
Calculate the due date range factor
Cmax = max {C1, C2 ... Cn)
R = d max-dmin/Cmax
Calculate I-ATC ranking index
Insert job i in schedule S such that        {0 = i = j}
        //apply Tabu Search
S0 = Initial schedule
```

```
            Let S0 = S;
            for n no of iterations
            Asp = Tardiness of s;
            for (size of S)
            {
    Neighbourhood Nsi = swap (I, j) from S0;
    evaluate NSi
    let local best = Tardiness of NS0
    if (local best >calculate tardiness of
    Nsi and I, j are not in Tabulist)
    then
    localbest = calculate tardiness of NSi
    else (if continuous non-improvement moves)
    sint ++;
    If (sint>5)
    {
    Remove jobs ji-sint till ji)
            (i be iteration number)
                                Reapply I-ATC
    Insert new index in place start from i-sint;
    }           If (local best <asp)
    Asp = local best
    Tabu list = I, j;
    End
//Machine selection
            For each job i ª S
    For each resource Rt ª R K = 1 ... n;
    If job i executable in Rj K = 1 ... n;
    Put i in the RSj for execution
    Break;
    else continue
```

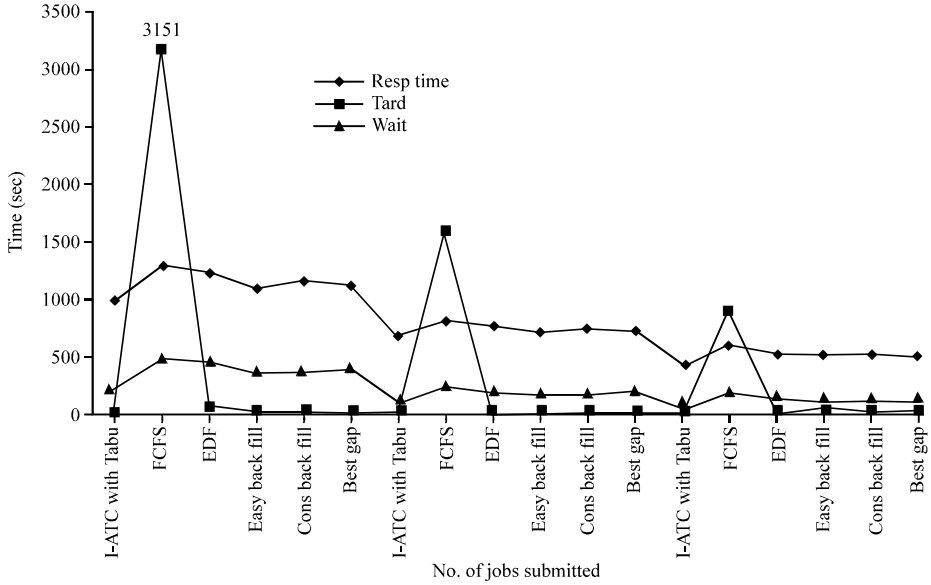
As tabulated the S0 is obtained from the I-ATC ranking index. The tardiness of the current schedule, i.e., S0 is calculated and assigned to aspiration criteria for optimization. For the definite number of iteration, the local optima in calculated.

On each iteration for a total number of jobs in the schedule, the local optimal solution is found. The neighborhood solution is generated by swapping adjacent moves i, j. For every neighborhood solution, the tardiness is calculated and it is compared against the local optima. If the moves are not in Tabu list then the local optima are confirmed. Then, the local optima are compared with the aspiration criteria which is global optima when local optima are lesser than a global one then current local optima are taken as aspiration criteria.

RESULTS AND DISCUSSION

The I-ATC with Tabu search rule for grid environment has been tested under the simulation environment created by Alea 3.0. We have used Intel Pentium core I3 processor, 2.8 Gz and 1 GB RAM processor. We have taken meta centrum research load traces with 5000, 10000 gridlets for simulation. These research load traces were publically available at grid forum for experimental purpose. The meta centrum research load traces contains >90000 traces in it, it also supplied with machine failure scenario which helps to realize a realistic grid environment.

