Journal of Zhejiang University SCIENCE A
ISSN 1673-565X (Print); ISSN 1862-1775 (Online)
www.zju.edu.cn/jzus; www.springerlink.com
E-mail: jzus@zju.edu.cn

A new heuristic for task scheduling in heterogeneous computing environment*

Ehsan Ullah MUNIR†[1,2], Jian-zhong LI[1], Sheng-fei SHI[1], Zhao-nian ZOU[1], Qaisar RASOOL[1]

([1]*School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, China*)

([2]*Department of Computer Science, COMSATS Institute of Information Technology, Wah Cantt 47040, Pakistan*)

†E-mail: ehsanmunnir@gmail.com

Received Jan. 2, 2008; revision accepted June 19, 2008; CrossCheck deposited Nov. 10, 2008

Abstract: Heterogeneous computing (HC) environment utilizes diverse resources with different computational capabilities to solve computing-intensive applications having diverse computational requirements and constraints. The task assignment problem in HC environment can be formally defined as for a given set of tasks and machines, assigning tasks to machines to achieve the minimum makespan. In this paper we propose a new task scheduling heuristic, high standard deviation first (HSTDF), which considers the standard deviation of the expected execution time of a task as a selection criterion. Standard deviation of the expected execution time of a task represents the amount of variation in task execution time on different machines. Our conclusion is that tasks having high standard deviation must be assigned first for scheduling. A large number of experiments were carried out to check the effectiveness of the proposed heuristic in different scenarios, and the comparison with the existing heuristics (Max-min, Sufferage, Segmented Min-average, Segmented Min-min, and Segmented Max-min) clearly reveals that the proposed heuristic outperforms all existing heuristics in terms of average makespan.

Key words: Heterogeneous computing, Task scheduling, Greedy heuristics, High standard deviation first (HSTDF) heuristic
doi:10.1631/jzus.A0820007 **Document code:** A **CLC number:** TP3

INTRODUCTION

Heterogeneous computing (HC) environment is composed of various resources with different computational capabilities to meet the demands of computing-intensive applications that have diverse computational requirements and constraints (Braun *et al.*, 2001). These scientific applications in HC environment usually consist of various computing-intensive tasks that must be performed on some resources in the HC environment. To achieve better performance via assignment of tasks to resources, an appropriate assignment strategy is very important for these scientific applications. Finding an efficient strategy for task assignment is an active topic of research and has been widely studied (Shivle *et al*., 2005; Luan *et al*., 2006; Kim *et al*., 2007) in other areas such as computational grids (Foster and Kesselman, 1998) and parallel program scheduling (Kwok and Ahmad, 1999a; 1999b). In HC environment, a computational job is designed and realized as a set of tasks, and the assignment of these tasks to the machines needs to be determined to achieve good performance. Formally, given a set of tasks $\{t_1, t_2, \ldots, t_m\}$, a set of machines $\{m_1, m_2, \ldots, m_n\}$, expected execution time of each task t_i on each machine m_j, e_{ij} ($1\leq i\leq m$, $1\leq j\leq n$), and a performance metric of task assignment strategy $f(X)$, we intend to find an assignment A of tasks $\{t_1, t_2, \ldots, t_m\}$ to machines $\{m_1, m_2, \ldots, m_n\}$ such that $f(A)$ is minimized. This problem is referred to as a 'scheduling problem' in the literature (Fernandez-Baca, 1989). The most common performance metric for task scheduling problems is the reduction in

* Project supported by the National Natural Science Foundation of China (No. 60703012), the National Basic Research Program (973) of China (No. 2006CB303000), and the Heilongjiang Provincial Scientific and Technological Special Fund for Young Scholars (No. QC06C033), China

makespan, which is defined as the maximum expected time required for completing the set of tasks using a particular assignment strategy.

The scheduling problem has been shown to be NP-complete (Ibarra and Kim, 1977; Fernandez-Baca, 1989), so it is not expected to be solved by using algorithms with polynomial time complexity. Hence other techniques such as greedy heuristics (Cormen *et al*., 2001), genetic algorithms (Wang *et al*., 1997), approximation algorithms (Vazirani, 2002) and ant-colony optimization (Ritchie and Levine, 2004) are used to find near-optimal solutions. In these techniques, greedy heuristics are widely used for task scheduling and are reported to be effective and efficient. Provision of solutions in real time makes them the logical candidate for scheduling problems. Greedy heuristics are an iterative process where at each step an individual task is chosen from the set of tasks based upon a certain criterion, and the selected tasks are assigned to respective machines based on some performance metric. This process is repeated until all the tasks are scheduled for processing on the machines. Recently a large number of heuristics have been proposed for task scheduling in HC environment (Maheswaran *et al*., 1999; Wu *et al*., 2000; Sakellariou and Zhao, 2004; Shivle *et al*., 2005; Luan *et al*., 2006; Kim *et al*., 2007). These heuristics use the minimum, maximum, mean or median of the expected execution time of tasks as the definitive selection criterion for scheduling.

