CHAPTER 5

GRID SCHEDULING USING MIN-MEAN HEURISTIC SCHEDULING ALGORITHM

Scheduling becomes one of the main challenging issues in the research area of the grid. Scheduling of metatasks to the resources in a grid environment proves to be a NP-complete problem, which can be solved by heuristic approaches (Batista and Fonseca 2011). This chapter proposes heuristic algorithm to solve the scheduling problem. The optimization criteria in scheduling are the makespan. The makespan measures the throughput of the system. The aim of the scheduling algorithm is to reduce the makespan.

Eleven heuristic approaches analyzed and implemented. The assumptions taken for analyzing the heuristic algorithms are (Braun et al 2001):

- An off-line or static mapping made.
- The tasks considered for mapping are independent metatask.
- Aim is to minimize the makespan of the metatasks.
- The tasks executes in order.
- The sizes of the tasks and the number of resources are static and known beforehand.
- The estimate of the expected execution time of each job on each resource represented within an ETC matrix of size...
m*n, where m-represents the number of tasks and n-represents the number of resources.

- \( T = \{t_1, t_2, t_3, \ldots, t_m\} \) represents the task set.
- \( R = \{r_1, r_2, r_3, \ldots, r_n\} \) represents the resource set.
- \( ETC_{ij} \)-expected execution time of task \( t_i \) on resource \( r_j \).
- \( CT_{ij} \)-expected completion time of task \( t_i \) on resource \( r_j \).
- \( BT_j \)-ready time of resource \( r_j \).
- \( CT_{ij} = ETC_{ij} + BT_j \).
- \( BT_j = ETC_{ij} + BT_j \).
- \( \text{Makespan} = \max (BT_j) \) or \( \max (CT_{ij}), 1 \leq i \leq m, 1 \leq j \leq n \).

Eleven heuristic algorithms are OLB, MET, MCT, Min-min, Max-min, Duplex, GA, SA, GSA, Tabu, and A*. The results show that the Min-min algorithm performs well compared to the other heuristic algorithms (Braun et al 2001).

In general, heuristic scheduling algorithm classified as online or batch mode scheduling algorithm (Xhafa and Abraham 2010). In an online mode scheduling, the task schedules to the available resources as soon it arrives. In the batch mode scheduling, tasks are independent. There is no order in the execution of the tasks, and the tasks schedules in batch every time. The batch mode scheduling technique fixes the number of tasks to be executed and the number of resources for executing the tasks. The batch mode scheduling considers that there is no task failure, and resources are available all the time. Batch mode scheduling considers the dynamics of the resource but does not consider the dynamics of the tasks. The advantage of batch mode
scheduling is that, it takes better chance in mapping the tasks to the available resources based on tasks and resource characteristics (Lee et al 2011). The proposed Min-mean heuristic algorithm is an algorithm for the batch mode scheduling.

To evaluate the performance of static mapping heuristics, an Expected Time to Compute (ETC) matrix used (Braun et al 2001).

The ETC matrix has m x n entries, where ‘m’ is the number of independent tasks to be scheduled, and ‘n’ is the number of available resources. The expected completion time of the task \( t_i \) on resource \( m_j \) is,

\[
CT(t_i,m_j) = BT(m_j) + ETC(t_i, m_j) \quad \text{(5.1)}
\]

\( BT(m_j) \) is the ready time of the resource \( m_j \).

The aim of the heuristic scheduling algorithm is to minimize the metatask execution time, and known as makespan.

Makespan in terms of the completion time of machines is as follows:

\[
\text{makespan} = \max \left[ \text{completion} \left[ i \right] \right] / i \in \text{machines} \quad \text{(5.2)}
\]

