On the Heterogeneity Bias of Cost Matrices when Assessing Scheduling Algorithms

Louis-Claude Canon, Laurent Philippe

INSTITUT FEMTO-ST

UMR CNRS 6174

On the Heterogeneity Bias of Cost Matrices when Assessing Scheduling Algorithms

Version 2

Louis-Claude CANON — Laurent PHILIPPE

Rapport de Recherche n° RR–FEMTO-ST–8663

DÉPARTEMENT DISC – November 7, 2016

On the Heterogeneity Bias of Cost Matrices when Assessing Scheduling Algorithms

Version 2

Louis-Claude CANON , Laurent PHILIPPE

Département DISC

Rapport de Recherche n° RR–FEMTO-ST–8663 November 7, 2016 (53 pages)

Abstract: Assessing the performance of scheduling heuristics through simulation requires one to generate synthetic instances of tasks and machines with well-identified properties. Carefully controlling these properties is mandatory to avoid any bias. We consider the scheduling problem consisting of allocating independent sequential tasks on unrelated machines while minimizing the maximum execution time. In this problem, the instance is a cost matrix that specifies the execution cost of any task on any machine. This report proposes two measures for quantifying the heterogeneity properties of a cost matrix. An analysis of two classical methods used in the literature reveals a bias in previous studies. Two new methods are proposed to generate instances with given heterogeneity properties and it is shown that they have a significant impact on several heuristics.

Key-words: Scheduling; Cost Matrix; Heterogeneity; Bias; Parallelism; Unrelated; Measure.

FEMTO-ST Institute, DISC research department
UFR Sciences - Route de Gray - F-25030 BESANÇON cedex FRANCE
Tel: (33 3) 81 66 65 15 – Fax: (33 3) 81 66 64 50 – e-mail: fvernill@femto-st.fr

Sur le biais d'hétérogénéité dans les matrices de coût lors de l'évaluation des algorithmes d'ordonnancement

Version 2

Résumé : Évaluer la performance des heuristiques d'ordonnancement avec des simulations nécessite de générer des instances synthétiques de tâches et de machines avec des propriétés clairement identifiées. Contrôler ces propriétés avec soin est indispensable pour éviter tout biais. Nous considérons le problème d'ordonnancement qui consiste à allouer des tâches séquentielles indépendantes sur des machines indépendantes tout en minimisant le temps d'exécution maximum. Dans ce problème, l'instance est une matrice de coût qui fournit le temps d'exécution de chaque tâche sur chaque machine. Ce rapport propose deux mesures pour quantifier les propriétés d'hétérogénéité d'une matrice de coût. Une analyse de deux méthodes classiques utilisées dans la littérature montre un biais dans les études précédentes. Deux nouvelles méthodes sont proposées pour générer des instances avec des propriétés d'hétérogénéité données et nous montrons qu'elles ont un impact significatif sur plusieurs heuristiques.

Mots-clés : ordonnancement; matrices de coût; hétérogénéité; biais; parallélisme; mesures.

FEMTO-ST Institute, DISC research department
UFR Sciences - Route de Gray - F-25030 BESANÇON cedex FRANCE
Tel: (33 3) 81 66 65 15 – Fax: (33 3) 81 66 64 50 – e-mail: fvernill@femto-st.fr

1 Introduction

Leveraging the parallelism of multi-core distributed platforms involves efficiently scheduling applications on several machines [105]. Current studies rely on performance evaluation to determine the best solution for any underlying problem. This process can be divided into several categories: formal analysis, experiments, simulations, etc. In the case of simulations, a scheduling strategy is tested in a virtual environment with a given workload. This paper focuses on the generation of synthetic instances.

Synthetic instances of workload allow a more general evaluation than with specific traces. They are particularly useful for sensitivity analysis [139], which consists in assessing the impact of the instance properties on the algorithms. However, the lack of control on the instance properties makes it difficult to confront the results of independent studies. For instance, although many papers have compared several scheduling heuristics [22, 26, 43, 109], predicting their performance is still an issue. These problems can be tackled by carefully controlling the instance properties.

Specifically, we consider the scheduling problem noted $R||C_{\max}$ in $\alpha|\beta|\gamma$ notation [63]. It consists in scheduling n independent sequential tasks on m unrelated machines. All tasks are available simultaneously and preemption is not possible. The instance is a *cost matrix* where each element $e_{i,j}$ is a positive integer that represents the execution cost of task i on machine j. The objective is to allocate each task to a machine such that the maximum execution time on any machine is minimized. More formally, we want to minimize $\max(\sum \pi(i,j) \times e_{i,j})$ where $\pi(i,j)$ is equal to one if task i is scheduled on machine j and zero otherwise.

For this problem, the range-based and CVB (Coefficient of Variation Based) methods proposed in [17, 18] are currently the standard methods used in the literature to generate instances. However, the properties of the matrices generated with these methods have never been formally analysed and previous studies may thus be exposed to a bias.

This paper provides the following contributions:[1]

- a statistical description of the use of the range-based and CVB methods in the literature (Section 3);
- a study of how to quantify the heterogeneity properties of a cost matrix (Section 4);
- a formal analysis of the range-based and CVB methods and the identification of a bias that impacts several studies (Section 4);
- a new method with control over heterogeneity properties (Section 5);
- and, an assessment of the impact of these properties on several heuristics (Section 6).

2 Related Work

The concept of heterogeneity was first introduced in the context of cost matrix by Armstrong [20]. He described the *heterogeneity quadrant* in which cost matrices are divided into four categories depending on their heterogeneity properties regarding tasks and machines: low/low, low/high, high/low, and high/high. For instance low/high refers to low task heterogeneity and high machine heterogeneity. However, no method for generating such matrices was proposed.

The range-based and CVB methods were first proposed to fill this gap in [8] and then in [17, 18]. However, task and machine heterogeneities were not formally defined and analyzed. The methods were assumed to generate matrices with the expected properties and only validated through some examples.

[1] The related code, data and analysis are available in [37].

The limits of these methods were later acknowledged in [5], which proposed to consider the average coefficient of variation[2], skewness and kurtosis of the costs for each task and for each machine. The proposed scheme (based on decision trees) uses these additional information to predict heuristic performance. Despite a wide experimentation plan, the study lacks discussion and interpretation in particular on the relative importance of the considered measures. Additionally, no formal analysis was provided. The exhibited decision trees suggest that the average coefficient of variation plays a significant role, which supports the current work.

The MPH (Machine Performance Homogeneity) is introduced in [3] for capturing the heterogeneity between the machines while its counterpart for the tasks, the TDH (Task Difficulty Homogeneity), appears in [2]. We discuss them more extensively in Section 4. In addition, the TMA (Task-Machine Affinity) is also defined in [3]: it quantifies the specialisation of the system (i.e., whether some machines are particularly efficient for some specific tasks). Although the three measures are applied to a real benchmark, no method is proposed for generating matrices with given MPH, TDH and TMA. It is thus unclear what is the impact of the proposed measures on heuristic performance. Finally, they show that the range-based and CVB methods do not cover the entire range of possible values for the MPH and the TMA, which is consistent with the conclusion of Section 4.

Friese et al [53] present a method for adding tasks in a given cost matrix while preserving some statistical properties on each column (mean, coefficient of variation, skewness and kurtosis). It ignores the properties on each row however.

A method for generating matrices with varying affinities (similar to the TMA) is proposed in [6]. It is similar to the noise-based method described in Section 5, but no formal analysis is provided.

Khemka et al [82] propose a method for changing the TMA of an existing matrix while keeping the same MPH and TDH. TMA is mentioned to be related to the correlation. Investigating the correlation properties is left for future work. There is also another field of studies dedicated to the generation of matrices with given correlation and covariance matrices [59].

3 Matrix Generation Methods

The most used methods for generating cost matrices are the range-based and the CVB (Coefficient of Variation Based) methods [8, 17, 18]. The most frequent notations are summarized in Appendix A.

3.1 Range-Based Method

The range-based method generates n vectors of m values that follow a uniform distribution in the range $[1, R_{mach}]$ (see Algorithm 1). Each row is then multiplied by a random value that follows a uniform distribution in the range $[1, R_{task}]$ (Line 2). The resulting cost matrix is similar to the following (where τ is a vector of n uniform values in $[1, R_{task}]$):

$$\begin{pmatrix} \tau[1]U(1,R_{mach}) & \cdots & \tau[1]U(1,R_{mach}) \\ \vdots & \ddots & \vdots \\ \tau[n]U(1,R_{mach}) & \cdots & \tau[n]U(1,R_{mach}) \end{pmatrix}$$

Proposition 1. *When used with parameters R_{task} and R_{mach}, the range-based method generates costs with expected value $\frac{1}{4}(R_{task}+1)(R_{mach}+1)$ and standard deviation $\frac{1}{12}[(R_{task}-1)^2(R_{mach}-1)^2+3(R_{mach}-1)^2(R_{task}+1)^2+3(R_{task}-1)^2(R_{mach}+1)^2]^{1/2}$.*

[2] Ratio of the standard deviation to the mean.

Algorithm 1 Range-based cost matrix generation with the uniform distribution

Input: n, m, R_{task}, R_{mach}
Output: a $n \times m$ cost matrix

```
for all 1 ≤ i ≤ n do                                  {Generate each row}
    τ[i] ← U(1, R_task)
    for all 1 ≤ j ≤ m do                              {Generate each value of the row}
        e_{i,j} ← τ[i] × U(1, R_mach)
    end for
end for
return {e_{i,j}}_{1≤i≤n,1≤j≤m}
```

Property	Value
Expected value	$\frac{1}{4}(R_{task}+1)(R_{mach}+1)$
Standard deviation	$\frac{1}{12}\sqrt{\begin{array}{r}(R_{task}-1)^2(R_{mach}-1)^2+3(R_{mach}-1)^2(R_{task}+1)^2+\\ 3(R_{task}-1)^2(R_{mach}+1)^2\end{array}}$
CV	$\frac{1}{3}\sqrt{\frac{(R_{task}-1)^2(R_{mach}-1)^2}{(R_{task}+1)^2(R_{mach}+1)^2}+3\frac{(R_{task}-1)^2}{(R_{task}+1)^2}+3\frac{(R_{mach}-1)^2}{(R_{mach}+1)^2}}$
Distribution	Product of two uniform laws
Asymptotic expected value	$\frac{1}{4}R_{task}R_{mach}$
Asymptotic standard deviation	$\frac{\sqrt{7}}{12}R_{task}R_{mach}$
Asymptotic CV	$\frac{\sqrt{7}}{3} \approx 0.88$

Table 1: Summary of the cost matrix properties with the range-based method. Asymptotic values are when both R_{task} and R_{mach} are large.

Proof. Each cost is the product of $\tau[i]$, which follows a uniform law in the range $[1, R_{task}]$, and a random variable that follows a uniform law in the range $[1, R_{mach}]$. Therefore, the expected value of the costs is the product of the expected values of both distributions, namely $(R_{task}+1)/2$ and $(R_{mach}+1)/2$.

The standard deviation of the product of two random variables with means μ_1 and μ_2, and standard deviations σ_1 and σ_2 is $\sqrt{\sigma_1^2\sigma_2^2+\mu_1^2\sigma_2^2+\sigma_1^2\mu_2^2}$. With a similar argument as for the expected value, we can derive the standard deviation of the costs. □

Table 1 summarizes the properties of this method. Except for low values of R_{task} and R_{mach}, the CV (Coefficient of Variation) remains close to a constant. For instance, when $R_{task} = R_{mach} = 100$, then the CV is around 0.86. As shown in Section 4, this method is not well-suited to control the heterogeneity of the resulting cost matrix. Also, given that this method is asymmetric, it may be expected to handle task heterogeneity differently from machine heterogeneity.

3.2 CVB Method

The CVB method is based on the same principle except it uses parameters that are distinct from the underlying distribution parameters. In particular, it requires two CV (V_{task} for the tasks and V_{mach} for the machines) and one mean (μ_{task} for the tasks). The random values follow a gamma distribution whose parameters are computed such that the provided CV and mean are respected.