Figure 1 shows the comparison of various algorithms on different job loads. The amount of jobs is like 5000, 10000. The Tabu base I-ATC out performs in all the



Resp_time	1000	1291	1225	1089	1156	1108	680.2	828.1	793.1	714.4	758.1	724.5	450.2	577.2	510.3	456.5	486.1	462.6
Tard	1.56	3151	54.54	33.76	2.69	21.54	4.81	1592	29.23	33.11	5.60	15.20	9.71	889.6	34.07	47.02	17.79	20.98
Wait	223.9	517.6	460.7	389.4	380.1	410.4	113.3	263.1	232.1	201.8	191.3	213.3	86.82	213.1	150.3	123.7	125.3	130.6

Fig. 1: Comparison of algorithms with different work load

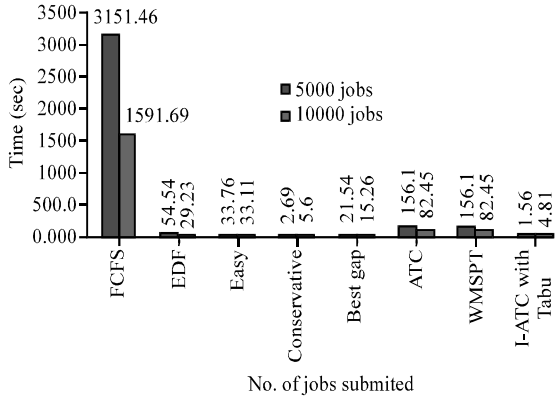


Fig. 2: Tardiness analysis

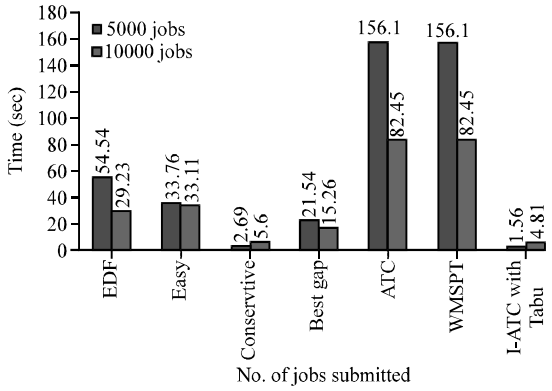


Fig. 3: Detailed analysis of tardiness performance

aspects of the comparison. The makespan does not differ much in our algorithm with other bench marks. The makespan and tardiness are important measures of schedule. The makespan is the measure of the efficiency of the schedule whereas tardiness is the measure of due date performance. The makespan may be identical on a schedule but tardiness will not be so. The tardiness has an impact in deciding the cost of computation. When tardiness reduces the computation cost will also be reduced considerably.

Figure 2 show cases the tardiness analysis. The tardiness of FCFS algorithm is 3151 sec whereas our I-ATC with Tabu could produce 1.6 sec tardiness which is lesser values compared with any other benchmarks.

Figure 3 brings the better visualization of I-ATC with Tabu algorithm, the tardiness is tremendously reduced compared to FCFS and EDF closely competing with conservative backfilling. The chart also shows the comparison between the base algorithms of I-ATC and performance withI-ATC with Tabu. The I-ATC with Tabu algorithm improves tardiness performance by 68% compared with traditional EDF and 60% compared with easy conservative algorithm. The best performer

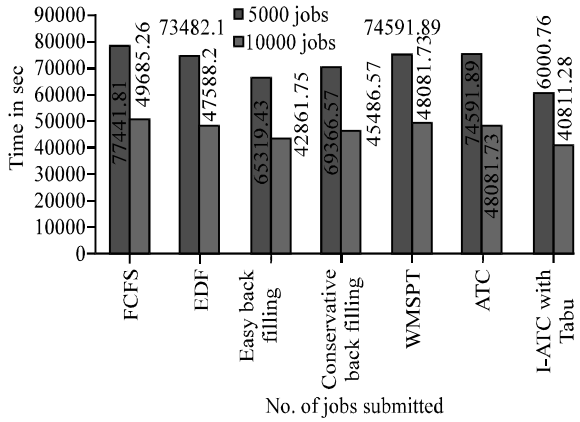


Fig. 4: Flow time analysis

conservative backfilling performs 10% lower than I-ATC with Tabu algorithm. It is observed that on some algorithm performance the tardiness produced on 10000 job load is <5000 job load. This is because of the rapid changes in the grid environment, machine availability and requirements of jobs being submitted.

Figure 4 shows the flow time comparison of the algorithms. The flow time is another important objective function of scheduling algorithms that represents the responsiveness of the schedule and algorithms. As the chart represents the I-ATC with Tabu gives abetter response when compared with other algorithms. As the chart represents I-ATC with Tabu improves responsiveness by 12 and 11% when compared with FCFS and EDF rules. It also improves 11 and 10% when compared with easy and conservative backfilling respectively; its performance is improved by 12% when compared with base ATC and WMSPT algorithms.

Composite rule I-ATC improves the responsiveness of the schedule. It concentrates, the due date, process time and arrival time into account while calculating the ranking index. The rule gives chooses the early submitted job with longer due range. The existing rules estimate the ranking index based on the current time which favours the short jobs and low due range jobs. The FCFS blindly favours the early jobs which make the schedule worse in makespan and tardiness analysis.

The wait time for the jobs in the schedule is compared in Fig. 5. It is shown that this Tabu search based algorithm could produce low wait time for jobs compared with all the other benchmarks. The improvement is 23% when compared with FCFS and 20% when compared with EDF and WMSPT, ATC algorithms. The easy and conservative backfilling produces high wait time for jobs, it is measured 17 and 16% higher wait time when compared with those two algorithms, respectively.

