IJCSI International Journal of Computer Science Issues, Vol. 7, Issue 4, No 5, July 2010
ISSN (Online): 1694-0784
ISSN (Print): 1694-0814



# A Double Min Min Algorithm for Task Metascheduler on Hypercubic P2P Grid Systems

**D.Doreen Hephzibah Miriam and K.S.Easwarakumar**

**Department of Computer Science and Engineering, Anna University, Chennai,
Tamil Nadu , 600025, India**

**Abstract**

Most of the existing solutions on task scheduling and resource management in grid computing are based on the traditional client/ server model, enforcing a homogeneous policy on making decisions and limiting the flexibility, unpredictable reliability and scalability of the system. Thus, we need well organized system architecture to provide high system availability with task scheduling scheme for Grid system. In this paper, we integrate Grid with P2P on to the extended Hypercube topology for task scheduling and load balancing, which gives optimal makespan and balances the load. We propose an efficient SPA based task scheduling algorithm named Double Min Min Algorithm which performs scheduling in order to enhance system performance in Hypercubic P2P Grid (HPGRID). The simulation result shows that the SPA based Double Min Min scheduling minimizes the makespan with load balancing and guarantees the high system availability in system performance. At last, the SPA based Double Min Min algorithm is compared with traditional Min Min and Max Min algorithm, by the experiment evaluation it shows that the new algorithm has a better quality of system load balancing and the utilization of system resources.

***Keywords:*** *Peer-to-Peer, Grid; Hypercube, Task Scheduling, Min Min, Max Min, Set Pair Analysis (SPA).*

## 1. Introduction

Several aspects of today's Grids are based on centralized or hierarchical services. However, as Grid size increase from tens to thousands of hosts, fault tolerance has been a key issue. To address this problem, functionalities should be decentralized to avoid bottlenecks and guarantees scalability. A way to ensure Grid scalability is to adopt P2P models and techniques to implement non-hierarchical decentralized Grid services and systems as discussed in [19]. As reason of that we expect strong possibility of the commonalities and synergies between these two technologies in terms of the connectivity, access services, resource discovery, task scheduling and fault tolerance.

P2P technology is applied to the Grid and we adopt P2P protocol to deal with Grid Computing in Hypercube topology, by means of joining some Cubic Grid Peer (CGP) into Grid and come into being a new Grid resource organization model called Hypercubic P2P Grid (HPGRID) model. It makes each CGP in the Hypercube topology to form one local Grid system and answer for dealing with local task. Various local Grid systems are connected by P2P technology. In this HPGRID model, for a group of clients, every CGP is used as a server, but it is equal between every two CGPs. The topology structure carries out the balance between centrality search efficiency; it also keeps the robustness of load balancing and distributes search methodologies. In the Grid information server that is based on HPGRID model, each participant can configure one or more nodes and operate as a CGP. The CGP in the HPGRID model will change the information of monitor and resource find, but for the nodes in the different CGP will exchange information by P2P manner. At the same time HPGRID model has many characteristics such as independent of concentrate control, distribute, extension and can adapt resource state change.

A new P2P Grid task scheduling algorithm is proposed and it is named as SPA based Double Min Min scheduling, in order to improve quality of the system load balancing and utilization of the system resources that gives better makespan. In this paper, we use binary connection number of Set Pair Analysis (SPA), a new soft computing method [22] to represent the uncertain Execution Time to Compute of tasks (ETC) and to process the synthetic uncertainty in task scheduling of a computing grid. The proposed SPA based Double Min Min scheduling algorithm is better logical and efficient than the conventional algorithms by simulation testing. This algorithm performs the load balancing by reselecting the task that are greater than mean Completion time (CT) and again schedule the reselected task using SPA based Min Min, which results in better load balancing and resource utilization.

The remainder of this paper is organized as follows. The section 2 presents the related works; the section 3 defines the Problem Formulation; the section 4 illustrates about the Set Pair Analysis Method; the section 5 describes the Hypercubic P2P Grid; the section 6 illustrates the SPA based Double Min Min Algorithm; the section 7 explains

the simulation testing model; the section 8 deals with performance evaluation; Finally, Section 9 gives the conclusion and future work.

## 2. Related Works

In this section, we will review and contrast a set of heuristic scheduling algorithms in heterogeneous computing (HC) system. First, we mention metatask concept: A meta-task is defined as a collection of independent tasks with no data dependences. A metatask is mapped onto machine statically; each machine executes a single task at a time. In general, we take into account a set of meta-tasks in the Grid environment. There are a large number of heuristic algorithms to be designed to schedule task to machines on heterogeneous computing system. Braun et al. [3] provides a comparison of 11 static heuristics for scheduling in HC environments. In this section, we will list several basic heuristics scheduling algorithms as follows:

**OLB** (Opportunistic Load Balancing) [11] assigns each job in arbitrary order to the processor with the shortest schedule, irrespective of the ETC on that processor.

**MET** (Minimum Execution Time) [10] assigns each job in arbitrary order to the processor on which it is expected to be executed fastest, regardless of the current load on that processor.

**MCT** (Minimum Completion Time) [10] assigns each job in arbitrary order to the processor with the minimum expected completion time (CT) for the job. The completion time of a job *j* on a processor *p* is simply the ETC of *j* on *p* added to *p's* current schedule length.

**Min-min** [15] establishes the minimum completion time for every unscheduled job (in the same way as MCT), and then assigns the job with the minimum completion time (hence Min-min) to the processor which offers it this time.

**Max-min** [15] is very similar to Min-min. Again the minimum completion time for each job is established, but the job with the maximum minimum completion time is assigned to the corresponding processor.

In **Sufferage** [15] for each task, the minimum and second minimum completion time are found in the first step. The difference between these two values is defined as the sufferage value. In the second step, the task with the maximum sufferage value is assigned to the corresponding machine with minimum completion time.

In **XSufferage**, Casanova et al gave an improvement to fix the problem, in Sufferage heuristic, when there is input and output data for the tasks and resources are clustered. Sufferage problems are described in [6].

Longest Job to Fastest Resource - Shortest Job to Fastest Resource **(LJFR-SJFR)** [1] heuristic begins with the set U of all unmapped tasks. Then, the set of MCT of task on machine is computed as, $M = min\ (CT\ (T_i, M_j))\ for\ (1 \le i \le n,\ 1 \le j \le m)$, is found the same as min-min. Next, the task with the overall minimum completion time from M is considered as the shortest job in the fastest resource (SJFR). Also the task with the overall maximum completion time from M is considered as the longest job in the fastest resource (LJFR).

In **Segmented Min-Min** heuristic described in [21] tasks are first ordered by their expected completion times. Then the ordered sequence is segmented and finally it applies Min-Min to these segments.