Algorithm 2 CVB cost matrix generation with the gamma distribution

Input: n, m, V_{task}, V_{mach}, μ_{task}
Output: a $n \times m$ cost matrix

1: $\alpha_{task} \leftarrow 1/V_{task}^2$
2: $\alpha_{mach} \leftarrow 1/V_{mach}^2$
3: $\beta_{task} \leftarrow \mu_{task}/\alpha_{task}$
4: **for all** $1 \leq i \leq n$ **do**
5: $\quad q[i] \leftarrow G(\alpha_{task}, \beta_{task})$
6: $\quad \beta_{mach}[i] \leftarrow q[i]/\alpha_{mach}$
7: $\quad$ **for all** $1 \leq j \leq m$ **do**
8: $\quad\quad e_{i,j} \leftarrow G(\alpha_{mach}, \beta_{mach}[i])$
9: $\quad$ **end for**
10: **end for**
11: **return** $\{e_{i,j}\}_{1 \leq i \leq n, 1 \leq j \leq m}$

Property	Value
Expected value	μ_{task}
CV	$\sqrt{V_{task}^2 V_{mach}^2 + V_{task}^2 + V_{mach}^2}$
Distribution	Product of two gamma laws

Table 2: Summary of the cost matrix properties with the CVB method.

Proposition 2. *When used with parameters V_{task}, V_{mach} and μ_{task}, the CVB method generates costs with expected value μ_{task} and coefficient of variation $\sqrt{V_{task}^2 V_{mach}^2 + V_{task}^2 + V_{mach}^2}$.*

Proof. In order to apply the same analysis as in the proof of Proposition 1, we need to prove that any cost is the product of two gamma distributions. More precisely, we need to prove that the random generation on Line 8 is equivalent to multiplying $q[i]$ by a gamma law with mean one and CV V_{mach}.

Each cost $e_{i,j}$ is a random variable that follows a gamma distribution with mean $q[i]$ and CV V_{mach}. The probability that $e_{i,j}$ is no more than x is given by $\frac{1}{\Gamma(\alpha)}\gamma(\alpha, \frac{x}{\beta})$ where $\alpha = 1/V_{mach}^2$, $\beta = q[i]/\alpha$, $\Gamma(\alpha)$ is the gamma function and $\Gamma(\alpha, \frac{x}{\beta})$ is the lower incomplete gamma function.

By contrast, let X be a random variable that follows a gamma distribution with mean one and CV V_{mach}. Then, the probability that $q[i]X$ is no more than x is the probability that X is no more than $x/q[i]$: $\frac{1}{\Gamma(\alpha)}\gamma(\alpha, \frac{x/q[i]}{\beta})$ where $\alpha = 1/V_{mach}^2$ and $\beta = 1/\alpha$. It is thus the same as for $e_{i,j}$.

Thus, Line 8 can be replaced by the product of $q[i]$ by a gamma law with mean one and CV V_{mach} (i.e., $e_{i,j} \leftarrow q[i]G(\alpha_{mach}, 1/\alpha_{mach})$), which is the product of two gamma distributions.

The proof is then analogous to the proof of Proposition 1. □

Table 2 summarizes the properties of this method, which is more adapted to control the heterogeneity of the resulting cost matrix. However, it is still asymmetric. Note that the CV is the same as with the range-based method when we replace V_{task} by the CV of the first uniform law, $\frac{\sqrt{12}}{6}\frac{R_{task}-1}{R_{task}+1}$, and V_{mach} by the CV of the second uniform law, $\frac{\sqrt{12}}{6}\frac{R_{mach}-1}{R_{mach}+1}$.

3.3 Consistency Extension

Both the previous methods produce cost matrices that may not be representative of realistic settings. For instance, the costs of a given task is not correlated to the costs of another task, which may often be the case in practice. The consistency extension consists in reordering the costs in the generated matrix to have an instance that is closer to the uniform case. Specifically, the rows of a submatrix of an rows and bm columns are sorted. Thus, a machine that is faster for a given task than another machine will likely be also faster for another task. Inconsistent matrices have $a = b = 0$ while consistent matrices have $a = b = 1$ (other matrices are either called semiconsistent or partially consistent).

3.4 Usage in the Literature

We covered the English articles that cite at least one of the references in which the methods were initially presented [8,17,18] and that were freely available. For each reference, we extracted all the distinct sets of parameters. Additionally, we differentiated between example cost matrices that illustrate the generation methods from cost matrices that are used in actual sets of experiments to study scheduling algorithms. However, the size was ignored as we only consider asymptotic properties (Section 4.6 assesses the impact of the size).

Some data were not specifically provided. The parameters that could be directly inferred from the article or from similar works are emphasized: this concerns mostly missing parameters for the consistency extension (the ones from the cited article were taken). Otherwise, they are treated as missing values (denoted by NA). Some articles lack enough information, which prevented any parameter extraction.

On the 160 analysed articles, 78 provide exploitable information on the cost matrix instances. The rest consists of 40 articles with no description, but which refer to instances described in other articles and 42 articles with unclear descriptions or approaches that do not fit the current study. The extracted data are provided in Appendix B and summarized below. While most articles fail to precisely describe the used method, only the range and CV parameters are crucial for reproducing similar instances. In the end, 342 sets of parameters were extracted in 78 articles for a total of 210 unique settings: 37 for the range-based method and 173 for the CVB one.

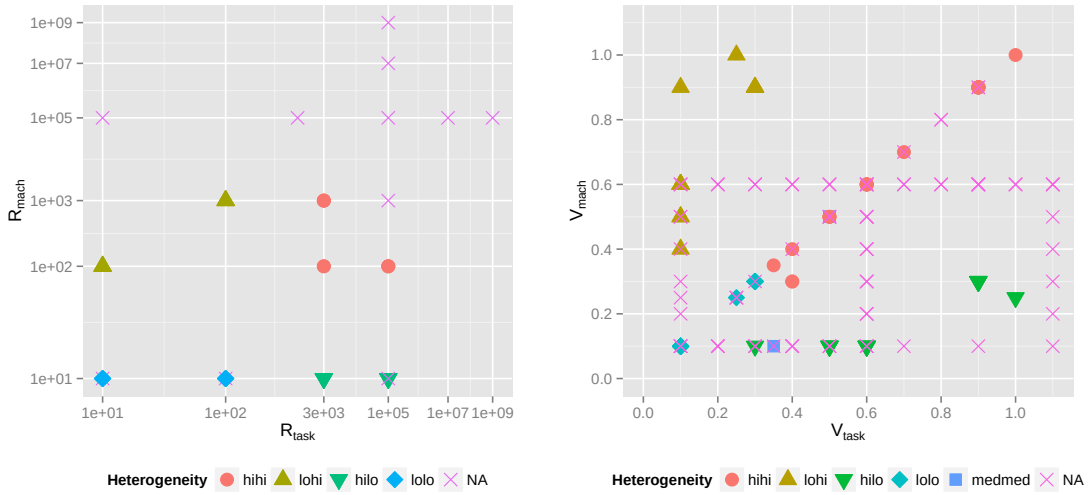
Figure 1 depicts the values used with both methods. Although there is no clear agreement on which precise parameters are the most relevant, there are some common tendencies. Values for low heterogeneity are usually 10 and 100 for the range-based method and .1, .25 and .3 for the CVB method. Values for high heterogeneity are usually 100, 1e3, 3e3 and 1e5 for the range-based method and .3, .35, .4, .5, .6, .7, .9, 1 and 2 for the CVB method.

4 Heterogeneity Measures

Assessing the impact of heterogeneity on heuristic performance requires a method for quantifying the heterogeneity of the generated cost matrices.

4.1 TDH and MPH

The closest related measures are the TDH (Task Difficulty Homogeneity) and the MPH (Machine Performance Homogeneity) [2, 3]. The TDH computation is described in Algorithm 3. The value $TD[i]$ represents the difficulty of task i, namely whether it has small costs. After the ordering, the final sum computes the average ratio between similar tasks in terms of difficulty (which lies in the interval $(0, 1]$). If this average is one, then tasks are all similar. If it is close to zero, then the task heterogeneity is large.



(a) Range-based method

(b) CVB method

Figure 1: Parameters used in the literature. Three points are not shown for the CVB method: $(1.4, 0.4)$, $(1.8, 0.4)$ and $(0.1, 2)$.

Algorithm 3 TDH computation

Input: a $n \times m$ cost matrix
Output: the TDH of this matrix
1: **for all** $1 \leq i \leq n$ **do**
2: $\quad TD[i] \leftarrow \sum_{j=1}^{m} \frac{1}{e_{i,j}}$
3: **end for**
4: sort TD in ascending order
5: **return** $\frac{1}{n-1} \sum_{i=1}^{n-1} \frac{TD[i]}{TD[i+1]}$

The MPH computation is analogous except that the sum on Line 2 is performed on each row instead of each column. This results in a measure of the machine heterogeneity.

These measures have three major shortcomings (as mentioned in Section 2). First, they are not intuitive (they require to invert costs, to order sums and to average ratios). Also, they do not rely on classical statistical measures, which makes deriving formal results more difficult. In particular, the ordering on Line 4 complicates formal analysis. A last notable problem is that the resulting values depend on the size of the matrix. In particular, it is close to one when the matrix is large (even if it is generated with the same parameters and has, intuitively, the same characteristics). For instance, if we consider only one machine, the following matrices (cost vectors in this case) have the same TDH: $[1, 2]$ and $[0.125, 0.25, 0.5, 1, 2, 4]$. The second vector, however, seems more heterogeneous. As another example, let the minimum TD be 1 and the maximum TD be 100. Given Proposition 3, the TDH is always greater than 0.60 when there are 10 tasks and it is always greater than 0.95 when there are 100 tasks. This measure is thus relevant only for comparing small cost matrices with similar sizes.

Proposition 3. *The TDH cannot be lower than* $e^{\log\left(\frac{\min(TD)}{\max(TD)}\right)/(n-1)}$.

Proof. The minimum TDH is achieved when the sum $\sum_{i=1}^{n-1} \frac{a_i}{a_{i+1}}$ where $a_i = TD[i]$ is minimum. Let $f : [a_1, a_n]^{n-2} \to (0, \infty)$ be the corresponding multivariate function with a_1 and a_n being

constant. Each value a_i for $1 < i < n$ is greater than or equal to a_1 because the a_i are ordered. As a_1 represents an average cost and is thus strictly greater than zero, all nominators and all denominators are strictly greater than zero. Therefore, f is a continuous function from the compact $[a_1, a_n]^{n-2}$. The extreme value theorem states that a continuous function from a non-empty compact space to a subset of the real numbers attains a maximum and a minimum. This proves the existence of a minimum.

We now show by contradiction that this minimum is achieved when the ratios $\frac{a_i}{a_{i+1}}$ are all equal for $1 \leq i < n$. Assume it is not the case and let i be the lowest value for which $\frac{a_i}{a_{i+1}} \neq \frac{a_{i+1}}{a_{i+2}}$, which can be rewritten as $a_{i+1} \neq \sqrt{a_i a_{i+2}}$. A lower value is attained when $a_{i+1} = \sqrt{a_i a_{i+2}}$ because the partial derivate of f with respect to a_{i+1} (i.e., $-\frac{a_{i-1}}{a_i^2} + \frac{1}{a_{i+1}}$) is zero with this value. Therefore, the minimum is achieved when all ratios $\frac{a_i}{a_{i+1}}$ are equal. This is the case when $a_i = e^{\log(TD[1]) + \frac{i-1}{n-1}\log\left(\frac{a_n}{a_1}\right)}$ for $1 \leq i \leq n$.

When replacing a_i by $TD[i]$, the TDH simplifies as $e^{\log\left(\frac{TD[1]}{TD[n]}\right)/(n-1)}$ or $e^{\log\left(\frac{\min(TD)}{\max(TD)}\right)/(n-1)}$ if the vector TD is not sorted. □

4.2 Intuitive Measures of Heterogeneity

We identify below two intuitive measures of task and machine heterogeneity that rely on classic properties:

- Assuming that the mean of each row represents a task weight, the task heterogeneity may be defined as the CV (Coefficient of Variation) of the means of the rows (noted $V\mu_{task}$). Analogously, the machine heterogeneity may be measured as the CV of the means of the columns (noted $V\mu_{mach}$).

$$CV \begin{Bmatrix} \mu_1 \\ \vdots \\ \mu_n \end{Bmatrix} \begin{pmatrix} \boxed{e_{1,1} \quad \cdots \quad e_{1,m}} \\ \vdots \quad \ddots \quad \vdots \\ \boxed{e_{n,1} \quad \cdots \quad e_{n,m}} \end{pmatrix}$$

- Alternatively, the CV of one column may represent the task heterogeneity for a given machine. Therefore, the mean of the CV of the columns may measure the task heterogeneity (noted μV_{task}) while the mean of the CV of the rows may measure the machine heterogeneity (noted μV_{mach}).

The first measure of task and machine heterogeneity has been criticized for small instances [2]. It is argued that the MPH is better than the CV as it is less sensitive to outliers. In this case, the CV can be replaced by the quartile coefficient of dispersion, which is a similar standard statistical measure but is more difficult to formally analyze. Finally, the decision trees in [5] suggest that varying this measure has an impact on the heuristic performance and is thus significant.

