Pak. j. life soc. Sci. (2011), 9(1): 70-79

Pakistan Journal of
Life and Social Sciences

# Comparative Study on Performance of Scheduling Heuristics in Heterogeneous Computing Environment

Ehsan Ullah Munir*, Sheraz Anjum[1], Muhammad Wasif Nisar, Waqas Anwar[2] and Kashif Ayyub

Department of Computer Science, COMSATS Institute of Information Technology, Quaid Avenue, Wah Cantt, Pakistan;

[1]Department of Computer Engineering, COMSATS Institute of Information Technology, Wah Cantt, Pakistan;

[2]Department of Computer Science, COMSATS Institute of Information Technology, Abbottabad, Pakistan

## Abstract

**Heterogeneous computing (HC) environment consists of different resources connected with high-speed links to provide a variety of computational capabilities for computing-intensive applications having multifarious computational requirements. The problem of optimal assignment of tasks to machines in HC environment is proven to be NP-complete requiring use of heuristics to find the near optimal solution. In this work we conduct a performance study of task scheduling heuristics in HC environment. In present study overall implemented 16 heuristics, among them 7 are proposed in this paper. The range bar for the average makespan of each heuristic shows a 95% confidence interval for the corresponding average makespan. From the values it is clear that for high values of $V_{machine}$ H16 is the best heuristic. And in all other cases one of the preoposed heuristic H2 or H5 outperforms all other heuristics. Based on experimental results, specify the circumstances under which one heuristic will outperform the others.**

**Keywords:** Heterogeneous computing, Task scheduling, Performance evaluation, Task Partitioning heuristic

## Introduction

Heterogeneous computing (HC) environment consists of different resources connected with high-speed links to provide a variety of computational capabilities for computing - intensive applications having multifarious computational requirements (Ali et al., 2005). In HC environment an application is decomposed into various tasks and each task is assigned to one of the machines, which is best suited for its execution to minimize the total execution time. Therefore, an efficient assignment scheme responsible for allocating the application tasks to the machines is needed; formally this problem is named task scheduling (El-Rewini et al., 1994). Developing such strategies is an important area of research and it has gained a lot of interest from researchers (Barbulescu et al., 2004; Shestak et al., 2005; Shivle et al., 2006). The problem of task scheduling has gained tremendous attention and has been extensively studied in other areas such as computational grids (Foster and Kesselman, 1999) and parallel programs (Kwok and Ahmad, 1999).

The problem of an optimal assignment of tasks to machines is proven to be NP-complete requiring use of heuristics to find the near-optimal solution (Baca, 1989; Ibarra and Kim, 1977). Plethora of heuristics has been proposed for assignment of tasks to machines in HC environment (Maheswaran et al., 1999; Wu et al., 2000; Sakellariou and Zhao, 2004; Kwok et al., 2006; Kim et al., 2007). Each heuristic has different underlying assumptions to produce near optimal solution however no work reports which heuristic should be used for a given set of tasks to be executed on different machines.

Provided with a set of tasks $\{t_1,t_2,...,t_m\}$, a set of machines $\{m_1,m_2,...,m_n\}$ and expected time to compute (ETC) of each task $t_i$ on each machine $m_j$, $ETC(t_i,m_j)$ $(1\le i\le m, 1\le j\le n)$, in the current study we find out the task assignment strategy that gives the minimum makespan.

For task selection in heterogeneous environment different criteria can be used, e.g. minimum, maximum or average of expected execution time across all machines. In current work we propose a new heuristic based on task partitioning, which

***Corresponding Author:** Ehsan Ullah Munir,
Department of Computer Science,
COMSATS Institute of Information Technology,
Wah Cantt , Pakistan
Email: ehsanmunir @comsats.edu.pk

consider minimum (min), maximum (max), average (avg), median (med) and standard deviation (std) of expected execution time of task on different machines as selection criteria. We call each selection criterion a key. Each heuristic uses only one key. Scheduling process for the proposed heuristics works like this; all the tasks are sorted in decreasing order of their key, then these tasks are partitioned into *k* segments and after this scheduling is performed in each segment.

A large number of experiments were conducted on synthetic datasets; Coefficient of Variation (COV) based method was used for generating synthetic datasets, which provides greater control over spread of heterogeneity (Ali et al., 2000). A comparison among existing heuristics is conducted and new heuristics are proposed. Extensive simulation results illustrate the circumstances when one heuristic would outperform other heuristics in terms of average makespan.

Let $T = \{t_1, t_2, ..., t_m\}$ be a set of tasks, $M = \{m_1, m_2, ..., m_n\}$ be a set of machines, and the expected time to compute (*ETC*) is a $m \times n$ matrix where the element $ETC_{ij}$ represents the expected execution time of task $t_i$ on machine $m_j$. For clarity, we denote $ETC_{ij}$ by $ETC(t_i, m_j)$ in the rest of the paper. Machine availability time, $MAT(m_j)$, is the earliest time machine $m_j$ can complete the execution of all the tasks that have previously been assigned to it (based on the *ETC* entries for those tasks). The completion time (*CT*) of task $t_i$ on machine $m_j$ is equal to the execution time of $t_i$ on $m_j$ plus the machine availability time of $m_j$ i.e.

$CT(t_i, m_j) = ETC(t_i, m_j) + MAT(m_j)$.

Makespan (*MS*) is equal to the maximum value of the completion time of all tasks i.e.

