J Supercomput
DOI 10.1007/s11227-011-0572-x

Statistical measures for quantifying task and machine heterogeneities

Abdulla M. Al-Qawasmeh · Anthony A. Maciejewski · Haonan Wang · Jay Smith · Howard Jay Siegel · Jerry Potter

Abstract We study heterogeneous computing (HC) systems that consist of a set of different machines that have varying capabilities. These machines are used to execute a set of heterogeneous tasks that vary in their computational complexity. Finding the optimal mapping of tasks to machines in an HC system has been shown to be, in general, an NP-complete problem. Therefore, heuristics have been used to find near-optimal mappings. The performance of allocation heuristics can be affected significantly by factors such as task and machine heterogeneities. In this paper, we identify different statistical measures used to quantify the heterogeneity of HC systems, and show the correlation between the performance of the heuristics and these measures

A.M. Al-Qawasmeh (✉) · A.A. Maciejewski · J. Smith · H.J. Siegel · J. Potter
Department of Electrical and Computer Engineering, Colorado State University, Fort Collins, CO, USA
e-mail: Abdulla.Al-Qawasmeh@colostate.edu

A.A. Maciejewski
e-mail: aam@colostate.edu

J. Smith
e-mail: jtsmith@digitalglobe.com

H.J. Siegel
e-mail: hj@colostate.edu

J. Potter
e-mail: jerry.potter@colostate.edu

H.J. Siegel
Department of Computer Science, Colorado State University, Fort Collins, CO, USA

H. Wang
Department of Statistics, Colorado State University, Fort Collins, CO, USA
e-mail: wanghn@stat.colostate.edu

J. Smith
DigitalGlobe Inc., Longmont, CO, USA

Published online: 22 February 2011

through simple mapping examples and synthetic data analysis. In addition, we illustrate how regression trees can be used to predict the most appropriate heuristic for an HC system based on its heterogeneity.

Keywords Distributed systems · Heterogeneity · Heterogeneous systems · Mapping heuristics · Task allocation · Regression trees

1 Introduction

We study heterogeneous computing (HC) systems that consist of a set of different machines that have varying capabilities. These machines are used to execute a set of heterogeneous tasks that vary in their computational complexity. Many of today's supercomputers are heterogeneous, e.g., the Extreme Scale Systems Center (ESSC) at Oak Ridge National Laboratory (ORNL). Furthermore, many of the homogeneous systems today may become heterogeneous in the future by adding new processing units with different capabilities than the existing ones.

The Estimated Time to Compute (*ETC*) each task on each machine in an HC system is arranged in an *ETC matrix*, where entry ETC(i, j) is the estimated execution time of task i on machine j when executed alone. The assumption of such ETC information is a common practice in resource allocation research (e.g., [6, 16, 21, 26, 29, 36, 39]). An ETC matrix for a given HC system can be obtained from user supplied information, experimental data, or task profiling and analytical benchmarking [2, 21, 29, 39].

Machine heterogeneity is the degree to which the execution time of a given task varies for different machines (the variation along the same row of an ETC matrix). Analogously, *task heterogeneity* is the degree to which the execution times of different tasks vary for the same machine (the variation along the same column) in the ETC matrix.

In an HC system, tasks should be mapped (allocated) to the available machines in a way that optimizes some performance objective (e.g., [6, 7, 9–11, 13, 30–33, 37]). Mapping tasks to machines in HC systems has been shown to be, in general, an NP-complete problem [14, 19, 24]. Hence, many heuristics have been developed for allocating tasks to machines in HC systems. The performance of allocation heuristics and the HC system is affected by several factors, one of which is the level of machine heterogeneity [9, 13]. Therefore, quantifying the heterogeneity of a given environment will allow the selection of a heuristic that is the most appropriate.

In most previous work (e.g., [5, 9, 12, 18, 23, 28]), either the range of the execution time values or their coefficient-of-variation (COV) was used as a measure of the heterogeneity to generate ETC matrices for simulation studies. These measures do not completely represent the possible variation in heterogeneity. For example, many ETC matrices with the same COV can have other statistical properties that are vastly different and that may be highly correlated with the performance of a mapping heuristic. Furthermore, these measures were intended for quantifying the heterogeneity of existing HC systems.

The contributions of this paper are the use of: (1) different statistical measures to quantify task and machine heterogeneities for existing HC systems, (2) simple mapping examples and synthetic data analysis that demonstrate the importance of these measures, and (3) regression trees to predict the most appropriate heuristic for a given HC system based on its heterogeneity. The regression trees demonstrate the importance of the heterogeneity measures in significantly reducing the error in predicting the most appropriate heuristic.

This paper is organized as follows. The procedures for each of the studied heuristics are given in Sect. 2. In Sect. 3, we describe different statistical measures for quantifying heterogeneity. Heuristic selection based on the heterogeneity measures is discussed in Sect. 4. Synthetic data analysis to demonstrate heuristic selection based on heterogeneity is given in Sect. 5. Section 6 discusses related work. Conclusions are given in Sect. 7.

2 Heuristics

The task mapping problem that we study in this paper is a static one (i.e., task mapping decisions are made before any task is executed). Further, there are no inter-task dependences. This mapping problem has been studied widely (e.g., [1, 24, 37, 38]). The makespan of a task mapping is the latest finish time among all machines, where a lower makespan is better. We study five heuristics to derive task mappings that minimize the makespan: Min-Min [24], Max-Min [24], Sufferage [31], MCT (*m*inimum *c*ompletion *t*ime) [1], and KPB (*k*-*p*ercent *b*est) [31]. The performance of the heuristics is quantified by the makespan.

The Min-Min heuristic passes through the unmapped tasks twice. In the first pass, for each task, it finds the machine that gives the *minimum* completion time. In the second pass, it finds the task with the *minimum* overall completion time and maps it to the minimum completion time machine identified in the first pass. The two passes are repeated until all tasks are assigned. The pseudocode of the Min-Min heuristic is given in Fig. 1. The Max-Min heuristic is similar to Min-Min except that in the second pass instead of mapping the task that has the *minimum* completion time it maps the task that has the *maximum* completion time (i.e., Max-Min starts by assigning the tasks with longer execution times).

