New Heuristic for Scheduling of Independent Tasks in Computational Grid

Anand K Chaturvedi, Rajendra Sahu

ABV- Indian Institute of Information Technology and Management, Gwalior India
chaturvedi101@gmail.com, rsahu@iiitm.ac.in

Abstract

The Scheduling of tasks on heterogeneous grid resources is known to be a NP-complete problem; therefore, to get a near optimal solution within finite duration, heuristics/meta-heuristics are used for task scheduling instead of exact optimization methods. In this paper, we proposed a new heuristic method for scheduling of independent tasks on heterogeneous grid resources and compared the result with ten other scheduling heuristics. Benchmark instances of Expected Time to Complete (ETC) model, suggested by Braun [3] are used to test the proposed heuristic. New heuristics give best Makespan as well as Flow-time values under much prevalent consistent resource and task heterogeneity conditions.

Keywords: *Computational Grid, Scheduling Heuristic*

1. Introduction

Connecting geographically distributed computational resources such as PCs, workstations, clusters, servers, and super computers, Computational Grids have emerged as a next generation computing platform for solving large-scale problems in academics, research and industry. Main step of Grid Resource Management is the task scheduling on grid resources. Grid resource management involves dealing with three classes of stakeholders - end users making use of grid resources, owners of resources, and grid administrators. Each class of stakeholders has their own perspective and preferences, which result in different, often contradictory, criteria for scheduling). To increase the level of satisfaction of these stakeholders grid management system must use the scheduling heuristic, which provides compromise solution (i.e. a compromise schedule) using the many conflicting objectives.

Scheduling of task on heterogeneous grid resources is known to be a NP-complete problem; therefore, to get a near optimal solution within finite duration, heuristics/meta-heuristics are used instead of exact optimization methods. In some real-world situations, the meta-heuristic methods are too difficult or inappropriate, for example in fully automated systems (where we cannot tune parameters manually) or where the execution time should be very short, or for extremely large problems, etc. Therefore using heuristics in such situations is an appropriate solution [7].

In this paper, we proposed a new heuristic method for scheduling of independent tasks on heterogeneous grid resources and compared the result with ten other scheduling heuristics. Twelve different benchmark instances of Expected Time to Complete (ETC) model, suggested by Braun [3] for grid simulation experiment is used to test proposed heuristic. New heuristics gave result better than other under consistent resource and task heterogeneity conditions.

2. Grid Scheduling Problem

Computational grid are complex combinations of Heterogeneous Computing resources, software and network components. Scheduling in grid involves mapping of each task on any one of the available computational resource in the grid. Each task needs only one computational resource for its complete processing. At a time, one resource can process only one task mapped on it. Tasks are processed on a resource one by one without preemption. Processing of a task, once start, cannot be stop or postponed until completion. Computational resources work in parallel and independent manner. Resources are assumed fully reliable and there is no failure of resource while processing group of tasks.

Present study considered only static scheduling, according to which the computational load of each task and the computing capacity of each resource in the grid is available a priori. This formulation is practically applicable as the computing capacity and computational needs can be known from user specifications, historic data or prediction. This aspect of grid scheduling is not explored in present research.

Let $T = \{T_1, T_2, ..., T_n\}$ denote the set of n tasks that are independent of each other to be scheduled to m resources $M = \{M_1, M_2,...,M_m\}$ within the Computational Grid environment. At time t=0 all m resources and all n independent tasks are available for processing. Resources take time to process task, which depends upon length of task as well as processing speed and suitability of the resource for that task. An Expected Time to Compute (ETC) matrix, of size (n x m), is used to store expected processing time of n tasks on m computational resources. Thus ETC (j, i) is expected time to compute task T_j on resource M_i. Completion time C_j of task T_j is

$$C_j = W_{i,j} + ETC(j,i) \quad (1)$$

Where $W_{i,j}$ is the work load of Resource M_i before processing of task T_j

The grid-scheduling problem is multi-objective in nature. Several performance measures and optimization criteria can be considered to evaluate the quality of a given schedule and overall grid system performance. Two most common objectives , Makespan and Flow-time used in this study are :