**QoS Guided Min-Min** shown in [12] adds a QoS constraint (QoS for a network by its bandwidth) to basic Min-Min heuristic. Its basic idea is that some tasks may require high network bandwidth, whereas others can be satisfied with low network bandwidth, so it assigns tasks with high QoS request first according to Min-Min heuristic. In the worst cases, where all tasks need either low QoS or high QoS, this heuristic will take $O(s^2m)$ time. In **Work Queue** [13] method the heuristic selects a task randomly and assigns it to the machine as soon as it becomes available (in other word the machine with minimum workload).

## 3. Problem Formulation

This section presents the problem of job scheduling in heterogeneous computing environment. The experimental study is based on a benchmark simulation model by Braun et al. [3]; in this model static mapping of meta-tasks is considered. Each machine executes one task at a time in the order in which tasks are allocated to the machines. For static mapping, the size of the meta-tasks and the number of machines in the heterogeneous computing environment is known a priori. Since there are static heuristics, the accurate estimate of the expected execution time for each task on each machine is known a priori to execution and is contained within an ETC (expected time to compute) matrix where $ETC\ (t_i, m_j)$ is the estimated execution time of task *i* on machine *j*. Using the ETC matrix model, the scheduling problem can be defined as follows:

- A number of independent jobs to be allocated to the available grid resources.
- Number of machines is available to participate in the allocation of tasks.
- The workload of each job (in millions of instructions).
- The Computing capacity of each resources(in MIPS)
- Ready m represents the ready time of the machine after completing the previously assigned jobs.

- ETC matrix of size t * m, where t represents the number of jobs and m represents the number of n machines.

The main aim of the scheduling algorithm is to minimize the makespan where makespan = max ($CT(t_i, m_j)$) where *CT* represents the completion time of task *i* on machine *j*.

## 4. Set Pair Analysis Method

Set Pair Analysis (SPA) is a new soft computing method which can express and process the synthetic uncertainty caused by fuzzy, random, indeterminate known uncertainty etc. It was first presented by Professor Zhao Ke-qin in 1989 [14]. After twenty years theory and application research about SPA, it gradually becomes a new uncertainty theory by which we can research certainty and uncertainty as a whole now. A Set Pair is a system containing two sets *(A,B)* where there are some similar attributes or tight relations, such as *(Control, Decision), (Products, Sell),* etc can be seen examples of Set Pair under specified conditions. The main thought of SPA is as follows: To two sets A, B under specified conditions, first analyzing their identical, discrepancy and contrary properties or attributes, and then describing them quantificational, expressing their relations by a formula called connection degree finally.

### 4.1 Connection Number

Let A and B be two sets, and both of them have N attributes. We define their connection degree denoted as μ *(A, B)*. For brevity, we will usually denote $\mu$ *(A, B)* simply as μ..

Thus $\mu = \frac{S}{N} + \frac{F}{N} i + \frac{P}{N} j$ (1)

where $S + F + P = N$ (2)

S is the number of their identical attributes, P is the number of their contrary attributes, $F = N - P - S$ is the number of their discrepancy attribute. $S/N$ , $F/N$ and $P/N$ are called identical degree, discrepancy (uncertain) degree, and contrary degree respectively. $j$ is the coefficient of the contrary degree, and is specified as -1. As the coefficient of the discrepancy degree, $i$ is an uncertain value between -1 and 1, i.e. $i \in [-1,1]$, in terms of various circumstances. The uncertainty of the discrepancy degree of two sets is eliminated when $i$ is specified as -1 or 1, and will increase when $i$ is approaching zero.

Let $a = S/N$, $b = F/N$, and $c = P/N$ , then the formula (1) and (2) can be rewritten as follows.

$\mu = a + bi + cj$ (3)

where $a + b + c = 1$ (4)

### 4.2 Binary Connection Number

We say that $\mu = a + bi + cj$ is a ternary connection number if a, b, c are arbitrary nonnegative real number, and i, j are discrepancy coefficient, contrary coefficient respectively. $\mu = a + bi$ is a binary connection number when c=0. Obviously, the connection number (binary or ternary) is an extension and generalization of connection degree by deleted the constraint condition $a + b + c = 1$, and the main theory meaning of the connection number is extended the conception of number. We only use binary connection number to represent the uncertain Execution Time to Compute (ETC) of grid tasks.

The heuristics algorithm discussed in related works has been mapped to SPA methodology for both batch mode and online mode in paper [8].

## 5. Hypercubic P2P Grid

### 5.1 Hypercube Topology

A hypercube can be represented as $Q_n^k$ , where $k$ is the number of nodes in each dimension and $n$ is the number of dimensions spanned by the hypercube. All the nodes in the hypercube have $(k-1)n$ neighbours, (in Figure 1, the hypercube has four dimensions and $k$ = 2). A complete hypercube graph consists of $N = k^n$ nodes, in which every peer has its own logical id, which is represented by base $k$ ($k$ is always 2 for hypercube), and it can be coded using $n$ bits as RGC (Reflected Gray Code). In RGC, the two adjacent nodes differ by only one bit in their coding. For instance, given $k$ = 2 and $n$ = 3, the RGC corresponds to the nodes are (000, 001, 010, 011, 100, 101,110,111). The hyper cube is always symmetric and the network diameter for a hypercube is $\Delta = \log_k N$. This is crucial for load balancing in the network, as the search begins from any node due to its symmetry.

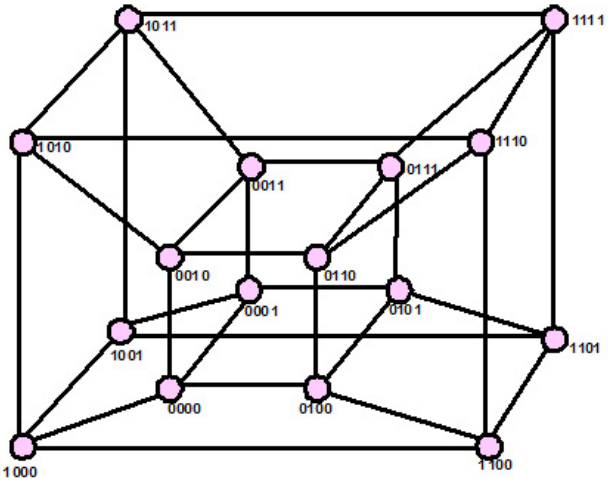


Fig 1: A Four dimensional hypercube System

## 5.2 Hypercubic P2P Grid Topology

The Hypercubic P2P Grid Topology is the hypercube structure with additional neighbourhood links. In short, we refer Hypercubic P2P Grid as HPGRID. The Hypercubic P2P Grid nodes have $(k-1)\times(n+1))$ neighbours. Let $l$ , $1 \leq \leq l \leq k^{n-2}$, be the layer of the HPGRID. Let $d$ be the set of nodes at each layer of the HPGRID, then $d = \{0, 1, 2, 3\}$. Also, the number of nodes in HPGRID is $k^n$, and the number of edges are $n2^{n-1}+k^{n-1}$ . The HPGRID Topology for $n = 3$ is depicted in Figure 2. There in, the dashed lines are the additional neighbourhood links.