The sufferage heuristic attempts to minimize the makespan by assigning every task to its best machine, but when multiple tasks want the same machine, it gives preference to the task that would "suffer" the most if it were not assigned to that machine. The *sufferage value* of a given task is the difference between its completion time on the machine with the second earliest completion time and the completion time

```
Do the following steps while there are unmapped tasks:
(1) For each unmapped task, determine the machine that gives the task its minimum completion time.
(2) Among the task-machine pairs determined in (1), map the task that has the minimum overall completion time to
    the corresponding machine.
(3) Update the ready times of each machine.
```

Fig. 1 The pseudocode of the Min-Min heuristic

```
Do the following while there are still unmapped tasks:
(1) For each machine, find the set of tasks that have their minimum completion time on this machine:
        (a) if the size of the set is one, then assign the corresponding task to the machine;
        (b) if the size of the set is greater than one, then assign the task with the highest sufferage
        value to the corresponding machine.
(2) Update the ready times for all the machines.
```

Fig. 2 The pseudocode of the sufferage heuristic

on the machine with the earliest completion time. The pseudocode of the sufferage heuristic is given in Fig. 2.

As the name implies, the MCT heuristic loops thought all of the tasks in arbitrary order and assigns each task to the machine that gives the minimum completion time for that task. The KPB heuristic also loops through the tasks in arbitrary order. However, for a given task, KPB considers a subset of machines for mapping that task. The subset is formed by picking the k-percent of all the machines that have the best (lowest) execution time for that task. Within the k-percent best machines the task is assigned to the machine with the minimum completion time. The k value is a parameter that can be set by the user. For the purpose of illustration, we used a k value of 0.75.

3 Measuring heterogeneity

3.1 Introduction

In this section, we illustrate how three statistical measures are used to quantify the heterogeneity of an HC system represented by an ETC matrix. These statistical measures are: (a) coefficient of variation, (b) skewness, and (c) kurtosis.

The following variables will be used in the calculation of each of the statistical measures: T is the number of tasks to be mapped, M is the number of machines in the system, and $\mu_i^{(\mathrm{t})}$ is the mean ETC of task i over all machines, given by $\mu_i^{(\mathrm{t})} = \frac{1}{M}\sum_{j=1}^{M} \mathrm{ETC}(i,j)$. The mean ETC of all tasks on machine j, $\mu_j^{(\mathrm{m})}$, is given by $\mu_j^{(\mathrm{m})} = \frac{1}{T}\sum_{i=1}^{T} \mathrm{ETC}(i,j)$. The standard deviation of the ETC of task i over all machines, $\sigma_i^{(\mathrm{t})}$, is given by $\sigma_i^{(\mathrm{t})} = \sqrt{\frac{1}{M}\sum_{j=1}^{M}(\mathrm{ETC}(i,j) - \mu_i^{(\mathrm{t})})^2}$. The standard deviation of the ETC of all tasks on machine j, $\sigma_j^{(\mathrm{m})}$, is given by $\sigma_j^{(\mathrm{m})} = \sqrt{\frac{1}{T}\sum_{i=1}^{T}(\mathrm{ETC}(i,j) - \mu_j^{(\mathrm{m})})^2}$.

3.2 Coefficient of variation

The COV has been used in [5] to generate ETC matrices with different task and machine heterogeneities.

For a set of values with standard deviation σ and mean μ, the *COV* is given by $\mathrm{COV} = \frac{\sigma}{\mu}$. Let $V_i^{(\mathrm{t})}$ for task i be the COV over all machines, given by $V_i^{(\mathrm{t})} = \frac{\sigma_i^{(\mathrm{t})}}{\mu_i^{(\mathrm{t})}}$.

Let $V_j^{(\mathrm{m})}$ for machine j be the COV over all tasks, given by $V_j^{(\mathrm{m})} = \frac{\sigma_j^{(\mathrm{m})}}{\mu_j^{(\mathrm{m})}}$. Task heterogeneity as measured by the *Average Task COV* (*ATC*) is given by $\mathrm{ATC} = [\sum_{j=1}^{M} V_j^{(\mathrm{m})}]/M$. Machine heterogeneity as measured by the *Average Machine COV* (*AMC*) is given by $\mathrm{AMC} = [\sum_{i=1}^{T} V_i^{(\mathrm{t})}]/T$.

Although both ATC and AMC quantify the variation of the execution time values, they do not indicate whether most of the values are less than or greater than the mean, and whether the variation is caused by many values having an average deviation from the mean, or a small number of values having a large deviation from the mean. These are quantified by skewness and kurtosis, respectively. Sections 4 and 5 describe how skewness and kurtosis may have an effect on the performance of heuristics. Using these measures we may be able to select a more appropriate heuristic for an HC system.

3.3 Skewness

The *skewness* of a set of values measures the degree of asymmetry of the values over the mean [35]. Skewness corresponds to the third moment of a distribution. Positive skewness means that most of the values are below the mean and negative skewness means that most of the values are greater than the mean.

Let $S_i^{(\mathrm{t})}$ for task i be the skewness over all machines, given by $S_i^{(\mathrm{t})} = [\frac{1}{M}\sum_{j=1}^{M} \times (\mathrm{ETC}(i,j) - \mu_i^{(\mathrm{t})})^3]/(\sigma_i^{(\mathrm{t})})^3$. Let $S_j^{(\mathrm{m})}$ for machine j be the skewness over all tasks, given by $S_j^{(\mathrm{m})} = [\frac{1}{T}\sum_{i=1}^{T}(\mathrm{ETC}(i,j) - \mu_j^{(\mathrm{m})})^3]/(\sigma_j^{(\mathrm{m})})^3$. Task heterogeneity as measured by the *Average Task Skewness* (*ATS*) is given by $\mathrm{ATS} = [\sum_{j=1}^{M} S_j^{(\mathrm{m})}]/M$. Machine heterogeneity as measured by the *Average Machine Skewness* (*AMS*) is given by $\mathrm{AMS} = [\sum_{i=1}^{T} S_i^{(\mathrm{t})}]/T$.

3.4 Kurtosis

The *kurtosis* of a set of values measures the extent to which the deviation is caused by a small number of values having extreme deviations from the mean versus a large number of values having modestly sized deviations [35]. Kurtosis corresponds to the fourth moment of a distribution. Higher values of kurtosis indicate that the standard deviation is caused by fewer values having extreme deviations. The definition of kurtosis that we use in this paper is the excess kurtosis [35]. Excess kurtosis equals kurtosis minus 3. This makes the excess kurtosis of the Gaussian (normal) distribution equal to 0.