In this work, a new task scheduling heuristic, high standard deviation first (HSTDF), is proposed, which considers a new selection criterion for task scheduling, i.e., the standard deviation of the expected execution time of tasks on all machines. The time complexity of HSTDF is $O(mn+m\log m+(m^2/k)n\log n)$, where m is the number of tasks, n is the number of machines and k is the number of groups. To the best of our knowledge, there has been no research to date considering the standard deviation of the expected task execution time as the selection criterion. This is the first paper taking this selection criterion into account.

A large number of experiments were conducted to show the effectiveness of the proposed heuristic, which outperforms other heuristics in terms of average makespan.

RELATED WORKS

A large number of heuristics have been proposed for task scheduling in HC environments. Min-min (Freund *et al*., 1998) and Max-min (Freund *et al*., 1998) heuristics are widely used for task scheduling in heterogeneous computing environments. Min-min gives the highest priority to the task for scheduling, which can be completed earliest. The idea behind Min-min heuristic is to finish each task as early as possible and hence, it schedules the tasks with the selection criterion of the minimum earliest completion time. Max-min heuristic is very similar to the Min-min, which gives the highest priority to the task with the maximum earliest completion time for scheduling. The idea behind Max-min is to overlap long-running tasks with short-running ones. In Segmented Min-min heuristic (Wu *et al*., 2000) the tasks are divided into four groups based on their minimum, maximum or average expected execution time, and then Min-min is applied to each group for scheduling. The Sufferage heuristic (Maheswaran *et al*., 1999) is based on the idea that, better mappings can be generated by assigning a task to the machine that would suffer most in terms of expected completion time if that particular machine is not assigned to the task. Sufferage assigns each task its priority according to its sufferage value. For each task, its sufferage value is equal to the difference between its best completion time and its second best completion time. Maheswaran *et al*.(1999) presented the detailed procedure and its comparison with some widely used heuristics, showing the superiority of Sufferage heuristic in most scenarios. Briceno *et al*.(2007) introduced a new criterion to minimize the completion time of non-makespan machines. It is noted that, although the completion time of a non-makespan machine can be reduced, the overall system makespan can be increased as well. Braun *et al*.(2001) compared 11 heuristics and declared that the Min-min heuristic was the best based on the makespan criterion. Kim *et al*.(2007) compared eight dynamic mapping heuristics, however, the problem domain considered there involves priorities and multiple deadlines.

The above-explored heuristics use the minimum, maximum, mean, or median of expected execution time as the decision parameter, respectively. These selection criteria may fail to account for the actual

distribution of the execution time of a task on all the machines. Subtle variations in the execution time on different machines can lead to an un-ideal assignment and an unexpected increase in the completion time of a given task set. Measuring the distribution of data in ECT matrix and allocating resources on the highest standard deviation first basis can alleviate this problem, which has not been investigated by any prior work.

PROBLEM DEFINITION

Prior to problem description, some fundamental concepts and definitions are given below.

Definition 1 (Expected computation time matrix) Given a set of tasks $T=\{t_1, t_2, \ldots, t_m\}$ and a set of machines $M=\{m_1, m_2, \ldots, m_n\}$, the expected computation time (ECT) matrix is an $m\times n$ matrix $\boldsymbol{E}=(e_{ij})$ $(1\le i\le m, 1\le j\le n)$, where entry e_{ij} is the estimated execution time of task t_i on machine m_j.

Definition 2 (Assignment function) Given a set of tasks $T=\{t_1, t_2, \ldots, t_m\}$ and a set of machines $M=\{m_1, m_2, \ldots, m_n\}$, an assignment function f: $T\times M\to\{1, 0\}$ is defined as

$$f(t_i, m_j)=\begin{cases}1, & \text{if } t_i \text{ is assigned to } m_j,\\ 0, & \text{otherwise,}\end{cases}$$
$$1\le i\le m,\ 1\le j\le n,$$

such that

$$\sum_{j=1}^{n} f(t_i, m_j)=1 \quad \text{for } 1\le i\le m.$$

Definition 2 indicates that a task can be assigned to one and only one machine.