The HPGRID system can be represented by an undirected graph $G=(V,E)$ where $V=\{v_{l.d},\ldots\ldots,v_{0.0}\}$ .

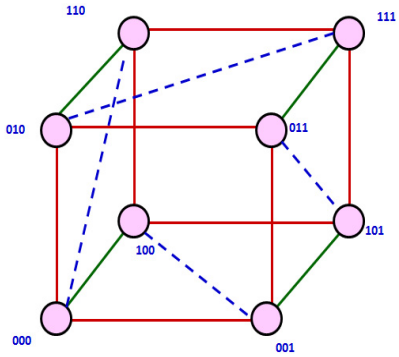


Fig 2: A 3D Hypercubic P2P Grid Topology

$$E=\bigcup_{l=1}^{2^{n-2}}\{(v_{l.0},v_{(l+1).2}),(v_{l.1},v_{(l+1).0}),(v_{l.2},v_{(l+1).3}),(v_{l.3},v_{(l+1).1})\}$$
$$\bigcup\{(p,q)\ni:|p\oplus q|_1=1\}$$

where $p$ and $q$ are the binary values of the nodes of HPGRID, and $|p|_1$ denotes the number of ones in $p$. For instance, consider the equivalent HPGRID of Figure 2 shown as a HPGRIP graph figure 3.

Here

$$V=\begin{Bmatrix}v_{1.0}(=000),v_{1.1}(=001),v_{1.2}(=010),v_{1.3}(=011)\\v_{2.0}(=100),v_{2.1}(=101),v_{2.2}(=110),v_{2.3}(=111)\end{Bmatrix}$$

*and*

$$V=\begin{Bmatrix}(v_{1.0},v_{2.2}),(v_{1.1},v_{2.0}),(v_{1.2},v_{2.3}),(v_{1.3},v_{2.1})\\(v_{1.0},v_{1.1}),(v_{1.1},v_{1.3}),(v_{1.3},v_{1.2}),(v_{1.2},v_{1.0})\\(v_{2.0},v_{2.1}),(v_{2.1},v_{2.3}),(v_{2.3},v_{2.2}),(v_{2.2},v_{2.0})\\(v_{1.0},v_{2.0}),(v_{1.1},v_{2.1}),(v_{1.2},v_{2.2}),(v_{1.3},v_{2.3})\end{Bmatrix}$$

In $E$, the first four edges are the additional neighbourhood links, and the remaining edges are the hypercubic edges.

## 5.3 Representation of HPGRID

Generally, the grid model integrated with P2P mode is composed of many Cubic GridPeers [20]. Each CubicGridPeer represents a super management domain. Each Cubic GridPeer controls the access of a group of local computing resources. It plays two roles: one as the resource provider and the other as resource consumer. The resource provider allows its free resources to other Cubic GridPeer (consumer), while the consumer arbitrarily uses its local resources or the free resources of other Cubic GridPeers to carry out its task. The resource management model for HPGRID is shown in figure 4. The bottom communities of the model using the traditional grid technologies, and the P2P mode are adapted to interact the information between Cubic Grid Peers. Here, Cubic GridPeer (CGP) is equivalent to a super node. When they search resources, the users first query the resources in the domain of Cubic GridPeer. If no query result, the search will be carried out through Cubic Grid- Peer to query the other Cubic GridPeers with P2P way.

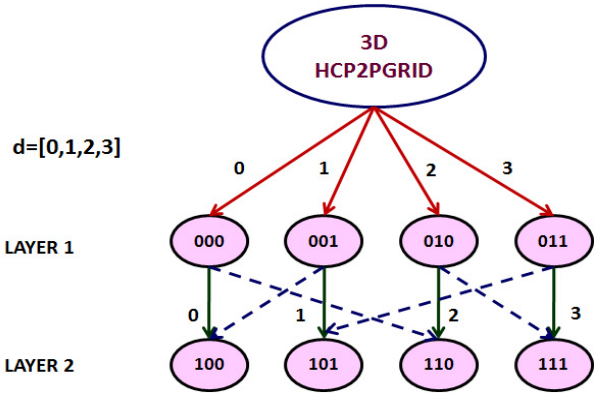


Fig 3: A 3D Hypercubic P2P Grid

In HPGRID, each node represents a *CGP* where each *CGP* is a collection of Grid Nodes *GNs*. The *GN* that belongs to a particular *CGP* is called Grid Community *GC*. Each Grid node is represented using its own identifier

and the identifier of the corresponding CGP. That is, grid node & is represented as $(ID_g, ID_{CGP(g)})$

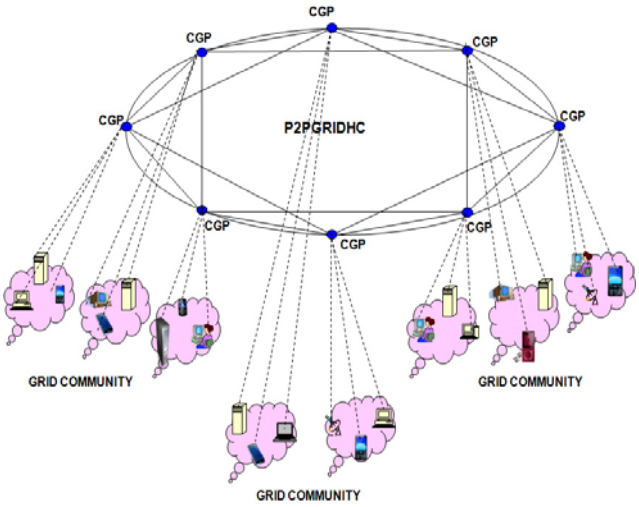


Fig. 4: Overview of HPGRID

At each CGP in the HPGRID system, it contains a CGP Header in the format represented in figure 5.

| CGP_ID | | Node_ID$_1$ | Node_ID$_2$ | | Node_ID$_n$ |
|---|---|---|---|---|---|
| DF | | Resrc_Avail | Resrc_Avail | | Resrc_Avail |
| RD | | Ptr to RT | Ptr to RT | ... | Ptr to RT |
| LF | Flag | No of CPUs | No of CPUs | | No of CPUs |

Fig 5: CGP Header Format

The CGP Header field description is as follows,

- $CGP_{ID}$ : Cubic Grid Peer Identifier,
- *DF :* Distance factor,
- R*D*: Resource Density,
- *LF* : Load Factor,
- Flag = 0,*CGP* is non-alive, Flag = 1,*CGP* is alive,
- $Node_{ID_i}$ : Nodes Identifier where i = 1, 2 ...*n.*
- *Resrc_Avail:* : The total number resources available at the node,
- *Ptr to RT* :Pointer to Resource Table,
- *No. of PEs* : Total number of processing element at the node.