2.1. Makespan

Most popular optimization criterion is minimization of Makespan i.e. the finishing time of the latest task. Makespan measures the throughput of grid system. It can be defined as:

$$Cmax = \max \{C_j , j=1,\ldots,n\} \quad (2)$$

2.2 Flow time

Flow-time is the sum of the finishing times of tasks. Flow time measures the Quality of Service of the grid system. It can be defined as:

$$F = \sum C_j , j=1,\ldots,n \quad (3)$$

Flow time is minimum when tasks are processed in ascending order of processing time on a particular grid resource. It is followed while calculating Flow-time.

3. Heuristics for Grid Scheduling

There are several heuristics for grid scheduling. This section describes ten popular heuristics, which are used for comparison purpose in this study. For details please refer [6], [2], [4], [3], [9], [10], [11], and [7].

3.1. Opportunistic Load Balancing (OLB)

Heuristic OLB selects earliest idle resource for task mapping without considering the task's execution time on the selected resource. If two or more resources are idle then resource is selected arbitrarily. In this method time required for scheduling is less and it keeps almost all the resources busy at all possible time [2], [4], [3] and [11]. Resulting schedule is not optimal. Problem is solvable using OLB in O(nm) time.

3.2. Minimum Execution Time (MET)

In this method minimum execution time is used to assign the task without considering the resource availability. Task is assigned to the resource on which it can be executed in minimum time [2], [4], [3] and [11]. Allocating task without considering resource availability results in load imbalance on grid resources. Heuristic MET can solve problem in O(nm) time.

3.3. Minimum Completion Time (MCT)

In MCT heuristic task is assigned to the resource that gives minimum completion time C_j (ready time of resource + task execution time on the selected resource) for the task [6], [2], [3] and [11]. Allocating task in this manner may result in execution of task on less faster grid resources. Problem is solvable using MCT in O(nm) time.

3.4. Switching Algorithm (SA)

This heuristic of scheduling combines the best features of MCT and MET methods of scheduling. Method tries to use better load balancing of MCT and execution on fastest resource of MET. Here the idea is to first use the MCT till a threshold of balance is reached followed by MET which creates the load unbalance by assigning tasks on faster resources. Here MCT and MET are used in cyclic manner [9] and [11]. Problem is solvable using OLB in O(nm) time.

3.5. k-Percent Best (kPB)

kPB heuristic also attempts to combine the best features of MCT and MET simultaneously but not in cyclic manner [9] and [11]. While assigning the tasks only k percentage of best resources, based on task execution time, are considered. Out of the k percent of best resources, for a task a resource is selected that gives minimum completion time for that task. For k=100 this method act similar to MCT while for k=100/m this method act similar to MET. Method has a serious drawback, while allocating tasks if k percentage resources are busy but other resource are free, in such situation kPB still allocates task only to one of those k percent best resources. This results in more idleness of resources in the generated schedule. Problem is solvable using MCT in O(nmlogm) time.

3.6. Work Queue (WQ)

Work Queue is a very simple heuristic of task allocation. Tasks are randomly selected from the list of unassigned tasks and assigned to the resource with minimum workload. Task assignment repeated in similar manner till list of unassigned tasks get exhausted [5].

3.7. Min-min

In Min-min method, completion time C_{ji} of all unassigned tasks ($1 \leq j \leq n$) on all the available resources ($1 \leq i \leq m$) is used to calculate the minimum completion time (MCT_j) of task Tj on resource Mi*. Then task which gives minimum of MCT_j is identified T*={ min(MCT_j) for ($1 \leq j \leq n$) } and assigned on the resource M*. Subsequently the task T* is removed from the list of unassigned task and workload of resource M* is updated. Above procedure is repeated till unassigned task list get exhausted [6], [2], [4], [3] and [10]. Problem is solvable using Min-min in $O(n^2m)$ time.