Definition 3 (Completion time) Given a set of tasks $T=\{t_1, t_2, \ldots, t_m\}$, a set of machines $M=\{m_1, m_2, \ldots, m_n\}$, an ECT matrix $\boldsymbol{E}=(e_{ij})$ where e_{ij} represents the estimated execution time of task t_i on machine m_j, and an assignment function f: $T\times M\to\{1, 0\}$, the completion time of a machine, i.e., the time to finish the tasks assigned to machine m_j, is defined as

$$T_c(m_j)=\sum_{i=1}^{m} e_{ij} f(t_i, m_j),\ 1\le j\le n.$$

Definition 4 (Makespan) Given a set of tasks $T=\{t_1, t_2, \ldots, t_m\}$ and a set of machines $M=\{m_1, m_2, \ldots, m_n\}$, the makespan of an assignment function f is defined as

$$makespan(f)=\max_{1\le j\le n} T_c(m_j).$$

Problem statement

Based upon the above definitions, the task assignment problem can be formulated as

Input:
A set of tasks $T=\{t_1, t_2, \ldots, t_m\}$;
A set of machines $M=\{m_1, m_2, \ldots, m_n\}$;
An ECT matrix $\boldsymbol{E}=(e_{ij})$, where e_{ij} represents the estimated execution time of task t_i on machine m_j.
Output:
An assignment function f such that
$makespan(f)=\min\{makespan(g) \mid g: T\times M\to\{1, 0\}$ is an assignment function$\}$.

HIGH STANDARD DEVIATION FIRST (HSTDF) HEURISTIC

The goal of HSTDF heuristic is to find a high quality assignment function, which achieves near-optimal solution. In HSTDF, we consider the standard deviation of the expected execution time of tasks as a selection criterion, i.e., the tasks having high standard deviation must be scheduled first. Intuitively, tasks having a low standard deviation of execution time have less variation in execution time on different machines and hence, their delayed assignment for scheduling will not affect the overall makespan much. Moreover, the tasks with a higher standard deviation of execution time exhibit more variation in their execution time on different machines. A delayed assignment of such tasks might reduce their chances of occupying faster machines as some other tasks might occupy these machines earlier. Such a scenario would result in an increase in the system makespan. Hence, the tasks having a high standard deviation must be scheduled first.

Given a set of tasks $T=\{t_1, t_2, \ldots, t_m\}$, a set of machines $M=\{m_1, m_2, \ldots, m_n\}$ and an ECT matrix, the solution can be found in three steps. Firstly we compute the standard deviation of the expected execution time of each task and then sort all the tasks in decreasing order of their standard deviation of execution time. Secondly we partition the tasks into k equally sized disjoint groups $g_1, g_2, \ldots, g_k$, where

$g_i \cap g_j = \varnothing$ ($i \neq j$) and $g_1 \cup g_2 \cup \ldots \cup g_k = T$. The first $k-1$ groups contain $\lfloor m/k \rfloor$ tasks each and g_k contains $m - \lfloor m/k \rfloor (k-1)$ tasks. Finally, for each group from g_1, g_2, …, g_k, we assign the tasks to machines iteratively, and in each iteration, for each task t_i in group g_x (x=1, 2, ..., k), the process is as follows:

Step 1: Calculate the estimated completion time of task t_i on machine m_j if task t_i is to be assigned to m_j, and denote this estimated completion time as c_{ij}—of course, $c_{ij}=e_{ij}+T_c(m_j)$.

Step 2: The machine m_y on which task t_i has the minimum estimated complete time (i.e., $c_{iy} = \min_{j=1,2,\ldots,n} c_{ij}$) is chosen as the candidate machine for assigning task t_i.

Step 3: If there is no task assigned to m_y in this iteration (i.e., m_y is available), assign t_i to m_y and remove t_i from g_x, since this assignment achieves the locally minimum completion time.

Step 4: If a task t_z has already been assigned to m_y in this iteration (i.e., m_y is not available at the time), compute the difference between the minimum estimated completion time on all machines and the second smallest estimated completion time on all machines for tasks t_i and t_z, respectively.

Step 4.1: If the difference value for task t_i is larger than that for t_z, assign t_i instead of t_z to m_y in this iteration, remove t_i from g_x, and add t_z to g_x.

Step 4.2: If the difference value for task t_i is less than that for t_z, keep the assignment.

Step 4.3: If the differences are equal, we compute the difference between the minimum estimated completion time and the third smallest estimated completion time for t_i and t_z, respectively.

Repeat Steps 4.1~4.3. Every time Step 4.3 is carried out, the difference between the minimum estimated completion time and the next estimated completion time (e.g., the fourth, the fifth, etc.) for t_i and t_z is computed, respectively. And if all the differences are the same, select the tasks deterministically, and thus the oldest task is chosen.

The process is outlined in Algorithm 1.