Each CGP contains a common resource table which has the details of all the available resources in its own *GC*. The Resource table format is described in Figure 6. It contains the resource name and number of resources corresponding to its resource number.

| Resrc. No | Resource | Total Number |
|---|---|---|
| 1 | Processor | (1 - 5) |
| 2 | Printer | (1 - 5) |
| . | . | . |
| . | . | . |
| R | . | (1 - 5) |

Fig 6: Resource Table

## 5.4 Parameter at Cubic Grid Peer

The parameters represent the state of a Cubic Grid Peer (*CGP*) that one must meet the following criteria: they must be small, they must facilitate identifying the fertility of a Cubic GridPeer and they must not divulge resource structure information. Based on parallel application characterization experience, we identified the following parameters.

### 5.4.1 Distance Factor (*DF*)

This gives an idea of how far the target *CGP is* from the home *CGP*. A home *CGP* is defined to be the *CGP* in which the program and input data are present and to which the output data will go. If it is separated by a large network distance, i.e., high latency and low bandwidth, the staging files and the arriving program and the input files to that *CGP* will be costly. Another reason why such a factor is important is that tasks in parallel programs might be scheduled on different *CGP*. Thus there will be some communication between *CGP*, even though such a situation will be reduced as far as possible by the search algorithm. For tightly coupled applications this may not always be possible and the scheduler might be forced to schedule them on different *CGP*. This parameter will make *CGP* between which there is large latency or low bandwidth less desirable to the *CGP* selector. A high value of this factor makes a *CGP* less desirable for scheduling.

$$DF = Min\_dist\{h(CGP), n(CGP)\} \quad (5)$$

where h(CGP) denotes home CGP and n(CGP) denotes neighbour CGP.

### 5.4.2 Resource Density (RD):

This parameter represents the intensity of computing power per unit communication bandwidth. The lower the value of RD, the more will be the bandwidth between every pair of nodes. This signifies that the resources in the CGP's are tightly coupled. For parallel programs that have a communicator in which a small group of processes communicate a lot, a *CGP* with a low value of RD is important. For example, a SMP will have low RD whereas a network of workstations will have high RD. A similar parameter has been used to represent the computation to communication ratio in schedulers of parallel programs.

$$Average \quad Resource \quad Density(ard) = \frac{\sum_{j=1}^{n} rd_j}{m} \quad (6)$$

where $Resource\ Density(rd) = \frac{\sum_i et_i - st_i}{T}$

where *i* represents *t* executed on *m*, $et_i$ is the end time of executing $t_i$ on resource $m_j$, $st_i$ is the start time of executing

$t_i$ on resource $m_j$ and $T$ is the total application execution time so far which can be calculated through the following relation:

$$T = max\ (et_i) - min\ (st_j)$$

5.4.3 Load Factor (LF)

This gives the overall load at some instant in that *CGP*. This is important to take care of the computation component of the parallel program. Thus parallel processes have a high computation aspect compared to communication which would prefer a better value for LF than for RD.

$$Load\ Factor = \left(1 - \frac{k}{ard}\right) \times 100\% \quad (7)$$

$$where\ k = \frac{\sqrt{\sum_{i=1}^{m} (ard - rd_j)^2}}{m}$$

These parameters would be number calculated from information about the state of resources in a particular *CGP*.

5.5 Node Arrival

The sequence of steps involved in node arrival is as follows.

1. The arriving node of the HPGRID will be placed in the waiting queue $W_q$ .
2. If there is a non-alive node present in a *CGP*, the incoming node is assigned to the available free space having the minimum identity.
    a. The *CGP* then fetches the leaf node's (*GN*'s) information from its neighbouring *CGP*s header from the same zone.
    b. In case of non availability of a *CGP* header, the required information will be obtained from the *CGP* header of the neighbouring zones.
    c. Suppose the required *CGP* header is not available even in the neighbouring zone, the incoming *CGP* probes each of its leaf nodes and creates a new *CGP* header.
3. If there is no non-alive node, the arriving node will be appended in the waiting queue $W_q$, until it finds a free space.

5.6 Node Departure

When a node decides to departure from the HPGRID, it does the following.

1. It transfers the *CGP* Header to its neighbouring *CGP* of the same zone, which will be exploited by the newly arrived node to fetch the information about its leaf nodes (*GN*'s) and makes the node's state as non-alive.
2. Suppose the neighbouring *CGP* is also non alive within the same zone, it transfers to the neighbouring zone by a single hop and makes the node's state as non-alive.

In exceptional cases, due to crashing, heavy traffic, etc., its state will be defined as non-alive.

## 6. Task Scheduling Algorithms

In this section, our proposed algorithm SPA based Double Min Min Algorithm is illustrated which outperforms the conventional algorithm Min-min and Max-min. The architecture of the metascheduler is given in Figure 7.

The metaschedulers takes as input the available resource list, users and the task to be scheduled in the HPGRID system. From the input values given by the user, the ETC matrix is generated as described in [3]. Now, the SPA based Double Min Min algorithm is executed, where it assigns the job on to the resource which minimizes the makespan and balances the load with effective resource utilization.

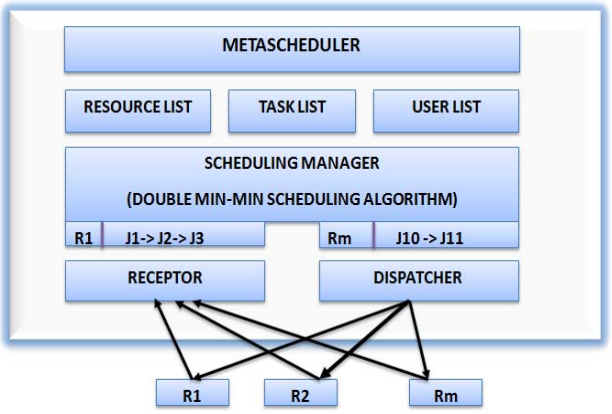


Fig 7: Architecture of Metascheduler

6.1 Task Scheduling

The metascheduler schedules the task as per SPA based Double Min Min algorithm, where it finds the minimum task on to the processor in the first step, in the following iteration it finds the minimum task on to the remaining processor leaving out the processor that has been assigned previously. The iteration again continues in the same method until the entire task has been mapped. The resulting scheduling set gives the task assigned to the processor; from the set the mean Completion Time (CT) is calculated. The task that are assigned to the processor with CT greater than mean are reselected and reschedule using

SPA based Min Min until all the CGPs values are less than or equal to Mean CT. Thus, task scheduling results in better makespan, load balancing and resource utilization.

### 6.2 Parameters Definition

1. $M_q$: On the one side it denotes the $q^{th}$ computing resource, on the other hand, it denotes the tasks set assigned to $M_q$ , and all tasks in the $M_q$ is on the order of its assigned time.
2. $S_q$: The time of resource $M_q$ to start a new task when $M_q$ finished all tasks assigned to it.
3. *E(p,q):* Expected execution time if task $t_p$, assigned to resource $M_q$.
4. *C(p,q):* Completion time of a new task $t_p$, assigned to resource $M_q$ , $C(p,q)= S_q + E(p,q)$
5. *CT:* a set of ordered pair $(t_p, M_q)$, it means task $t_p$ is going to be assigned to resource $M_q$ and $C(p,q)$ = min $C(p,q)$ such that k=1,2,.....,m in a loop of scheduling algorithm.
6. *grp(CT$_p$,machines)*: The function "grp" is used to group all the tasks and machines that has minimum completion time. The best minimum task/machine pair $(p,q)$ is selected from the Group.
7. *Mean_CT* :is used to find the mean completion of all the machines.

