IJCAT International Journal of Computing and Technology, Volume 1, Issue 2, March 2014
ISSN : 2348 - 6090
www.IJCAT.org

User Deadline Based Job Scheduling in Grid Computing

[1] **S.Gokul Dev,** [2] **R.Lalith Kumar**

[1]Associate Professor, Department of computer science and engineering, SNS College of engineering, Coimbatore-641035,Tamilnadu.

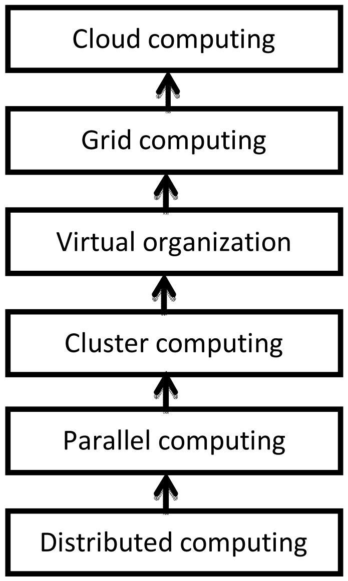
[2]UG Scholar, Department of computer science and engineering, SNS College of engineering, Coimbatore-641035, Tamilnadu.

Abstract - Grid computing is a form of distributed computing that co-ordinates and provides the facility of resource sharing over various geographical locations. Resource scheduling in Grid computing is a complex task due to the heterogeneous and dynamic nature of the resources. Bacterial foraging has recently emerged as a global optimization algorithm for distributed optimization and control. Since virtualization, despite its benefits incurs a performance penalty, which could be significant for systems dealing with uncertainty such as High Performance Computing (HPC) applications where jobs have tight deadlines and have dependencies on other jobs before they could run. We present a novel approach to optimize job deadlines when run in virtual machines by developing a deadline-aware algorithm that responds to job execution delays in real time, and dynamically optimizes jobs to meet their deadline obligations. A novel bacterial foraging based hyper-heuristic resource scheduling algorithm based on deadline has been designed to effectively schedule the jobs on available resources in a Grid environment. The performance of the proposed algorithm has been evaluated with the existing bacterial foraging based hyper-heuristic based scheduling algorithms through the GridSim toolkit. The experimental results show that the proposed algorithm outperforms the existing algorithms by minimizing cost and makespan of user applications submitted to the Grid.

Keywords - **Grid computing, Job scheduling, Meta heuristic, Hyper heuristic, User deadline.**

1. Introduction

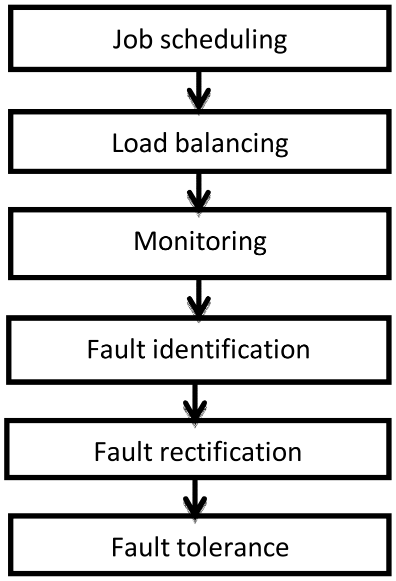
Distributed computing, which has several processors and each processor has separate memory was developed first. If a particular component fails then the job assigned to that component will not be executed. This disadvantage let to the development of the parallel computing when a single memory is shared among all processors and even if a processor fails then the job assigned to that processor can be reassigned to another processor. Since parallel computing can perform only less number of jobs and considering its limitations, cluster computing was developed in which it clusters several parallel computers connecting them with high speed network such as LAN or WAN. A virtual organization in grid computing is the dynamic way of organizing the cluster to define the resource sharing rules. This virtual organization when combined with cluster computing forms the grid computing. The advancement in grid allowing it to perform online by service oriented method is called as cloud computing.



One fine example of grid computing is the project GARUDA which connects nearly 45 institutions in India including all IIT’s. It helps share information’s among these institutions at a faster rate and with very low cost. A grid computing is highly secured and it can either be used within an institute or an organization or a particular community. When grid computing needs to be used for a wider range then it combines several cluster computers to cover those area. A grid computing uses CPU computation and not the service like in cloud computing. Grid technologies promise to tackle complex computational issues. Grid computing permits the virtualization of distributed computing and data resources such as processing, network information measure and storage capacity to form a single system image, granting users and applications seamless access to large IT capabilities. At its core, grid computing relies on associate degree open set of standards and protocols e.g., open Grid Services Architecture (OGSA) that change communication across heterogeneous, geographically distributed environments. When you deploy a grid, it will be to meet a collection of customer necessities. To raise match grid computing capabilities to those necessities, it is useful to keep in mind the reasons for victimization grid computing. This section describes the foremost vital capabilities of grid computing.