Let $K_i^{(\mathrm{t})}$ for task i be the kurtosis over all machines, given by $K_i^{(\mathrm{t})} = [(\frac{1}{M}\sum_{j=1}^{M} \times (\mathrm{ETC}(i,j) - \mu_i^{(\mathrm{t})})^4)/(\sigma_i^{(\mathrm{t})})^4] - 3$. Let $K_j^{(\mathrm{m})}$ for machine j be the kurtosis over all tasks, given by $K_j^{(\mathrm{m})} = [(\frac{1}{T}\sum_{i=1}^{T}(\mathrm{ETC}(i,j) - \mu_j^{(\mathrm{m})})^4)/(\sigma_j^{(\mathrm{m})})^4] - 3$. Task heterogeneity as measured by the *Average Task Kurtosis* (*ATK*) is given by $\mathrm{ATK} = \frac{1}{M}\sum_{j=1}^{M} K_j^{(\mathrm{m})}$. Machine heterogeneity as measured by the *Average Machine Kurtosis* (*AMK*) is given by $\mathrm{AMK} = \frac{1}{T}\sum_{i=1}^{T} K_i^{(\mathrm{t})}$.

	m_1	m_2
t_1	6	10
t_2	14	8
t_3	40	36

AMC = 0.19
ATC = 0.72
AMS = 0
ATS = 0.62
AMK = − 2
ATK = − 1.5

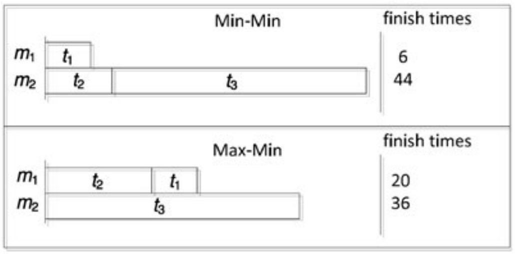


Fig. 3 An example that illustrates a situation where the Max-Min heuristic outperforms the Min-Min heuristic for an ETC matrix with high positive ATS. Shown in the figure are the ETC matrix, the values of the statistical measures for the matrix, and a pictorial representation of the mappings produced by the Max-Min and the Min-Min heuristics

4 Heuristic selection

4.1 Overview

This section illustrates how the heterogeneity measures can be used to make heuristic selection decisions based on the heterogeneity of an HC system. Section 4.2 illustrates the correlation that ATS and ATK may have with the performance of Max-Min and Min-Min heuristics by using simple mapping examples. Heuristic selection decisions based on a single heterogeneity measure and based on multiple heterogeneity measures are illustrated in Sects. 4.3 and 4.4, respectively.

4.2 Simple Mapping examples

The Min-Min and Max-Min heuristics have been studied widely (e.g., [3, 9, 10, 17, 20, 24, 25, 27, 31, 37]). Therefore, in this section, we selected these two heuristics to illustrate how skewness and kurtosis may affect the performance of the two heuristics using some simple task mapping examples.

Figure 3 shows a scenario in which the Max-Min heuristic outperforms the Min-Min heuristic for high values of ATS. The ETC shown in Fig. 3 has a positive ATS value of 0.62. A pictorial representation of the assignments made by each heuristic is given in the figure. The makespan of the mapping produced by the Min-Min heuristic is 44 and the makespan of the mapping produced by Max-Min is 36.

Although Min-Min performs better than Max-Min in most cases (e.g., [9, 10]), Max-Min usually has better performance than Min-Min when there are many shorter tasks than there are longer ones, i.e., the corresponding ETC matrix has high positive task skewness. This is because Max-Min starts by assigning the longer tasks to their best machine.

An example where Min-Min outperforms Max-Min for an ETC matrix with negative ATS is given in Fig. 4. The ETC in the figure has a negative ATS value of −0.23. A pictorial representation of the mapping produced by each heuristic is given in the figure. The makespan of the mapping produced by Min-Min is 46 and the makespan of the mapping produced by Max-Min is 56.

To show how kurtosis of an ETC matrix may impact the performance of the two heuristics, we compare the examples given in Figs. 5 and 6. The ETC matrices in both

	m_1	m_2
t_1	6	12
t_2	56	24
t_3	40	35

AMC = 0.27
ATC = 0.51
AMS = 0
ATS = − 0.23
AMK = − 2
ATK = − 1.5

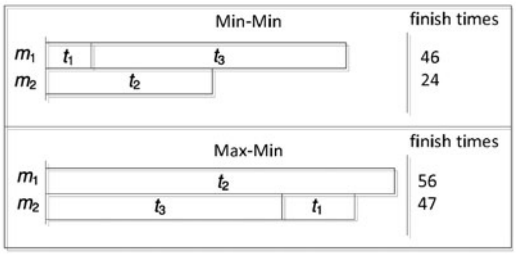


Fig. 4 An example that illustrates a situation where the Min-Min heuristic outperforms the Max-Min heuristic for an ETC matrix with negative ATS. Shown in the figure are the ETC matrix, the values of the statistical measures for the matrix, and a pictorial representation of the mappings produced by the Min-Min and the Max-Min heuristics

	m_1	m_2
t_1	3	5
t_2	55	53
t_3	49	47
t_4	50	52
t_5	54	56
t_6	45	51
t_7	54	50
t_8	105	93

AMC = 0.06
ATC = 0.47
AMS = 0
ATS = 0
AMK = − 2
ATK = 0.92

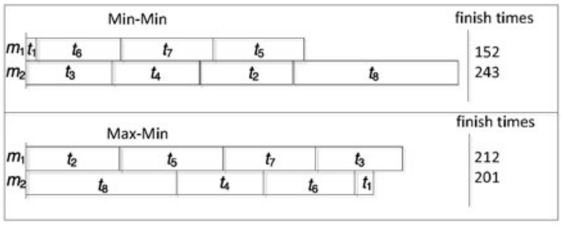
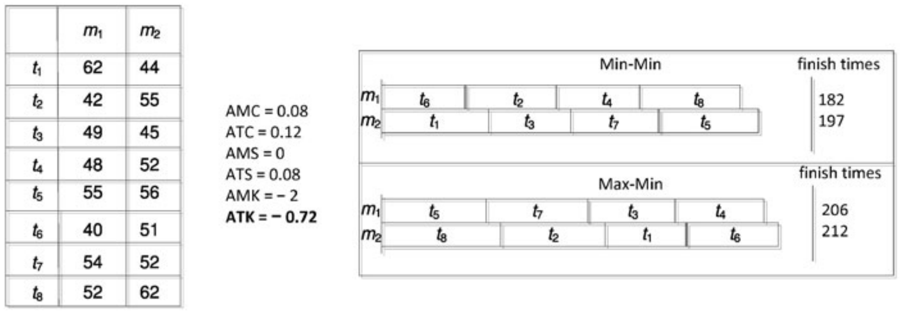


Fig. 5 An example that illustrates the situation where the Max-Min heuristic outperforms the Min-Min heuristic for an ETC matrix with high ATK and low ATS. Shown in the figure are the ETC matrix, the values of the statistical measures for the matrix, and a pictorial representation of the mappings produced by the Max-Min and the Min-Min heuristics

figures have an ATS value that is 0 or close to 0. Therefore, using the ATS will result in the same heuristic selection decision. However, using the ATK, better decisions can be made.