5.1 COMPARISON OF ELEVEN HEURISTIC ALGORITHMS

The comparison of eleven heuristic algorithms based on the three characteristics resource heterogeneity, task heterogeneity, and consistency. The characteristics of the ETC matrices were varied in an attempt to represent a range of possible heterogeneous computing environment for the simulation studies. To further vary the ETC matrix in an attempt to capture more aspect of realistic mapping situations, a different ETC matrix consistency were used.
<table>
<thead>
<tr>
<th>Consistency</th>
<th>Performance order of eleven heuristics from best case to worst case</th>
<th>Performance order of eleven heuristics from best case to worst case</th>
</tr>
</thead>
<tbody>
<tr>
<td>Both cases of High Resource Heterogeneity (ie., High Task High Resource Heterogeneity and Low Task High Resource Heterogeneity)</td>
<td>Both cases of Low Resource Heterogeneity (ie., High Task Low Resource Heterogeneity and Low Task Low Resource Heterogeneity)</td>
<td></td>
</tr>
</tbody>
</table>
The performance order of the eleven heuristics from the best case to the worst case is listed in the Table 5.1. From the Table 5.1, it shows clearly that GA provides the best mapping results. The mapping decision based on performance of the Min-min solution. The best solution of GA is the populations seeded with the Min-min heuristic solution.

The Min-min heuristic algorithm gives the second best mapping results and performed extremely well for consistent ETC matrices and partially-consistent matrices. Min-min algorithm can make intelligent decisions in mapping metatasks to machines. The mapping minimizes the task completion time, which leads to good resource utilization, and good makespan (Braun et al 2001, Ali et al 2000).

5.2 MIN-MIN HEURISTIC SCHEDULING ALGORITHM

One of the best heuristic scheduling algorithms for mapping the metatasks to the resources is the Min-min algorithm (Braun et al 2001). Min-min algorithm starts with a set of all unmapped tasks. All unmapped tasks, placed in a task set ‘T’. Min-min algorithm works in two phases. In phase 1, the algorithm calculates the expected completion time for each task on each resource. It calculates the minimum completion time for each task. In phase 2, the algorithm selects the task with the overall minimum completion time and allocates to the resource. The process repeats until all the tasks get mapped.

Min-min heuristic scheduling algorithms executes all small tasks first, and then executes the long task. The demerit of Min-min algorithms is that, it is unable to balance the load well since it usually allocates the smaller task first. The shorter task will complete the execution earlier, and some of the resources would be sitting idle while executing the longer task.
5.3 MIN-MEAN HEURISTIC SCHEDULING ALGORITHM

The proposed Min-mean heuristic scheduling algorithm works in two phases. In the first phase, Min-mean heuristic scheduling algorithm starts with a set of all unmapped tasks. The algorithm calculates the completion time for each task on each resource and finds the minimum completion time for each task. From that group, the algorithm selects the task with the overall minimum completion time and allocates to the appropriate resource. Removes the task from the task set. This process repeats until all the tasks get mapped. The algorithm calculates the total completion time of all the resources and the mean completion time. In the second phase, the algorithm selects the resource whose completion time is greater than mean completion time. The algorithm arranges the selected resources in the decreasing order of the completion time. The algorithm reschedules the tasks assigned on the selected resources to the resource, whose completion time is less than the mean completion time. Compute makespan.

The proposed Min-mean algorithm reschedules the selected tasks and achieves better makespan than the Min-min algorithm.

Algorithm 5.1: Min-mean

Variables

- ETC\(t_i, r_j\) - Expected execution time
- BT\(r_j\) - burst time of resource
- R - Number of resources
- T - Number of tasks
- CT\(i,j\) - Completion time of task
- MinCT\(j\) - Minimum completion time of task
- TotalCT - Total completion time of resources
- MeanCT - Mean completion time of resources
- nooftasks\(R\) - Number of tasks assigned to a resource R
- Tasklist - represents the tasks \(t_i\) assigned to a resource \(r_j\)
Noofselmac - Number of selected resources whose BT > MeanCT
Selectedmachine - represents the list of selected resource
Previous - Expected execution time of the selected task on the first resource \( r_0 \)

**Input**

\( R, T, \) and ETC

**Output**

\( \text{makespan} \)