### 6.3 SPA based Double Min Min Algorithm

**Input**: Tasks set T, Computing resource set M, Dimension of HPGRID System, ETC Matrix based on connection number and $S_q$ .
**Output**: Scheduling results $M_1, M_2 \ldots, M_m$.

### 6.4 SPA based Double Min Min Algorithm Description

1. For each task, find the Completion time with respect to each resource.
2. Compare all the completion times of that task and select the resource which provides the minimum earliest completion time.
3. Repeat the process for all tasks.
4. From the resultant set, find the task which has the minimum of the minimum completion times.
5. Assign that task with the corresponding resource.
6. Repeat the process until task set becomes empty.
7. The mean of all machines completion time is taken.
8. The task mapped to machine whose completion time is greater than the mean value is selected.
9. Perform SPA based Min Min Scheduling to enhance load balancing moving overload task to CGPs with lesser load.
10. Tasks allocated to the selected machines are reallocated according to grp function.

**Algorithm 1**: SPA based Double Min Min Algorithm

1. for every $M_q \in M$
   $M_q = 0$;
   end for
   Repeat the following steps until (T = *0*).
2. $CT = 0$ ;
3. for every $t_p \in T$
4. for every $M_q \in M$
   $C(p,q) = S_q + E(p,q)$;
   end for
5. Find the minimum earliest completion time of $t_p$ and the corresponding resource $M_q$ .
   Let $CT = CT \cup \{< t_p , M_q >\}$;
6. end for
7. $C_{\min} = \min\left\{C(p,q) \middle| \langle t_p, M_q \rangle \in CT\right\}$
   $M^{\min} = \left\{\{M_k | C(p,k) = C_{\min}\} and \langle t_p, M_k \rangle \in CT\right\}$
8. Choose any one $< t_p, M_q >$ from $M^{min}$;
   Let $M_k = M_k \cup t_r$ ;
   $T = T - t_r$;
   $S_k = S_k + E(r,k)$;
9. Calculate *Mean_CT* = $(\Sigma S_k)$ */No. of Machines;*
10. for all Machine $M_k$
    if ($S_k \geq$ *Mean_CT)*
    Select tasks reserved on the machine
    end for
11. for every $t_p \in T$ reselected
12. Repeat Step 4 to Step 8
13. Compute *NewCT* = CT $\cup \{<t_p, M_q >\}$;
    if (*NewCT* $\leq$ *Mean_CT)*
    *Group* ($CT_{p,machines}$) = *grp*($CT_{p,1}, CT_{p,2}..$)
    end for
14. Reschedule ($t_p$ on $M_k$)
    Compute New$CT_{p,k}$
    end for

## 7. Simulation Model

### 7.1 Types of ETC Matrix

ETC Matrix has been generated as defined in [3]. To further vary the characteristics of the ETC matrices in an attempt to capture more aspects of realistic mapping situations, different ETC matrix consistencies were used. An ETC matrix is said to be consistent if whenever a machine $m_j$ executes any task $t_k$ faster than machine $m_k$ , then machine $m_j$ executes all tasks faster than

machine $m_k$ . **Consistent matrices** were generated by sorting each row of the ETC matrix independently, with machine $m_0$ always being the fastest and machine $m_{(\mu-1)}$ the slowest. In contrast, **inconsistent matrices** characterize the situation where machine $m_j$ may be faster than machine $m_k$ for some tasks and slower for others. These matrices are left in the unordered, random state in which they were generated (i.e., no consistency is enforced). **Semi-consistent matrices** are inconsistent matrices that include a consistent sub matrix of a predefined size. For the partially-consistent matrices used here, the row elements in column positions [0, 2, 4, ...] of row i are extracted, sorted, and replaced in order, while the row elements in column positions [1, 3, 5, ...] remain unordered (i.e., the even column are consistent and the odd columns are, in general, inconsistent).

### 7.2 SPA on ETC matrix

Set-pair analysis represents uncertainty in grid. SPA concept is applied in ETC matrix generation. Expected time to compute (ETC) is represented as $a + bi$, where $a$ is normal finish time, $b$ is possible delay time or saving time, $i$ $[-1, 1]$ is the uncertainty coefficient which determines b.

Example of SPA based ETC Matrix [3 × 3]

$$\begin{pmatrix} 30+5i & 24+3i & 28+6i \\ 14+10i & 15+2i & 16+3i \\ 19+8i & 21+14i & 18+5i \end{pmatrix}$$

For example $30 + 5i \Rightarrow 30$ represents seconds to complete the task, where $i = \{-1, 1\}$, specifies $5i \Rightarrow \{-5i+ 5\}$ seconds saved / delayed.
Example of SPA based ETC Matrix types.

**Inconsistent Matrix**

$$\begin{pmatrix} 30+5i & 24+3i & 28+6i \\ 14+10i & 15+2i & 16+3i \\ 19+8i & 21+14i & 18+5i \end{pmatrix}$$

**Semi Consistent Matrix**

$$\begin{pmatrix} 28+6i & 24+3i & 30+5i \\ 14+10i & 15+2i & 16+3i \\ 18+5i & 21+14i & 19+8i \end{pmatrix}$$

**Consistent Matrix**

$$\begin{pmatrix} 24+3i & 28+6i & 30+5i \\ 14+10i & 15+2i & 16+3i \\ 18+5i & 19+8i & 21+14i \end{pmatrix}$$

Figure 8, shows the part of the ETC Consistent matrices generated for 512 tasks on 16 machines using Gridsim[4]toolkit.