The ETC matrix in the Fig. 5 has a high ATK value of 0.92 (compared to the kurtosis of the normal distribution which is 0 and the uniform distribution which is -1.2). In Fig. 5, the Max-Min heuristic outperforms Min-Min. A pictorial representation of the mapping produced by both heuristics is given in the figure. The makespan for the mapping produced by Max-Min is 212 and the makespan for the mapping produced by Min-Min is 243. Although ATS for the ETC matrix in this figure is 0, the ETC matrix still has the property that there are few tasks that have a high execution time compared to the rest of the other tasks.

An example where Min-Min outperforms Max-Min for an ETC matrix with low ATK is given in Fig. 6. The ETC in the figure has a low ATK value of -0.72. A representation of the mappings produced by the two heuristics is given in the figure. The makespan of the mapping produced by Min-Min is 197 and the makespan of the mapping produced by Max-Min is 212.



	m_1	m_2
t_1	62	44
t_2	42	55
t_3	49	45
t_4	48	52
t_5	55	56
t_6	40	51
t_7	54	52
t_8	52	62

AMC = 0.08
ATC = 0.12
AMS = 0
ATS = 0.08
AMK = − 2
ATK = − 0.72

Min-Min
finish times
m_1: t_6 t_2 t_4 t_8 — 182
m_2: t_1 t_3 t_7 t_5 — 197
Max-Min
finish times
m_1: t_5 t_7 t_3 t_4 — 206
m_2: t_8 t_2 t_1 t_6 — 212

Fig. 6 An example ETC matrix that illustrates the situation where the Min-Min heuristic outperforms the Max-Min heuristic for an ETC matrix with low kurtosis. Shown in the figure are the ETC matrix, the values of the statistical measures for the matrix, and a pictorial representation of the mappings produced by the Max-Min and the Min-Min heuristics

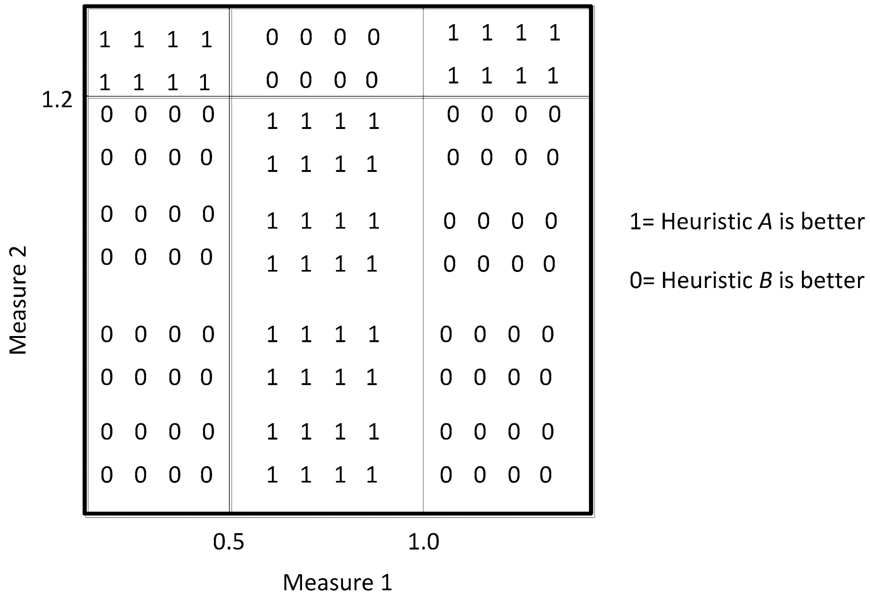


Fig. 7 An example to illustrate heuristic selection decisions

4.3 Selection based on one measure

One way to make heuristic selections is to identify a single measure that has the most correlation with the performance of the studied heuristics, and then identify the ranges of values within which a heuristic performs better. This method is a simple method and the selection decisions based on it are straightforward. However, in some cases this method may lead to a high number of wrong decisions. Consider the example in Fig. 7. This figure represents the performance of two heuristics A and B relative to the values of two heterogeneity measures 1 and 2. For selections based on a single measure, the best measure to use is measure 1 and the best decisions can be made if heuristic A is used when the value of measure 1 is between 0.5 and 1.0, and heuristic B is used otherwise. The number of wrong decisions in this case will be 24. The next section illustrates how a regression tree can be used to make better decisions based on multiple measures, which will result in no wrong decisions if it is used to combine both measures 1 and 2 to make heuristic selection decisions for Fig. 7.

4.4 Selection based on multiple measures

A regression tree [8] is a technique used to predict the outcome of a dependent variable based on a number of input variables. Regression trees are binary. The nodes of the tree represent yes/no questions about the input variables. If the answer to a question is yes, then the path along the left edge is followed, otherwise, the path along the right edge is followed. The question at each subsequent node is answered until a leaf node is reached. The leaf node represents a prediction of the value of the dependent variable based on the input variables. The prediction is simply the average of all the values of the dependent variable that belong to that node.

Following the CART technique proposed in [8], a regression tree is constructed by recursive partitioning of the data. At the beginning of the partitioning procedure, the question that will lead to the least error (quantified by the mean square error of the prediction) in the prediction is considered the root node of the tree. After the first partition, two sub-problems are created. Each sub-problem is solved in a similar manner as the original problem. A node is considered for partitioning if the following two conditions are true: (1) the number of values that belong to that node is greater than the *minimum node size*, and (2) there exists a split that leads to a decrease in the overall mean square error greater than or equal to the *minimum decrease in error*. Both minimum node size and minimum decrease in error are parameters set by the user. The minimum node size value that we have used is 500 and the minimum decrease in errorvalue that we have used is 0.01.

The regression tree for the example in Fig. 7 is given in Fig. 8. Measure 1 is at the root of the tree because it is the one that has the most correlation with the heuristics' performance, i.e., choosing the first split of based on the measure 1 results in the least error. This regression tree has 100% accuracy for the values in Fig. 7.