Algorithm 1

Input: G_k.
Output: assignment function f.

```
1   while (Gk≠∅) do
2     for (each ti∈Gk) do
3       for (j=1, 2, …, n) do
4         cij=eij+Tc(mj);
5       endfor
6       sort ci1, ci2, …, cin in increasing order, denoted as
          ci(1), ci(2), ..., ci(n);
7     endfor
8     mark m1, m2, …, mn as unassigned;
9     for (each ti∈Gk) do
10      find a machine my such that ciy = min_j cij;
11      if (my is not assigned)
12        f(ti, my)=1;
13        Gk=Gk−{ti};
14      else
15        get the latest task tz which is assigned to my;
16        for (j=2, 3, …, n)
17          if (ci(j) − ci(1) > cz(j) − cz(i))
18            f(tz, my)=0;
19            f(ti, my)=1;
20            Gk=Gk∪{tz}−{ti};
21            break;
22          endif
23        endfor
24      endif
25    endfor
26  endwhile
27  return f;
```

In Algorithm 1, lines 3~5 take $O(n)$ to update c_{i1}, c_{i2}, …, c_{in}. Line 6 needs $O(n\log n)$ to sort the completion time c_{i1}, c_{i2}, …, c_{in} in increasing order. Thus, lines 2~7 take $O(pn\log n)$, where p is the number of current tasks in G_k. In the worst case, lines 16~23 finish after $n-1$ loops. Thus, the complexity of lines 9~25 is $O(pn)$. Lines 2~25 take $O(pn\log n)$. Since in each iteration of the while loop, at least one task can be assigned to machines, the time complexity of Algorithm 1 is thus $O(tn\log n)+O((t-1)n\log n)+O((t-2)n\log n)+\ldots+O(n\log n)=O(t^2 n\log n)$, where t is the number of tasks in G_k.

The proposed algorithm HSTDF is shown in Algorithm 2.

Algorithm 2 HSTDF heuristic

Input: $T=\{t_1, t_2, \ldots, t_m\}$, $M=\{m_1, m_2, \ldots, m_n\}$, ECT matrix.
Output: assignment function f.

```
1   for (i=1, 2, …, n) do
2     compute the standard deviation of ECT values for each
        task;
3   endfor
4   sort the tasks in decreasing order of their standard deviation
      of execution time;
5   divide the tasks evenly into k groups;
6   for (each group Gk)
7     apply Algorithm 1 for assignment;
8   endfor
9   return f;
```

In the HSTDF heuristic, lines 1~3 take $O(nm)$ to compute the standard deviation of the expected execution time of tasks. Line 4 takes $O(m\log m)$ to sort the tasks in decreasing order. Lines 6~8 need $O(k(m/k)n^2\log n)=O((m^2/k)n\log n)$ time to complete. Therefore, the time complexity of the HSTDF heuristic is $O(mn+m\log m+(m^2/k)n\log n)$.

A simple example is given to show the comparison of the proposed heuristic with Min-min, Max-min, Sufferage and Segmented Min-min. The execution time of nine tasks on three machines is recorded in Table 1.

All the machines are assumed to be idle for this case. In this example the minimum makespan produced by all other heuristics (Min-min, Max-min, Sufferage, and Segmented Min-min) is 29 and the makespan produced by the proposed heuristic is 22, which clearly shows that the proposed heuristic outperforms all the heuristics considered here for comparison. Task division is given in Table 2 (here we divide the tasks into three groups). Table 3 explains how the results are derived.

Table 1 Element of the ECT matrix

Task	Execution time		
	m_1	m_2	m_3
t_1	9	11	17
t_2	4	5	4
t_3	5	9	9
t_4	19	13	16
t_5	11	7	6
t_6	11	9	8
t_7	14	9	12
t_8	4	5	5
t_9	11	8	8

Table 2 Division of tasks in three groups

Task	Execution time			Std. Dev.
	m_1	m_2	m_3	
t_1	9	11	17	4.1633
t_4	19	13	16	3.0000
t_5	11	7	6	2.6458
t_7	14	9	12	2.5166
t_3	5	9	9	2.3094
t_9	11	8	8	1.7321
t_6	11	9	8	1.5275
t_2	4	5	4	0.5774
t_8	4	5	5	0.5774

Table 3 Execution process of Algorithm 1 on each group

Group	Iterations in each group		Minimum completion time	Difference	Machine
1	1st pass	$t_1\rightarrow m_1$	9	2	m_1
		$t_4\rightarrow m_2$	13	3	m_2
		$t_5\rightarrow m_3$	6	1	m_3
2	1st pass	$t_3\rightarrow m_1$	14	1	m_1
		$t_9\rightarrow m_3$	14	6	m_3
	2nd pass	$t_7\rightarrow m_2$	22	6	m_2
3	1st pass	$t_6\rightarrow m_3$	22	3	m_3
		$t_8\rightarrow m_1$	18	1	m_1
	2nd pass	$t_2\rightarrow m_1$	22	4	m_1

RESULTS AND DISCUSSION

Dataset

In our experiments, coefficient-of-variation (COV) based ECT generation method was used to simulate different HC environments by changing the parameters μ_{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 a greater control over the spread of the execution time values than the common range-based method (Braun *et al.*, 2001; Ritchie and Levine, 2004; Shivle *et al.*, 2005).