**Phase 1:**

Initialize \( BT[r_j] = 0, \, r_j \in R, \, 1 \leq j \leq m \)

while there are tasks to be scheduled do

for each task \( t_i \) to be scheduled \( t_i \in T \) do

for each resource \( r_j \) available, \( r_j \in R, \, 1 \leq j \leq m \) do

// Calculate the completion time of task \( t_i \) on resource \( r_j \)

\[ CT_{ij} = ETC[t_i, r_j] + BT[r_j] \]

end for

end for

// Find the minimum completion time of task \( t_i \) on resource \( r_j \)

for each task \( t_i \) to be scheduled \( t_i \in T \) do

MinCT\(_i\)=\( CT[t_i,r_0] \)

end for

for each task \( t_i \) to be scheduled \( t_i \in T \) do

for each resource \( r_j \) available, \( r_j \in R, \, 1 \leq j \leq m \) do

if (MinCT\(_i\) > CT\(_{ij}\))

\[ \text{MinCT}_i = CT_{ij} \]

end if

end for

resource=j, task=i
end for
end for

// Update the ready time of resource
Update BT\textsubscript{resource}=BT\textsubscript{resource} + ETC[task, resource]
Increment nooftasks[resource]
Update tasklist[resource][nooftasks[resource]]=task
Allocate task $t_i$ to the resource $r_j$
Remove task $t_i$ from task set T
Repeat until all tasks are assigned to resources
end while

// Calculate the makespan
makespan = BT[r_{0}]
for each resource $r_j$ available, $r_j \in R, 1 \leq j \leq m$
do
if( makespan < (BT_{j}))
    makespan = BT_{j}
end for

Phase 2:

// Calculate the total completion time of the resource
for each resource $r_j$ available, $r_j \in R, 1 \leq j \leq m$
do
    TotalCT=TotalCT+ BT_{j}
end for

// Calculate the mean completion time of the resource
MeanCT=TotalCT/No. of resources

// Select resources whose $BT_{j} > MeanCT$
for each resource $r_j$ available, $r_j \in R, 1 \leq j \leq m$
do
    if (BT\(_j\) > MeanCT)
        selectedmachine\([i]\) = j
        i++
        noofselmac++
    end for

// Order the selected resources \(r_k\) in the decreasing order of CT\(_j\)
for each resource \(r_k\) selected, 1 \(\leq\) noofselmac
    do
        for \(k \leq j \leq \) noofselmac
            do
                if (BT[selectedmachine[k]] > BT[selectedmachine[j]])
                    temp = selectedmachine[k]
                    selectedmachine[k] = selectedmachine[j]
                    selectedmachine[j] = temp
                end if
            end for
        end for
    end for

// Find the optimum schedule for the selected machines
for each resource \(r_k\) selected, 1 \(\leq\) noofselmac
    do
        l = selectedmachine[k]
        for each task \(t_i\) of resource \(r_k\) to be scheduled, 1 \(\leq\) nooftasks[j]
            do
                Previous = ET[tasklist[k][i]][0]
                for each resource \(r_j\) available, \(r_j \in R, 1 \leq j \leq m\)
                    if ((BT\(_j\) + ET[tasklist[k][i]][j]) < MeanCT) &&
                        ((BT\(_j\) + ET[tasklist[k][i]][j] < BT\(_l\)) &&
                            (ET[tasklist[k][i]][j] < Previous)
                end if
            end for
        end do
    end do
Update Previous = ET\{tasklist[k][i][j]\] 
\text{newmac} = j 
end if 
end for 

\text{BT}_{\text{newmac}} = \text{BT}_{\text{newmac}} + ET\{tasklist[k][i][\text{newmac}]\] 
\text{BT}_i = \text{BT}_i - ET\{tasklist[l][i][l]\] 
Assign task \( t_i \) to resource \( r_{\text{newmac}} \) 
end for 
end for 

// Calculate the makespan 
\text{makespan} = \text{BT}\{r_0\] 
for each resource \( r_j \) available, \( r_j \in \mathcal{R}, 1 \leq j \leq m \) do 
if( \text{makespan} < (\text{BT}_j)) 
\text{makespan} = \text{BT}_j 
end if 
end for 

5.4 EXPERIMENTAL RESULTS AND DISCUSSION

In this thesis, the Min-mean heuristic scheduling algorithm tested with the simulated benchmark instances. 100 instances of the benchmark model and each with twelve possible types of ETC matrices based on three following metrics: task heterogeneity, resource heterogeneity, and consistency used. This simulation model assumes 16 resources currently available in the grid, and 512 tasks to be scheduled. The thesis uses \( 100 \times 12 \) matrices to test the existing and proposed algorithms.