2. Various Stages of Grid Computing

The first process in grid computing is to schedule the incoming jobs so that the order in which they should be executed to obtain a better result can be determined. A load balancing process is used to balance the input load to the server based on its capacity. A monitoring is done to ensure that the process that is the jobs and the resources are running smoothly without any glitch. In this paper the monitoring work is done by the portal itself. Once if a fault has been occurred, then that fault needs to be identified and it should be rectified as soon as possible. After the rectification process the method to tolerate further occurrence of fault must be implemented. In this paper only the first stage is handled by a method that gives the best result in scheduling of jobs.



3. Related Works

[1] Describes the uses of gridsim and the functions in which they can be executed and most of the work in implementation is described in this package. [2][3] Are survey papers which help understand the concepts of grid computing and the scheduling strategies. It also describes the difference between grid and various other computing concepts. Various algorithms are compared and their results are graphed and the optimal solution is found. [4] Implements the conception of particle swarm optimization that suggests Meta heuristic search. It selects the optimum solution in an exceedingly single search based on the fitness value by repeating those values into classes like population test and global test. [5] Introduces an idea in genetic rule that is once more a Meta heuristic like choice, cross over, fitness and mutation. [6] Planned suffrage algorithm within which the roles are assigned to the resources and if the roles cannot be executed in that resource than that jobs suffers pretty much. Suffrage value calculated is employed to perform the task. [7] Planned an idea of bacterial foraging optimization which supports the fitness functions. The roles are allotted to the resources of which solely minimum fitness worth relies, deleting the utmost fitness value.

4. Existing Reference Algorithms

4.1 Particle Swarm Optimization (PSO)

Particle Swarm optimization (PSO) simulates the behaviors of bird flocking. PSO learned from the state of affairs and used it to resolve the optimization issues. In PSO, each single resolution may be a "bird" within the search house, called as "particle". All of particles have fitness values which square measure evaluated by the fitness function to be optimized, and have velocities which direct the flying of the particles. The particles fly through the problem house by following the current optimum particles. PSO is initialized with a bunch of random particles (solutions) then searches for optima by updating generations.

4.2 Genetic Algorithm (GA)

Genetic algorithmic rule primarily based meta-heuristics follow the Darwin’s natural selection law i.e. only the fittest will survive. GA a population-based meta-heuristic, was created by John Netherlands and produces consequent generation with the techniques inspired by evolutionary biology, like inheritance, mutation, crossover, and selection. GA considers associate degrees as an organism; therefore better the standard of the answer higher is the survival chance, through crossover (also called recombination) and mutation. GA will escape from the local best to search for the global best. In this paper, we have a tendency to propose a genetic algorithmic rule for job planning to address the heterogeneousness of security mechanism in a procedure grid.

4.3 Suffrage Algorithm (SA)

Suffrage is that a task should be appointed to a certain resource and if it doesn't attend that resource, it will suffer the foremost. For each task, its franchise worth is defined because the difference between its best Minimum Completion Time (MCT) and its second-best MCT. Tasks with high franchise worth take precedence throughout programing. In franchise, the minimum and second minimum completion time for each job square measure found in first step. The difference between these 2 worth's is defined as franchise value. In second step, the task with maximum franchise worth is appointed to corresponding machine with minimum completion time.

4.4 Bacterial Foraging Optimization (BFO)

The Bacteria Foraging Optimization (BFO) algorithmic program was planned by Passino et al.It's a population-based numerical improvement algorithmic program supported hunting behavior of E. coli bacterium. E.coli bacterium features a system that directs its behavior in food hunting. In the hunting theory, the objective of the animal is to look and acquire nutrients during a fashion that energy intake per unit time (E/T) is maximized. Hunting is a process during which a group of bacterium moves in search of food during a region, they decide whether to enter into an attainable food region and seek for a new food region so on to get a high quality of nutrients. The Bacteria Foraging Optimization process consists of 4 main mechanisms: taxis, Swarming, replica and Elimination-dispersal event.