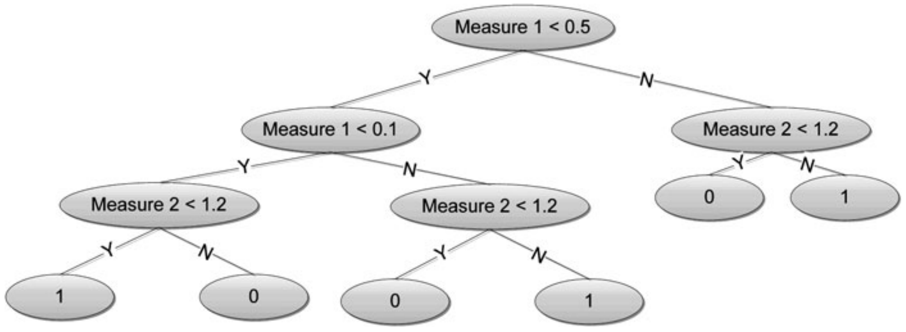


Fig. 8 The regression tree that corresponds for the example given in Fig. 7

To show how the performance of other heuristics is correlated with the statistical measures, we studied ETC matrices generated randomly and that have more tasks and machines than the matrices presented in this section. In the next section, we describe the result of the study.

5 Synthetic data

5.1 ETC matrix generation

Each of the ETC matrices that was used in this section was generated via the coefficient-of-variation-based method (CVB) proposed in [5]. The CVB method uses the COV to represent task and machine heterogeneity. To generate an ETC matrix, the CVB method takes three parameters: (a) task COV, (b) machine COV, and (c) the mean task execution time.

The CVB method uses the Gamma distribution [22] to generate the ETC values of an ETC matrix. The gamma distribution has a positive skewness. Furthermore, the COV, skewness, and kurtosis of the gamma distribution are correlated. Therefore, to show the effect of different combinations of COV, skewness, and kurtosis we have used the following seven distributions to generate the ETC values: (1) uniform, (2) gamma, (3) exponential, (4) Chi-square, (5) Cauchy, (6) normal, and (7) modified gamma distribution (described later in this section). Each of the distributions has different correlations between the COV, skewness, and kurtosis. The parameters of each of the distributions can be calculated based on the mean and COV values. The mean and COV values used to generate the ETC matrices are different for different simulations. The Cauchy distribution does not have a mean value. Therefore, the median was used as an estimate of the mean value. Any generated random value that is less than or equal to 0 is discarded.

The modified gamma distribution is obtained from the gamma distribution by truncating it at an upper limit value, normalizing it, and then inverting it. The truncation and normalization are done by discarding any generated values that have values higher than the upper limit. The inversion is done by subtracting the generated values from the upper limit value. Therefore, the modified gamma distribution will have negative skewness. The upper limit value equals the mean of the gamma distribution multiplied by an upper limit multiple u that is greater than 1. For our studies, we used $u = 2$. The procedure for generating the modified gamma distribution is depicted in Fig. 9. Each ETC matrix used in the simulations has 128 tasks and eight machines.

After each ETC matrix was generated using a specific mean, task COV, and machine COV, we calculated the statistical measures of the generated ETC matrix to obtain their actual values. The actual average machine and average task COV values of the generated ETC may differ from those used to generate the ETC matrix due to a finite number of values being generated. The maximum value of the COV was selected to be 1.5, based on preliminary experiments that showed no significant changes in performance among the heuristics studied for ETCs with larger COVs.

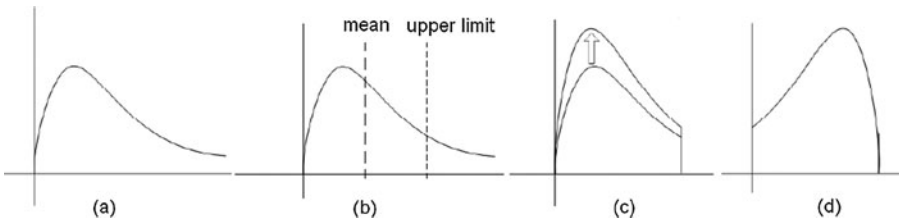


Fig. 9 The procedure for generating the modified gamma distribution: (**a**) the gamma distribution, (**b**) the truncation of the gamma distribution at the upper limit value based on $u = 2$ (i.e. the upper limit value will be twice the mean), (**c**) the normalization of the truncated distribution, and (**d**) the final modified gamma distribution (inverted)

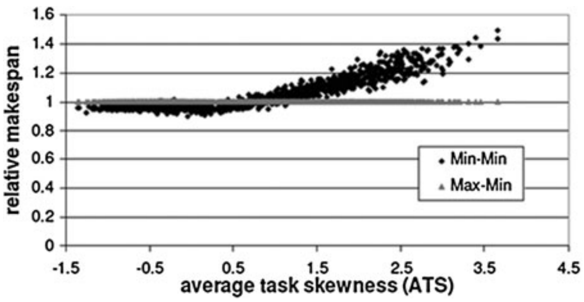


Fig. 10 Scatterplot of the relative makespan of Min-Min to Max-Min for 1445 ETC matrices generated using both the gamma distribution and the modified gamma distribution. The CVB machine COV in this figure is fixed at 0.1 and the CVB task COV is increased from 0.01 to 1.5 for both distributions

5.2 Data analysis: scatterplots

The scatterplots shown in this section represent a way to make heuristic selection decisions based on one measure. Figure 10 shows the correlation between the relative makespan of Min-Min to Max-Min and the ATS. Because of the correlation between the COV and the skewness of the gamma distribution, we do not know if ATS affects the relative makespan, or if the effect of ATS is a result of it being correlated with ATC and only ATC having the effect on the relative makespan. Therefore, in Fig. 10, we used both the modified gamma distribution and the gamma distribution to generate the ETC matrices. These two distributions have different correlations between their skewness and COV. Some of the ETC matrices (generated using the modified gamma distribution) have negative ATS values and have high ATC values; for those ETC matrices, Min-Min performs better than Max-Min. However, the ETC matrices (generated using the gamma distribution) that have high positive ATS values also have high ATC values; for those ETC matrices, Max-Min performs better. Therefore, we can see that there is a correlation between the ATS and the relative makespan of Min-Min to Max-Min.