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

Comparative performance evaluation

The performance of the heuristic algorithm was evaluated by the average makespan of 1 000 results on the 1 000 ECTs generated by the same parameters. In

all the experiments, the size of ECTs was 512×16, k=3, μ_{task}=1 000, $V_{task}\in$[0.1, 1.1], $V_{machine}\in$[0.1, 0.6]. The heterogeneous ranges were chosen to reflect the fact that in real situations there is more variability across the execution time for different tasks on a given machine than that across the execution time for a single task on different machines (Luo *et al*., 2007).

The range bar for the average makespan of each heuristic shows a 95% confidence interval for the corresponding average makespan. That is, if another ECT matrix (of the same type) is generated, and the specified heuristic generates a task assignment, the makespan of the task assignment would be within the given interval with 95% certainty. In some cases such as the first subfigure of Fig.1a, the entire confidence interval is too short at the current scale used in plotting the results, and thus it is difficult to differentiate upper and lower bounds from the confidence interval in such cases.

The proposed heuristic was compared with five existing heuristics. In the results H1=Max-min, H2=Sufferage, H3=Segmented Min-average, H4=Segmented Min-min, H5=Segmented Max-min, and H6=HSTDF.

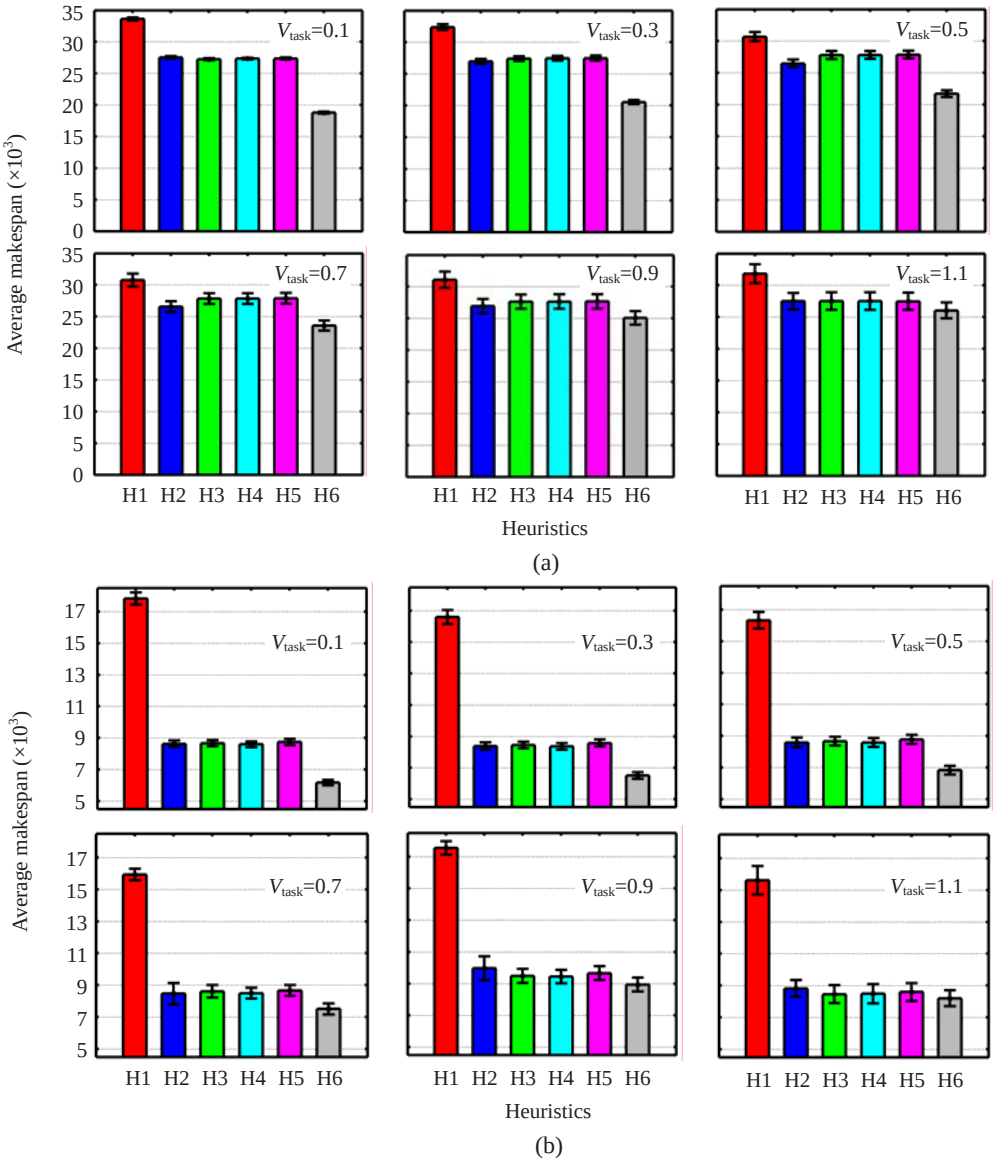

Fig.1 Average makespan of the heuristics for fixed $V_{machine}$ and variable V_{task} when ECT matrices are inconsistent. H1=Max-min, H2=Sufferage, H3=Segmented Min-average, H4=Segmented Min-min, H5=Segmented Max-min, H6=HSTDF. (a) $V_{machine}$=0.1; (b) $V_{machine}$=0.6