5. Architecture

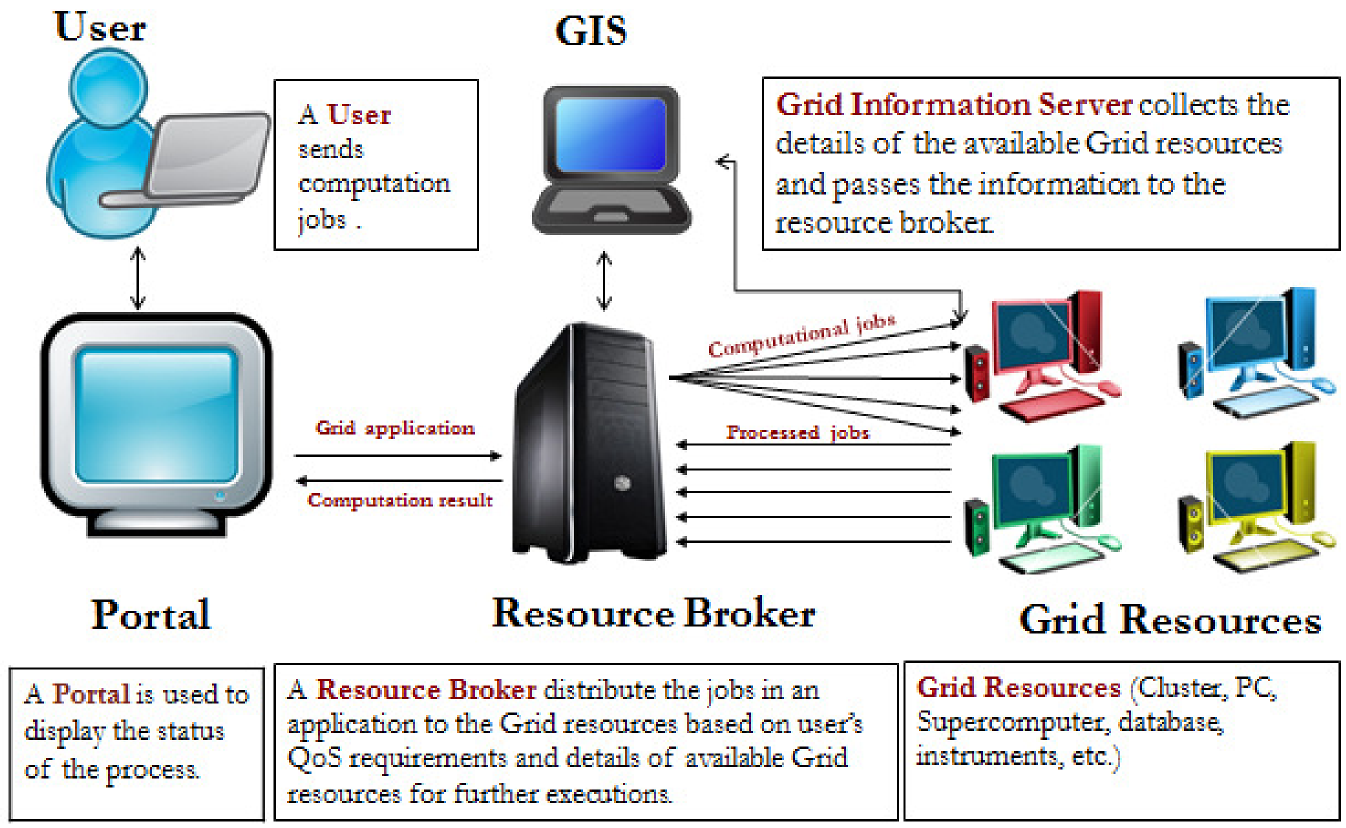

User
A User sends computation jobs .
GIS
Grid Information Server collects the details of the available Grid resources and passes the information to the resource broker.
Computational jobs
Grid application
Computation result
Processed jobs
Portal
Resource Broker
Grid Resources
A Portal is used to display the status of the process.
A Resource Broker distribute the jobs in an application to the Grid resources based on user's QoS requirements and details of available Grid resources for further executions.
Grid Resources (Cluster, PC, Supercomputer, database, instruments, etc.)

Fig.1 Architecture for grid scheduling

The programing in grid environment where the roles are allocated to the resources is shown within the fig. at first the user provides the input of variety of jobs and therefore the variety of resources to the portal which is used to show the standing of the programing process. In order of any system failure or shy resource the user is warned by the portal. The portal sends data to the resource broker which performs the programing process by obtaining the information regarding the resources current standing from Grid information Server (GIS) which periodically updates its information regarding the resources. Once the information's are gathered the resource broker sends the machine jobs to the resources the roles are processed and therefore the computed results are sent back to the broker who again sends the result to the portal from which the user gets the result.

Once job is processed the GIS is updated and therefore the resource broker allocates the resource to ensuing job. During this environment the hyper heuristic method is implemented in which the resource broker performs multiple gathering of data from the GIS and therefore the best one is chosen for the work.