With both measures, it is possible to use the standard deviation instead of the CV. However, the CV provides a relative measure that is independent from the cost mean. If an absolute measure is deemed more meaningful, the proposed measures can be adapted by using the standard deviation.

4.3 Coherence with the Uniform Model

The previous measures do not only rely on intuition, they are also consistent with the expectation when we consider the uniform model. In this model, the cost of executing a task i on a machine j is given by the product of the task weight, w_i, and the cycle time, b_j. The concept of task

and machine heterogeneity is easy to grasp in the uniform model: it is given by the statistical dispersion of the weights and the speeds, respectively. We assume that the CV of the weights, noted CV_{task}, is a relevant measure of the task heterogeneity. Analogously, the CV of the speeds, noted CV_{mach}, represents the machine heterogeneity.

It is possible to convert an instance of the uniform model to the unrelated model because this last model is more general. The cost matrix is generated by combining both vectors $\{w_i\}_{1\leq i\leq n}$ and $\{b_j\}_{1\leq j\leq m}$ such that $e_{i,j} = w_i b_j$. As we know the heterogeneity properties of a uniform instance, we expect our proposed measures for the unrelated model to be consistent when applied on the converted instance.

Proposition 4. *Let $U = (\{w_i\}_{1\leq i\leq n}, \{b_j\}_{1\leq j\leq m})$ be a uniform instance and $E = \{e_{i,j}\}_{1\leq i\leq n, 1\leq j\leq m}$ be the corresponding unrelated instance such that $e_{i,j} = w_i b_j$. Then, $CV_{task}(U) = V\mu_{task}(E) = \mu V_{task}(E)$ and $CV_{mach}(U) = V\mu_{mach}(E) = \mu V_{mach}(E)$.*

Proof. By definition, $CV_{task}(U) = \frac{\sqrt{\sum_{i=1}^{n} w_i^2/n - (\sum_{i=1}^{n} w_i/n)^2}}{\sum_{i=1}^{n} w_i/n}$ whereas $V\mu_{task}(E)$ is the CV of the means of the rows. The mean of row i is $\sum_{j=1}^{m} e_{i,j}/m = w_i/m \sum_{j=1}^{m} b_j$. Then, $V\mu_{task}(E) = \frac{\sqrt{\sum_{i=1}^{n} (w_i\phi)^2/n - (\sum_{i=1}^{n} w_i\phi/n)^2}}{\sum_{i=1}^{n} w_i\phi/n}$ where $\phi = \sum_{j=1}^{m} b_j/m$ is the mean of the inverse speeds. Therefore, $V\mu_{task}(E) = CV_{task}(U)$.

Remember that $\mu V_{task}(E)$ is the mean of the CV of the columns. The CV of column j is

$$\frac{\sqrt{\sum_{i=1}^{n} e_{i,j}^2/n - (\sum_{i=1}^{n} e_{i,j}/n)^2}}{\sum_{i=1}^{n} e_{i,j}/n} = \frac{\sqrt{\sum_{i=1}^{n} (w_i b_j)^2/n - (\sum_{i=1}^{n} (w_i b_j)/n)^2}}{\sum_{i=1}^{n} (w_i b_j)/n} = CV_{task}(U)$$

The mean of these CV is thus also $CV_{task}(U)$.

The demonstration is analogous for the machine heterogeneity measures. □

Proposition 4 shows that our proposed measures are consistent with the intuition on uniform instances.

4.4 Heterogeneity of the Range-Based and CVB Methods

We analyse the asymptotic heterogeneity properties of the CVB method with the proposed measures depending on the parameters V_{task} and V_{mach}. An estimator T converges to θ when the expected value of T tends to θ as the number of samples (n and m in our case) tends to ∞.

Proposition 5. *The measure $V\mu_{task}$ of a cost matrix generated using the CVB method with the parameters V_{task} and V_{mach} converges to V_{task} as $n \to \infty$ and $m \to \infty$.*

Proof. This proof assumes that the mean of a set of n samples (called the sample mean) of a random variable with mean μ and standard deviation σ is a random variable with mean μ and standard deviation $\frac{\sigma}{\sqrt{n}}$. Moreover, the CV of a set of n samples (called the sample CV) of a random variable with CV V converges to V as $n \to \infty$.

Let μ_i be the sample mean of the costs on row i. This row is the product of $q[i]$, which is a random variable that follows a distribution with mean μ_{task} and CV V_{task}, and m values that follow a distribution with mean one and CV V_{mach}. μ_i is thus also the product of the first random variable and the sample mean of the other m values, which follows a random variable with mean one and CV $\frac{V_{mach}}{\sqrt{m}}$. Therefore, the mean of μ_i is μ_{task} and its CV is $\sqrt{V_{task}^2 \frac{V_{mach}^2}{m} + \frac{V_{mach}^2}{m} + V_{task}^2}$, which tends to V_{task} as $m \to \infty$. The consistency properties have no impact on μ_i because only values on the same row are ordered. □

Proposition 6. *The measure $V\mu_{mach}$ of a cost matrix generated using the CVB method with the parameters V_{task} and V_{mach} converges to $a\sqrt{b}V_{mach}$ as $n \to \infty$ and $m \to \infty$.*

Proof. Let μ_j be the sample mean of the costs on column j. The measure $V\mu_{mach}$ is the ratio of the sample standard deviation of all μ_j, noted $\sigma\mu_{mach}$, to the sample mean of all μ_j, noted $\mu\mu_{mach}$.

Let's distinguish the columns where the costs are consistent ($1 \leq j \leq bm$) from the inconsistent columns ($bm < j \leq m$). For the inconsistent columns, μ_j is the sample mean of n values that follow a product between a distribution with mean μ_{task} and CV V_{task}, and a distribution with mean one and CV V_{mach}. Thus, μ_j follows a distribution with mean μ_{task} and CV $\sqrt{\frac{V_{task}^2 V_{mach}^2 + V_{task}^2 + V_{mach}^2}{n}}$ for $bm < j \leq m$. Therefore, the sample mean of μ_j converges to μ_{task} and its sample standard deviation converges to zero as $n \to \infty$ for $bm < j \leq m$.

For the consistent columns, an rows are sorted. Let q_p denotes the p-quantile of a distribution with mean one and CV V_{mach} (it is the value x for which $F(x) = p$ where F is the cumulative distribution function). Note that $e_{i,j} \to q[i]q_{j/(bm)}$ as $m \to \infty$ for $1 \leq i \leq an$ and $1 \leq j \leq bm$. Therefore, μ_j can be decomposed as a weighted sum of sample means (one for the sorted rows and another for the last rows): the first sample mean follows a distribution with mean $\mu_{task}q_{j/(bm)}$ and CV $\frac{V_{task}}{\sqrt{an}}$ while the second follows a distribution with mean μ_{task} and CV $\sqrt{\frac{V_{task}^2 V_{mach}^2 + V_{task}^2 + V_{mach}^2}{(1-a)n}}$. Therefore, the sample mean of μ_j converges to $a\mu_{task}q_{j/(bm)} + (1-a)\mu_{task}$ and its sample standard deviation converges to zero as $n \to \infty$ for $1 \leq j \leq bm$.

On one hand, $\mu\mu_{mach} = \frac{1}{m}\sum_{j=1}^{m}\mu_j = \frac{1}{m}(\sum_{j=1}^{bm}(a\mu_{task}q_{j/(bm)} + (1-a)\mu_{task}) + (1-b)m\mu_{task}) = ab\mu_{task}\frac{1}{bm}\sum_{j=1}^{bm}q_{j/(bm)} + (1-a)b\mu_{task} + (1-b)\mu_{task}$ as $n \to \infty$. Note that $\frac{1}{bm}\sum_{j=1}^{bm}q_{j/(bm)} = \int_0^1 q_p dp = 1$ as $m \to \infty$. Thus, $\mu\mu_{mach} = \mu_{task}$ as $n \to \infty$ and $m \to \infty$. On the other hand, as $n \to \infty$ and $m \to \infty$:

$$
\begin{aligned}
\sigma\mu_{mach} &= \sqrt{\frac{1}{m}\sum_{j=1}^{m}\mu_j^2 - \left(\frac{1}{m}\sum_{j=1}^{m}\mu_j\right)^2} \\
&= \sqrt{\frac{1}{m}\sum_{j=1}^{bm}\mu_j^2 + \frac{1}{m}\sum_{j=bm+1}^{m}\mu_j^2 - \mu_{task}^2} \\
&= \sqrt{\frac{1}{m}\sum_{j=1}^{bm}(a\mu_{task}q_{j/(bm)} + (1-a)\mu_{task})^2 + (1-b)\mu_{task}^2 - \mu_{task}^2} \\
&= \mu_{task}\sqrt{\frac{1}{m}\sum_{j=1}^{bm}(a^2q_{j/(bm)}^2 + 2aq_{j/(bm)}(1-a) + (1-a)^2) - b} \\
&= \mu_{task}\sqrt{a^2b\frac{1}{bm}\sum_{j=1}^{bm}q_{j/(bm)}^2 + 2a(1-a)b\frac{1}{bm}\sum_{j=1}^{bm}q_{j/(bm)} + (1-a)^2b - b} \\
&= a\sqrt{b}\mu_{task}\sqrt{\frac{1}{bm}\sum_{j=1}^{bm}q_{j/(bm)}^2 - 1}
\end{aligned}
$$

Note that $\frac{1}{bm}\sum_{j=1}^{bm}q_{j/(bm)}^2 = \int_0^1 q_p^2 dp = \int_{-\infty}^{\infty}x^2 f(x)dx = V_{mach}^2 + 1$ as $m \to \infty$ with the substitution $p = F(x)$ and $dp = f(x)dx$ where f is the probability density function of a distribution with mean one and CV V_{mach}. This requires the distribution to be continuous,

Measure	Value
$V\mu_{task}$	V_{task}
μV_{task}	$\begin{cases}\Phi = \sqrt{V_{task}^2 V_{mach}^2 + V_{task}^2 + V_{mach}^2} & \text{if } a = 0 \\ bV_{task} + (1-b)\Phi & \text{if } a = 1\end{cases}$
$V\mu_{mach}$	$a\sqrt{b}V_{mach}$
μV_{mach}	V_{mach}

Table 3: Summary of the heterogeneity properties of the CVB method.

which is the case for the gamma distribution. Therefore, $\sigma\mu_{mach} = a\sqrt{b}\mu_{task}V_{mach}$ and $V\mu_{mach} = a\sqrt{b}V_{mach}$ as $n \to \infty$ and $m \to \infty$. □

Proposition 7. *The measure μV_{task} of a cost matrix generated using the CVB method with the parameters V_{task} and V_{mach} converges to $\sqrt{V_{task}^2 V_{mach}^2 + V_{task}^2 + V_{mach}^2}$ as $n \to \infty$ if the matrix is inconsistent and to $bV_{task} + (1-b)\sqrt{V_{task}^2 V_{mach}^2 + V_{task}^2 + V_{mach}^2}$ as $n \to \infty$ and $m \to \infty$ if $a = 1$.*

Proof. Let V_j be the sample CV of column j. When $a = 0$, the values on column j follow a distribution that is the product of a distribution with mean μ_{task} and CV V_{task}, and a distribution with mean one and CV V_{mach}. Therefore, V_j converges to $\sqrt{V_{task}^2 V_{mach}^2 + V_{task}^2 + V_{mach}^2}$ as $n \to \infty$. Since this value does not depends on j, μV_{task} (the sample mean of these sample CV) also converges to $\sqrt{V_{task}^2 V_{mach}^2 + V_{task}^2 + V_{mach}^2}$ as $n \to \infty$.