The experiments were performed for inconsistent, consistent and partially consistent ECT matrices. For each type of ECT matrix, experiments were performed in two phases. In the first phase, $V_{machine}$ was fixed with different values representing low (0.1) and high (0.6) levels and V_{task} increased in steps from low (0.1) to high (1.1). In the second phase, V_{task} was varied from low (0.1) to high (1.1) and $V_{machine}$ increased in steps from low (0.1) to high (0.6).

Inconsistent ECTs

In the first set of experiments, inconsistent ECTs were obtained using the COV-based ECT generation method without sorting any rows or columns, and the performance of each heuristic was measured for various combinations of $V_{machine}$ and V_{task}. Two phases of experiments are given below for inconsistent ECTs.

1. Fixed machine heterogeneity & variable task heterogeneity

Figs.1a and 1b depict the scenarios when $V_{machine}$ was fixed at 0.1 and 0.6 respectively and V_{task} increased from 0.1 to 1.1 with a step of 0.2 to cover a wide range of task heterogeneity. It is clear from the results that the proposed heuristic outperforms all other heuristics in terms of average makespan in all cases.

2. Fixed task heterogeneity & variable machine heterogeneity

In this phase three experimental situations were considered. V_{task} was fixed at 0.1, 0.6 and 1.1 respectively and the machine heterogeneity was varied from 0.1 to 0.6 with a step of 0.1. Fig.2 (see the next page) shows the results, from which it is evident that the makespan of the proposed heuristic is smaller than those of all other heuristics. But for high machine heterogeneity the difference is not much.

Consistent ECTs

The following experimental results were obtained using the consistent ECT matrices. Consistent ECT matrices were generated by sorting the e_{ij} values such that $e_{i1}<e_{i2}<\ldots<e_{in}$ for $1\leq i\leq m$. Here also experiments were devised in two phases.

1. Fixed machine heterogeneity & variable task heterogeneity

In this phase two scenarios were checked. $V_{machine}$ was fixed at 0.1 and 0.6, respectively, and V_{task} was varied from 0.1 to 1.1 with a step of 0.2. The experimental results show that the average makespan of the proposed heuristic is smaller than those of all other heuristics, with an exception for high task heterogeneity (V_{task}=1.1) when $V_{machine}$=0.6, where the proposed heuristic is the second best.

2. Fixed task heterogeneity & variable machine heterogeneity

During the second phase, V_{task} was fixed at 0.1, 0.6 and 1.1 respectively and $V_{machine}$ was varied from 0.1 to 0.6 with a step of 0.1. Experimental results show that the average makespan of the proposed heuristic is much smaller than those of all other heuristics when V_{task}=0.1, outperforms all other heuristics when V_{task}=0.6, and is comparable to the smallest average makespan (although not much smaller than the average of makespans) of other heuristics when V_{task}=1.1.

Partially consistent ECTs

In the final part of experiments we considered partially consistent ECTs, which were obtained by sorting half of the values of e_{ij} in each row or column.

1. Fixed machine heterogeneity & variable task heterogeneity

$V_{machine}$ was fixed at 0.1 and 0.6 respectively and V_{task} was varied from 0.1 to 1.1 with a step of 0.2. The experimental results show that the proposed heuristic outperforms all other heuristics in terms of average makespan when $V_{machine}$=0.1, and that the proposed heuristic performs well for low V_{task} when $V_{machine}$= 0.6.

2. Fixed task heterogeneity & variable machine heterogeneity

V_{task} was fixed at 0.1, 0.6 and 1.1, respectively, and $V_{machine}$ was varied from 0.1 to 0.6 with a step of 0.1. The experimental results show that the average makespan of the proposed heuristic is much smaller than those of all other heuristics when V_{task}=0.1 and 0.6, and that the average makespan of the proposed heuristic is comparable to those of other heuristics when V_{task}=1.1.

Due to space limitations, the results of consistent and partially consistent ECT matrices are not shown here.