3.8. Max-min

In this method completion time C_{ji} of all unassigned tasks ($1 \leq j \leq n$) on all the available resources ($1 \leq i \leq m$) is used to calculate the minimum completion time (MCT_j) of task Tj on resource Mi*. Then task which gives maximum of MCT_j is identified T*={ max(MCT_j) for ($1 \leq j \leq n$) } and assigned on the resource M*. Subsequently the task T* is removed from the list of unassigned task and workload of resource M* is updated. Above procedure is repeated till unassigned task list get exhausted [6], [2], [4], [3] and [10]. Heuristic Max-min can solve problem in $O(n^2m)$ time**.**

3.9. LJFR-SJFR

Largest Job (task) on Fastest Resource – Shortest Job (task) on Fastest Resource (LJFR-SJFR) method allocates largest task to fastest resource to reduce the makespan and allocates smallest task to fastest resource to reduce the flow time of the schedule [1] and [10].

In the first stage m number of tasks are assigned on m number of unallocated resources using steps of Max-min method as described above.

In second stage, remaining (n-m) tasks are assigned using min-min method and max-min method alternatively i.e. smallest task on fastest resource followed by largest task on fastest resource. Procedure is followed till the list of unassigned tasks get exhausted. Heuristic LJFR-SJFR can solve problem in $O(n^2m)$ time.

3.10. Suffrage

Suffrage for a task is the difference between second minimum completion time and first minimum completion time for that task [9] and [10]. Suffrage method tries to allocate most suffered tasks in terms of expected completion time first. In this method suffrage is calculated for all unassigned tasks and the task which has maximum suffrage value is assigned to the resource which gives first minimum completion time. Then task is removed from unassigned task list, resource workload updated and above cycle of

task allocation repeated till list of unassigned tasks get exhausted. Problem is solvable using Suffrage in $O(n^2m)$ time.

4. Proposed Heuristics:

Scheduling is the main step of grid resource management. Satisfaction of all stakeholders of computational grid depends upon quality of schedule generated by resource scheduler. Theoretically, we may get the best schedule if we are able to assign the tasks to the resource on which it can be processed in minimum possible processing time. MET heuristic method, as discussed earlier, do this but assigning tasks according to Minimum Execution Time creates large load imbalance in the computational grid. Some resources become over burdened while some become idle. This is the major drawback of MET resulting in very larger value of Makespan and Flow time for the generated schedule.

Proposed heuristic has two stages. In the first stage, tasks are assigned to resource which takes minimum execution time. Here tasks are assigned without considering the availability or load imbalance on resources in the grid similar to MET method. To remove the load imbalance, tasks are transferred from maximally loaded resource to minimally loaded resource in the second stage. A potential task for transfer must be loaded on maximally loaded resource and if it is transferred to minimally loaded resource then penalty should be minimum. Further task completion time should not be much larger than the average resource completion time .

Detailed procedure followed in proposed heuristic is enumerated in algorithm given below. In step 1-2 tasks are assigned to the resources taking minimum time in executing that task. Step 3-4 identify resources in increasing order of execution time for each unassigned task. For any job j the Sort Index SI[j, k], where $1 \leq k \leq m$, stores the identification number of resource in increasing order of processing time. For the task j sort index SI[j, 1] gives identification of the fastest resource and SI[j, m] gives identification of slowest resource. Step 5 is a loop which repeats step 6-9 for maximum n repetitions or till a stopping condition is satisfied (which ever is earlier). Here n is the number of task to be assigned on m number of resources. Stopping condition is true if there is no improvement in makespan or flow-time for large number of repetitions.

Step 6 identifies maximum loaded resource (A) and minimum loaded resource (B) and completion time of last task assigned to them C_A and C_B respectively. To reduce the imbalance of load generated due to assignment of task as per MET here maximum loaded and minimum loaded resources are identified and a task, as identified in step 7, is transferred from maximum loaded resource to minimum loaded resource. In the worst case grid scheduling problem is solvable using proposed heuristic in $O(n^2m)$ time.