When $a = 1$, V_j still converges to $\sqrt{V_{task}^2 V_{mach}^2 + V_{task}^2 + V_{mach}^2}$ as $n \to \infty$ for $bm < j \leq m$. However, μ_j (the sample mean of column j) converges to $\mu_{task}q_{j/(bm)}$ as $n \to \infty$ and $m \to \infty$ while σ_j (the sample standard deviation of column j) converges to $\mu_{task}V_{task}q_{j/(bm)}$ as $n \to \infty$ and $m \to \infty$ for $1 \leq j \leq bm$. Thus, V_j converges to V_{task} as $n \to \infty$ and $m \to \infty$ for $1 \leq j \leq bm$. Therefore, μV_{task} converges to $bCV_{task} + (1-b)\sqrt{V_{task}^2 V_{mach}^2 + V_{task}^2 + V_{mach}^2}$ as $n \to \infty$ and $m \to \infty$. □

Proposition 8. *The measure μV_{mach} of a cost matrix generated using the CVB method with the parameters V_{task} and V_{mach} converges to V_{mach} as $m \to \infty$.*

Proof. Let V_i be the sample CV of row i. The values on row i follow a distribution that is the product of $q[i]$ and a distribution with mean one and CV V_{mach}. Therefore, V_i converges to V_{mach} as $m \to \infty$. Since this value does not depend on i, μV_{mach} (the sample mean of these sample CV) also converges to V_{mach} as $m \to \infty$. □

Table 3 synthesises the previous formal results. They can be extended to the range-based method by replacing V_{task} by the CV of the first random variable ($\frac{\sqrt{12}}{6}\frac{R_{task}-1}{R_{task}+1}$) and V_{mach} by the CV of the second one ($\frac{\sqrt{12}}{6}\frac{R_{mach}-1}{R_{mach}+1}$). Indeed, the proofs only use the mean and the CV of the underlying distributions. Moreover, the uniform distribution is also continuous. Although the formal analysis of μV_{task} for arbitrary values of a was unsuccessful, the following formula provides a close estimate: $a^2bV_{task} + (1-a^2b)\sqrt{V_{task}^2 V_{mach}^2 + V_{task}^2 + V_{mach}^2}$.

In the case of complete consistency (i.e., when $a = b = 1$), $V\mu_{task} = \mu V_{task} = V_{task}$ and $V\mu_{mach} = \mu V_{mach} = V_{mach}$, which supports the proposed heterogeneity measures. This special

case is due to the fact that consistent cost matrices are closer to uniform instances than inconsistent ones and both measures are equivalent for uniform instances.

However, the CVB method has two issues. As a consequence of the asymmetry of the generation method, the task heterogeneity is not symmetric to the machine heterogeneity. For instance, we have $\mu V_{task} = \sqrt{V_{task}^2 V_{mach}^2 + V_{task}^2 + V_{mach}^2}$, whereas $V\mu_{mach} = V_{mach}$ for inconsistent matrices. This makes the generation method less direct as the parameters must be chosen such as to circumvent this asymmetry. In particular, if a high machine heterogeneity is required, then the task heterogeneity will also be high.

The second issue is related to the impact of the consistency parameters on the heterogeneity properties. It biases comparisons of scheduling methods when cost matrices are used with different consistency settings because these matrices will also have different heterogeneity properties. The range-based method presents an even stronger bias as both V_{task} and V_{mach} tend to $\frac{\sqrt{12}}{6}$ as $R_{task} \to \infty$ and $R_{mach} \to \infty$ (the heterogeneity properties are thus often similar).

4.5 Task and Machine Heterogeneity in Previous Studies

For each of the instances summarized in Section 3, we computed both heterogeneity measures using the formulas of Table 3 and the input parameters: R_{task} and R_{mach} for the range-based method; V_{task} and V_{mach} for the CVB method; and the consistency parameters, a and b, for both methods. For the case when $0 < a < 1$, μV_{task} was measured on a single 1000×1000 cost matrix that was generated with the range-based or the CVB method. When the consistency values are missing, matrices are assumed to be inconsistent. Finally, the mean is set to 1 when it is not given with the CVB method because it has no impact on any measure.

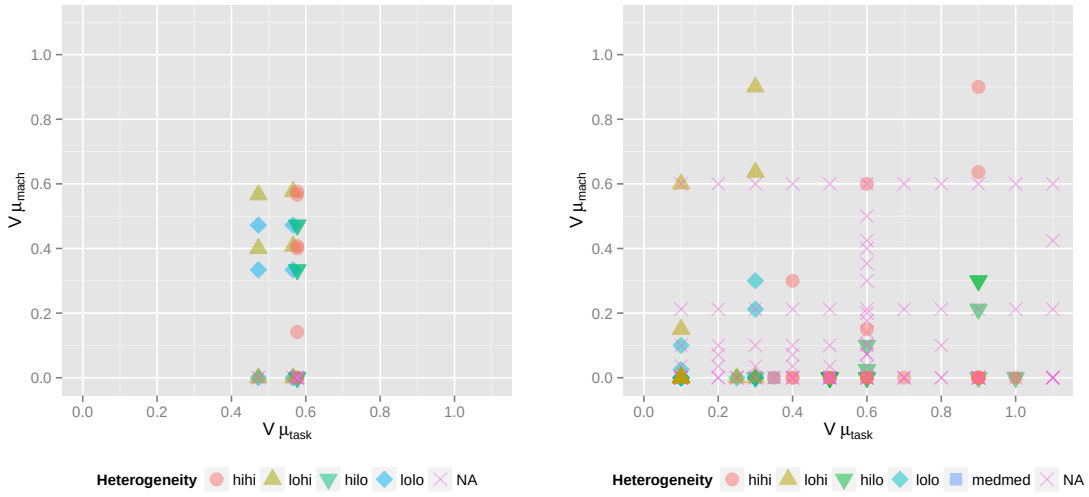
Figures 2 and 3 depict the values for the measures proposed above. The range-based method has a clear bias because many heterogeneity properties have never been obtained. Also, the consistency parameters invalidate the claimed properties of the cost matrices relatively to the heterogeneity quadrant for both heterogeneity measures: some *hihi* instances have the same machine heterogeneity as *lolo* instances on Figure 2, whereas some *lohi* instances have the same task heterogeneity as *hilo* instances on Figure 3.

This analysis is also consistent with the observation made in [3] about the fact that the range-based and CVB methods do not cover the entire range of possible values for the MPH.

As mentioned in Section 4, both proposed heterogeneity measures are relative. This allows a direct comparison between each heterogeneity value. Using the standard deviation instead would require normalizing them for this analysis.

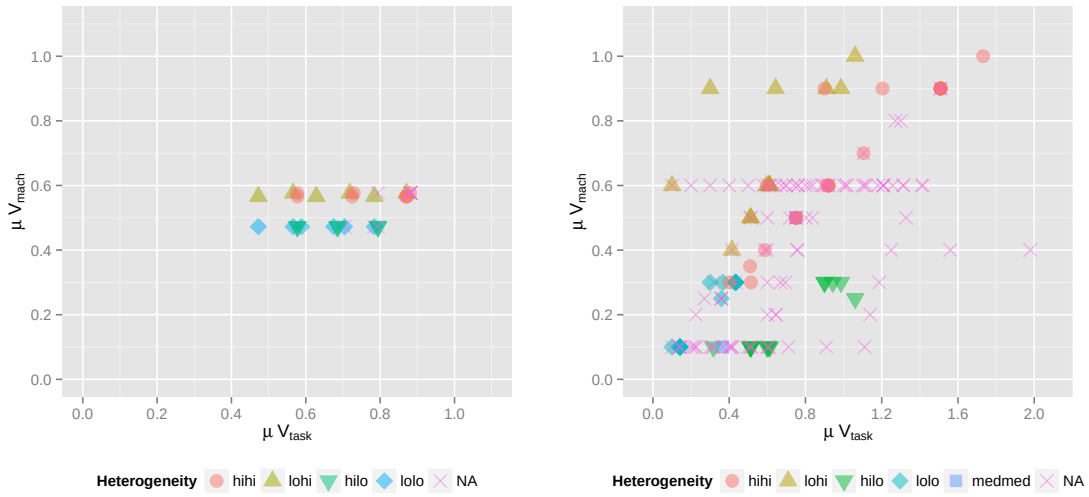
4.6 Non-asymptotic properties

The previous analysis only considers asymptotic heterogeneity properties (i.e., when $n \to \infty$ and $m \to \infty$). We study how far are these properties from the asymptotic ones by generating several small cost matrices for each setting used in the literature and by applying all the measures on them. Figure 4 depicts the heterogeneous properties of cost matrices with different sizes. We see that the column corresponding to 256 tasks with 64 machines is close to the asymptotic properties presented in Figures 2 and 3. Although the properties of smaller cost matrices are more dispersed, their central tendency remain similar to the asymptotic ones and the previous biases also occur with small matrices.



(a) Range-based method (b) CVB method

Figure 2: Heterogeneity properties ($V\mu_{task}$ and $V\mu_{mach}$) of cost matrices used in the literature. Two points are not shown for the CVB method: $(1.4, 0)$ and $(1.8, 0)$.



(a) Range-based method (b) CVB method

Figure 3: Heterogeneity properties (μV_{task} and μV_{mach}) of cost matrices used in the literature. The x-scale is twice as large as in Figure 2 for the CVB method because large values of V_{mach} tends to increase the measure μV_{task}. One point is not shown for the CVB method: $(2.01, 2)$.

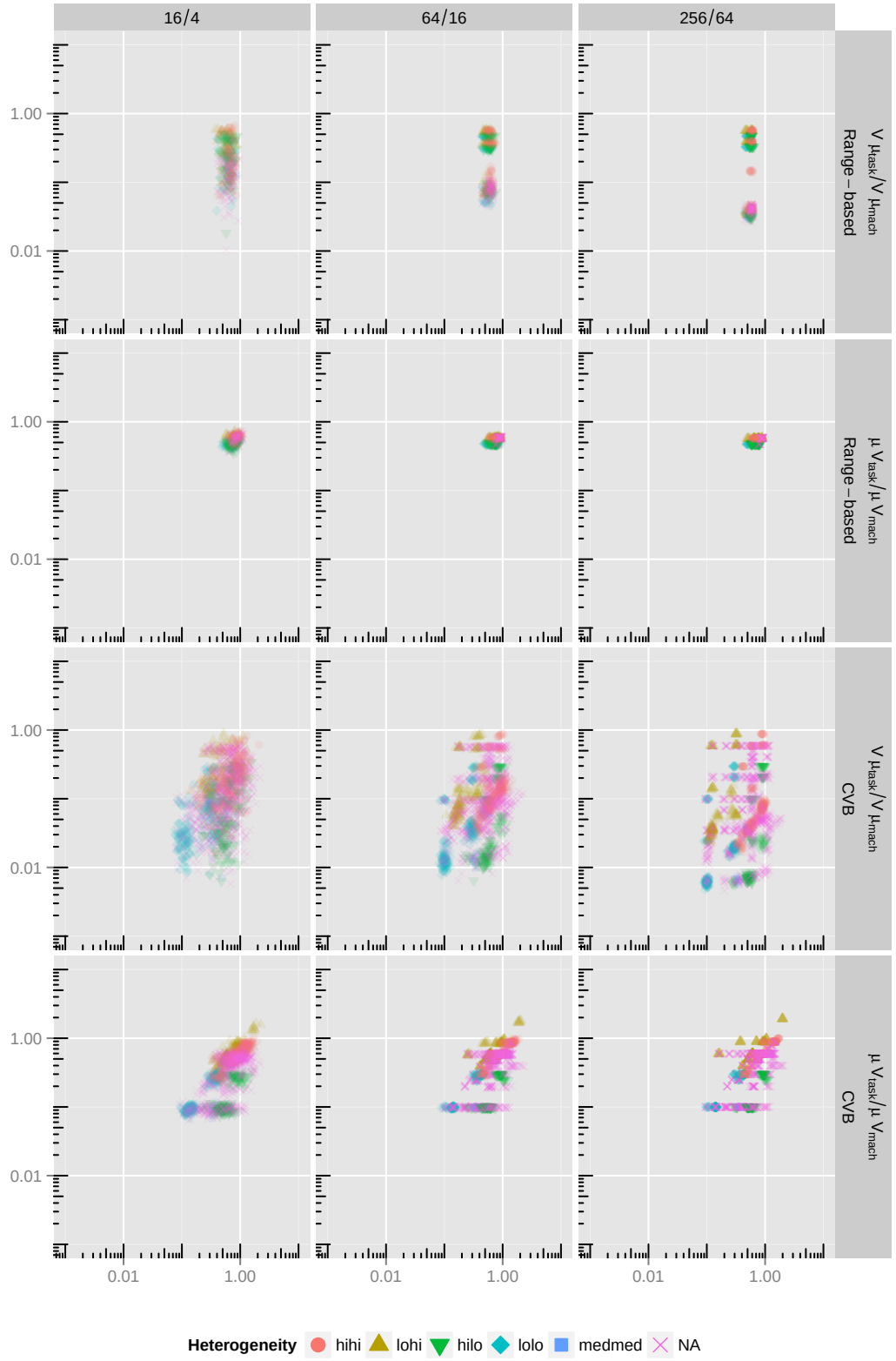


Figure 4: Heterogeneity properties (with respect to all measures) of cost matrices used in the literature. For each setting, 10 cost matrices are generated with different sizes: 16 tasks and 4 machines; 64 tasks and 16 machines; and, 256 tasks and 64 machines. The logarithmic scale is the same that is used in the figures of Section 6.