CONSISTENT MATRIX

| | M1 | M2 | M3 | M4 | M5 | M6 | M7 | M8 | M9 | M10 | M11 | M12 | M13 | M14 | M15 | M16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| T1 | 120+i 21 | 126+i 18 | 132+i 18 | 132+i 10 | 138+i 10 | 138+i 19 | 144+i 25 | 144+i 15 | 150+i 28 | 162+i 16 | 162+i 13 | 162+i 22 | 162+i 19 | 162+i 24 | 168+i 30 | 180+i 12 |
| T2 | 1342+i 92 | 1403+i 93 | 1403+i 72 | 1464+i 50 | 1525+i 89 | 1525+i 78 | 1586+i 88 | 1586+i 99 | 1647+i 99 | 1647+i 60 | 1708+i 59 | 1769+i 67 | 1769+i 76 | 1830+i 75 | 1830+i 87 | 1830+i 69 |
| T3 | 1320+i 76 | 1320+i 56 | 1386+i 73 | 1386+i 89 | 1386+i 55 | 1386+i 64 | 1518+i 70 | 1518+i 71 | 1584+i 69 | 1650+i 69 | 1716+i 84 | 1716+i 68 | 1782+i 60 | 1782+i 85 | 1914+i 97 | 1914+i 75 |
| T4 | 60+i 15 | 63+i 23 | 63+i 29 | 66+i 26 | 66+i 20 | 66+i 15 | 66+i 21 | 72+i 30 | 75+i 17 | 75+i 15 | 78+i 17 | 81+i 19 | 81+i 11 | 81+i 27 | 81+i 10 | 84+i 24 |
| T5 | 660+i 72 | 693+i 54 | 726+i 77 | 726+i 88 | 759+i 79 | 759+i 52 | 759+i 61 | 759+i 69 | 792+i 72 | 792+i 68 | 792+i 72 | 858+i 62 | 858+i 77 | 891+i 83 | 957+i 67 | 990+i 56 |
| T6 | 420+i 87 | 420+i 88 | 441+i 75 | 441+i 52 | 462+i 53 | 483+i 77 | 483+i 75 | 483+i 93 | 483+i 52 | 483+i 82 | 483+i 95 | 504+i 84 | 525+i 78 | 525+i 100 | 567+i 56 | 630+i 89 |
| T7 | 140+i 17 | 147+i 12 | 147+i 23 | 161+i 19 | 168+i 26 | 168+i 12 | 175+i 21 | 175+i 16 | 182+i 16 | 182+i 10 | 196+i 29 | 196+i 27 | 196+i 10 | 203+i 28 | 203+i 18 | 210+i 22 |
| T8 | 567+i 53 | 567+i 64 | 567+i 87 | 567+i 56 | 594+i 66 | 594+i 55 | 648+i 81 | 702+i 59 | 702+i 78 | 702+i 56 | 702+i 86 | 729+i 65 | 756+i 85 | 756+i 57 | 783+i 88 | 810+i 88 |
| T9 | 504+i 86 | 504+i 68 | 528+i 68 | 528+i 94 | 552+i 73 | 576+i 64 | 576+i 72 | 600+i 81 | 600+i 93 | 624+i 60 | 648+i 72 | 648+i 71 | 648+i 79 | 648+i 55 | 672+i 93 | 696+i 56 |
| T10 | 42+i 20 | 42+i 26 | 42+i 18 | 44+i 23 | 46+i 21 | 46+i 17 | 48+i 20 | 50+i 10 | 56+i 17 | 56+i 15 | 56+i 16 | 56+i 18 | 58+i 29 | 58+i 28 | 58+i 30 | 60+i 24 |

Fig. 8: Part of Consistent Matrix for 512 X 16

## 8. Performance Analysis

This section deals about the performance analysis using the simulator Gridsim toolkit, the simulation environment used for implementation and experimental results has been analysed.

### 8.1 Simulation Environment

To evaluate and compare our SPA based Double Min Min scheduling algorithm with its two basic heuristics Min-Min and Max-Min, a simulation environments known as Gridsim toolkit [4] had been used. There are several grid simulators that allow evaluating a new grid scheduling algorithm, such as Bricks [2], MicroGrid [18] and SimGrid [5]. But Gridsim has some good which are listed below:

- It allows modelling of heterogeneous types of resources.
- Resource capability can be defined in the form of MIPS (Million Instructions Per Second) as per SPEC (Standard Performance Evaluation Corporation) benchmark.
- There is no limit on the number of application jobs that can be submitted to a resource.
- It supports simulation of both static and dynamic schedulers.

Gridsim had been used in many researches to evaluate the results such as [[7], [9], [16], [17]].

### 8.2 Experimental Results and Evaluation

Using Gridsim toolkit, the scheduling algorithm SPA based Double Min Min along with the basic heuristic Min-Min and Max-Min based on SPA, is simulated. Specifically, the proposed algorithm is compared with Min Min algorithm using SPA which has given minimum makespan as discussed in [8]. The experimental results represent the proposed algorithm generates even better

makespan when compared with the makespan generated for the conventional algorithm Min-Min and Max-Min based on SPA. For each heuristic all the three types of ETC matrix is generated and the results are evaluated.
The simulation has been executed 10 times for 256 tasks on 3D HPGRID system for all three types of ETC matrix Consistent, Semi - Consistent, Inconsistent and performs SPA Double Min Min for which the average values are computed to generate their corresponding makespan values. From the results obtained, the proposed method is evaluated which gives good results.

Figures 9, 10 and 11 shows value of average makespan generated by SPA based Double Min Min outperforms the SPA based Min Min and Max Min for Consistent, Semi-consistent and inconsistent ETC matrices executed for 256 tasks on 8 machines. The results shows that SPA based Double Min Min gives better makespan even as the number of tasks and machines are increased in the HPGRID system

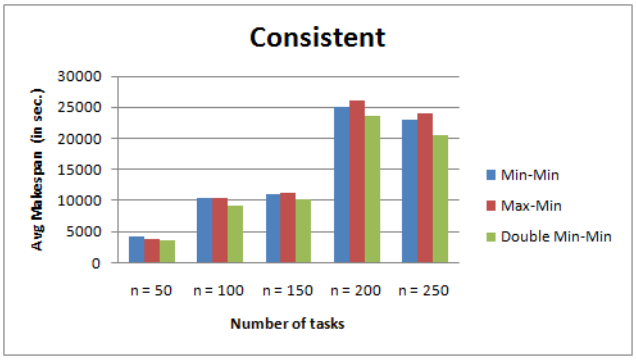


Fig. 9: SPA Double Min Min Scheduling for 256 tasks on 8 CGPs for Consistent ETC

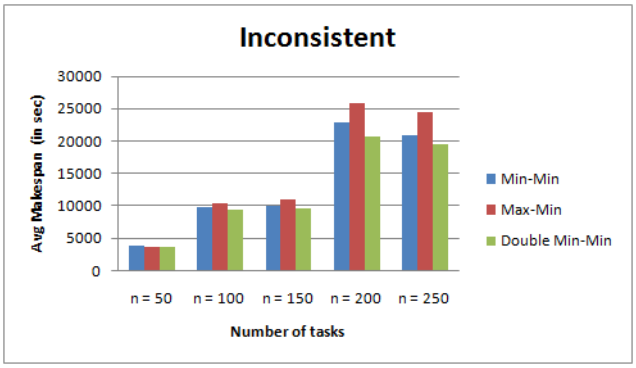


Fig. 10: SPA Double Min Min Scheduling for 256 tasks on 8 CGPs for Inconsistent ETC

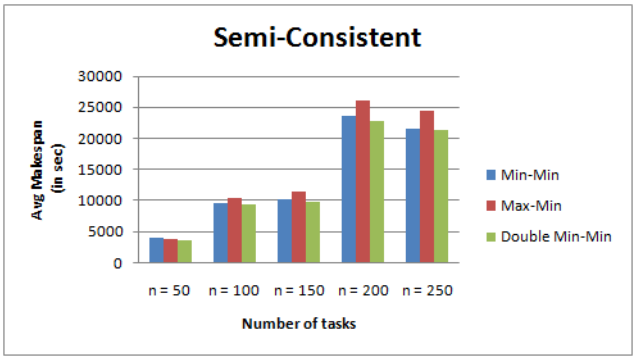


Fig. 11: SPA Double Min Min Scheduling for 256 tasks on 8 CGPs for Semi-Consistent ETC

The resource density for all the 3 types of Matrix has been analyzed on executing 256 tasks on 8 CGPs. The following figures 12, 13 and 14 shows the graph generated from the values obtained during simulation process. For all the three Matrix types, the proposed algorithm SPA Double Min Min gives better resource density than the SPA Min Min and SPA Max Min. Figures 15, 16 and 17 shows value of load factor in percentage generated by SPA based Double Min Min which outperforms the SPA based Min Min and Max Min for Consistent, inconsistent and Semi-consistent ETC matrices executed for 256 tasks on 8 machines.