$MS = \max\ MAT(m_j)$ for $(1 \le j \le n)$

Provided with *T*, *M* and *ETC* our objective is to find the task assignment strategy which minimizes makespan.

## Materials and Methods

### Task partitioning heuristic

In heterogeneous environment for task selection different criteria can be used, examples are minimum, maximum or average of expected execution time across all machines. In task partitioning heuristic we use minimum (min), maximum (max), average (avg), median (med) and standard deviation (std) of expected execution time of task on different machines as selection criteria; hereafter referred to as key. Given a set of tasks $T = \{t_1, t_2, ..., t_m\}$, a set of machines $M = \{m_1, m_2, ..., m_n\}$, expected time to compute (*ETC*) matrix then the working of proposed heuristic can be explained as follows: we compute the sorting key for each task (for each heuristic only one key will be used for sorting), then we sort the tasks in decreasing order of their sorting key. Next the tasks are partitioned into *k* disjoint equal sized groups. In last, tasks are scheduled in each group $g_x$ using the following procedure:

**Procedure 1**

a) for each task $t_i$ in a group $g_x$ find machine $m_j$ which completes the task at earliest.

b) If machine $m_j$ is available i.e. no task is assigned to machine then assign task to machine and remove it from list of tasks.

c) If there is already task $t_k$ assigned to machine i.e. machine $m_j$ is not available then compute the difference between the minimum earliest completion time and the second smallest earliest completion time on all machines for $t_i$ and $t_k$ respectively.

1. If the difference value for $t_i$ is larger than that of $t_k$ then $t_i$ is assigned to machine $m_j$.
2. If the difference value for $t_i$ is less than that of $t_k$, then no changes to the assignment .
3. If the differences are equal, we compute the difference between the minimum earliest completion time and the third smallest earliest completion time for $t_i$ and $t_k$ respectively. And repeat 1-3. Every time if step 3 is selected, the difference between the minimum earliest completion time and the next earliest completion time (e.g. the fourth, the fifth…) for $t_i$ and $t_k$ are computed respectively. If all the differences are the same then the task is selected deterministically i.e. the oldest task is chosen.

Now the proposed Task partitioning algorithm can be summed up in the following steps:

**Task Partitioning Heuristic**

1. Compute the sorting key for each task:

   Sub-policy1 (avg): Compute the average value of each row in ETC matrix

   $key_i = \sum_j ETC(t_i, m_j) / n.$

   Sub-policy2 (min): Compute the minimum value of each row in ETC matrix

   $key_i = \min_j ETC(t_i, m_j).$

   Sub-policy3 (max): Compute the maximum value of each row in ETC matrix

   $key_i = \max_j ETC(t_i, m_j).$

   Sub-policy4 (med): Compute the median value of each row in ETC matrix

   $key_i = \underset{j}{med}\, ETC(t_i, m_j).$

   Sub-policy5 (std): Compute the standard deviation value of each row in ETC matrix

   $key_i = \underset{j}{std}\, ETC(t_i, m_j).$

2. Sort the tasks in decreasing order of their sorting key (for each heuristic only one key will be used for sorting).
3. Partition the tasks evenly into $k$ segments.
4. Apply the procedure 1 for scheduling each segment.

**Table 1 Scenario ETC matrix**

| Task no | m1 | m2 | m3 | m4 |
|---|---|---|---|---|
| $t_1$ | 17 | 19 | 31 | 17 |
| $t_2$ | 2 | 4 | 2 | 5 |
| $t_3$ | 18 | 11 | 12 | 7 |
| $t_4$ | 3 | 4 | 6 | 13 |
| $t_5$ | 4 | 2 | 2 | 3 |
| $t_6$ | 10 | 9 | 11 | 7 |
| $t_7$ | 13 | 26 | 28 | 10 |
| $t_8$ | 9 | 6 | 4 | 4 |
| $t_9$ | 10 | 13 | 8 | 5 |
| $t_{10}$ | 5 | 4 | 7 | 9 |
| $t_{11}$ | 7 | 9 | 6 | 13 |
| $t_{12}$ | 14 | 6 | 12 | 8 |
| $t_{13}$ | 14 | 8 | 12 | 20 |
| $t_{14}$ | 16 | 9 | 16 | 15 |
| $t_{15}$ | 18 | 11 | 5 | 7 |

**Table 2 Task partitioning**

| Task no | m1 | m2 | m3 | m4 | Avg |
|---|---|---|---|---|---|
| $t_1$ | 17 | 19 | 31 | 17 | 21.00 |
| $t_7$ | 13 | 26 | 28 | 10 | 19.25 |
| $t_{14}$ | 16 | 9 | 16 | 15 | 14.00 |
| $t_{13}$ | 14 | 8 | 12 | 20 | 13.50 |
| $t_3$ | 18 | 11 | 12 | 7 | 12.00 |
| $t_{15}$ | 18 | 11 | 5 | 7 | 10.25 |
| $t_{12}$ | 14 | 6 | 12 | 8 | 10.00 |
| $t_6$ | 10 | 9 | 11 | 7 | 9.25 |
| $t_9$ | 10 | 13 | 8 | 5 | 9.00 |
| $t_{11}$ | 7 | 9 | 6 | 13 | 8.75 |
| $t_4$ | 3 | 4 | 6 | 13 | 6.50 |
| $t_{10}$ | 5 | 4 | 7 | 9 | 6.25 |
| $t_8$ | 9 | 6 | 4 | 4 | 5.75 |
| $t_2$ | 2 | 4 | 2 | 5 | 3.25 |
| $t_5$ | 4 | 2 | 2 | 3 | 2.75 |