5 Controlling the Heterogeneity

We are interested in generating cost matrices that have specific heterogeneity properties according to the measures introduced in Section 4. We propose two methods that both alter a cost matrix generated from uniform instances for which we control the task and machine heterogeneities. These cost matrices have specific properties in terms of consistency and correlation between each row and each column, and the proposed methods introduce some randomness in it. They both possess the same time complexity (i.e., $O(nm)$).

5.1 Shuffling Method

The first proposed method shuffles the costs in the matrix that corresponds to a uniform instance (see Algorithm 4). It first generates the task weights on Line 2 and the inverse of the machine speeds on Line 5. The corresponding matrix is computed on Line 9 before starting the shuffling part. For each cost $e_{i,j}$, another cost $e_{i',j'}$ is selected on a different row and column (Lines 14 and 15). The same amount is then removed from these costs and is added to two other costs, $e_{i,j'}$ and $e_{i',j}$ (one that is on the same row as the first cost and on the same column as the second, and another one that is on the same row as the second cost and on the same column as the first). This step (Lines 25 to 28) preserves the mean of each row and the mean of each column. The heterogeneity properties thus remain the same.

The transferred amount is the largest value (in absolute) such that no cost among the four considered costs becomes lower than the minimum one among them (this prevents costs to be arbitrarily low). For instance, if $e_{i,j}$ is the minimum cost (i.e., $e_{i,j} = \min(e_{i,j}, e_{i',j}, e_{i,j'}, e_{i',j'})$), there are two cases: if $e_{i,j'} < e_{i',j}$, then $e_{i,j'}$ becomes the new minimum and the added value to $e_{i,j}$ and to $e_{i',j'}$ is $e_{i,j'} - e_{i,j}$; otherwise, it is $e_{i',j} - e_{i,j}$.

Maintaining both the minimum and the maximum cost is not possible because the cost matrix is generated from a uniform instance. This method focuses only on preventing costs to be arbitrarily low because it is critical to guarantee positive costs.

Proposition 9. *When used with parameters V_{task} and V_{mach}, the shuffling method generates costs with expected value* 1.

Proof. Costs in the matrix corresponding to the uniform matrix follow a distribution that is the product of two distributions with mean one. Therefore, the expected value of the costs in the matrix before the shuffling step is also one. The shuffling step does not change the expected value of the costs because the amount that is taken on any cost is given to another cost. □

Proposition 10. *The measure $V\mu_{task}$ of a cost matrix generated using the shuffling method with the parameters V_{task} and V_{mach} converges to V_{task} as $n \to \infty$.*

Proof. Analogously to the proof of Proposition 9, the shuffling step has no impact on the mean of each row and each column. The measure $V\mu_{task}$ is thus the same for the final cost matrix as for the intermediate matrix that corresponds to a uniform instance.

As a corollary of Proposition 4, the sample CV of the sample means of all rows in this intermediate matrix is equal to the sample CV of the vector $\{w_i\}_{1 \leq i \leq n}$. This last sample CV converges to V_{task} as $n \to \infty$. □

Proposition 11. *The measure $V\mu_{mach}$ of a cost matrix generated using the shuffling method with the parameters V_{task} and V_{mach} converges to V_{mach} as $m \to \infty$.*

Proof. Due to the symmetry of the shuffling method, the proof is analogous to the proof of Proposition 10. □

Table 4 summarizes the formal results related to the shuffling method.

Algorithm 4 Shuffling cost matrix generation with gamma distribution

Input: n, m, V_{task}, V_{mach}
Output: a $n \times m$ cost matrix

1: **for all** $1 \leq i \leq n$ **do**
2: $w_i \leftarrow G(1/V_{task}^2, V_{task}^2)$
3: **end for**
4: **for all** $1 \leq j \leq m$ **do**
5: $b_j \leftarrow G(1/V_{mach}^2, V_{mach}^2)$
6: **end for**
7: **for all** $1 \leq i \leq n$ **do**
8: **for all** $1 \leq j \leq m$ **do**
9: $e_{i,j} \leftarrow w_i b_j$
10: **end for**
11: **end for**
12: **for all** $1 \leq i \leq n$ **do**
13: **for all** $1 \leq j \leq m$ **do**
14: $i' \leftarrow (U(1, n-1) + i - 1 \mod n) + 1$
15: $j' \leftarrow (U(1, n-1) + j - 1 \mod m) + 1$
16: **if** $e_{i,j} = \min(e_{i,j}, e_{i',j}, e_{i,j'}, e_{i',j'})$ **then**
17: $d \leftarrow \min(e_{i',j} - e_{i,j}, e_{i,j'} - e_{i,j})$
18: **else if** $e_{i',j} = \min(e_{i',j}, e_{i,j'}, e_{i',j'})$ **then**
19: $d \leftarrow -\min(e_{i,j} - e_{i',j}, e_{i',j'} - e_{i',j})$
20: **else if** $e_{i,j'} = \min(e_{i,j'}, e_{i',j'})$ **then**
21: $d \leftarrow -\min(e_{i,j} - e_{i,j'}, e_{i',j'} - e_{i,j'})$
22: **else**
23: $d \leftarrow \min(e_{i',j} - e_{i',j'}, e_{i,j'} - e_{i',j'})$
24: **end if**
25: $e_{i,j} \leftarrow e_{i,j} + d$
26: $e_{i',j} \leftarrow e_{i',j} - d$
27: $e_{i,j'} \leftarrow e_{i,j'} - d$
28: $e_{i',j'} \leftarrow e_{i',j'} + d$
29: **end for**
30: **end for**
31: **return** $\{e_{i,j}\}_{1 \leq i \leq n, 1 \leq j \leq m}$

Property	Value
Expected value	1
$V\mu_{task}$	V_{task}
$V\mu_{mach}$	V_{mach}

Table 4: Summary of the cost matrix properties with the shuffling method.

5.2 Noise-Based Method

The second method, described in Algorithm 5, relies on a simple idea, which was also used in [6]: each cost of a matrix, which corresponds to a uniform instance, is multiplied by a matrix with a random variable with mean one (Line 9).

Algorithm 5 Noise-based cost matrix generation with gamma distribution

Input: n, m, V_{task}, V_{mach}, V_{noise}
Output: a $n \times m$ cost matrix

for all $1 \leq i \leq n$ **do**
  $w_i \leftarrow G(1/V_{task}^2, V_{task}^2)$
end for
for all $1 \leq j \leq m$ **do**
  $b_j \leftarrow G(1/V_{mach}^2, V_{mach}^2)$
end for
for all $1 \leq i \leq n$ **do**
  for all $1 \leq j \leq m$ **do**
    $e_{i,j} \leftarrow w_i b_j \times G(1/V_{noise}^2, V_{noise}^2)$
  end for
end for
return $\{e_{i,j}\}_{1 \leq i \leq n, 1 \leq j \leq m}$

Proposition 12. *When used with parameters V_{task}, V_{mach} and V_{noise}, the noise-based method generates costs with expected value one and CV $\sqrt{V_{task}^2 V_{mach}^2 V_{noise}^2 + V_{task}^2 V_{mach}^2 + V_{task}^2 V_{noise}^2 + V_{mach}^2 V_{noise}^2 + V_{task}^2 + V_{mach}^2 + V_{noise}^2}$.*

Proof. Each cost is the product of three random variables that have all the same mean one. Additionally, their CV (and standard deviations in this case) are V_{task}, V_{mach} and V_{noise}. The global CV can be derived by remarking that the CV of the product of two random variables with CV V_1 and V_2 is $\sqrt{V_1^2 V_2^2 + V_1^2 + V_2^2}$. □

Proposition 13. *The measure $V\mu_{task}$ of a cost matrix generated using the noise-based method with the parameters V_{task}, V_{mach} and V_{noise} converges to V_{task} as $n \to \infty$ and $m \to \infty$.*

Proof. Let μ_i be the sample mean of row i. This row is the product of w_i, which follows a distribution with mean one and CV V_{task}, and m values that are each the product of a random variable with mean one and CV V_{mach} and a random variable with mean one and CV V_{noise}. μ_i is thus also the product of w_i and the sample mean of the other m values, which follows a random variable with mean one and CV $\sqrt{\frac{V_{mach}^2 V_{noise}^2 + V_{mach}^2 + V_{noise}^2}{m}}$. Therefore, the mean of μ_i is one and its CV is $\sqrt{V_{task}^2 \frac{V_{mach}^2 V_{noise}^2 + V_{mach}^2 + V_{noise}^2}{m} + V_{task}^2 + \frac{V_{mach}^2 V_{noise}^2 + V_{mach}^2 + V_{noise}^2}{m}}$, which tends to V_{task} as $m \to \infty$. Therefore, the sample CV of all μ_i converges to V_{task} as $n \to \infty$ and $m \to \infty$. □

Proposition 14. *The measure $V\mu_{mach}$ of a cost matrix generated using the noise-based method with the parameters V_{task}, V_{mach} and V_{noise} converges to V_{mach} as $n \to \infty$ and $m \to \infty$.*

Proof. Due to the symmetry of the noise-based method, the proof is analogous to the proof of Proposition 13. □

Proposition 15. *The measure μV_{task} of a cost matrix generated using the noise-based method with the parameters V_{task}, V_{mach} and V_{noise} converges to $\sqrt{V_{task}^2 V_{noise}^2 + V_{task}^2 + V_{noise}^2}$ as $n \to \infty$.*

Proof. Let V_j be the sample CV of column j. Each column is the product of b_j and n values that are each the product of a random variable with mean one and CV V_{task} and a random variable with mean one and CV V_{noise}. Thus, V_j converges to the CV of this product

Property	Value
Expected value	1
CV	$\sqrt{V_{task}^2V_{mach}^2V_{noise}^2 + V_{task}^2V_{mach}^2 + V_{task}^2V_{noise}^2 + V_{mach}^2V_{noise}^2 + V_{task}^2 + V_{mach}^2 + V_{noise}^2}$
Distribution	Product of three gamma laws
$V\mu_{task}$	V_{task}
μV_{task}	$\sqrt{V_{task}^2V_{noise}^2 + V_{task}^2 + V_{noise}^2}$
$V\mu_{mach}$	V_{mach}
μV_{mach}	$\sqrt{V_{mach}^2V_{noise}^2 + V_{mach}^2 + V_{noise}^2}$

Table 5: Summary of the cost matrix properties with the noise-based method.

(i.e., $\sqrt{V_{task}^2V_{noise}^2 + V_{task}^2 + V_{noise}^2}$) as $n \to \infty$. Therefore, the measure μV_{task} converges to $\sqrt{V_{task}^2V_{noise}^2 + V_{task}^2 + V_{noise}^2}$ as $n \to \infty$. □

Proposition 16. *The measure μV_{mach} of a cost matrix generated using the noise-based method with the parameters V_{task}, V_{mach} and V_{noise} converges to $\sqrt{V_{mach}^2V_{noise}^2 + V_{mach}^2 + V_{noise}^2}$ as $m \to \infty$.*

Proof. Due to the symmetry of the noise-based method, the proof is analogous to the proof of Proposition 15. □

Table 5 summarizes the formal results related to the noise-based method.

This method requires one additional parameter: V_{noise}. When the objective is to have cost matrices with specific values of $V\mu_{task}$ and $V\mu_{mach}$ (for large n and m), we propose to set V_{noise} to $\min(V_{task}, V_{mach})$. This limits the amount of noise in the costs.

Contrary to the shuffling method, the noise-based method can also generate cost matrices with specific values of μV_{task} and μV_{mach} (asymptotically). The parameters can be fixed as follow: if $\mu V_{task} < \mu V_{mach}$, then $V_{task} = 0$, $V_{noise} = \mu V_{task}$ and $V_{mach} = \sqrt{(\mu V_{mach}^2 - \mu V_{task}^2)/(\mu V_{task}^2 + 1)}$; otherwise, $V_{mach} = 0$, $V_{noise} = \mu V_{mach}$ and $V_{task} = \sqrt{(\mu V_{task}^2 - \mu V_{mach}^2)/(\mu V_{mach}^2 + 1)}$. This setting maximizes the amount of noise.

Even though the shuffling method has less formal results (probably due to its combinatoric operations), the noise-based method has two drawbacks: the additional parameter is not trivial to determine and the method introduces more variation in the costs than the shuffling method. This makes this method more complex to use.