6. Problems of the Existing System

The problem of finding the most effective resources for a selected job is incredibly tedious particularly in meta heuristic technique where just one search is performed however this is overcome in hyper heuristic technique where multiple searches are done that chooses the most effective resources for a job therefore improves the resource utilization. By Hyper heuristic technique, the resource utilization may be achieved nice however the time to settle on the most effective resource is drastically raised therefore it also allocates the job with a resource utilizing high information measure. This can raise the price to execute a job and also raises the hardware execution time.

The hyper heuristic technique maybe smart for low number if inputs however when the jobs and resources are high, then the execution runs out of time and also grid system is usually utilized for a colossal processing of jobs and they don't persist with small works. The client that provides the request needs to do its work with minimum cost however the server that allocates its resources to the job needs to minimize the makespan. By using hyper heuristic technique, mostly the servers are benefited than the client. Therefore the hyper heuristic technique tries to beat the disadvantage by calculative the fitness price and therefore the higher values are eliminated and only] y the lower values are thought-about. On the other hand the time consumption cannot be reduced. This can be overcome by using the point in time primarily based hyper heuristic technique.

7. Proposed Algorithm

We have bestowed a rule referred to as point in time based mostly Hyper Heuristic methodology. In this rule a replacement idea referred to as the point in time is employed, wherever the user requests the utmost completion time for his job and servers apportion the roles to the resources specified, those jobs are executed among the period. There could also be cases once bound jobs cannot be executed among that time amount, thus those jobs can cross the limit fixed for a number of extents. This methodology can improve both resource utilization and cut back the C.P.U. execution time.

7.1 Procedural Steps

1. Each resource to be regular for application's execution has a unique id.
2. Jobs are executed independently.
3. Arrival of jobs for execution of application is random and jobs are placed in a queue of unexpected jobs.
4. The computing capacity/speed of the resources is measured in multiple instructions Per Second (MIPS) as per the standard Performance analysis Corporation (SPEC) benchmark.
5. The processing requirement of job is measured in Million instructions (MI).

8. Tools Used

The GridSim toolkit provides a comprehensive facility for simulation of different classes of heterogeneous resources, users, applications, resource brokers, and schedulers. It can be used to simulate application schedulers for single or multiple administrative domain distributed computing systems such as clusters and Grids. Application schedulers in the Grid environment, called resource brokers, perform resource discovery, selection, and aggregation of a diverse set of distributed resources for an individual user. This means that each user has his or her own private resource broker and hence it can be targeted to optimize for the requirements and objectives of its owner. In contrast, schedulers, managing resources such as clusters in a single administrative domain, have complete control over the policy used for allocation of resources. This means that all users need to submit their jobs to the central scheduler, which can be targeted to perform global optimization such as higher system utilization and overall user satisfaction depending on resource allocation policy or optimize for high priority users.

9. Experimental Results and Comparison

In this paper we have compared the results of various algorithms like Genetic Algorithm (GA), Suffrage Algorithm (SA), Particle Swarm Optimization (PSO), Bacterial Foraging Optimization (BFO) and User Deadline Algorithm (UDA). In these entire algorithms the first step involves the resource allocation process and then initializing the gridsim package. The Grid Information Server (GIS) is created and the routers are initialized. The resources are identified by a unique Resource ID.

In Genetic Algorithm (GA), the resources are grouped and the fitness value is calculated. The resource group is crossover and the fitness value is found. The results of the fitness functions are sorted and the resources with least fitness value are chosen to execute the job.

In a Suffrage Algorithm (SA) the expected execution time and suffrage value are calculated. Based on the suffrage value the jobs are allocated to the resources. In Partial Swarm Optimization (PSO) the pbest value which is the fitness value is calculated and the best fitness value is called as gbest and with that value the resources are allocated to the job.

A Bacterial Foraging Optimization (BFO) is similar to PSO but the fitness function for the same resource and jobs is performed several times in order to find the best one. The number of searches that should be performed I specified by the user. The lowest fitness value is taken.
In User Deadline Algorithm (UDA), the user specifies the tie within which the jobs should be executed. If any jobs cannot be executed within that time then the time may exceed for a certain limit. These five algorithms are compared and the results conclude that UDA gives better results than the rest.