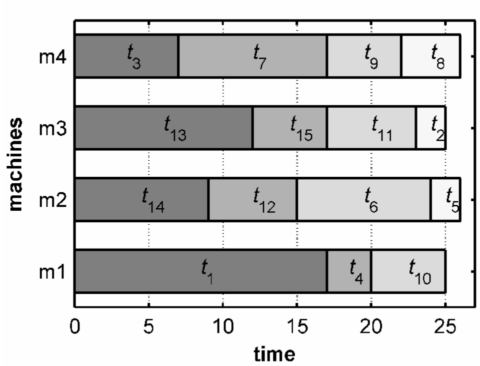


**Figure 1 Visual representation of task assignment in task partitioning heuristic**

A scenario of ETC is given in Table 1 to describe the working of proposed heuristic. All machines are assumed to be idle for this case. Sorting key used for the algorithm is average (avg) i.e. tasks are sorted in decreasing order of their average value. Table 2 shows the task partitioning; tasks are partitioned into three segments which implies $k$ = 3. Table 3 shows how the results are derived using procedure 1. Figure 1 gives the visual representation of task assignment for proposed heuristic.

**Heuristics notation**

In task partitioning heuristic tasks are sorted based on average, minimum, maximum, median and standard deviation and each heuristic is named as TPAvg, TPMin, TPMax, TPMed and TPStd. The algorithms Segmented min-min (med) and Segmented min-min (std) are also implemented for the evaluation purpose. The naming conventions and source information for all existing and proposed heuristics are detailed in Table 4.

**Table 3 Execution process of Procedure 1 on each group**

| Execution process on group 1 | | |
|---|---|---|
| 1st pass | min. CT | difference |
| $t_1 \rightarrow$ m1 | 17 | 0 |
| $t_{14} \rightarrow$ m2 | 9 | 6 |
| $t_3 \rightarrow$ m4 | 7 | 4 |
| 2nd pass | min. CT | difference |
| $t_7 \rightarrow$ m4 | 17 | 11 |
| $t_{13} \rightarrow$ m3 | 12 | 5 |
| execution process on group 2 | | |
| 1st pass | min. CT | difference |
| $t_{15} \rightarrow$ m3 | 17 | 3 |
| $t_{12} \rightarrow$ m2 | 15 | 9 |
| 2nd pass | min. CT | difference |
| $t_6 \rightarrow$ m2 | 24 | 0 |
| $t_9 \rightarrow$ m4 | 22 | 3 |
| $t_{11} \rightarrow$ m3 | 23 | 1 |
| execution process on group 3 | | |
| 1st pass | min. CT | difference |
| $t_4 \rightarrow$ m1 | 20 | 8 |
| 2nd pass | min. CT | difference |
| $t_{10} \rightarrow$ m1 | 25 | 3 |
| $t_8 \rightarrow$ m4 | 26 | 1 |
| 3rd pass | min. CT | difference |
| $t_2 \rightarrow$ m3 | 25 | 2 |
| 4th pass | min. CT | difference |
| $t_5 \rightarrow$ m2 | 26 | 1 |

## Results and Discussion

**Dataset**

In the experiments, COV based ETC generation method is used to simulate different HC environments by changing the parameters $\mu_{task}$, $V_{task}$ *and* $V_{machine}$, which represent the mean task execution time, the task heterogeneity, and the machine heterogeneity, respectively. The COV based method provides greater control over the spread of the execution time values than the common range-based method used previously (Braun et al., 2001; Ritchie and Levine, 2003; Shivle et al., 2005).

The COV-based ETC generation method works as follows (Ali et al., 2000): First, a task vector, *q*, of expected execution times with the desired task heterogeneity is generated following gamma distribution with mean $\mu_{task}$ and standard deviation $\mu_{task} * V_{task}$. The input parameter $\mu_{task}$ is used to set the average of the values in *q*. The input parameter $V_{task}$ is the desired coefficient of variation of the values in *q*. The value of $V_{task}$ quantifies task heterogeneity, and is larger for high task heterogeneity. Each element of the task vector *q* is then used to produce one row of the ETC matrix following gamma distribution with mean $q[i]$ and standard deviation $q[i] * V_{machine}$ such that the desired coefficient of variation of values in each row is $V_{machine}$, another input parameter. The value of $V_{machine}$ quantifies machine heterogeneity, and is larger for high machine heterogeneity.

**Comparative performance evaluation**

The performance of the heuristic algorithm is evaluated by the average makespan of 1000 results on 1000 ETCs generated by the same parameters. In all the experiments, the size of ETCs is $512 \times 16$, the value of $k = 3$, the mean of task execution time $\mu_{task}$ is 1000, and the task COV $V_{task}$ is in $[0.1, 2]$ while the machine COV $V_{machine}$ is in $[0.1, 1.1]$.

The motivation behind choosing such heterogeneous ranges is that in real situation there is more variability across execution times for different tasks on a given machine than the execution time for a single task across different machines.

The range bar for the average makespan of each heuristic shows a 95% confidence interval for the corresponding average makespan. This interval represents the likelihood that makespans of task assignment for that type of heuristic fall within the specified range. That is, if another ETC matrix (of the same type) is generated, and the specified heuristic generates a task assignment, then the makespan of the task assignment would be within the given interval with 95% certainty. In our experiments we have also considered two metrics in comparison of heuristics. Such metrics have also been considered by (Sakellariou and Zhao, 2004)

- The number of best solutions (denoted by NB) is the number of times a particular method was the only one that produced the shortest makespan.
- The number of best solutions equal with another method (denoted by NEB), which counts those cases where a particular method produced the shortest makespan but at least one other method also achieved the same makespan. NEB is the complement to NB.