6 Impact on Scheduling Heuristics

This section assesses the impact of the heterogeneity properties defined in Section 4 on the relative performance of some classic heuristics.

6.1 Scheduling Heuristics

Our intention here is not to find the best heuristic but rather to show the impact of the cost matrix generation method on the performance results. We use classical heuristics from the

Name	Ref	Complexity	Remark	Algo
OLB	[26]	nm	Opportunistic Load Balancing	6
MET	[26]	nm	Minimum Execution Time	7
MCT	[26]	nm	Minimum Completion Time	8
Min-min	[26]	n^2m	Earliest finish time of smallest task	9
Max-min	[26]	n^2m	Earliest finish time of largest task	10
Suff	[38]	n^2m	Task that will suffer most first	11
GA	[26]	–	Genetic Algorithm	12
HLPT	[62]	$nm + n\log(n)$	Heterogeneous version of LPT	13
GreedySuff		$nm\log(m)$	Greedy allocation based on sufferage	14
BalSuff		–	Reconsider MET mapping	15
BalEFT		–	Reconsider MET mapping	16

Table 6: Summary of the scheduling heuristics for the $R||C_{\max}$ problem.

literature summarized in Table 6. Most of them (OLB, MET, MCT, Min-min, Max-min, HLPT[3], Suff) are list-based algorithms. The Genetic Algorithm (GA) relies on an initial population containing a solution obtained with Min-min. In addition to these classic heuristics, we added two more elaborated methods (the Bal prefixed methods) that try to reconsider an initial mapping obtained from MET (Minimum Execution Time) mapping: any task is moved to the machine that will finish it the earliest if it does not increase the maximum completion time. These heuristics are described in Appendix C.

Getting reference values (lower bounds on the makespan) for our performance measures is not straightforward in practice due to the heterogeneity of the problem. We thus rely on a variation of the genetic algorithm to provide an estimation of these values. The initial population is initialized, in addition to other random individuals, with all the solutions obtained by the other algorithms. The population evolution is based on the algorithm description given in [26]. An elite chromosome is maintained so that the resulting solution cannot be worse than any of the initial solutions and thus the genetic algorithm is no worse than any of the other algorithms.

6.2 Settings

Cost matrices are generated with three different methods: the shuffling method and the noise-based method with two approaches to set the noise (see Section 5.2). In all cases, there are two parameters: $V\mu_{task}$ and $V\mu_{proc}$ for the first two methods and μV_{task} and μV_{proc} for the last one. These two parameters are distributed in the range $[0.001, 10]$ with 30 equidistant values using a probit scale (i.e., 0.001, 0.0014, 0.0019, 0.0026, ..., 5.3, 7.3, 10).

For each pair of parameters, 200 cost matrices are generated with $n = 100$ tasks and $m = 30$ machines. For each scenario, we compute the makespan of each heuristic. We only consider the relative difference from the reference makespan: $|C - C_{\min}|/C_{\min}$ where C is the makespan of a given heuristic and $C_{\min}$ is the best makespan we obtained (the genetic algorithm initialized with all the solutions obtained by the other heuristics). The closer to zero, the better the performance.

6.3 Results

Figures 5 to 7 are heat maps of the relative performance for each algorithm. On each figure, we use a logarithmic scale on both axes: the x-axis gives the heterogeneity value for the tasks

[3]The variant HLPT (mean) is equivalent to HEFT [154].

($V\mu_{task}$ or μV_{task}) while the y-axis gives the heterogeneity value for the machines ($V\mu_{mach}$ or μV_{mach}). The bottom-left area represents almost homogeneous instances (same cost for each execution) while the top-right area is the most heterogeneous one. The heterogeneity values covered by the range-based and CVB methods in the literature are represented with dark rectangles on each sub-figure.

The scales on each heat map start at 0.001. We assume that an heterogeneity that is below this value may be considered negligible and that a heuristic that is closer to the reference makespan than this value is good enough. For instance, BalSuff may be considered near-optimal when the heterogeneity values are below 1%.

In Figure 7, both V_{task} and V_{mach} are zero on the diagonal, which may cause the irregularity for almost all heuristics. We suspect that using larger matrices would smooth this effect.

Figure 8 plots the best heuristic depending on the heterogeneity properties. Contour lines show the number of heuristics which performance is closer to the best heuristics than 0.001. For instance, there are at least 5 heuristics whose relative performances are almost equivalent when task heterogeneity is high (i.e., if the best heuristic average relative difference from the reference value is 0.004, then at least 5 other heuristics have a relative difference lower than 0.005).

The heuristics are ordered by the number of instances for which no other heuristic produces a better solution. When several heuristics are equivalent for a given tile, the appearing heuristic is the one that is the best the least often. This allows one to see even the settings for which the worst heuristics may be good.

Note that GA is close to Min-min (we observed that it returns the same solution in more than half of the cases) but improves it when the task heterogeneity is high. This proximity is explained by the inclusion of the solution provided by Min-min in the initial population of GA.

6.4 Analysis

The settings cover a large part of the possible instances for the $R||C_{\max}$ problem. Some areas on the figures may be associated to specific scheduling problems: the $Q|p_i = p|C_{\max}$ problem (top-left area), the $P|p_i = p|C_{\max}$ problem (bottom-left area) and the $P||C_{\max}$ problem (bottom-right area). While the first two problems can be solved in polynomial time, the last problem is NP-complete.

The heat maps suggest that the area where the heterogeneity values are between 0.1 and 1 is more challenging for most heuristics (areas in purple on the heat maps are 30% far from the reference). This is confirmed by Figure 8 where there is often a single best heuristic with these settings. Oppositely, many heuristics are close to the best one when the task heterogeneity is low or high, or when the machine heterogeneity is high. On one hand, execution costs are quite similar when the coefficient of variation is below 0.1. A non-optimal allocation will thus have a lower impact than with higher heterogeneity. On the other hand, most execution costs are close to zero when the coefficient of variation is higher than 1 and bad allocations may be easy to avoid because there are few allocations that are extremely critical while most of them are not. It is thus easier to generate a reasonable schedule.

When the machine heterogeneity is low (with medium task heterogeneity), there is often a single best heuristic. This suggests that these settings leads to difficult instances. As mentioned above, this is close to the $P||C_{\max}$ problem. We may conclude that dealing with heterogeneous tasks is more difficult than with heterogeneous machines, which is also supported by the asymmetry of the heat maps.

Finally, Figure 8 shows the best heuristics: BalSuff when both heterogeneity properties are comparable, BalEFT when the machine heterogeneity is higher than the task heterogeneity and HLPT when the task heterogeneity is high.

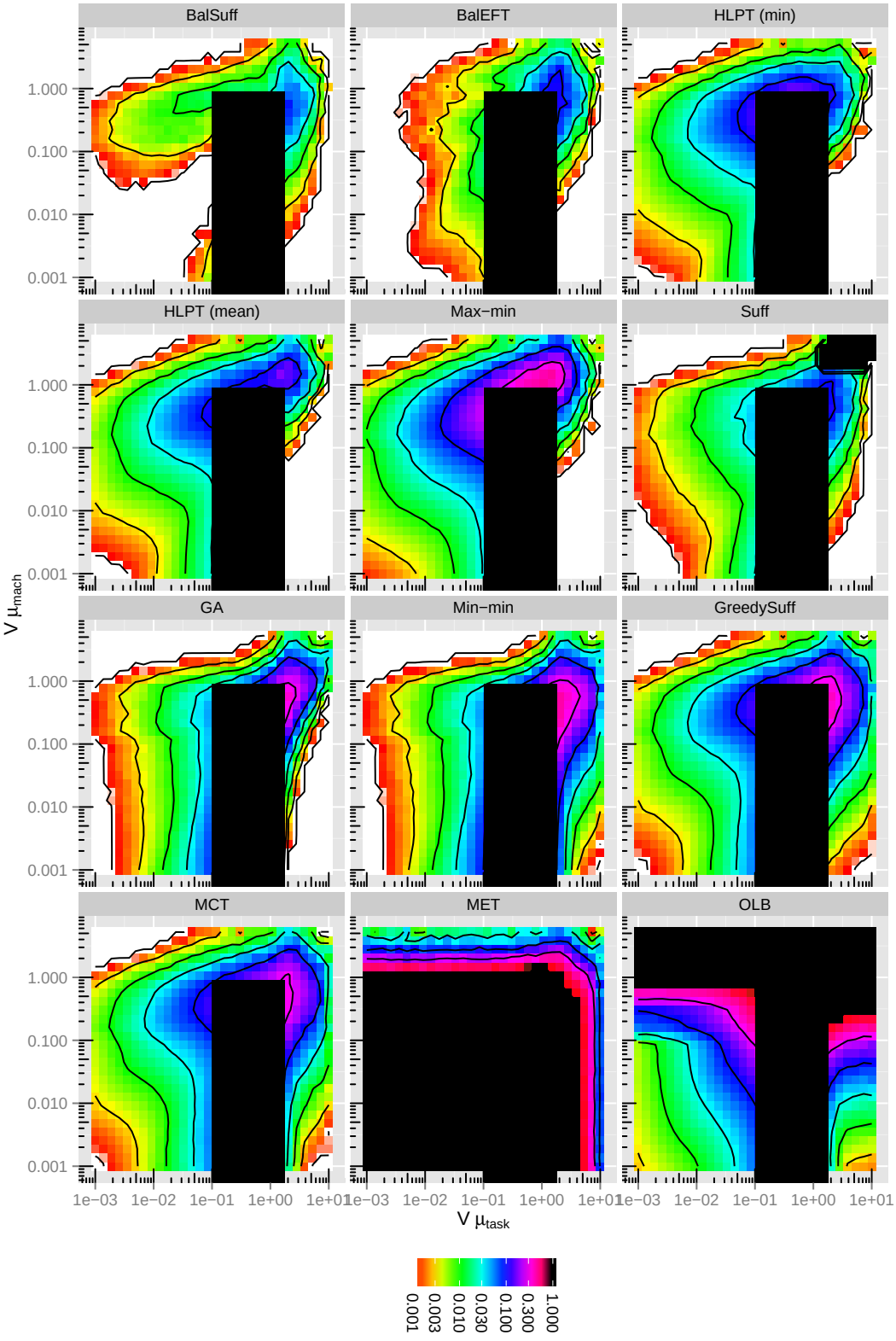


Figure 5: Heuristic performance relatively to the best case with the shuffling method. Values below 0.001 are shown in white and values above 1 are shown in black. Contour lines correspond to the levels in the legend (0.001, 0.003, ...). The dark rectangles correspond to the properties covered by the range-based and CVB methods in the literature (see Figure 2).