For both distributions, the mean value was fixed at 20, the machine COV was fixed at 0.1, and the task COV was increased from 0.01 to 1.5. After the ETC matrices were generated, the actual ATS value was calculated for each ETC matrix. As shown in the

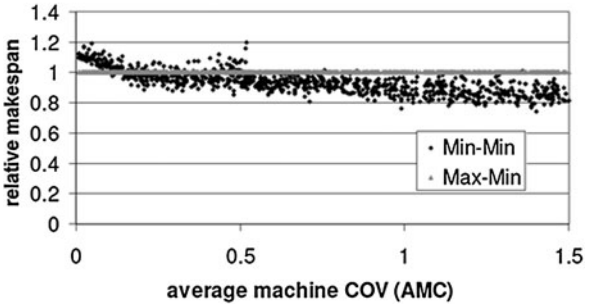


Fig. 11 Scatterplot of the relativemakespan of Min-Min to Max-Min for 925 ETC matrices generated using the gamma distribution. The task COV is fixed at 0.7, and the machine COV is increased from 0.01 to 1.5

figure, Max-Min outperforms Min-Min for ATS values greater than 1.4. However, for ATS values less than 0.5, Min-Min always outperforms Max-Min. We do not see a decrease in the relative makespan of Min-Min for negative ATS ETC matrices. This is because ETC matrices with negative ATS have very few tasks that have low execution times and the majority of the tasks have execution time values close to the mean task execution time. Therefore, making better choices of assigning the few low execution time tasks will not lead to a large performance gain.

The effect of AMC on the relative makespan of Min-Min to Max-Min is shown in Fig. 11. The gamma distribution was used to generate the ETC matrices in the figure. The mean ETC value was fixed at 20, the task COV was fixed at 0.7, and the machine COV was increased from 0.01 to 1.5. Min-Min almost always outperforms Max-Min for AMC values greater than 0.5. The reason Min-Min's performance relative to Max-Min's performance becomes better as the AMC value increases is because as the AMC increases the average difference between the best performing machine and the worst one becomes larger. Therefore, a task that has a lower execution time for a specific machine will have a higher ratio between the execution time on the best machine and the execution time on other machines. Assigning those tasks first (which Min-Min does) will result in a lower makespan.

The scatterplots in this section represent a simple way to compare the relative makespan and make heuristic selection decisions based on only one measure. In the next section, we show how regression trees can be used to compare the relative makespan of heuristics based on all of the six heterogeneity measures.

5.3 Data analysis: regression trees

In our study, we have used regression trees to predict whether a heuristic is better than another for a specific ETC matrix based on the values of the heterogeneity measures for that matrix. We have generated 100,000 ETC matrices randomly from the seven distributions described in Sect. 5.1. The mean ETC value used to generate the matrices is 500. The machine COV and the task COV values used to generate each matrix were sampled from a uniform distribution between the values of 0 and 1.5. For each matrix, the makespan of each of the five studied heuristics is calculated.

For any two heuristics A and B, let b be a variable that may have a value of either 0 or 1, such that if the makespan of B is less than A, then $b = 0$; otherwise, $b = 1$. Each regression tree in this section attempts to predict the value of b for two

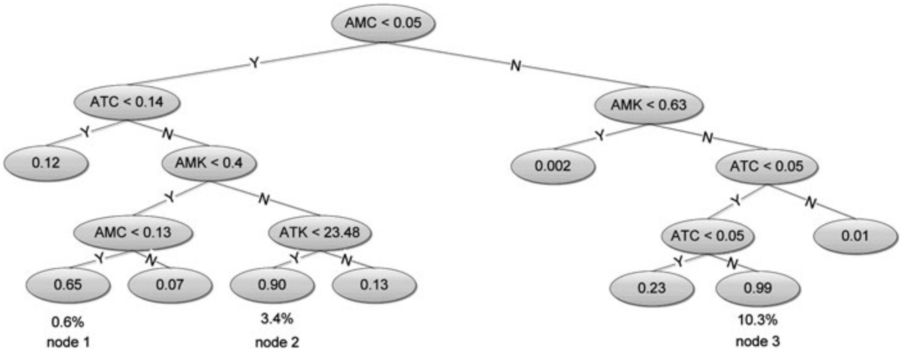


Fig. 12 The regression tree for Max-Min vs. sufferage. The percentages below some of the leaf nodes represent the average decrease in the makespan achieved by using Max-Min rather than sufferage for each ETC matrix that belongs to that node. Each of the values in the nodes is rounded to two decimal places except when that results in a 0 value

heuristics based on the heterogeneity values of the corresponding ETC matrices. The values at the leaf nodes of each tree represent the average of b for the ETC matrices that belong to that node. The closer the values are to 0 or 1 the better the prediction. For a given leaf node, if the predicted value of b is greater than 0.5, then we will consider using heuristic A for all ETC matrices that belong to that node; otherwise, we will use heuristic B.

Figure 12 shows the regression tree that attempts to predict the value of b when heuristic A is Max-Min and heuristic B is sufferage. The value in each of the leaf nodes represents the ratio of the number of times that Max-Min was better or equal to sufferage for all ETC matrices that belong to that node. Therefore, the number of wrong predictions of the tree can be calculated as follows. First, for each of the leaf nodes that have a value less than 0.5, multiply the node value by the number of ETC matrices that belong to that node, then sum across these leaf nodes. This value represents the number of times that we have chosen sufferage when Max-Min had better or equal performance. Second, for each of the leaf nodes that have a value greater than 0.5, multiply $(1 - \text{node value})$ by the number of ETC matrices that belong to that node, and then sum across these leaf nodes. This value represents the number of times that we have chosen Max-Min when we should have chosen sufferage. The total number of wrong predictions is the sum of the values obtained by the previous two steps. For the regression tree in Fig. 12, the total number of wrong predictions is 1064.

Among all of the 100,000 ETC matrices, sufferage is better than Max-Min for 95,252 of them. Therefore, without any knowledge about the values of the heterogeneity measures, a reasonable prediction would be to always use sufferage which will lead to 4748 wrong predictions. However, if we use the tree in Fig. 12, the number of wrong predictions can be reduced by 78%. In addition, for each of the leaf nodes 1, 2, and 3, choosing Max-Min rather than sufferage decreased the makespan. Node 3 has the highest average decrease in makespan for each ETC matrix that belongs to that node; the average decrease is 10.3%. Nodes 1 and 2 have less average decrease in the makespan. Note that the skewness measures did not have significant