The proposed heuristics are compared with 11 existing heuristics. Experiments are performed with different ranges of task and machine heterogeneity.

In the first experiment we have fixed the value of $V_{task} = 2$ and then increase the value of $V_{machine}$ from 0.1 to 1.1 with increment of 0.2 in each step. The results of NB and NEB are shown in the Table 5. From the values it is clear that for high values of $V_{machine}$ H16 is the best heuristic. And in all other cases one of the proposed heuristic H2 or H5 outperforms all other heuristics. Figure 2 gives the comparison of average makespan of the all heuristics considered.

In the second experiment we have fixed the value of $V_{task}$ = 1.1 and then increase the value of $V_{machine}$ from 0.1 to 1.1 with increment of 0.2 in each step. The results of NB and NEB are shown in the Table 6. From the values it is clear that here in all the cases one of the proposed heuristic H2 or H5 is best. Figure 3 gives the comparison of average makespan of all the heuristics consider here.

**Table 4 Summary of compared heuristics**

| No | Name | Reference | No | Name | Reference |
|---|---|---|---|---|---|
| H1 | TPAvg | New | H9 | Smm-avg | (Wu, M.Y et al) |
| H2 | TPMin | New | H10 | Smm-min | (Wu, M.Y et al) |
| H3 | TPMax | New | H11 | Smm-max | (Wu, M.Y et al) |
| H4 | TPMed | New | H12 | Smm-med | New |
| H5 | TPStd | New | H13 | Smm-std | New |
| H6 | Min-min | (Freund R.F et al) | H14 | MCT | (Maheswaran, M et al) |
| H7 | Max-min | (Freund R.F et al) | H15 | minSD | (Luo, P et al) |
| H8 | Sufferage | (Maheswaran, M et al) | H16 | HTF | (Yarmolenko, V et al) |

**Table 5 NB and NEB values table when fix $V_{task}$ = 2**

| Cov of tasks | | H1 | H2 | H3 | H4 | H5 | H6 | H7 | H8 | H9 | H10 | H11 | H12 | H13 | H14 | H15 | H16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0.1 | NB | 86 | 197 | 169 | 78 | 245 | 0 | 0 | 96 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 4 |
| | NEB | 97 | 27 | 48 | 92 | 29 | 0 | 2 | 18 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 |
| 0.3 | NB | 101 | 252 | 112 | 132 | 90 | 0 | 0 | 213 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| | NEB | 62 | 54 | 48 | 62 | 52 | 0 | 1 | 49 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 4 |
| 0.5 | NB | 101 | 352 | 98 | 106 | 65 | 0 | 0 | 92 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 19 |
| | NEB | 105 | 84 | 104 | 103 | 99 | 0 | 1 | 90 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 10 |
| 0.7 | NB | 82 | 350 | 62 | 89 | 47 | 0 | 0 | 45 | 1 | 2 | 4 | 1 | 2 | 0 | 0 | 146 |
| | NEB | 100 | 59 | 98 | 96 | 99 | 0 | 2 | 89 | 0 | 0 | 2 | 1 | 1 | 0 | 0 | 32 |
| 0.9 | NB | 60 | 199 | 43 | 62 | 44 | 0 | 0 | 11 | 5 | 2 | 2 | 4 | 0 | 0 | 0 | 381 |
| | NEB | 103 | 78 | 115 | 103 | 110 | 0 | 14 | 94 | 1 | 0 | 2 | 0 | 1 | 2 | 0 | 90 |
| 1.1 | NB | 17 | 69 | 22 | 21 | 16 | 0 | 0 | 9 | 0 | 1 | 0 | 3 | 1 | 0 | 0 | 575 |
| | NEB | 167 | 156 | 160 | 163 | 160 | 0 | 47 | 156 | 1 | 0 | 3 | 1 | 2 | 5 | 0 | 202 |