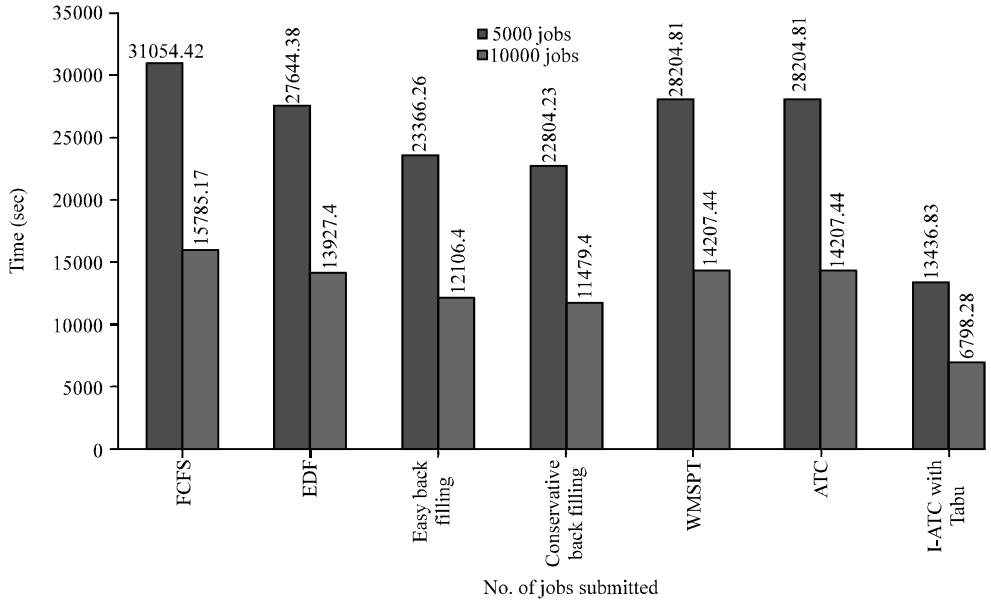


Fig. 5: Wait time analysis

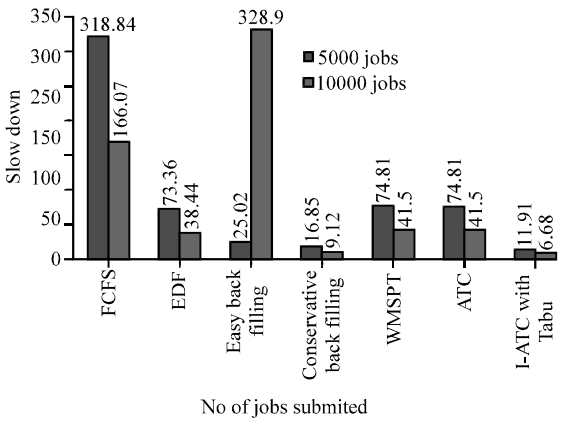


Fig. 6: Slowdown analysis

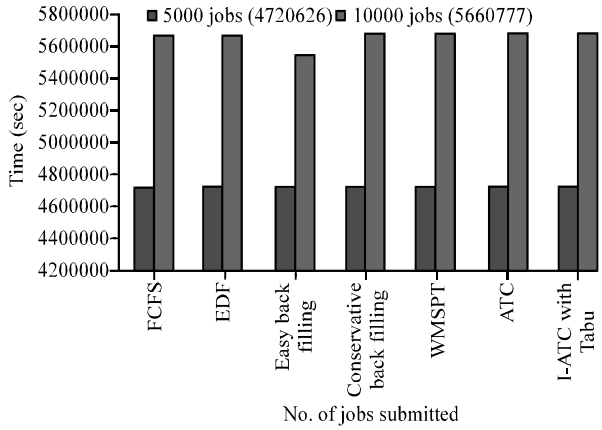


Fig. 7: Makespan analysis

Figure 6 shows slowdown imposed by various algorithms. The slowdown is a performance degradation faced by the cluster being shared by other application. The slowdown factor is a normalized frequency which determines the process speed at therun time. The Tabu implemented I-ATC has imposed a very low slowdown when compared with other algorithms.

Figure 7 show cases the algorithms makespan performance under 5000 and 10000 job submissions respectively. Every algorithm performs equally under 5000 job submission scenario and only easy backfilling produces trivial improvement on submission of 10000 jobs. It shows 10% improvement when compared with other algorithms.