5. Test Problem

For fair comparison of different scheduling heuristic, ETC model of benchmark simulation experiments by Braun [3] is used in our study. This model is based on Expected Time to Complete (ETC) matrix for 512 tasks and 16 machines (resources).

Several researchers considered twelve possible scenarios of grid scheduling based on task heterogeneity, resource heterogeneity, and consistency combinations as mentioned in Table 1. Braun suggested twelve benchmark instances in the form of twelve ETC matrices (512x16) for these combinations of task heterogeneity, resource heterogeneity, and consistency.

Resource heterogeneity represents the variation of execution times for a given task across the resources. An environment having similar resources will be represented by low resource heterogeneity, while high resource heterogeneity represents computing resources of different type and power. Task heterogeneity represents the degree of variation among the execution times of tasks for a given resource. In High task heterogeneity different types of applications are submitted to execute in the system, from simple programs to large and complex tasks which require large CPU times to be performed. An ETC matrix is considered consistent if a resource m_1 executes task j faster than resource m_2, then m_1 executes all the tasks faster than m_2. Inconsistency is there if a resource is faster for some tasks and slower for others. An ETC matrix is considered semi-consistent if it contains a consistent sub-matrix.

Instances of ETC matrix are labeled as u_x_yyzz. Different symbols used in this label are:

- Symbol u means uniform distribution which is used in generating the ETC matrix.
- Symbol x means the type of consistency

  (c – consistent, i – inconsistent and s– semi-consistent).
- Symbol yy indicates the heterogeneity of the tasks

  (hi means high, and lo means low).
- Symbol zz indicates the heterogeneity of the resources

  (hi means high, and lo means low).

These benchmark instances are considered one of the most demanding for the scheduling problems in heterogeneous grid computing environment by the large number of researchers. The many researchers have used these instances for testing their scheduling Heuristics [3], [10], [11], [8] and [7].

Algorithm: Proposed Heuristic

1. For task j =1 to n
   a. Assign tasks j on the resource i which takes minimum execution time
2. End for
3. For task j =1 to n
   a. For k =1 to m
      i. For task j identify resource in increasing order of processing time & store this in SI[j, k]

   // for a task j Sort index SI[j, k=1] is fastest SI[j, k=1] SI[j, k=m] is slowest resource
   b. End for
4. End for
5. Do : while ((# of repetition < =n) AND (Not Stopping condition))
6. Find maximum and minimum loaded resource (A) & (B) and their last completion time C_A & C_B
7. For k =1 to m
   a. For task j =1 to n
      i. If task j is not assigned on A then Go to step 7a

ii. If for task j resource SI[j, k] is not B then Go to step 7a
iii.
iv. If $(C_B + ETC[j, B]) > \left(\frac{C_A + C_B}{2}\right)$ *then go to step 7a*
v. $\text{penalty} = \frac{ETC[J, B] - ETC[J, A]}{ETC[J, A]}$
vi. Identify task (J_{min}) with minimum penalty

b. End for
8. End for
9. Transfer task J_{min} from resource A to resource B
10. End Do : Go to 5
11. Stop

6. Computational Results

In this section we present the results obtained for scheduling of twelve instances of test problem using the ten existing heuristics and one proposed heuristic for grid scheduling.

6.1. Computation Program

A computer program in C++ language is developed for ten existing and proposed heuristic methods mentioned above. Program can produces schedule for mapping of tasks to available resources and calculate the objectives based on the ETC matrix supplied to it. Twelve different ETC matrix suggested by [3] for different scenarios mentioned in Table I is used by the computer program. Program is executed on Intel (R) Core 2 Duo CPU T5550 @ 1.83 GHz, 1.83 GHz with 2 GB RAM and Window Vista operating system. Results obtained are discussed as follows.

6.2. Result

Makespan and Flow-time obtained using all heuristic for all twelve instances of grid scheduling problem are shown in Table 3 and Table 4.