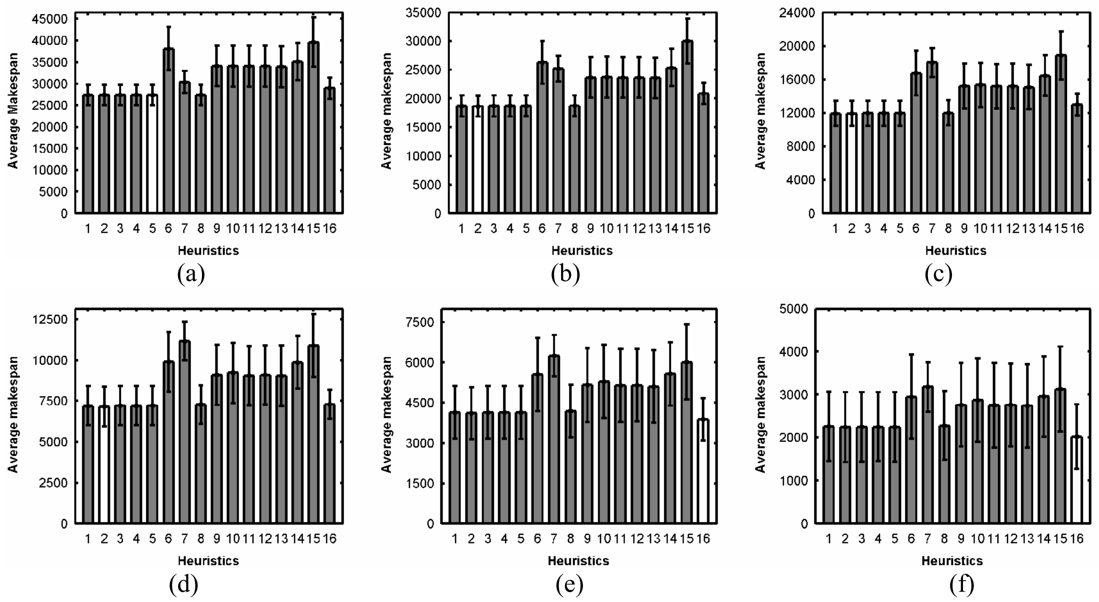


**Figure 2 Average makespan of the heuristics when $V_{task}$ = 2 and $V_{machine}$ = (a) $V_{machine}$= 0.1, (b) $V_{machine}$ = 0.3, (c) $V_{machine}$ = 0.5, (d) $V_{machine}$ = 0.7, (e) $V_{machine}$ = 0.9, (f) $V_{machine}$ = 1.1.**

**Table 6 NB and NEB values table when fix $V_{task}$ = 1.1**

| Cov of tasks | | H1 | H2 | H3 | H4 | H5 | H6 | H7 | H8 | H9 | H10 | H11 | H12 | H13 | H14 | H15 | H16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0.1 | NB | 141 | 159 | 150 | 150 | **372** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | NEB | 24 | 2 | 5 | 21 | 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0.3 | NB | 139 | **284** | 199 | 161 | 211 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | NEB | 2 | 4 | 2 | 3 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0.5 | NB | 129 | **445** | 154 | 127 | 142 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | NEB | 1 | 2 | 2 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0.7 | NB | 84 | **613** | 97 | 82 | 102 | 0 | 0 | 0 | 3 | 10 | 1 | 2 | 0 | 0 | 0 | 0 |
| | NEB | 3 | 2 | 4 | 3 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0.9 | NB | 78 | **586** | 80 | 63 | 91 | 0 | 0 | 0 | 8 | 59 | 5 | 14 | 1 | 0 | 0 | 2 |
| | NEB | 6 | 8 | 6 | 7 | 4 | 0 | 0 | 1 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 1 |
| 1.1 | NB | 66 | **505** | 76 | 73 | 63 | 0 | 0 | 1 | 28 | 24 | 4 | 24 | 4 | 0 | 0 | 92 |
| | NEB | 20 | 24 | 17 | 16 | 14 | 0 | 0 | 10 | 3 | 0 | 1 | 1 | 1 | 0 | 0 | 11 |

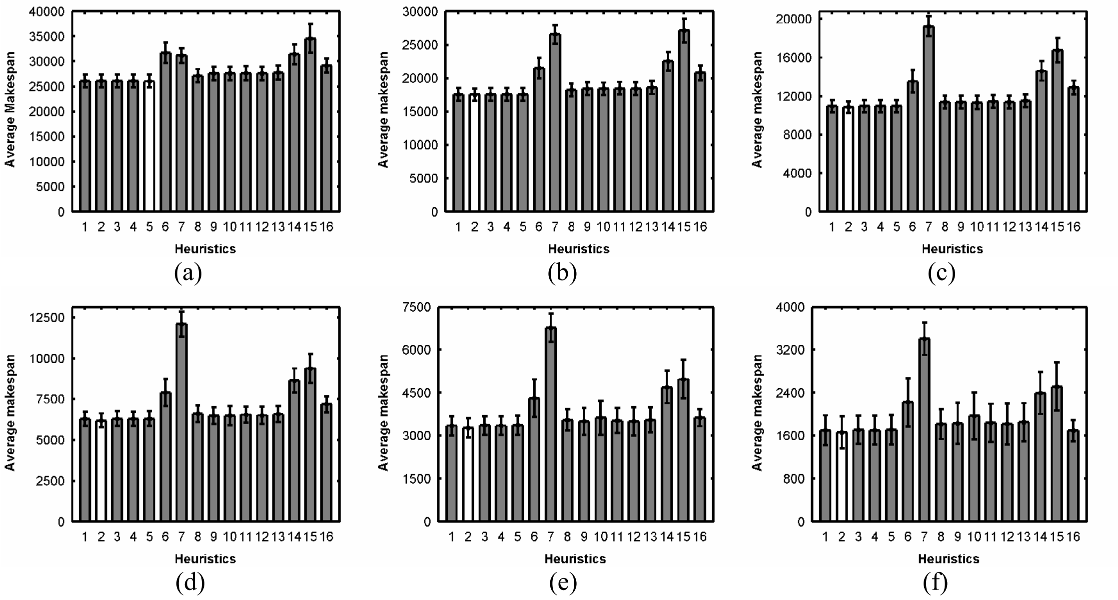


**Figure 3 Average makespan of the heuristics when $V_{task}$ = 2 and $V_{machine}$ = (a) $V_{machine}$= 0.1, (b) $V_{machine}$ = 0.3, (c) $V_{machine}$ = 0.5, (d) $V_{machine}$ = 0.7, (e) $V_{machine}$ = 0.9, (f) $V_{machine}$ = 1.1.**