The instances labeled u_x_yyzz.k described as follows:

‘u’ means uniform distribution used to generate ETC matrices.
‘x’ represents the consistency.
‘c’ consistent - if any one task $t_i$ executes faster in resource $r_j$ than resource $r_k$, than resource $r_j$ executes all tasks faster than resource $r_k$.

‘ic’ inconsistent - if a resource $r_j$ executes faster than resource $r_k$ for few tasks, and resource $r_j$ executes slower than resource $r_k$ for other tasks.

‘pc’ partially consistent - includes a consistent sub matrix.

‘yy’ - task heterogeneity is the variation in the execution time of all the tasks on a given resource (hi-high, lo-low).

‘zz’ - resource heterogeneity is the variation in the execution time of a task on all resources (hi-high, lo-low).

### 5.4.1 Simulation Parameters

The simulation parameters (Braun et al 2001) measure the performance of Min-mean heuristic scheduling algorithm and shown in Table 5.2.

#### Table 5.2 Simulation Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of Tasks</td>
<td>512</td>
</tr>
<tr>
<td>No. of Resources</td>
<td>16</td>
</tr>
<tr>
<td>Task Heterogeneity</td>
<td></td>
</tr>
<tr>
<td>Upper Bound- $\Phi_b$</td>
<td>3000</td>
</tr>
<tr>
<td>Lower Bound- $\Phi_b$</td>
<td>100</td>
</tr>
<tr>
<td>Resource Heterogeneity</td>
<td></td>
</tr>
<tr>
<td>Upper Bound- $\Phi_r$</td>
<td>1000</td>
</tr>
<tr>
<td>Lower Bound- $\Phi_r$</td>
<td>10</td>
</tr>
</tbody>
</table>

Table 5.3 gives a sample ETC matrix, the expected execution time of three tasks ($t_1, t_2, t_3$) on two machines ($m_1, m_2$).
Table 5.3 Expected Execution Time of Three Tasks on Two Machines

<table>
<thead>
<tr>
<th></th>
<th>m1</th>
<th>m2</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>t2</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>t3</td>
<td>5</td>
<td>9</td>
</tr>
</tbody>
</table>

This sample ETC matrix clearly explains how proposed Min-mean heuristic scheduling algorithm performs better than the Min-min algorithm. The graph in Figure 5.1 represents the static mapping of tasks to machines based on Min-min. Min-min algorithm gives a makespan of 8 seconds.

Figure 5.1 Results of Min-min Algorithm

The steps of the execution of Min-min algorithm and the proposed Min-mean heuristic scheduling algorithm are as follows:
The Min-mean heuristic scheduling algorithm works as follows:

**Step 1:** The mean completion time for the sample ETC matrix can be calculated using the following relation:

\[ \text{MeanCT} = \text{CT}_{m1} + \text{CT}_{m2} \]

where,

\[ \text{CT}_{m1}: \text{Completion time of all tasks on machine m1} \]

\[ \text{CT}_{m2}: \text{Completion time of all tasks on machine m2.} \]

MeanCT = 4 seconds.

**Step 2:** The algorithm selects the tasks on machine m1 as shown in Figure 5.2 because, CTm1 > MeanCT.

![Figure 5.2 Selected Tasks on Machine m1](image)
**Step 3:** The algorithm reschedules those tasks on machine m1 to the machine m2 because the expected execution time of the tasks on machine m2 is less than the MeanCT.

![Figure 5.3 Results of Min-mean Algorithm](image)

The graph in Figure 5.1 and 5.3 reveal clearly that Min-mean algorithm reduces makespan than the Min-min algorithm. Thus, the Min-mean heuristic scheduling algorithm performs better than Min-min algorithm.

The comparison results of Figure 5.1 and 5.3 as following:

- The free times of the machine m2 reduced.
- The load balances well, for the machines m1 and m2.

The result of the proposed Min-mean heuristic scheduling algorithm compared with the existing MET, MCT, Min-min, Max-min heuristic scheduling algorithms.