Figure 8 represents the runtime performances of algorithms. Since, Tabu takes several rounds to decide

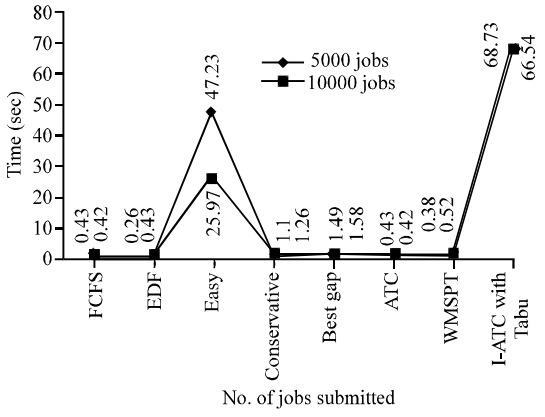


Fig. 8: Analysis of runtime of the algorithm

local optimum then global optima are found among local optima, its run time is always high. This fact is reflected in Fig. 8. Tabu associated algorithm takes higher run time when compared with any other algorithm. It is noted that on varying number of rounds in global optima, selection does not have any impact on run time of thealgorithm. The CPU load does not have any impact on the run time of Tabu associated algorithm. The easy backfilling has higher run time when compared with other algorithms except I-ATC with Tabu algorithm. It shows variations in the run time when the load is varied.

CONCLUSION

The computational results are compared with various benchmark algorithms and excellence of our algorithm is show cased in the results and discussion study.

REFERENCES

Beausoleil, R.P., 2011. A tabu search approach for the weighted tardiness with sequence-dependent setups in one-machine problem. J. Math. Theory Appl., 9: 35-46.

Chen, J.S., J.C.H. Pan and C.K. Wu, 2008. Hybrid tabu search for re-entrant permutation flow-shop scheduling problem. Expert Syst. Appl., 34: 1924-1930.

Etminani, K. and M. Naghibzadeh, 2007. A min-min max-min selective algorihtm for grid task scheduling. Proceedings of the 3rd IEEE/IFIP International Conference in Central Asia on Internet, ICI 2007, September 26-28, 2007, IEEE, Tashkent, Uzbekistan, ISBN: 978-1-4244-1007-1, pp: 1-7.

Foster, I. and C. Kesselman, 2003. The grid2: Blueprint for a New Computing Infrastructure (The Elsevier Series in Grid Computing). 2nd Edn., Morgan Kaufmann, San Francisco, CA, USA..

Foster, I., C. Kesselman and S. Tuecke, 2001. The anatomy of the grid: Enabling scalable virtual organizations. Int. J. High Perform. Comput. Applic., 15: 200-222.

Fujimoto, N. and K. Hagihara, 2004. A comparison among grid scheduling algorithms for independent coarse-grained tasks. Proceedings of the 2004 International Symposium on Applications and the Internet Workshops, SAINT 2004 Workshops, January 26-30, 2004, IEEE, USA., ISBN: 0-7695-2050-2, pp: 674-680.

Goswami, R., T.K. Ghosh and S. Barman, 2011. Local search based approach in grid scheduling using simulated annealing. Proceedings of the 2011 2nd International Conference on Computer and Communication Technology (ICCCT), September 15-17, 2011, IEEE, Allahabad, India, ISBN: 978-1-4577-1385-9, pp: 340-345.

Graham, R.L., E.L. Lawler, J.K. Lenstra and A.H.G. Rinnooy Kan, 1979. Optimization and approximation in deterministic sequencing and scheduling theory: A survey. Ann. Discrete Math., 5: 287-326.

Maheswaran, M., S. Ali, H.J. Siegel, D. Hensgen and R.F. Freund, 1999. Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems. Proceedings of the 8th Heterogeneous Computing Workshop, April 12, 1999, San Juan, USA., pp: 30-44.

Matsumura, Y., K. Ohkura, Y. Matsuura, M. Oiso and N. Fujimoto et al., 2007. Grid task scheduling algorithm R3Q for evolution strategies. Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2007, September 25-28, 2007, IEEE, Singapore, ISBN: 978-1-4244-1339-3, pp: 1756-1763.

Pfund, M., J.W. Fowler, A. Gadkari and Y. Chen, 2008. Scheduling jobs on parallel machines with setup times and ready times. Comput. Ind., 54: 764-782.

Pinedo, M., 2008. Scheduling: Theory, Algorithms and Systems. 3rd Edn., Springer, New York, ISBN 0387789340.

Shah, S.N.M., A.K.B. Mahmood and A. Oxley, 2011. Dynamic multilevel hybrid scheduling algorithms for grid computing. Proc. Comput. Sci., 4: 402-411.

Tseng, L.Y., Y.H. Chin and S.C. Wang, 2009. The anatomy study of high performance task scheduling algorithm for grid computing system. Comput. Stand. Interfaces, 31: 713-722.

Vepsalainen, A.P.J. and T.E. Morton, 1987. Priority rules for job shops with weighted tardiness costs. Manage. Sci., 33: 1035-1047.