9.1 Requirements

The parameters and their values required for the scheduling process are,

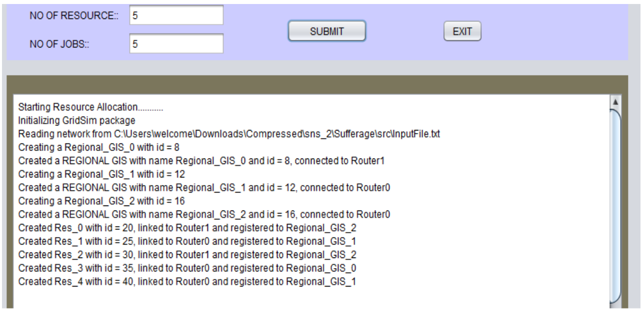
Table 1 Parameter requirements

PARAMETERS	VALUES
Gridlet size	100000
Gridlet length	42000000
Cost per second	1
No. of routers assigned	2
No. of GIS assigned	3
No. of grid users	7

The fitness function is defined by the amount of instructions transferred per second and the gridlets are the packages that contain all the information about the job and the resources. The processing rate of gridlets is measured in Million Instructions (MI). The number of resources and the jobs are user defined. The file size is not limited; they can be of any size.

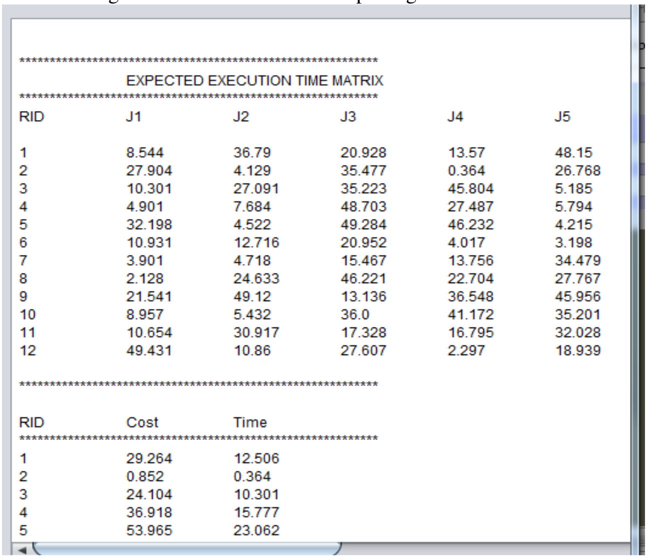
The various existing algorithms are compared by defining the number of resources and jobs to be 5 and their results are tabulated.

9.2 Screen Shots of Bacterial Foraging Algorithm



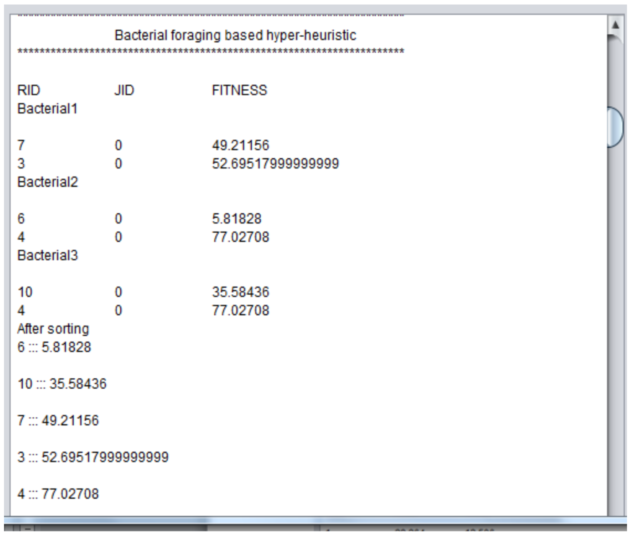
```
Starting Resource Allocation..........
Initializing GridSim package
Reading network from C:\Users\welcome\Downloads\Compressed\sns_2\Sufferage\src\InputFile.txt
Creating a Regional_GIS_0 with id = 8
Created a REGIONAL GIS with name Regional_GIS_0 and id = 8, connected to Router1
Creating a Regional_GIS_1 with id = 12
Created a REGIONAL GIS with name Regional_GIS_1 and id = 12, connected to Router0
Creating a Regional_GIS_2 with id = 16
Created a REGIONAL GIS with name Regional_GIS_2 and id = 16, connected to Router0
Created Res_0 with id = 20, linked to Router1 and registered to Regional_GIS_2
Created Res_1 with id = 25, linked to Router0 and registered to Regional_GIS_1
Created Res_2 with id = 30, linked to Router1 and registered to Regional_GIS_2
Created Res_3 with id = 35, linked to Router0 and registered to Regional_GIS_0
Created Res_4 with id = 40, linked to Router0 and registered to Regional_GIS_1
```