**Table 7 NB and NEB values when fix $V_{task}$ = 0.6**

| Cov of tasks | | H1 | H2 | H3 | H4 | H5 | H6 | H7 | H8 | H9 | H10 | H11 | H12 | H13 | H14 | H15 | H16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0.1 | NB | 81 | 80 | 78 | 79 | 682 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | NEB | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0.3 | NB | 73 | 42 | 143 | 76 | 663 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | NEB | 1 | 1 | 3 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0.5 | NB | 84 | 20 | 254 | 118 | 520 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | NEB | 3 | 0 | 2 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0.7 | NB | 127 | 13 | 285 | 130 | 441 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | NEB | 2 | 0 | 3 | 1 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0.9 | NB | 150 | 33 | 313 | 144 | 354 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | NEB | 2 | 0 | 2 | 4 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1.1 | NB | 138 | 124 | 245 | 158 | 313 | 0 | 0 | 0 | 0 | 6 | 0 | 0 | 0 | 0 | 0 | 1 |
| | NEB | 4 | 9 | 5 | 8 | 5 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

In the third experiment we have fixed the value of $V_{task}$ = 0.6 and then increase the value of $V_{machine}$ from 0.1 to 1.1 with increment of 0.2 in each step. The results of NB and NEB are shown in the Table 7. From the values it is clear that here in all the cases proposed heuristic H5 outperforms all other heuristics. Figure 4 gives the comparison of average makespan of all the heuristics.

In the fourth experiment we have fixed the value of $V_{task}$ = 0.1 and then increase the value of $V_{machine}$ from 0.1 to 1.1 with increment of 0.2 in each step. The results of NB and NEB are shown in the Table 8. From the values it is clear that here in all the cases proposed heuristic H5 outperforms all other heuristics. Figure 5 gives the comparison of the average makespan of all the heuristics.

**Algorithm to find best heuristic**

Based on the values of $V_{task}$ and $V_{machine}$ we divide *ETC* into three different regions. If the values of $V_{task}$ and $V_{machine}$ are high (here $V_{task}$ = 2 and 0.9 <= $V_{machine}$ <= 1.1) then *ETC* falls in the region 1, if either of them is medium (here $V_{task}$ = 1.1 or 0.3 <= $V_{machine}$ < =0.7) then it falls in region 2 and if either of them is low (here 0.1 <= $V_{task}$ <= 0.6 or 0.1 <= $V_{machine}$ <= 0.2) then it falls in region 3. Fig. 6 shows the three regions and best heuristic for each region.

The procedure for finding a best heuristic is given below in Algorithm Best Heuristic, which suggests the best heuristic depending on ETC type.

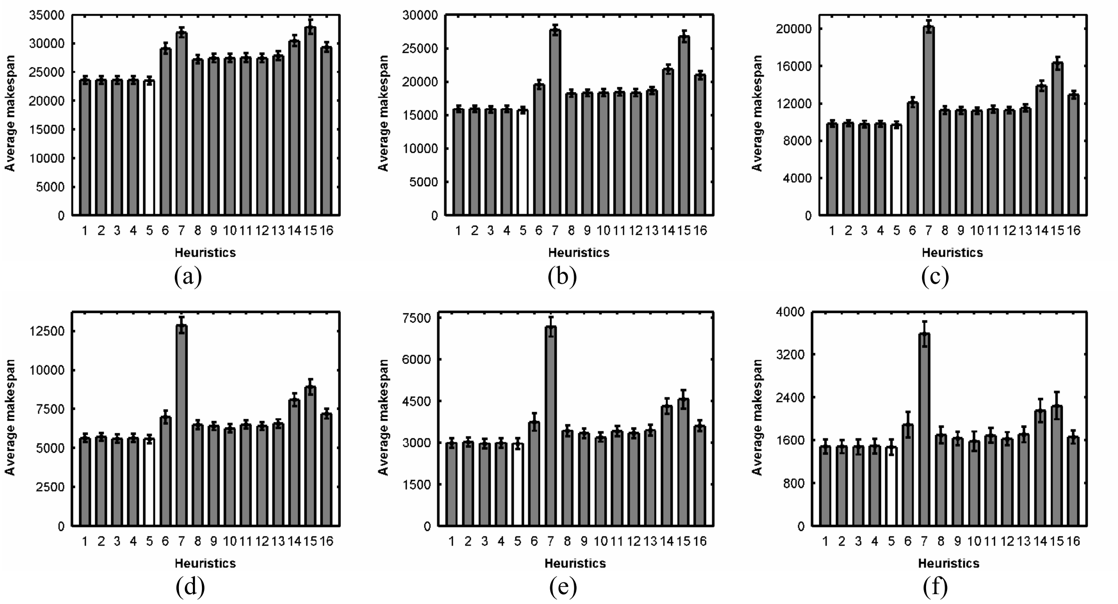


**Figure 4 Average makespan of the heuristics when $V_{task}$ = 2 and $V_{machine}$ = (a) $V_{machine}$= 0.1, (b) $V_{machine}$ = 0.3, (c) $V_{machine}$ = 0.5, (d) $V_{machine}$ = 0.7, (e) $V_{machine}$ = 0.9, (f) $V_{machine}$ = 1.1.**

**Table 8 NB and NEB values when fix $V_{task}$ = 0.1**

| Cov of tasks | | H1 | H2 | H3 | H4 | H5 | H6 | H7 | H8 | H9 | H10 | H11 | H12 | H13 | H14 | H15 | H16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0.1 | NB | 0 | 0 | 0 | 0 | 1000 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | NEB | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0.3 | NB | 0 | 0 | 0 | 0 | 1000 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | NEB | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0.5 | NB | 0 | 0 | 14 | 0 | 986 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | NEB | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0.7 | NB | 0 | 0 | 84 | 5 | 910 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | NEB | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0.9 | NB | 8 | 0 | 215 | 10 | 763 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | NEB | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1.1 | NB | 41 | 0 | 311 | 28 | 619 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | NEB | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Best heuristic**