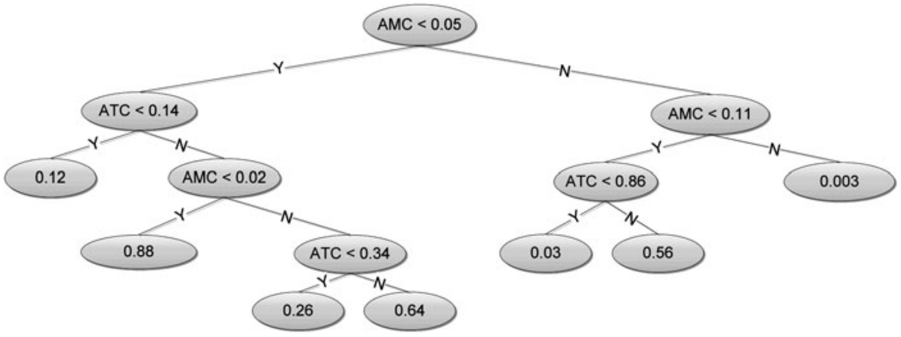


Fig. 13 The regression tree for Max-Min vs. sufferage when only the COV measures are used

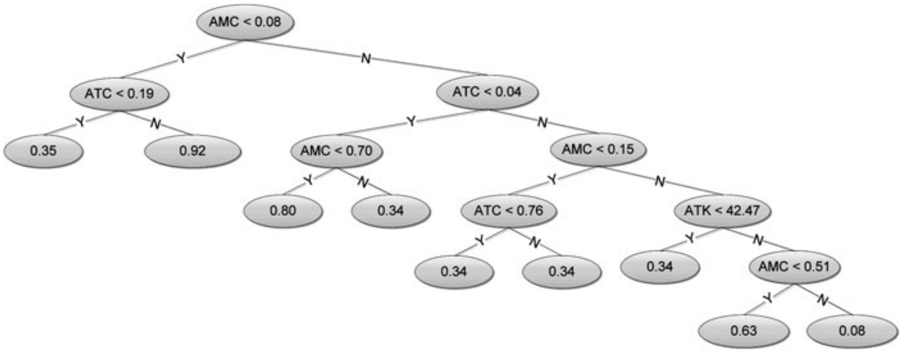


Fig. 14 The regression tree for Max-Min vs. MCT

effect on the decision of which of the two heuristics to use. Therefore, they were not used to partition any node.

Figure 13 shows the regression tree for the same b used in Fig. 12. However, in that tree, only the COV measures were used to make predictions. The number of wrong predictions in that tree is 2255. Therefore, using the kurtosis measures in Fig. 12 reduced the number of wrong predictions by 53%.

We have also built a regression tree for predicting the value of b when heuristic A is Max-Min and heuristic B is MCT. This tree is given in Fig. 14. For this tree, only three of the measures where used to make decisions, namely, AMC, ATC, and ATK. The regression tree when heuristic A is Max-Min and heuristic B is KPB is identical to the one in Fig. 14. This is due to the similarity between the procedures of both heuristics.

It is important to note that the results shown in this section apply for the distributions used to generate the ETC matrices. Other HC systems that have ETC matrices that belong to different underlying distributions may have different regression trees. For such systems, the average percentage decrease in the makespan when using a regression tree compared to using a heuristic H all the time can be computed as follows. First, multiply the average percentage decrease in makespan at each leaf node where H is not chosen by the number of ETC matrices that belong to that node. Then, take the average across all leaf nodes where H is not chosen. Finally, divide the result by the total number of ETC matrices used to construct the tree.

6 Related work

ETC matrices were previously used with different degrees of heterogeneity (e.g., [2, 6, 9, 10, 31, 37]). Most of these ETC matrices were generated by the range-based method described in [9] and the CVB method described in [5]. Therefore, depending on which method was used, heterogeneity was assumed to be either the range of the execution time values, or the COV. In [4], singular value decomposition of the ETC matrix was used to quantify task and machine affinity, and the average ratio of the sums of the columns of the ETC matrices was used to quantity machine performance homogeneity. This paper differs from [4] in that here we consider the COV, skewness, and kurtosis as heterogeneity measures, and show their impact on the performance of allocation heuristics.

Regression trees have been used widely in machine learning, decision making, pattern recognition, and data mining. For example, in [15], a number of machine learning techniques including regression trees have been used to predict the performance of total order broadcast algorithms in systems running heterogeneous workloads. A number of workload and system characteristics were used as input variables for predicting the performance in terms of message delivery latency and throughput. Another example of using regression trees is given in [34]. In that work, the authors propose the use of regression trees to predict the performance of transactional memory workloads based on different hardware transactional memory design dimensions and multicore microarchitecture configuration.

7 Conclusions

In this paper, we proposed a number of statistical measures to quantify the heterogeneity of HC systems. A method to calculate each of the measures for an existing ETC matrix was described. The impact that the heterogeneity measures may have on the performance of five different heuristics was demonstrated through simple examples. In addition, ETC matrices generated randomly via seven different distributions have been used to show how using regression trees to analyze the information provided by the heterogeneity measures allows us to make better heuristic selection decisions.

Acknowledgements The authors thank Arnold Rosenberg, Paul Maxwell, Luis Briceño, and Ron Chiang for their valuable comments on this work. A preliminary version of portions of this material was presented at the 2009 International Conference on Parallel and Distributed Processing Techniques and Applications. This research was supported by the NSF under Grants Nos. CNS-0615170 and CNS-0905399, and by the Colorado State University George T. Abell Endowment.