Fig. 2 Resource allocation and package initialliation



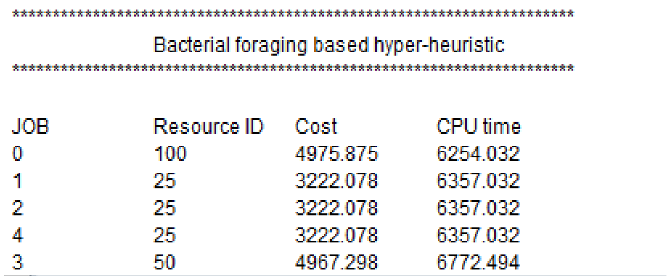
```
************************************************************
           EXPECTED EXECUTION TIME MATRIX
************************************************************
RID        J1         J2         J3         J4         J5

1          8.544      36.79      20.928     13.57      48.15
2          27.904     4.129      35.477     0.364      26.768
3          10.301     27.091     35.223     45.804     5.185
4          4.901      7.684      48.703     27.487     5.794
5          32.198     4.522      49.284     46.232     4.215
6          10.931     12.716     20.952     4.017      3.198
7          3.901      4.718      15.467     13.756     34.479
8          2.128      24.633     46.221     22.704     27.767
9          21.541     49.12      13.136     36.548     45.956
10         8.957      5.432      36.0       41.172     35.201
11         10.654     30.917     17.328     16.795     32.028
12         49.431     10.86      27.607     2.297      18.939

************************************************************

RID        Cost       Time
************************************************************
1          29.264     12.506
2          0.852      0.364
3          24.104     10.301
4          36.918     15.777
5          53.965     23.062
```

Fig. 3 Calculating expected execution time and best execution time



```
******************************************************************
            Bacterial foraging based hyper-heuristic
******************************************************************

RID         JID          FITNESS
Bacterial1

7           0            49.21156
3           0            52.69517999999999
Bacterial2

6           0            5.81828
4           0            77.02708
Bacterial3

10          0            35.58436
4           0            77.02708
After sorting
6 ::: 5.81828

10 ::: 35.58436

7 ::: 49.21156

3 ::: 52.69517999999999

4 ::: 77.02708
```

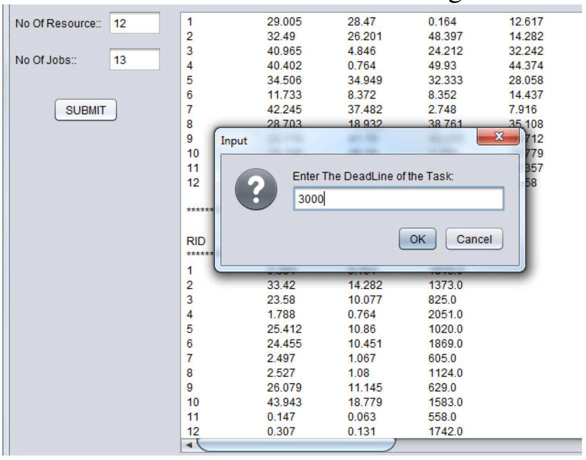
Fig. 4 Various searches performed with job 0 on resources and sorting those results



```
*****************************************************************************
           Bacterial foraging based hyper-heuristic
*****************************************************************************

JOB          Resource ID     Cost          CPU time
0            100             4975.875      6254.032
1            25              3222.078      6357.032
2            25              3222.078      6357.032
4            25              3222.078      6357.032
3            50              4967.298      6772.494
```