Values of objectives obtained by the ten existing and one new heuristic are compared for twelve different scenarios of grid scheduling. Based these values of Makespan and Flow-time heuristics are ranked. Top three heuristics for Makespan and Flow-time for different scenario are given Table 2.

Out of twelve scenarios, proposed method gives best makespan in four scenario, second best in four scenario and third best in four scenarios. Suffrages and Min-min gives the rank I, II and III in five, two & five and three, six & three scenarios respectively.

Out of twelve scenarios proposed method gives best flow-time in the four scenario, second best in one scenario and third best in six scenarios. Min-min gives the rank I, in seven and rank II in five scenarios. Suffrage gives No rank I, rank II in six and rak III in six scenarios.

Makespan as well as flow- time result is better especially under consistent condition for different combinations of task heterogeneity and resource heterogeneity.

7. Conclusion & Future Work

There is no single heuristic that is excellent in satisfying fully all the conflicting expectations of different class of stakeholders of computational grid.

Consistent scenario is more prevalent and obvious scenario in Computing Grid environment. Task processing time has strong negative correlation with speed of computing resource. Usually if a processor is then it takes lesser time for processing all the asks whether large or small. Proposed heurist is giving best values of makespan as well as flow-time i.e. is more suitable under consistent task and resource heterogeneity for both the objectives Makespan as well as Flow-time. Min-min is better in providing Flow-time while Suffrage provides better Makespan

Heuristics can be segregated into two groups. One group is suitable for immediate mode consisting OLB, MCT, MET, SA, kPB, and WQ and another is suitable for batch mode scheduling consisting Max-Min, LJFR-SJFR, Suffrage, Min-Min, and proposed heuristic.

Table 1: Heterogeneity and consistency combinations in the ETC model

Heterogeneity		Consistency		
Task	Resource	Consistent	In-consistent	Semi-consistent
high	High	u_c_hihi	u_i_hihi	u_s_hihi
High	Low	u_c_hilo	u_i_hilo	u_s_hilo
low	High	u_c_lohi	u_i_lohi	u_s_lohi
low	Low	u_c_lolo	u_i_lolo	u_s_lolo

Table 2 : Rank of heuristics based on makespan & flowtime value of respective schedule for different instances

	Makespan			Flowtime		
Instances	**I**	**II**	**III**	**I**	**II**	**III**
u_c_hihi	New Heuristic	Min-Min	Suffrage	New Heuristic	Min-Min	Suffrage
u_c_hilo	New Heuristic	Min-Min	Suffrage	New Heuristic	Min-Min	Suffrage
u_c_lohi	New Heuristic	Min-Min	Suffrage	New Heuristic	Min-Min	Suffrage
u_c_lolo	Min-Min	New Heuristic	Suffrage	New Heuristic	Min-Min	Suffrage
u_i_hihi	Suffrage	New Heuristic	Min-Min	Min-Min	Suffrage	New Heuristic
u_i_hilo	New Heuristic	Suffrage	Min-Min	Min-Min	Suffrage	New Heuristic
u_i_lohi	Suffrage	Min-Min	New Heuristic	Min-Min	Suffrage	New Heuristic
u_i_lolo	Suffrage	New Heuristic	Min-Min	MET	Min-Min	Suffrage
u_s_hihi	Suffrage	Min-Min	New Heuristic	Min-Min	Suffrage	New Heuristic
u_s_hilo	Suffrage	Min-Min	New Heuristic	Min-Min	Suffrage	New Heuristic
u_s_lohi	Min-Min	Suffrage	New Heuristic	Min-Min	Suffrage	New Heuristic
u_s_lolo	Min-Min	New Heuristic	Suffrage	Min-Min	New Heuristic	Suffrage