References

1. Armstrong R, Hensgen D, Kidd T (1998) The relative performance of various mapping algorithms is independent of sizable variances in run-time predictions. In: 7th IEEE heterogeneous computing workshop (HCW '98), 1998, pp 79–87
2. Ali S, Braun TD, Siegel HJ, Maciejewski AA, Beck N, Bölöni L, Maheswaran M, Reuther AI, Robertson JP, Theys MD, Yao B (2005) Characterizing resource allocation heuristics for heterogeneous computing systems. In: Parallel, distributed, and pervasive computing. Advances in computers, vol 63, pp 91–128
3. Ali S, Kim JK, Siegel HJ, Maciejewski AA (2008) Static heuristics for robust resource allocation of continuously executing applications. J Parallel Distrib Comput 68(8):1070–1080
4. Al-Qawasmeh AM, Maciejewski AA, Siegel HJ (2010) Characterizing heterogeneous computing environments using singular value decomposition. In: 19th heterogeneity in computing workshop (HCW 2010), 24th international parallel and distributed processing symposium, workshops and PhD forum (IPDPSW 2010), Apr 2010
5. Ali S, Siegel HJ, Maheswaran M, Hensgen D, Ali S (2000) Representing task and machine heterogeneities for heterogeneous computing systems. Tamkang J Sci Eng 3(3):195–207. Special 50th anniversary issue
6. Barada H, Sait SM, Baig N (2001) Task matching and scheduling in heterogeneous systems using simulated evolution. In: 10th heterogeneous computing workshop (HCW 2001), 15th IEEE international parallel and distributed processing symposium (IPDPS 2001), Apr 2001
7. Banicescu I, Velusamy V (2001) Performance of scheduling scientific applications with adaptive weighted factoring. In: 10th heterogeneous computing workshop (HCW 2001), 15th IEEE international parallel and distributed processing symposium (IPDPS 2001), Apr 2001
8. Breiman L, Friedman J, Olshen R, Stone C (1984) Classification and regression trees. Wadsworth, Belmont
9. Braun TD, Siegel HJ, Beck N, Bölöni L, Freund RF, Hensgen D, Maheswaran M, Reuther AI, Robertson JP, Theys MD, Yao B (2001) A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems. J Parallel Distrib Comput 61(6):810–837
10. Braun TD, Siegel HJ, Maciejewski AA, Hong Y (2008) Static mapping heuristics for tasks with dependencies, priorities, deadlines, and multiple versions in heterogeneous environments. J Parallel Distrib Comput 68(11):1504–1516
11. Burns A, Punnekkat S, Littlewood B, Wright D (1997) Probabilistic guarantees for fault tolerant real-time systems. Technical Report Design for Validation (DeVa) TR No. 44, Esprit Long Term Research Project No. 20072, Department of Computer Science, University of Newcastle upon Tyne, UK
12. Canon L, Jeannot E (2009) Precise evaluation of the efficiency and the robustness of stochastic DAG schedules. Research Report 6895, INRIA, April
13. Chiang RC, Maciejewski AA, Rosenberg AL, Siegel HJ (2010) Statistical predictors of computing power in heterogeneous clusters. In: 19th heterogeneity in computing workshop (HCW 2010), 24th International parallel and distributed processing symposium, workshops and PhD forum (IPDPSW 2010), Apr 2010
14. Coffman EG (ed) (1976) Computer and job-shop scheduling theory. Wiley, New York
15. Couceiro M, Romano P, Rodrigues L (2010) A machine learning approach to performance prediction of total order broadcast protocols. In: 4th IEEE international conference on self-adaptive and self-organizing systems (SASO), 2010
16. Dhodhi MK, Ahmad I, Yatama A (2002) An integrated technique for task matching and scheduling onto distributed heterogeneous computing systems. J Parallel Distrib Comput 62:1338–1361
17. Ding Q, Chen G (2001) A benefit function mapping heuristic for a class of meta-tasks in grid environments. In: CCGRID '01: 1st international symposium on cluster computing and the grid, May 2001
18. Eslamnour B, Ali S (2009) Measuring robustness of computing systems. Simul Model Pract Theory 17(9):1457–1467
19. Fernandez-Baca D (1989) Allocating modules to processors in a distributed system. IEEE Trans Softw Eng 15(11):1427–1436
20. Ghanbari S, Meybodi MR (2005) On-line mapping algorithms in highly heterogeneous computational grids: a learning automata approach. In: International conference on information and knowledge technology (IKT '05), May 2005

21. Ghafoor A, Yang J (1993) A distributed heterogeneous supercomputing management system. IEEE Trans Comput 26(6):78–86
22. Gubner JA (2006) Probability and random processes for electrical and computer engineering. Cambridge University Press, Cambridge
23. Huang D, Yuan Y, Zhang L, Zhao K (2009) Research on tasks scheduling algorithms for dynamic and uncertain computing grid based on a + bi connection number of SPA. J Softw 4(10):1102–1109
24. Ibarra OH, Kim CE (1977) Heuristic algorithms for scheduling independent tasks on nonidentical processors. J ACM 24(2):280–289
25. Jinquan Z, Lina N, Changjun J (2005) A heuristic scheduling strategy for independent tasks on grid. In: 8th international conference on high-performance computing in Asia–Pacific region, Nov 2005
26. Kafil M, Ahmad I (1998) Optimal task assignment in heterogeneous distributed computing systems. IEEE Concurr 6(3):42–51
27. Kaya K, Ucar B, Aykanat C (2007) Heuristics for scheduling file-sharing tasks on heterogeneous systems with distributed repositories. J Parallel Distrib Comput 67(3):271–285
28. Khan SU, Ahmad I (2006) Non-cooperative, semi-cooperative, and cooperative games-based grid resource allocation. In: 20th international parallel and distributed processing symposium (IPDPS 2006), Apr 2006
29. Khokhar A, Prasanna VK, Shaaban ME, Wang C (1993) Heterogeneous computing: challenges and opportunities. IEEE Trans Comput 26(6):18–27
30. Kim J-K, Hensgen DA, Kidd T, Siegel HJ, John DS, Irvine C, Levin T, Porter NW, Prasanna VK, Freund RF (2006) A flexible multi-dimensional QoS performance measure framework for distributed heterogeneous systems. Cluster Comput 6(3):281–296. Special issue on cluster computing in science and engineering
31. Maheswaran M, Ali S, Siegel HJ, Hensgen D, Freund RF (1999) Dynamic mapping of a class of independent tasks onto heterogeneous computing systems. J Parallel Distrib Comput 59(2):107–121
32. Mehta AM, Smith J, Siegel HJ, Maciejewski AA, Jayaseelan A, Ye B (2007) Dynamic resource allocation heuristics that manage trade-off between makespan and robustness. J Supercomput 42(1):33–58. Special issue on grid technology
33. Michalewicz Z, Fogel DB (eds) (2000) How to solve it: modern heuristics. Springer, New York
34. Poe J, Cho C-B, Li T (2008) Using analytical models to efficiently explore hardware transactional memory and multi-core co-design. In: 20th international symposium on computer architecture and high performance computing, 2008
35. Press WH, Flannery BP, Teukolsky SA, Vetterling WT (1988) Numerical recipes in C. Cambridge University Press, Cambridge
36. Singh H, Youssef A (1996) Mapping and scheduling heterogeneous task graphs using genetic algorithms. In: 5th IEEE heterogeneous computing workshop (HCW '96), 1996, pp 86–97
37. Wu M, Shu W, Zhang H (2000) Segmented Min-Min: a static mapping algorithm for meta-tasks on heterogeneous computing systems. In: 9th IEEE heterogeneous computing workshop, Mar 2000, pp 375–385
38. Wu M, Shu W (2001) A high-performance mapping algorithm for heterogeneous computing systems. In: 15th international parallel and distributed processing symposium (IPDPS 2001), Apr 2001
39. Xu D, Nahrstedt K, Wichadakul D (2001) QoS and contention-aware multi-resource reservation. Cluster Comput 4(2):95–107