Fig.5 Calculating cost and CPU time

9.3 Screen shorts of User Deadline Algorithm

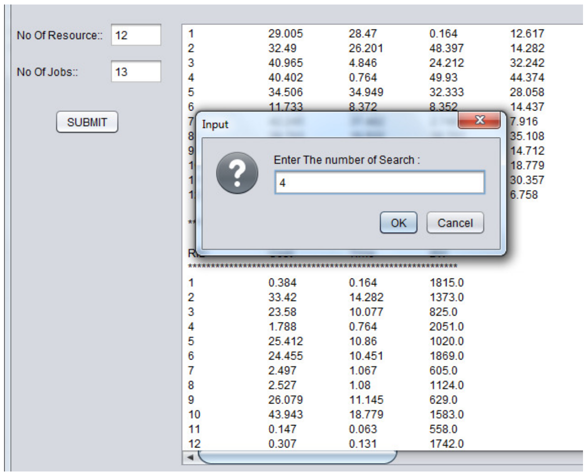


No Of Resource:: 12

No Of Jobs:: 13

```
1          29.005     28.47      0.164      12.617
2          32.49      26.201     48.397     14.282
3          40.965     4.846      24.212     32.242
4          40.402     0.764      49.93      44.374
5          34.506     34.949     32.333     28.058
6          11.733     8.372      8.352      14.437
7          42.245     37.482     2.748      7.916
8          28.703     18.932     38.761     35.108
9                                            712
10                                           779
11                                           357
12                                           58

******
Enter The DeadLine of the Task:
3000
RID
******
1
2          33.42      14.282     1373.0
3          23.58      10.077     825.0
4          1.788      0.764      2051.0
5          25.412     10.86      1020.0
6          24.455     10.451     1869.0
7          2.497      1.067      605.0
8          2.527      1.08       1124.0
9          26.079     11.145     629.0
10         43.943     18.779     1583.0
11         0.147      0.063      558.0
12         0.307      0.131      1742.0
```

Fig.6 User Deadline value as input

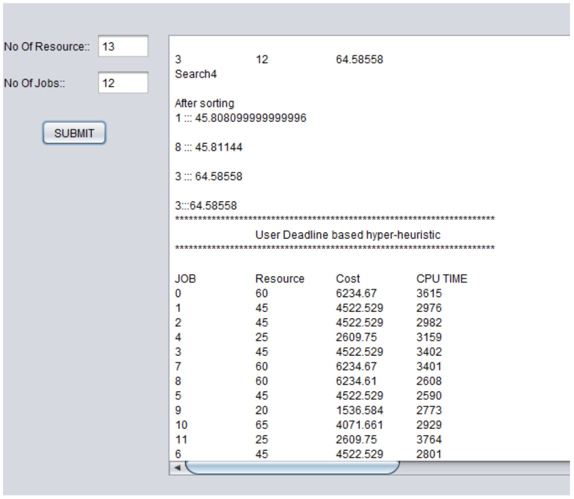


No Of Resource:: 12

No Of Jobs:: 13

```
1          29.005     28.47      0.164      12.617
2          32.49      26.201     48.397     14.282
3          40.965     4.846      24.212     32.242
4          40.402     0.764      49.93      44.374
5          34.506     34.949     32.333     28.058
6          11.733     8.372      8.352      14.437
7                                            7.916
8                                            35.108
9                                            14.712
1                                            18.779
1                                            30.357
1                                            6.758

**
Enter The number of Search :
4
R
****************************************************************
1          0.384      0.164      1815.0
2          33.42      14.282     1373.0
3          23.58      10.077     825.0
4          1.788      0.764      2051.0
5          25.412     10.86      1020.0
6          24.455     10.451     1869.0
7          2.497      1.067      605.0
8          2.527      1.08       1124.0
9          26.079     11.145     629.0
10         43.943     18.779     1583.0
11         0.147      0.063      558.0
12         0.307      0.131      1742.0
```

Fig.7 Number of Searchs to be done



No Of Resource:: 13

No Of Jobs:: 12

```
3          12         64.58558
Search4

After sorting
1 ::: 45.808099999999996

8 ::: 45.81144

3 ::: 64.58558

3:::64.58558
**********************************************************************
           User Deadline based hyper-heuristic
**********************************************************************

JOB        Resource   Cost       CPU TIME
0          60         6234.67    3615
1          45         4522.529   2976
2          45         4522.529   2982
4          25         2609.75    3159
3          45         4522.529   3402
7          60         6234.67    3401
8          60         6234.61    2608
5          45         4522.529   2590
9          20         1536.584   2773
10         65         4071.661   2929
11         25         2609.75    3764
6          45         4522.529   2801
```

Fig.8 calculating cost and CPU time for User Deadline Algorithm

9.4 CPU Time

Table 2Time Comparison Table

PARAMETER	GA	SA	PSO	BFOH	UDHH
J_1	7264	7316	7178	6411	3408
J_2	7156	7316	7717	6423	2666
J_3	8228	7934	7717	7171	3119
J_4	8255	8188	7801	7181	3700
J_5	9264	8972	7655	6312	2977

9.5 Time Graph

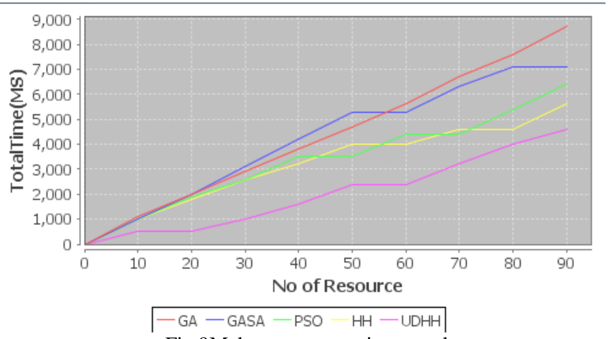


Fig.9Makespan comparison result

9.6 Cost

Table 3Cost Comparison Table

PARAMETER	GA	SA	PSO	BFOHH	UDHH
J_1	6585	203794	8928	5864	4080
J_2	6585	150819	7815	3818	6511
J_3	7123	181416	8155	5161	3429
J_4	4792	181416	5762	4827	5391
J_5	7895	169187	9282	6177	5427