Input: expected time to compute matrix (ETC)
Output: best heuristic
Compute the $V_{task}$ and $V_{machine}$
if $V_{task}$ is high and $V_{machine}$ is high then
    ETC belongs to region1
if $V_{task}$ is medium or $V_{machine}$ is medium then
    ETC belongs to region2
if $V_{task}$ is low or $V_{machine}$ is low then
    ETC belongs to region3
end if switch(region)
    case region1: return H16
    case region2: return H2
    case region3: return H5
end switch

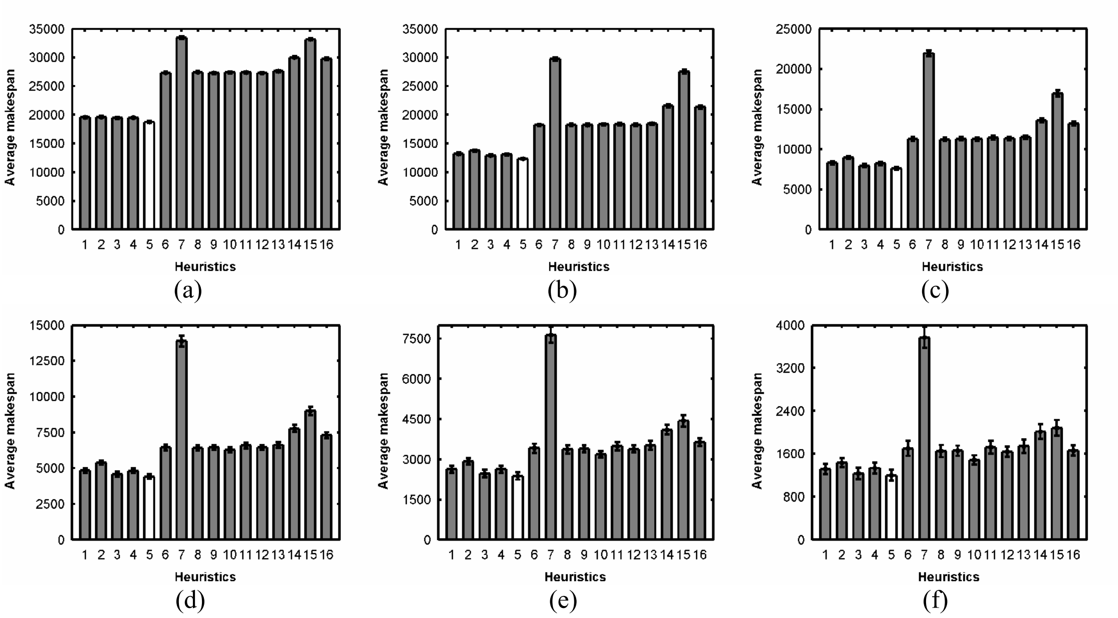


**Figure 5 Average makespan of the heuristics when $V_{task}$ = 2 and $V_{machine}$ = (a) $V_{machine}$= 0.1, (b) $V_{machine}$ = 0.3, (c) $V_{machine}$ = 0.5, (d) $V_{machine}$ = 0.7, (e) $V_{machine}$ = 0.9, (f) $V_{machine}$ = 1.1.**

| Cov of Tasks | COV of Machines | | | | | | |
|---|---|---|---|---|---|---|---|
| | | 0.1 | 0.3 | 0.5 | 0.7 | 0.9 | 1.1 |
| | 2 | H5 | H2 | H2 | H2 | H16 | H16 |
| | 1.1 | H5 | H2 | H2 | H2 | H2 | H2 |
| | 0.6 | H5 | H5 | H5 | H5 | H5 | H5 |
| | 0.1 | H5 | H5 | H5 | H5 | H5 | H5 |

Region 1

Region 2

Region 3

**Figure 6 Division of ETC in different regions**

## Conclusions

Optimal assignment of tasks to machines in a HC environment has been proven to be a NP-complete problem. It requires the use of efficient heuristics to find near optimal solutions. In this paper, we have proposed, analyzed and implemented seven new heuristics. A comparison of the proposed heuristics with the existing heuristics was also performed in order to identify the circumstances in which one heuristic outperforms the others. The experimental results demonstrate that in most of the circumstances one of the proposed heuristics H2 or H5 outperforms all the existing heuristics. Based on these experimental results, we are also able to suggest, given an ETC, which heuristic should be used to achieve the minimum makespan.

## Acknowledgements

The authors are thankful to COMSATS Institute of Information Technology for providing the support for this research. A preliminary version of portions of this document appear in the proceedings of the Modelling, Computation and Optimization in Information Systems and Management Sciences (MCO'08).

## References

Ali S, TD Braun, HJ Siegel, M Maciejewski, AA Beck, N Boloni, L Maheswaran, M Reuther, AI Robertson, JP Theys and MD Yao, 2005. Characterizing source allocation heuristics for heterogeneous computing systems. In: A.R. Hurson (Ed.), Advances in Computers, vol. 63: Parallel, Distributed, and Pervasive Computing, Elsevier, Amsterdam, The Netherlands, pp: 91-128.