Table 3 : Makespan Value

Instances	New Heuristic	Min-Min	Suffrage	LJFR-SJFR	Max-Min	WQ	kPB	SA	MET	MCT	OLB
u_c_hihi	**7718740**	8460675	10249173	12112938	12385672	16242463	12496864	12613221	47472299	11422624	14376662
u_c_hilo	**158113.6**	161805.4	168982.6	199252.4	204054.6	210935.7	201154	194549.8	1185093	185887.4	221051.8
u_c_lohi	**260359.1**	275837.4	337121.5	398733.1	392566.7	476788.5	400291.1	426271.4	1453098	378303.6	477357
u_c_lolo	5480.133	**5441.428**	5658.538	6745.371	6945.362	7392.327	6846.273	8167.052	39582.3	6360.055	7306.596
u_i_hihi	3502023	3513919	**3306819**	6713596	8018378	25668847	4508656	4692192	4508507	4413583	26102018
u_i_hilo	**76654.02**	80755.68	77589.1	130346.9	151923.8	277150.3	93005.9	102981	96610.48	94855.91	272785.2
u_i_lohi	124021.8	120517.7	**114578.9**	213484.6	251528.8	842589.3	143816.1	143905.2	185694.6	143816.1	833605.7
u_i_lolo	2662.64	2785.645	**2639.318**	4397.988	5177.709	9420.533	3122.956	3485.29	3399.285	3137.35	8938.027
u_s_hihi	5538803	5160343	**5121954**	8339410	9208811	20043901	6514162	7127730	25162058	6693924	19464876
u_s_hilo	110232.1	104375.2	**102499.9**	154962.9	172822.7	239122.9	123543.8	149050.3	605363.8	126587.6	250362.1
u_s_lohi	164003	**140284.5**	150297.1	246097.7	282085.7	639839.1	187956	194318.4	674689.5	186151.3	603231.5
u_s_lolo	3828.391	**3806.828**	3846.469	5527.161	6232.242	8593.204	4405.247	5836.963	21042.41	4436.118	8938.389

Table 4 : Flow-time Value

Instances	New Heuristic	Min-Min	Suffrage	LJFR-SJFR	Max-Min	WQ	kPB	SA	MET	MCT	OLB
u_c_hihi	**1.98E+09**	2.14E+09	2.64E+09	3.14E+09	3.16E+09	3.62E+09	3.10E+09	2.86E+09	1.18E+10	2.84E+09	3.51E+09
u_c_hilo	**3.71E+07**	3.75E+07	4.23E+07	4.82E+07	4.90E+07	4.87E+07	4.80E+07	4.44E+07	2.83E+08	4.43E+07	4.95E+07
u_c_lohi	**5.77E+07**	5.74E+07	8.37E+07	9.71E+07	9.01E+07	1.04E+08	9.61E+07	9.07E+07	3.01E+08	8.99E+07	9.78E+07
u_c_lolo	1.36E+06	**1.37E+06**	1.46E+06	1.70E+06	1.79E+06	1.78E+06	1.73E+06	1.63E+06	9.96E+06	1.59E+06	1.73E+06
u_i_hihi	8.95E+08	8.12E+08	**8.49E+08**	1.81E+09	2.19E+09	6.51E+09	1.15E+09	1.12E+09	8.20E+08	1.12E+09	6.44E+09
u_i_hilo	**1.74E+07**	1.67E+07	1.72E+07	2.82E+07	3.43E+07	5.92E+07	2.04E+07	2.03E+07	1.70E+07	2.07E+07	5.90E+07
u_i_lohi	2.54E+07	2.13E+07	**2.23E+07**	4.48E+07	5.56E+07	1.88E+08	2.94E+07	2.94E+07	2.14E+07	2.95E+07	1.81E+08
u_i_lolo	6.88E+05	6.66E+05	**6.74E+05**	1.10E+06	1.36E+06	2.38E+06	8.08E+05	7.11E+05	6.64E+05	8.09E+05	2.33E+06
u_s_hihi	1.41E+09	1.16E+09	**1.32E+09**	2.16E+09	2.43E+09	5.05E+09	1.58E+09	1.62E+09	3.68E+09	1.61E+09	4.98E+09
u_s_hilo	2.55E+07	2.35E+07	**2.50E+07**	3.66E+07	4.17E+07	5.61E+07	2.86E+07	2.87E+07	8.58E+07	2.88E+07	5.61E+07
u_s_lohi	3.46E+07	**2.69E+07**	3.46E+07	5.69E+07	6.33E+07	1.44E+08	4.14E+07	4.20E+07	8.39E+07	4.24E+07	1.34E+08
u_s_lolo	6.60E+05	**5.97E+05**	7.16E+05	9.09E+05	1.12E+06	1.53E+06	8.25E+05	8.31E+05	1.97E+06	8.40E+05	1.57E+06