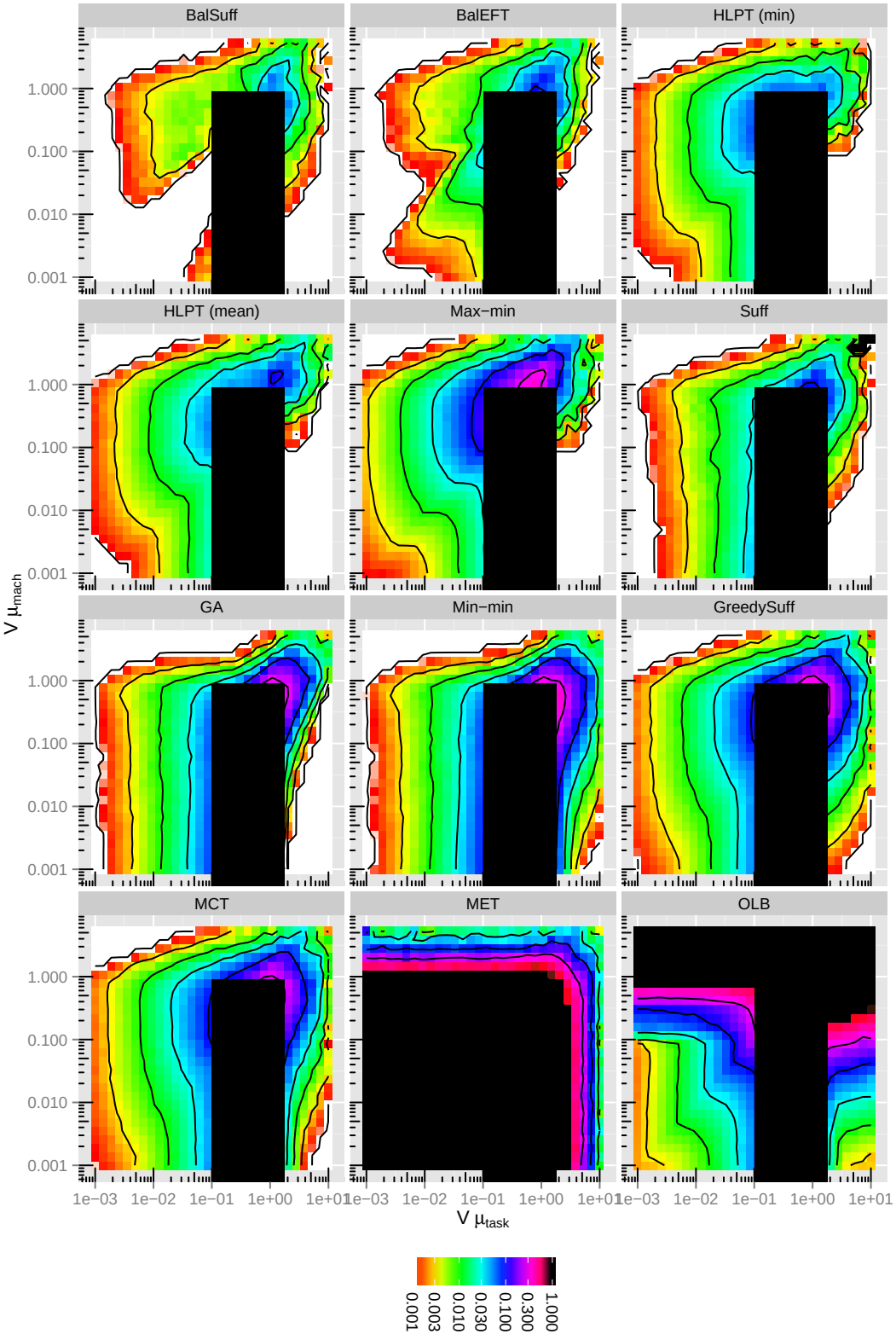


Figure 6: Heuristic performance relatively to the best case with the noise-based method with $V\mu_{task}$ and $V\mu_{mach}$ as parameters. Values below 0.001 are shown in white and values above 1 are shown in black. Contour lines correspond to the levels in the legend (0.001, 0.003, ...). The dark rectangles correspond to the properties covered by the range-based and CVB methods in the literature (see Figure 2).

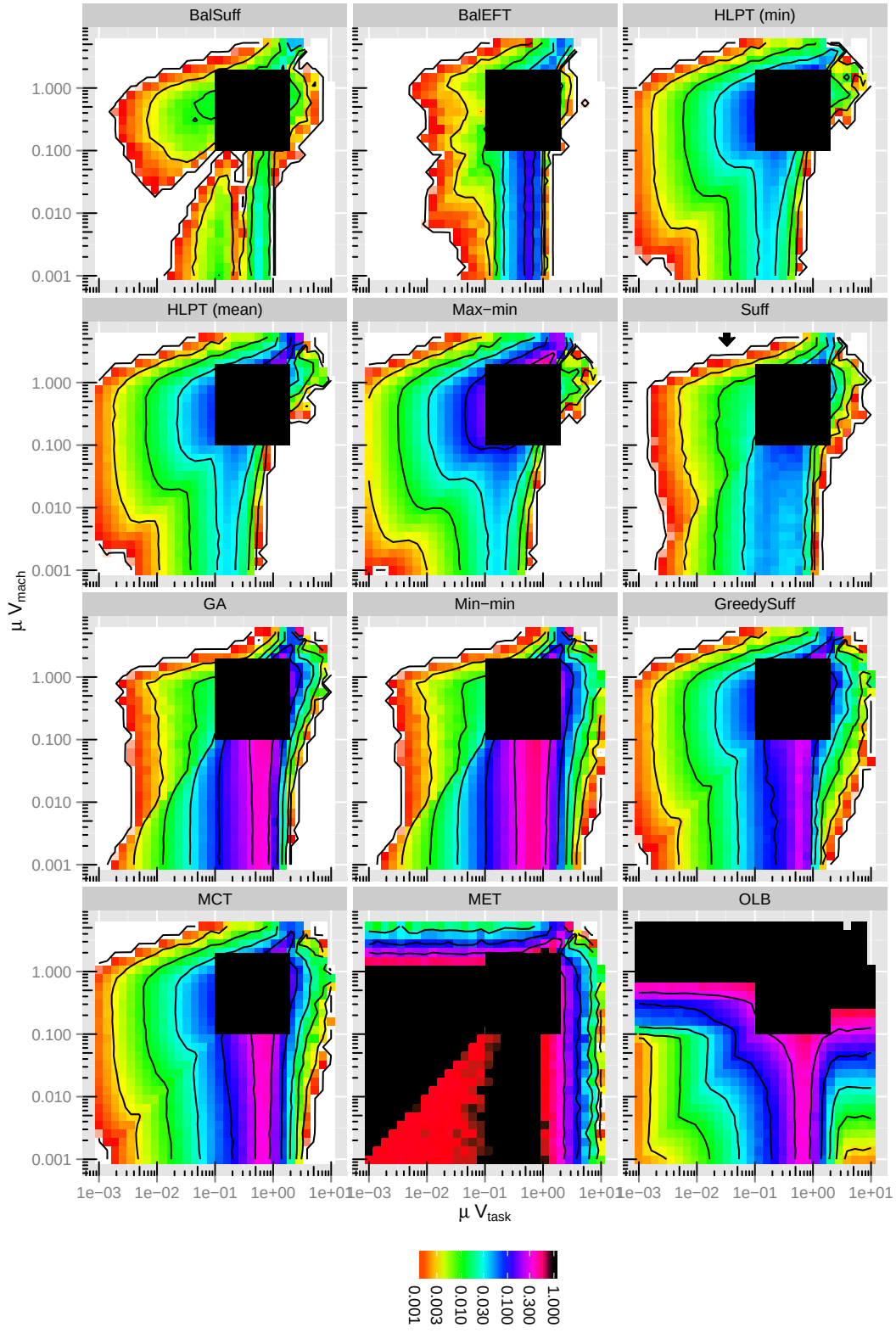


Figure 7: Heuristic performance relatively to the best case with the noise-based method with μV_{task} and μV_{mach} as parameters. Values below 0.001 are shown in white and values above 1 are shown in black. Contour lines correspond to the levels in the legend (0.001, 0.003, ...). The dark rectangles correspond to the properties covered by the range-based and CVB methods in the literature (see Figure 3).

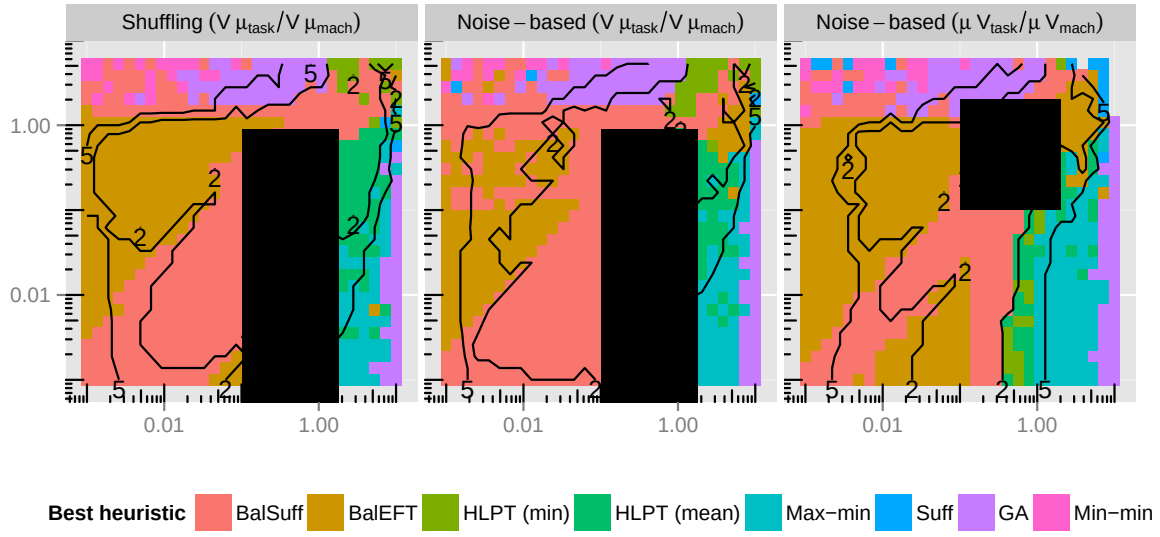


Figure 8: Best heuristic in the average case with the shuffling and the noise-based method with $V\mu_{task}$ and $V\mu_{mach}$ or μV_{task} and μV_{mach} as parameters. Contour lines correspond to the number of heuristics with a performance closer to the best heuristic performance than 0.001. The dark rectangles correspond to the properties covered by the range-based and CVB methods in the literature (see Figure 3).

Overall, we used two generation methods and two heterogeneity measures (one with the shuffling method and two with the noise-based method) and this analysis stands in all cases.

The range-based and CVB generation methods used in the literature could not provide these results due to two factors: the heterogeneity properties of the generated instances have a limited coverage (shown by the dark rectangles) and the erroneous claimed properties of these matrices prevent an unbiased analysis.

6.5 Discussion

This study focuses on the impact of some measures (either $V\mu_{task}$ and $V\mu_{mach}$, or μV_{task} and μV_{mach}) on the performance of several heuristics. However, there are many other properties that could be measured. If we consider the skewness and the kurtosis as in [5], we can think of 4×4 measures for the rows and as many for the columns. The main limitation of this study is to ignore the effect of all these possible measures. In addition, this study cannot be directly extended to assess all the possible interactions between them.

Another limitation is related to the effect of outliers. For large instances, the law of large number applies and the measures proposed in Section 4 correspond to the characteristics of the cost matrices. However, for small instances, we suggest switching to robust measures such as the median, the interquartile range and the quartile coefficient of dispersion instead of the mean, the standard deviation and the CV, respectively.

7 Conclusion

This study shows that the methods used in the literature for generating cost matrices are biased: the claimed heterogeneity properties of these instances are invalidated by the two measures we proposed to quantify them. We also show that the range of instances that has been used are restricted. It is specifically the case for the range-based method that covers only a minor fraction of all the possible settings in terms of heterogeneity. By providing new cost matrix generation methods, we show that heuristics for the $R||C_{\max}$ problem have interesting behavior outside this restriction. For instance, BalEFT is the best heuristic when the task heterogeneity is low and this could not have been shown with the instances used in the literature.

In addition to all the possible measures mentioned in Section 6.5, we plan to analyse other properties, in particular the correlation. It would also be interesting to see if the conclusions hold for some variations of the $R||C_{\max}$ problem such as considering arrival times or online scheduling.

Acknowledgments & Contributions

The authors would like to thank Pierre-Cyrille Héam for his helpful comments on the proof of Proposition 3.

Computations have been performed on the supercomputer facilities of the Mésocentre de calcul de Franche-Comté.

Analyzed previous generation methods: LCC. Designed generation methods: LCC. Implemented heuristics: LP. Performed the experiments: LP. Analyzed the data: LCC LP. Wrote the paper: LCC LP.

A Notation

Symbol	Definition
i	index of the tasks
j	index of the machines
n	number of tasks
m	number of machines
$e_{i,j}$	execution cost of task i on machine j
w_i	weight of task i
b_j	inverse speed of machine j
$U(A,B)$	uniform distribution between A and B
$G(\alpha,\beta)$	gamma distribution with shape α and scale β
R_{task}	parameter for the range-based method
R_{mach}	parameter for the range-based method
V_{task}	parameter for the CVB, shuffling and noise-based methods
V_{mach}	parameter for the CVB, shuffling and noise-based methods
V_{noise}	parameter for the noise-based method
a	fraction of the consistent rows
b	fraction of the consistent columns
$V\mu_{task}$	first measure of task heterogeneity
$V\mu_{mach}$	first measure of machine heterogeneity
μV_{task}	second measure of task heterogeneity
μV_{mach}	second measure of machine heterogeneity

Table 7: List of notations

B Usage of the Range-Based and CVB Methods in the Literature

B.1 Range-Based Method

The following table summarizes the studies that used the range-based method for generating cost matrices. Each row correspond to a cost matrix instance generated with the range-based method. I (resp. T) denotes matrices that are used for Illustration (resp. Testing algorithm performance). The range and consistency columns are the input parameters. The last columns represent the expected heterogeneity properties with two possible levels: low and high.