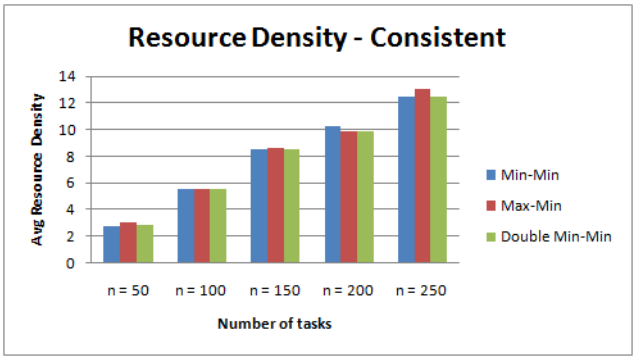


Fig. 12: Resource Density on SPA Double Min Min Scheduling for 256 tasks on 8 CGPs for Consistent ETC

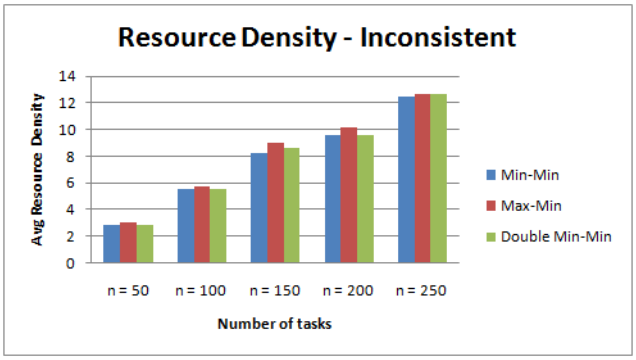


Fig. 13: Resource Density on SPA Double Min Min Scheduling for 256 tasks on 8 CGPs for Inconsistent ETC

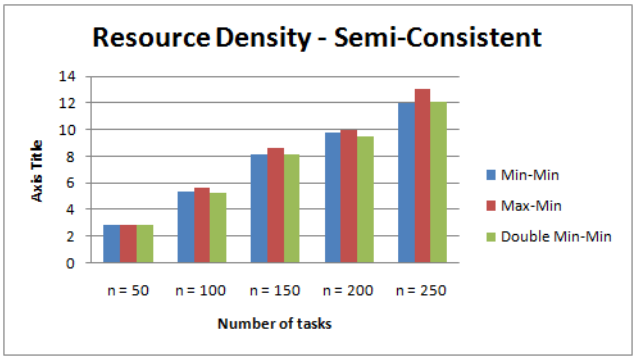


Fig. 14: Resource Density on SPA Double Min Min Scheduling for 256 tasks on 8 CGPs for Semi-Consistent ETC

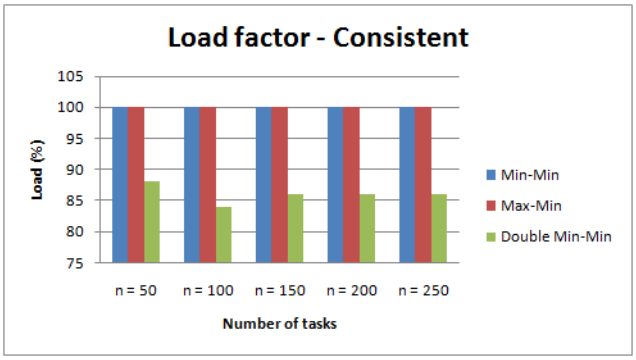


Fig. 15: Load Factor on SPA Double Min Min Scheduling for 256 tasks on 8 CGPs for Consistent ETC

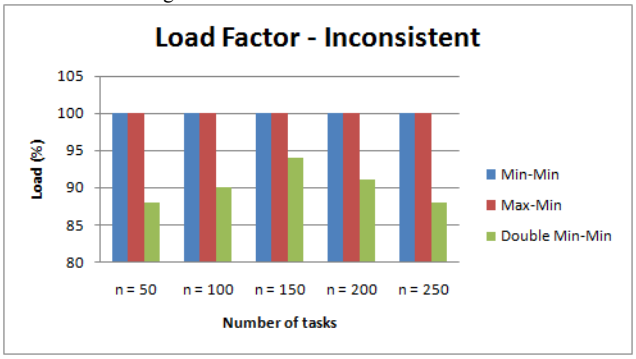


Fig. 16: Load Factor on SPA Double Min Min Scheduling for 256 tasks on 8 CGPs for Inconsistent ETC

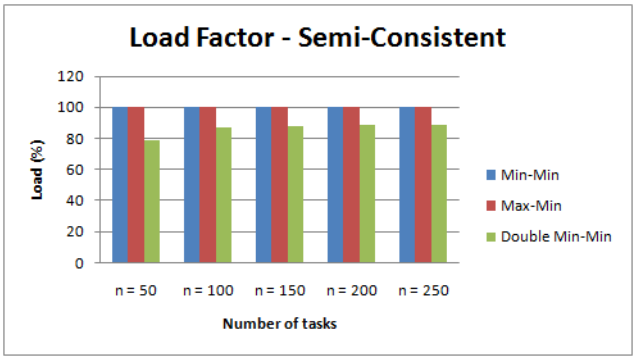


Fig. 17: Load Factor on SPA Double Min Min Scheduling for 256 tasks on 8 CGPs for Semi-Consistent ETC

## 9. Conclusions and Future Work

To achieve high throughput computing in a Hypercubic P2P Grid environment, the SPA Double Min Min scheduling algorithm was proposed. To enhance Load balancing, we find the Mean CT for the scheduled results and reselect the tasks that are greater than Mean CT. For the reselected task, the SPA based Min Min is computed. We have separately computed the makespan value for all the three types of ETC Matrix, where we concluded that the proposed algorithm gives better results than the conventional algorithms, SPA based Min-Min and Max-Min. The Resource Density and Load factor is also calculated for SPA Double Min Min Average, when compared the proposed work gives better value than the SPA based Min Min and Max Min for all the three sets. Evaluation of our new heuristic was done through a simulation environment called Gridsim. The experimental results show that the SPA Double Min Min outperforms the SPA based Min-Min and Max-Min heuristics.