The proposed method runs all the 100×12 instances, and the average values of 100 instances listed in Table 5.4.
Table 5.4 Makespan Values of MET, MCT, Min-min, Max-min, and Min-mean Heuristic Scheduling Algorithms

<table>
<thead>
<tr>
<th>Instances</th>
<th>MET</th>
<th>MCT</th>
<th>Min-min</th>
<th>Max-min</th>
<th>Min-mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>u-c-hihi</td>
<td>4.58E+07</td>
<td>1.16E+07</td>
<td>8298107.34</td>
<td>1.38E+07</td>
<td>8166690.12</td>
</tr>
<tr>
<td>u-c-hilo</td>
<td>401179.15</td>
<td>110423</td>
<td>79940.04</td>
<td>136134.73</td>
<td>79818.75</td>
</tr>
<tr>
<td>u-c-lohi</td>
<td>1490832.68</td>
<td>387649.23</td>
<td>267044.88</td>
<td>454895.74</td>
<td>263117.09</td>
</tr>
<tr>
<td>u-c-lolo</td>
<td>14372.99</td>
<td>3584.36</td>
<td>2600.80</td>
<td>4556.56</td>
<td>2575.61</td>
</tr>
<tr>
<td>u-ic-hihi</td>
<td>5394238.98</td>
<td>4184438.56</td>
<td>3565661.22</td>
<td>2.16E+07</td>
<td>3402895.84</td>
</tr>
<tr>
<td>u-ic-hilo</td>
<td>39398.43</td>
<td>37616.09</td>
<td>32412.49</td>
<td>218925.21</td>
<td>30211.03</td>
</tr>
<tr>
<td>u-ic-lohi</td>
<td>143873.82</td>
<td>142816.69</td>
<td>125061.74</td>
<td>682872.51</td>
<td>118455.87</td>
</tr>
<tr>
<td>u-ic-lolo</td>
<td>1347.46</td>
<td>1300.89</td>
<td>1062.34</td>
<td>7289.06</td>
<td>1011.65</td>
</tr>
<tr>
<td>u-pc-hihi</td>
<td>2.13E+07</td>
<td>6295863.23</td>
<td>4602969.66</td>
<td>1.85E+07</td>
<td>4520263.85</td>
</tr>
<tr>
<td>u-pc-hilo</td>
<td>215401.68</td>
<td>60204.34</td>
<td>44979.51</td>
<td>180125.79</td>
<td>44743.87</td>
</tr>
<tr>
<td>u-pc-lohi</td>
<td>740790.39</td>
<td>211425.70</td>
<td>169090.74</td>
<td>575055.24</td>
<td>163015.57</td>
</tr>
<tr>
<td>u-pc-lolo</td>
<td>7244.55</td>
<td>1967.45</td>
<td>1586.50</td>
<td>6254.76</td>
<td>1573.33</td>
</tr>
</tbody>
</table>

Figure 5.4 Comparison of the Makespan Values obtained by MET, MCT, Min-min, Max-min, and Min-mean Heuristic Algorithms
Makespan computed for the existing and the proposed heuristic scheduling algorithm using the benchmark model listed in Table 5.4. Figure 5.4 represents the graphical representation of all types of instances’ average makespan values. From the Table 5.4 and Figure 5.4, it is evident that the proposed Min-mean heuristic scheduling algorithm produces better results for all the twelve types of ETC matrices than the existing heuristic scheduling algorithms. Figures 5.5 to 5.8 illustrates the comparison of the average makespan values of MET, MCT, Min-min, Max-min, and Min-mean Heuristic Algorithms in high task high machine, high task low machine, low task high machine and low task low machine heterogeneity respectively.

![Comparison of Makespan Values](image)

Figure 5.5  Comparison of the Makespan Values obtained by MET, MCT, Min-min, Max-min, and Min-mean Heuristic Algorithms for High Task and High Machine Heterogeneity
Figure 5.6  Comparison of the Makespan Values obtained by MET, MCT, Min-min, Max-min, and Min-mean Heuristic Algorithms for High Task and Low Machine Heterogeneity

Figure 5.7  Comparison of the Makespan Values obtained by MET, MCT, Min-min, Max-min, and Min-mean Heuristic Algorithms for Low Task and High Machine Heterogeneity
From Figures 5.5 to 5.8 the performance of the existing and proposed methods can be easily analyzed with respect to the makespan. The above results show that the proposed algorithm produces makespan better than the other exiting approaches for all twelve classes of ETC matrix.