Ali S, HJ Siegel, M Maheswaran, S Ali, and D Hensgen, 2000. Task Execution Time Modeling for Heterogeneous Computing Systems, Proceedings of the 9th Heterogeneous Computing Workshop, pp:185–199.

Barbulescu L, LD Whitley and AE Howe, 2004. Leap Before You Look: An Effective Strategy in an Oversubscribed Scheduling Problem. Proceedings of the 19th National Conference on Artificial Intelligence, ppL:143–148.

Baca, DF., (1989). Allocating modules to processors in a distributed system, IEEE Transactions on Software Engineering 15(11):1427–1436.

Braun TD, HJ Siegel, N Beck, LL Bölöni, M Maheswaran, AL Reuther, JP Robertson, MD Theys, B Yao, D Hensgen and RF Freund, 2001. 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.

Briceno LD, M Oltikar, HJ Siegel and AA Maciejewski, 2007. Study of an Iterative Technique to Minimize Completion Times of Non-Makespan Machines. Proceedings of the 17th Heterogeneous Computing Workshop pp: 1-14

El-Rewini H, TG Lewis and HH Ali, 1994. Task Scheduling in Parallel and Distributed Systems, PTR Prentice Hall, New Jersey, USA.

Foster I, and C Kesselman, 1998. The Grid: Blueprint for a New Computing Infrastructure, Morgan Kaufman Publishers, San Francisco, CA, USA.

Freund RF, M Gherrity, S Ambrosius, M Campbell, M Halderman, D Hensgen, E Keith, T Kidd, M Kussow, JD Lima, M Mirabile, L Moore, B Rust, and HJ Siegel, 1998. Scheduling Resources in Multi-User, Heterogeneous, Computing Environments with Smartnet, Proceedings of the 7th Heterogeneous Computing Workshop pp.184-199.

Ritchie G and J Levine, 2004. A hybrid ant algorithm for scheduling independent jobs in heterogeneous computing environments. Proceedings of the 23rd Workshop of the UK Planning and Scheduling Special Interest Group.

Ritchie G and J Levine, 2003. A fast, effective local search for scheduling independent jobs in heterogeneous computing environments, Proceedings of the 22nd Workshop of the UK Planning and Scheduling Special Interest Group, pp.178-183.

Ibarra OH and CE Kim, 1977. Heuristic algorithms for scheduling independent tasks on non-identical processors. Journal of association of computing machinery 24 (2):280–289.

Kim JK, S Shivle, HJ Siegel, AA Maciejewski, TD Braun, M Schneider, S Tideman, R Chitta, RB Dilmaghani, R Joshi, A Kaul, A Sharma, S Sripada, P Vangari and SS Yellampalli, 2007. Dynamically mapping tasks with priorities and multiple deadlines in a heterogeneous environment. Journal of Parallel and Distributed Computing, 67(2):154-169.

Kwok Y and K I Ahmad, 1999. Static scheduling algorithms for allocating directed task graphs to multiprocessors, association of computing machinery Computing Surveys, 31(4):406–471.

Kwok YK, AA Maciejewski, HJ Siegel, I Ahmad and A Ghafoor, 2006. A semi-static approach to mapping dynamic iterative tasks onto heterogeneous computing system. Journal of Parallel and Distributed Computing 66(1):77-98.

Luo P, K Lu and ZZ Shi, 2007. A revisit of fast greedy heuristics for mapping a class of independent tasks onto heterogeneous computing systems. Journal of Parallel and Distributed Computing 67 (6): 695-714.

Maheswaran M, S Ali, HJ Siegel, D Hensgen, and RF Freund, 1999. Dynamic mapping of a class of independent tasks onto heterogeneous computing system. Journal of Parallel and Distributed Computing 59(2): 107–131.

Sakellariou R and H Zhao 2004. A Hybrid Heuristic for Dag Scheduling on Heterogeneous Systems, Proceedings of the 13th Heterogeneous Computing Workshop pp:111-123

Shestak V, EKP Chong, AA Maciejewski, HJ Siegel, L Bemohamed, I Wang and R Daley 2005. Resource Allocation for Periodic Applications in a Shipboard Environment. Proceedings of the 14th Heterogeneous Computing Workshop. pp:124-130

Shivle S, HJ Siegel, AA Maciejewski, P Sugavanam, T Banka, R Castain, K Chindam, S Dussinger, P Pichumani, P Satyasekaran, W Saylor, D Sendek, J Sousa, J Sridharan and J.,Velazco 2006. Static allocation of resources to communicating subtasks in a heterogeneous ad hoc grid environment. Journal of Parallel and Distributed Computing, 66(4): 600–611.

Shivle S, P Sugavanam, HJ Siegel, AA Maciejewski, T Banka, K Chindam, S Dussinger, A Kutruff, P Penumarthy, P Pichumani, P Satyasekaran, D Sendek, J Smith, J Sousa, J Sridharan and J Velazco 2005. Mapping subtasks with multiple versions on an adhoc grid. Parallel Computing. Special Issue on Heterogeneous Computing. 31(7):671– 690.

Wu MY, W Shu and H Zhnag 2000. Segmented min-min: A Static Mapping Algorithm for Meta-Tasks on Heterogeneous Computing Systems. Proceedings of the 9th Heterogeneous Computing Workshop, pp.375–385.

Yarmolenko V, J Duato, DK Panda, and P Sadayappan, 2000. Characterization and Enhancement of Static Mapping Heuristics for Heterogeneous Systems. International Conference on Parallel Processing, pp: 437-444.