The future work will focus on the research on scheduling occasion, resource description, fault tolerant processing and other aspects, in order to further optimize and improve the system. In addition to this, many issues remain open, like deadline of task, QoS of task, execution cost on each resource, communication cost etc. Some of the above mentioned issues are under consideration as a part of further work.

## References

[1] A. Abraham, Rajkumar Buyya, and Baikunth Nath. Nature's heuristics for scheduling jobs on computational grids. In *Proceedings of 8th IEEE International Conference on Advanced Computing and Communications (ADCOM 2000)*, pages 45–52, 2000.

[2]K. Aida, Atsuko Takefusa, Hidemoto Nakada, Satoshi Matsuoka, Satoshi Sekiguchi, and Umpei Nagashima. Performance evaluation model for scheduling in global computing systems. *International Journal of High Performance Computing Applications*, 14(3):268–279, 2000.

[3]T.D Braun , Howard Jay Siegel, Noah Beck, Ladislau L. Boloni, Muthucumaru Maheswaran, Albert I. Reuther, James P. Robertson, Mitchell D. Theys, Bin Yao, Debra Hensgen, and Richard F. Freund. et al, 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):810–837, 2001.

[4]R. Buyya and Manzur Murshed. Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing. *Concurrency and Computation: Practice and Experience (CCPE)*, 14(13):1175–1220, 2002.

[5]H. Casanova. Simgrid: A toolkit for the simulation of application scheduling. In *CCGRID '01: Proceedings of the 1st International Symposium on Cluster Computing and the*

*Grid*, page 430 - 437, Washington, DC, USA, 2001. IEEE Computer Society.

[6]H. Casanova, Dmitrii Zagorodnov, Francine Berman, and Arnaud Legrand. Heuristics for scheduling parameter sweep applications in grid environments. In *HCW '00: Proceedings of the 9th Heterogeneous Computing Workshop*, page 349 - 363, Washington, DC, USA, 2000. IEEE Computer Society.

[7] S.S Chauhan and R.C. Joshi. A weighted mean time min-min max-min selective scheduling strategy for independent tasks on grid. In *Advance Computing Conference (IACC), 2010 IEEE 2nd International*, pages 4 –9, 19-20 2010.

[8]H. Decai, Yuan Yuan, Zhang Li-jun, and Zhao Ke-qin. Research on tasks scheduling algorithms for dynamic and uncertain computing grid based on a+bi connection number of SPA. *Journal of Software*, 4(10):1102–1109, 2009.

[9]K. Etminani and M. Naghibzadeh. A min min max-min selective algorithm for grid task scheduling. In *Proceedings of the 3rd IEEE/IFIP International Conference in Central Asia*, 2007.

[10]R. Freund, Michael Gherrity, Stephen Ambrosius, Mark Campbell, Mike Halderman, Debra Hensgen, Elaine Keith, Taylor Kidd, Matt Kussow, John D. Lima, Francesca Mirabile, Lantz Moore, Brad Rust, and H. J. Siegel. Scheduling resources in multi-user, heterogeneous, computing environments with smartnet. In *7TH IEEE Heterogeneous Computing Workshop (HCW 98)*, pages 184–199, 1998.

[11]R. Freund and Taylor Kidd Nccosc. Smartnet: A scheduling framework for heterogeneous computing. In *Proceedings of the International Symposium on Parallel Architectures, Algorithms, and Networks (ISPAN-96)*, pages 514–521. IEEE Computer Society Press, 1996.

[12]X. He, XianHe Sun, and Gregor von Laszewski. QoS guided min-min heuristic for grid task scheduling. *Journal of Computer Science and Technology*, 18(4):442–451, 2003.

[13]H. Izakian, Ajith Abraham, and Vaclav Snasel. Comparison of heuristics for scheduling independent tasks on heterogeneous distributed environments. In *CSO '09: Proceedings of the 2009 International Joint Conference on Computational Sciences and Optimization*, pages 8–12, Washington, DC, USA, 2009. IEEE Computer Society.

[14] Z. Ke-Qin and Xuan Ai-li. Set pair theory- a new theory method of non-define and its applications. *Systems engineering,*, 1:18–24, 1996.

[15]M. Maheswaran Shoukat, Muthucumaru Maheswaran, Shoukat Ali, Howard Jay Siegel, Debra Hensgen, and Richard F. Freund. Dynamic mapping of a class of independent tasks onto heterogeneous computing systems. *Journal of Parallel and Distributed Computing*, 59:107–131, 1999.

[16]J. Sherwani, Nosheen Ali, Nausheen Lotia, Zahra Hayat, and Rajkumar Buyya. Libra: a computational economy-based job scheduling system for clusters. *Journal of Software: Practice and Experience*, 34(6):573–590, 2004.

[17] C. Shin Yeo and Rajkumar Buyya. Pricing for utility-driven resource management and allocation in clusters. *International Journal of High Performance Computing Applications*, 21(4):405–418, 2007.

[18] H. J. Song, X.Liu, D.Jakobsen, R.Bhagwan, X.Zhang, K.Taura, and A.Chien. The Micro-Grid: a scientific tool for modeling computational grids. *Scientific Programming,* 8(3):127– 141, 2000.

[19]P. Trunfio and Domenico Talia. Toward a synergy between P2P and Grids. *IEEE Internet Computing*, 7(4):94–96, 2003.

[20]X. Wen, Wei Zhao, and Fan xing Meng. Researchof Grid Scheduling Algorithm based on P2P GridModel. In *International Conference on Electronic Commerce and Business Intelligence*,pages 41–44, 2009.

[21]M. Wu, Wei Shu, and Hong Zhang. Segmented min-min: A static mapping algorithm for metatasks on heterogeneous computing systems. In *HCW '00: Proceedings of the 9th Heterogeneous Computing Workshop*, page 375, Washington, DC, USA, 2000. IEEE Computer Society.

[22] J. Yun-Liang and Xu Cong-Fu. Advances in set pair analysis theory and its applications. Computer Science. *Computer Science*,33(1):205–209, 2006.

**Doreen Hephzibah Miriam** is currently a Research Scholar at the Department of Computer Science and Engineering at Anna University, Chennai. She received her B.Tech in Information Technology from Madras University, Chennai, and M.E degree in Computer Science and Engineering from Anna University, Chennai. Her research interests include parallel and distributed computing, peer to peer computing and grid computing.

**K.S. Easwarakumar** is a Professor & Head at the Department of Computer Science and Engineering at Anna University, Chennai. He received his M.Tech in Computer and Information Sciences from Cochin University of Science and Technology, Cochin and Ph.D in Computer Science and Engineering from Indian Institute of Technology, Madras. His research interests include parallel and distributed computing, Data Structures and Algorithms, Graph Algorithms, Parallel Algorithms, Computational Geometry, Theoretical Computer Science and Molecular computing.