9.7 Cost Graph

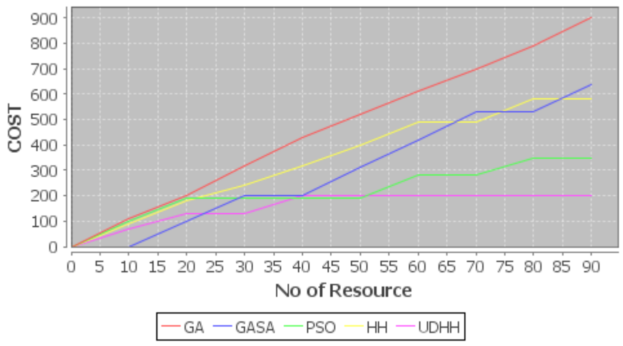


Fig.10 Cost comparison result

Thus from the various results derived we can conclude that hyper heuristic gives much better results than the other algorithms since it uses multiple search strategies. Even the cost is reduced drastically and the bandwidth occupied is also very less. They transfer bits at a very high speed when compared to the other algorithms. PSO ranks the second position in terms of both cost and time. The rest of the algorithms like the genetic algorithm (GA), suffrage algorithm (SA) are far behind and there is no proper resource allocation to the jobs.

10. Conclusion and Future Work

Grid resource scheduling rule outperforms the hybrid heuristics in all the cases. The projected rule not solely minimizes cost however it conjointly minimizes the makespan. Thus from the results of various scheduling algorithms are compared and Bacterial algorithm is found to be the best one and this can be further enhanced by using user deadline hyper heuristic where the user specifies the time to complete the jobs. The algorithm for User Deadline based Hyper Heuristic gives the best result than the Bacterial Foraging Optimization where the users request to complete the process within their specified time. This can be verified from the table and graph generated.The main disadvantage of USD is low reliability where there is a chance of deadlock occurrence which will be implemented in the future work.

References

[1] R.Buyya, M.Murshed,"Gridsim: a tool kit for the modeling and simulation of distributed resource management and scheduling for grid computing concurrency and computation", 2002.

[2] G.Jaspher W. Kathrine, MansoorIlaghi U, "job scheduling algorithms in grid computing survey", International journal of engineering research and technology (IJERT), ISSN: 2278-0181, vol. issue 7, September 2012.

[3] Rakesh Sharma, VishnukantSoni, Manoj Kumar Mishra, PrachetBhuyan, "A survey of job scheduling and resource management in grid computing", world academy of science, engineering and technology 40 2010.

[4] T.R. Srinivasan, R. Shanmugalakshmi, "Neural approach for resource selection with PSO for grid scheduling", International Journal of computer application volume no 11, September 2012.

[5] R. Kashyap, D.P. Vidyarthi, "Security driven scheduling model for computational grid using Genetic Algorithm", WCECS 2011, October 19-21, 2011, San Francisco, USA.

[6] Kamaligupta, manpreetsingh, "Heuristic based task scheduling in grid", International journal of engineering and technology (IJET), ISSN: 0975-4024, vol. 4no 4 Aug-Sep 2012.

[7] Rajni, InderveerChana, "Bacterial foraging based Hyper-heuristic for resource scheduling in grid computing", future generation computer systems, Elsevier, 14 September 2012.

Author's Profile

Prof. S. Gokuldev obtained his Bachelor's degree and Master's degree in Computer Science from Bharathiar University in the year 1999 and 2003, He obtained his Master of Philosophy in Computer Science from Bharathidasan University in 2004, Master of Engineering in Software Engineering from Anna University, Chennai in 2008 and currently pursuing PhD in Anna University, Chennai. He has around ten years of academic and teaching experience. Presently working as an Associate Professor in Department of Computer Science & Engineering, SNS College of Engineering, Coimbatore. His areas of interest are Distributed and Grid Computing. He had published around 6 papers in international journals and more than 15 papers in few reputed International and National level conferences.

Lalith Kumar is an UG Scholar in Computer Science and Engineering from SNS college of Engineering, affiliated to Anna University, India. His area of interest lies in grid and cloud computing and scheduling in grid computing. His research interest includes heuristic algorithms in grid scheduling.