References

[1] A. Abraham, R. Buyya, B. Nath, "Nature's heuristics for scheduling jobs on computational grids", The 8th IEEE International Conference on Advanced Computing and Communications, 2000.

[2] R. Armstrong, D. Hensgen, and T. Kidd, "The relative performance of various mapping algorithms is independent of sizable varie,nces in run-time predictions", 7th IEEE Heterogeneous Computing Workshop (HCW '98), Mar. 1998, pp. 79-87.

[3] T.D. Braun, H.J. Siegel, N. Beck, L.L, Boloni, M. Maheswaran, A.L. Reuther, J.P. Robertson, M.D. Theys, B. Yao, "A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems", Journal of Parallel and Distributed Computing 61 (6) (2001) 810-837.

[4] R. F. Freund, 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. Siegel, "Scheduling resources in multi-user, heterogeneous, computing environments with SmartNet", 7th IEEE Heterogeneous Computing Ubrkshop (HCW '98), Mar. 1998, pp. 184-199.

[5] T. Hagerup, "Allocating Independent Tasks to Parallel Processors: An Experimental Study", Journal of Parallel and Distributed Computing, 47, 1997, pp. 185-197.

[6] O. Ibarra and C. Kim, "Heuristic algorithms for scheduling independent tasks on non-identical processors", Journal of the ACM, 24(2):280{289, 1977. ISSN 0004-5411.

[7] Hesam Izakian, Ajith Abraham and Vaclav Snasel, "Performance Comparison of Six Efficient Pure Heuristics for Scheduling Meta-Tasks on Heterogeneous Distributed Environments", Neural Network World, Volume 19, Issue 6, pp. 695-710, 2009.

[8] K. Kousalya and P. Balasubramanie, "An Enhanced Ant Algorithm for Grid Scheduling Problem", IJCSNS International Journal of Computer Science and Network Security, VOL.8 No.4, April 2008.

[9] M. Maheswaran, S. Ali, H.J. Siegel, D. Hensgen, R.F. Freund, "Dynamic mapping of a class of independent tasks onto heterogeneous computing systems", Journal of Parallel and Distributed Computing 59 (2) (1999) 107_131.

[10] F. Xhafa, L. Barolli and A. Durresi, "Batch Mode Schedulers for Grid Systems. International Journal of Web and Grid Services", Vol. 3, No. 1, 19-37, 2007a.

[11] F. Xhafa, J. Carretero, L. Barolli and A. Durresi, "Immediate Mode Scheduling in Grid Systems", International Journal of Web and Grid Services, Vol.3 No.2, 219-236, 2007b.

Authors

Anand K Chaturvedi obtained his B.E. from University of Rajasthan, M.Tech. in Industrial Engineering from IIT Delhi and currently pursuing PhD at ABV-IIITM Gwalior, India. He has over 15 years of Industrial and teaching experience in Operations Management, Operations Research and Computer Programming.

Prof R. Sahu received his B.Sc. (Engg.), M.Sc. (Engg.) from Sambalpur University and PhD from IIT, Kharagpur, India. He has over 25 years of teaching experience in Systems Approach to Management, Business Process Management, Business Analytics, e-Commerce, and IT enabled Services. He has been closely associated with industry, business and government consultancy projects and has published more than 90 papers in his areas of proficiency and interest.

Dr Sahu is Professor at ABV- Indian Institute of Information Technology and Management, Gwalior India. He is currently the Director, of AP-Indian Institute of Information Technology, Basar, India