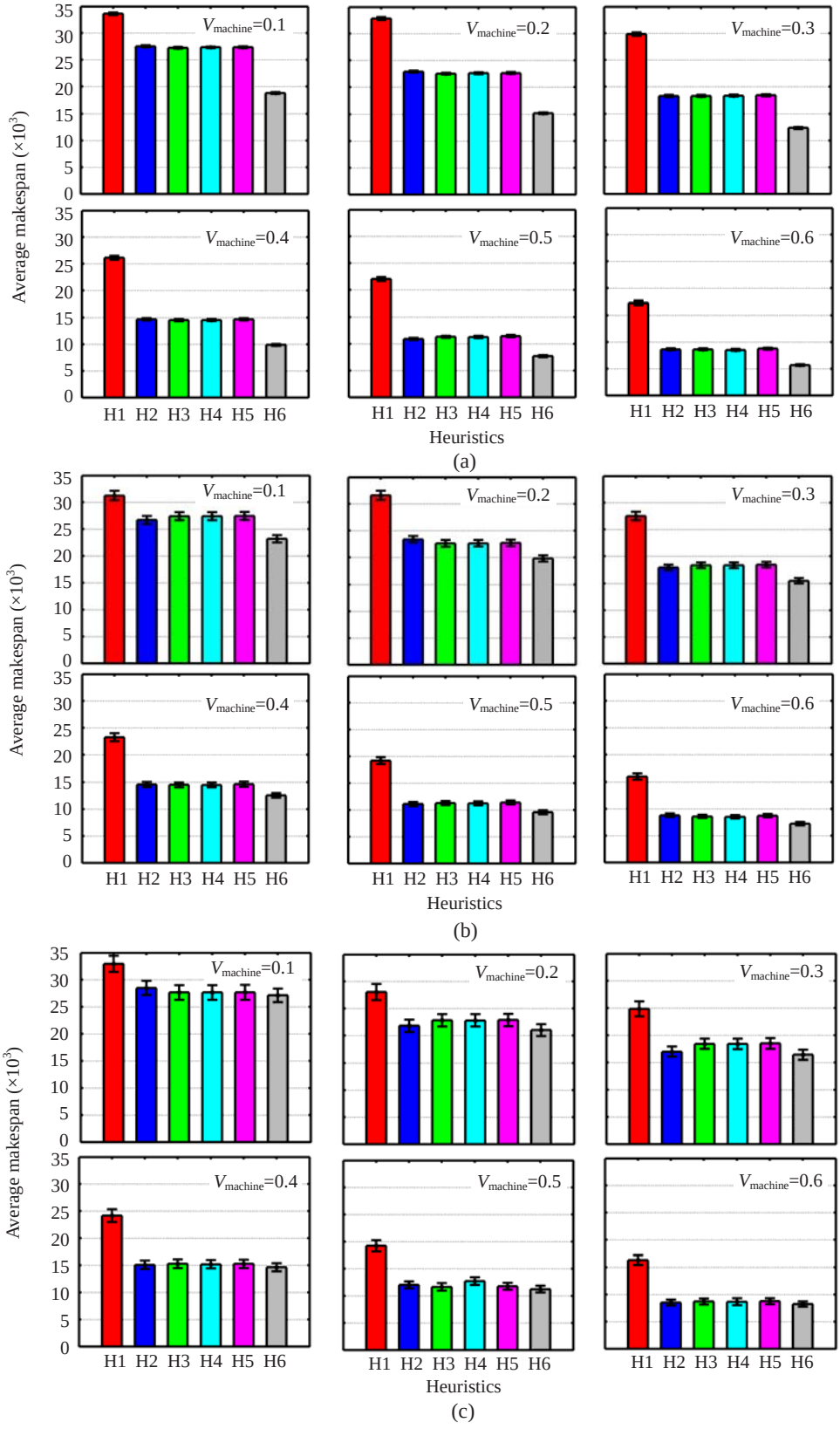

Fig.2 Average makespan of the heuristics for fixed V_{task} and variable $V_{machine}$ when ECT matrices are inconsistent. H1=Max-min, H2=Sufferage, H3=Segmented Min-average, H4=Segmented Min-min, H5=Segmented Max-min, H6=HSTDF. (a) V_{task}=0.1; (b) V_{task}=0.6; (c) V_{task}=1.1

CONCLUSION

In this paper we proposed a new task scheduling heuristic, HSTDF, for independent task scheduling in HC environments, taking into account the standard deviation of the expected execution time of a task as the selection criterion. We proved that this trait is of significant importance and must be considered for task scheduling in HC environments. A large number of experiments were carried out to check the effectiveness of the proposed heuristic in different scenarios. The experimental results clearly revealed that the proposed heuristic outperforms all existing heuristics in terms of average makespan.