Table 5.5 show the percentage of improvement of the proposed Min-mean heuristic scheduling algorithm over Min-min and the graphical illustration shown in Figure 5.9.
### Table 5.5 Percentage of Improvement of Min-mean over Min-min

<table>
<thead>
<tr>
<th>Consistency</th>
<th>Improvement over Min-min</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Inconsistent</strong></td>
<td></td>
</tr>
<tr>
<td>High-High</td>
<td>4.56%</td>
</tr>
<tr>
<td>High-Low</td>
<td>6.79%</td>
</tr>
<tr>
<td>Low-High</td>
<td>5.28%</td>
</tr>
<tr>
<td>Low-Low</td>
<td>4.77%</td>
</tr>
<tr>
<td><strong>Consistent</strong></td>
<td></td>
</tr>
<tr>
<td>High-High</td>
<td>1.58%</td>
</tr>
<tr>
<td>High-Low</td>
<td>0.15%</td>
</tr>
<tr>
<td>Low-High</td>
<td>1.47%</td>
</tr>
<tr>
<td>Low-Low</td>
<td>0.97%</td>
</tr>
<tr>
<td><strong>Partially Consistent</strong></td>
<td></td>
</tr>
<tr>
<td>High-High</td>
<td>1.80%</td>
</tr>
<tr>
<td>High-Low</td>
<td>0.52%</td>
</tr>
<tr>
<td>Low-High</td>
<td>3.59%</td>
</tr>
<tr>
<td>Low-Low</td>
<td>0.83%</td>
</tr>
</tbody>
</table>

![Figure 5.9](image)  

**Figure 5.9** Percentage of Improvement of Min-mean Algorithm over Min-min Algorithm
Figures 5.10 to 5.12 represents the percentage of improvement of Min-mean heuristic scheduling algorithm over Min-min in all 12 different types of instances based on three metrics: Job, machine heterogeneity and consistency.

![Inconsistent](image)

**Figure 5.10** Percentage of Improvement of Min-mean over Min-min for Inconsistent Task Machine Heterogeneity

![Consistent](image)

**Figure 5.11** Percentage of Improvement of Min-mean over Min-min for Consistent Task Machine Heterogeneity
Figure 5.12 Percentage of Improvement of Min-mean over Min-min for Partially Consistent Task Machine Heterogeneity

From Figures 5.10 to 5.12, the illustration clearly proves that the Min-mean heuristic scheduling algorithm produces better makespan than the Min-min in all twelve types of ETC matrices.

Figures 5.13 to 5.16 show the percentage of improvement of Min-mean Heuristic Algorithm over Min-min Algorithm in high task high machine, high task low machine, low task high machine, and low task low machine heterogeneity respectively.

Figure 5.13 Percentage of Improvement of Min-mean over Min-min for High Task High Machine Heterogeneity
Figure 5.14  Percentage of Improvement of Min-mean over Min-min for High Task Low Machine Heterogeneity

Figure 5.15  Percentage of Improvement of Min-mean over Min-min for Low Task High Machine Heterogeneity

Figure 5.16  Percentage of Improvement of Min-mean over Min-min for Low Task Low Machine Heterogeneity
From Figures 5.13 to 5.16, it becomes evident that the proposed Min-mean Heuristic Scheduling Algorithm performs well for all Task Machine Heterogeneity.

5.5 CONCLUSION

This chapter proposed Min-mean heuristic scheduling algorithm. The performance of the proposed algorithm measures with the help of makespan. The results compare with the existing algorithms with the help of the simulation benchmark instances. The proposed algorithm produced better results than the other algorithms. The makespan value of the proposed Min-mean heuristic scheduling algorithm was less than the existing heuristic scheduling algorithm.

The Min-min algorithm performs well compared to the other heuristic algorithms. The proposed Min-mean heuristic scheduling algorithm produces better makespan than the Min-min algorithm. The Min-min algorithm takes $O(n^2m)$ time, where ‘n’ represents the number of tasks and ‘m’ represents the number of machines. The Min-mean heuristic scheduling algorithm also takes $O(n^2m)$ time. The proposed Min-mean algorithm reschedules the selected tasks and achieves better makespan than the Min-min algorithm within the same efficiency class. With respect to the optimization criteria, makespan, the proposed Min-mean well suits for the grid scheduling.