References

Ali, S., Siegel, H.J., Maheswaran, M., Ali, S., Hensgen, D., 2000. Task Execution Time Modeling for Heterogeneous Computing Systems. Proc. 9th Heterogeneous Computing Workshop, p.185-200. [doi:10.1109/HCW.2000.843743]

Braun, T.D., Siegel, H.J., Beck, N., Bölöni, L.L., Maheswaran, M., Reuther, A.I., Robertson, J.P., Theys, M.D., Yao, B., Hensgen, D., *et al*., 2001. A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems. *J. Parall. Distrib. Comput.*, **61**(6):810-837. [doi:10.1006/jpdc.2000.1714]

Briceno, L.D., Oltikar, M., Siegel, H.J., Maciejewski, A.A., 2007. Study of an Iterative Technique to Minimize Completion Times of Non-makespan Machines. Proc. 17th Heterogeneous Computing Workshop, p.1-14. [doi:10.1109/IPDPS.2007.370325]

Cormen, T.H., Leirson, C.E., Rivest, R.L., 2001. Introduction to Algorithms. MIT Press, Cambridge, MA.

Fernandez-Baca, D., 1989. Allocating modules to processors in a distributed system. *IEEE Trans. on Software Eng.*, **15**(11):1427-1436. [doi:10.1109/32.41334]

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

Freund, R.F., Gherrity, M., Ambrosius, S., Campbell, M., Halderman, M., Hensgen, D., Keith, E., Kidd, T., Kussow, M., Lima, J.D., *et al*., 1998. Scheduling Resources in Multi-user, Heterogeneous, Computing Environments with Smartnet. Proc. 7th Heterogeneous Computing Workshop, p.184-199. [doi:10.1109/HCW.1998.666558]

Ibarra, O.H., Kim, C.E., 1977. Heuristic algorithms for scheduling independent tasks on non-identical processors. *J. ACM*, **24**(2):280-289. [doi:10.1145/322003.322011]

Kim, J.K., Shivle, S., Siegel, H.J., Maciejewski, A.A., Braun, T.D., Schneider, M., Tideman S., Chitta, R., Dilmaghani, R.B., Joshi, R., *et al*., 2007. Dynamically mapping tasks with priorities and multiple deadlines in a heterogeneous environment. *J. Parall. Distrib. Comput.*, **67**(2):154-169. [doi:10.1016/j.jpdc.2006.06.005]

Kwok, Y.K., Ahmad, I., 1999a. Benchmarking and comparison of the task graph scheduling algorithms. *J. Parall. Distrib. Comput.*, **59**(3):381-422. [doi:10.1006/jpdc.1999.1578]

Kwok, Y.K., Ahmad, I., 1999b. Static scheduling algorithms for allocating directed task graphs to multiprocessors. *ACM Comput. Surv.*, **31**(4):406-471. [doi:10.1145/344588.344618]

Luan, C.J., Song, G.H., Zheng, Y., 2006. Application-adaptive resource scheduling in a computational grid. *J. Zhejiang Univ. Sci. A*, **7**(10):1634-1641. [doi:10.1631/jzus.2006.A1634]

Luo, P., Lu, K., Shi, Z.Z., 2007. A revisit of fast greedy heuristics for mapping a class of independent tasks onto heterogeneous computing systems. *J. Parall. Distrib. Comput.*, **67**(6):695-714. [doi:10.1016/j.jpdc.2007.03.003]

Maheswaran, M., Ali, S., Siegel, H.J., Hensgen, D., Freund, R.F., 1999. Dynamic mapping of a class of independent tasks onto heterogeneous computing system. *J. Parall. Distrib. Comput.*, **59**(2):107-131. [doi:10.1006/jpdc.1999.1581]

Ritchie, G., Levine, J., 2004. A Hybrid Ant Algorithm for Scheduling Independent Jobs in Heterogeneous Computing Environments. Proc. 23rd Workshop of the UK Planning and Scheduling Special Interest Group, p.1-7.

Sakellariou, R., Zhao, H., 2004. A Hybrid Heuristic for DAG Scheduling on Heterogeneous Systems. Proc. 18th Parallel and Distributed Processing Symp., p.111-123. [doi:10.1109/IPDPS.2004.1303065]

Shivle, S., Sugavanam, P., Siegel, H.J., Maciejewski, A.A., Banka, T., Chindam, K., Dussinger, S., Kutruff, A., Penumarthy, P., Pichumani, P., *et al*., 2005. Mapping subtasks with multiple versions on an ad hoc grid. *Parall. Comput., Special Issue Heterog. Comput.*, **31**(7):671-690. [doi:10.1016/j.parco.2005.04.003]

Vazirani, V.V., 2002. Approximation Algorithms. Springer, Berlin, Germany.

Wang, L., Siegel, H.J., Roychowdhury, V.P., Maciejewski, A.A., 1997. Task matching and scheduling in heterogeneous computing environments using a genetic-algorithm-based approach. *J. Parall. Distrib. Comput.*, **47**(1):8-22. [doi:10.1006/jpdc.1997.1392]

Wu, M.Y., Shu, W., Zhang, H., 2000. Segmented Min-min: A Static Mapping Algorithm for Meta-tasks on Heterogeneous Computing Systems. Proc. 9th Heterogeneous Computing Workshop, p.375-385. [doi:10.1109/HCW.2000.843759]