Context		Range		Consistency		Heterogeneity	
Ref.	Use	Task	Mach.	Task	Mach.	Task	Mach.
[8,17,18]	I	[1;10]	[1;10]	0	0	low	low
[170]	T	[1;10]	[1;10]	0	0	low	low
[36,171,172]	T	[1;10]	[1;10]	NA	NA	NA	NA
[170]	T	[1;10]	[1;10]	1	.5	low	low
[170]	T	[1;10]	[1;10]	1	1	low	low
[8,17,18]	I	[1;10]	[1;100]	0	0	low	high

Table 8 – *Continued from previous page*

Context		Range		Consistency		Heterogeneity	
Ref.	Use	Task	Mach.	Task	Mach.	Task	Mach.
[170]	T	[1;10]	[1;100]	0	0	low	high
[170]	T	[1;10]	[1;100]	1	.5	low	high
[170]	T	[1;10]	[1;100]	1	1	low	high
[156, 157]	T	[1;10]	*[1;1e5]*	NA	NA	NA	NA
[25, 153]	I	[1;100]	[1;10]	0	0	low	low
[25, 26]	T	[1;100]	[1;10]	0	0	low	low
[23]	T	[1;100]	[1;10]	0	0	NA	NA
[25, 26]	T	[1;100]	[1;10]	1	.5	low	low
[25, 26]	T	[1;100]	[1;10]	1	1	low	low
[25]	I	[1;100]	[1;1e3]	0	0	low	high
[25, 26, 153]	T	[1;100]	[1;1e3]	0	0	low	high
[25, 26, 153]	T	[1;100]	[1;1e3]	1	.5	low	high
[25, 26, 153]	T	[1;100]	[1;1e3]	1	1	low	high
[156, 157]	T	[1;1e3]	*[1;1e5]*	NA	NA	NA	NA
[25]	I	[1;3e3]	[1;10]	0	0	high	low
[8, 25, 26, 147]	T	[1;3e3]	[1;10]	0	0	high	low
[25, 26]	T	[1;3e3]	[1;10]	1	.5	high	low
[25, 26]	T	[1;3e3]	[1;10]	1	1	high	low
[8, 110, 111, 147]	T	[1;3e3]	[1;100]	0	0	high	high
[8]	T	[1;3e3]	[1;100]	*.5*	*.25*	high	high
[110, 111]	T	[1;3e3]	[1;100]	.5	.25	high	high
[25, 26, 153]	I	[1;3e3]	[1;1e3]	0	0	high	high
[25, 26]	T	[1;3e3]	[1;1e3]	0	0	high	high
[26]	I	[1;3e3]	[1;1e3]	1	.5	high	high
[25, 26]	T	[1;3e3]	[1;1e3]	1	.5	high	high
[26]	I	[1;3e3]	[1;1e3]	1	1	high	high
[25, 26]	T	[1;3e3]	[1;1e3]	1	1	high	high
[8, 17, 18]	I	[1;1e5]	[1;10]	0	0	high	low
[170]	T	[1;1e5]	[1;10]	0	0	high	low
[170]	T	[1;1e5]	[1;10]	1	.5	high	low
[170]	T	[1;1e5]	[1;10]	1	1	high	low
[156, 157]	T	*[1;1e5]*	[1;10]	NA	NA	NA	NA
[8, 17, 18]	I	[1;1e5]	[1;100]	0	0	high	high
[170]	T	[1;1e5]	[1;100]	0	0	high	high
[170]	T	[1;1e5]	[1;100]	1	.5	high	high
[170]	T	[1;1e5]	[1;100]	1	1	high	high
[156, 157]	T	*[1;1e5]*	[1;1e3]	NA	NA	NA	NA
[156, 157]	T	[1;1e5]	[1;1e5]	NA	NA	NA	NA
[156, 157]	T	*[1;1e5]*	[1;1e7]	NA	NA	NA	NA
[156, 157]	T	*[1;1e5]*	[1;1e9]	NA	NA	NA	NA
[156, 157]	T	[1;1e7]	*[1;1e5]*	NA	NA	NA	NA
[156, 157]	T	[1;1e9]	*[1;1e5]*	NA	NA	NA	NA

B.2 CVB Method

This second table is for the CVB method with the gamma distribution. There is one more parameter, the mean, and the expected heterogeneity has a new level: *med*, for medium.

Context		CV		Global	Consistency		Heterogeneity	
Ref.	Use	Task	Mach.	Mean	Task	Mach.	Task	Mach.
[8, 17, 18]	I	.1	.1	*1e3*	0	0	low	low
[41, 42, 116]	T	.1	.1	100	0	0	low	low
[41, 42]	T	.1	.1	100	*.5*	*.25*	low	low
[41, 42]	T	.1	.1	100	1	1	low	low
[107]	T	.1	.1	10	NA	NA	low	low
[58]	T	.1	.1	50	NA	NA	low	low
[68, 164]	T	.1	.1	100	NA	NA	low	low
[57]	T	.1	.1	1e3	NA	NA	low	low
[3]	T	.1	.1	NA	NA	NA	low	low
[109, 115]	T	.1	.1	1e3	0	0	NA	NA
[109]	T	.1	.1	1e3	.5	.5	NA	NA
[109]	T	.1	.1	1e3	1	1	NA	NA
[78, 80]	T	.1	.1	2e12	NA	NA	NA	NA
[115]	T	.1	.2	1e3	0	0	NA	NA
[52]	T	*.1*	*.25*	10	NA	NA	NA	NA
[115]	T	.1	.3	1e3	0	0	NA	NA
[31]	T	.1	.4	7.5	NA	NA	low	high
[115]	T	.1	.4	1e3	0	0	NA	NA
[58]	T	.1	.5	30	NA	NA	low	high
[57]	T	.1	.5	1e3	NA	NA	low	high
[115]	T	.1	.5	1e3	0	0	NA	NA
[78, 80]	T	.1	.5	2e12	NA	NA	NA	NA
[41, 42, 116]	T	.1	.6	100	0	0	low	high
[8, 17, 18]	I	.1	.6	*1e3*	0	0	low	high
[41, 42]	T	.1	.6	100	*.5*	*.25*	low	high
[41, 42]	T	.1	.6	100	1	1	low	high
[68, 164]	T	.1	.6	100	NA	NA	low	high
[109, 115]	T	.1	.6	1e3	0	0	NA	NA
[109]	T	.1	.6	1e3	.5	.5	NA	NA
[109]	T	.1	.6	1e3	1	1	NA	NA
[3]	T	.1	.9	NA	NA	NA	low	high
[8, 17, 18]	I	.1	2	*1e3*	0	0	low	high
[43, 109]	T	.2	.1	1e3	0	0	NA	NA
[109]	T	.2	.1	1e3	.5	.5	NA	NA
[43]	T	.2	.1	1e3	*1*	*.5*	NA	NA
[109]	T	.2	.1	1e3	1	1	NA	NA
[109]	T	.2	.6	1e3	0	0	NA	NA
[109]	T	.2	.6	1e3	.5	.5	NA	NA
[109]	T	.2	.6	1e3	1	1	NA	NA
[49, 50]	T	.25	.25	100	NA	NA	low	low
[167]	T	.25	.25	750	0	0	NA	NA
[168]	T	.25	.25	NA	0	0	NA	NA

Table 9 – *Continued from previous page*

Context		CV		Global	Consistency		Heterogeneity	
Ref.	Use	Task	Mach.	Mean	Task	Mach.	Task	Mach.
[166]	T	.25	.25	750	NA	NA	NA	NA
[49]	T	.25	1	100	NA	NA	low	high
[8, 17, 18]	I	.3	.1	*1e3*	0	0	high	low
[109, 115]	T	.3	.1	1e3	0	0	NA	NA
[109]	T	.3	.1	1e3	.5	.5	NA	NA
[109]	T	.3	.1	1e3	1	1	NA	NA
[150, 151]	T	.3	.3	30	0	0	low	low
[113, 114]	T	.3	.3	100	0	0	low	low
[112, 114]	T	.3	.3	120	0	0	low	low
[71]	T	.3	.3	1e3	0	0	low	low
[71]	T	.3	.3	1e3	1	.5	low	low
[71]	T	.3	.3	1e3	1	1	low	low
[84, 85]	T	.3	.3	3	NA	NA	low	low
[12]	T	.3	.3	NA	NA	NA	NA	NA
[109, 115]	T	.3	.6	1e3	0	0	NA	NA
[109]	T	.3	.6	1e3	.5	.5	NA	NA
[109]	T	.3	.6	1e3	1	1	NA	NA
[71]	T	.3	.9	1e3	0	0	low	high
[71]	T	.3	.9	1e3	1	.5	low	high
[71]	T	.3	.9	1e3	1	1	low	high
[108]	T	.35	.1	10	NA	NA	*med*	*med*
[107]	T	.35	.1	10	NA	NA	NA	NA
[107, 108]	T	.35	.35	10	NA	NA	high	high
[43, 109]	T	.4	.1	1e3	0	0	NA	NA
[109]	T	.4	.1	1e3	.5	.5	NA	NA
[43]	T	.4	.1	1e3	*1*	*.5*	NA	NA
[109]	T	.4	.1	1e3	1	1	NA	NA
[32]	T	.4	.3	1	1	1	high	high
[33]	T	.4	.3	1	NA	NA	high	high
[10]	T	.4	.4	12	NA	NA	high	high
[159]	T	.4	.4	20	*0*	*0*	NA	NA
[109]	T	.4	.6	1e3	0	0	NA	NA
[109]	T	.4	.6	1e3	.5	.5	NA	NA
[109]	T	.4	.6	1e3	1	1	NA	NA
[29]	I	.5	.1	100	0	0	high	low
[8, 17, 18]	I	.5	.1	*1e3*	0	0	high	low
[58]	T	.5	.1	50	NA	NA	high	low
[57]	T	.5	.1	1e3	NA	NA	high	low
[109, 115]	T	.5	.1	1e3	0	0	NA	NA
[109]	T	.5	.1	1e3	.5	.5	NA	NA
[109]	T	.5	.1	1e3	1	1	NA	NA
[142, 143]	T	.5	.5	20	NA	NA	med	med
[3]	T	.5	.5	NA	NA	NA	med	med
[77]	T	.5	.5	10	NA	NA	high	high
[58]	T	.5	.5	50	NA	NA	high	high

Table 9 – *Continued from previous page*

Context		CV		Global	Consistency		Heterogeneity	
Ref.	Use	Task	Mach.	Mean	Task	Mach.	Task	Mach.
[57]	T	.5	.5	1e3	NA	NA	high	high
[35]	T	.5	.5	20	NA	NA	NA	NA
[61, 67]	T	.5	.5	100	NA	NA	NA	NA
[169]	T	.5	.5	200	NA	NA	NA	NA
[109, 115]	T	.5	.6	1e3	0	0	NA	NA
[109]	T	.5	.6	1e3	.5	.5	NA	NA
[109]	T	.5	.6	1e3	1	1	NA	NA
[41, 42, 116]	T	.6	.1	100	0	0	high	low
[41, 42]	T	.6	.1	100	*.5*	*.25*	high	low
[41, 42]	T	.6	.1	100	1	1	high	low
[68, 164]	T	.6	.1	100	NA	NA	high	low
[109, 115]	T	.6	.1	1e3	0	0	NA	NA
[109]	T	.6	.1	1e3	.5	.5	NA	NA
[109]	T	.6	.1	1e3	1	1	NA	NA
[109, 115]	T	.6	.2	1e3	0	0	NA	NA
[109]	T	.6	.2	1e3	.5	.5	NA	NA
[109]	T	.6	.2	1e3	1	1	NA	NA
[109, 115]	T	.6	.3	1e3	0	0	NA	NA
[109]	T	.6	.3	1e3	.5	.5	NA	NA
[109]	T	.6	.3	1e3	1	1	NA	NA
[109, 115]	T	.6	.4	1e3	0	0	NA	NA
[109]	T	.6	.4	1e3	.5	.5	NA	NA
[109]	T	.6	.4	1e3	1	1	NA	NA
[43, 109, 115]	T	.6	.5	1e3	0	0	NA	NA
[109]	T	.6	.5	1e3	.5	.5	NA	NA
[43]	T	.6	.5	1e3	*1*	*.5*	NA	NA
[109]	T	.6	.5	1e3	1	1	NA	NA
[41, 42, 116]	T	.6	.6	100	0	0	high	high
[8, 17, 18]	I	.6	.6	*1e3*	0	0	high	high
[41, 42]	T	.6	.6	100	*.5*	*.25*	high	high
[41, 42]	T	.6	.6	100	1	1	high	high
[68, 164]	T	.6	.6	100	NA	NA	high	high
[109, 115]	T	.6	.6	1e3	0	0	NA	NA
[28]	T	.6	.6	1e3	.25	.25	NA	NA
[109]	T	.6	.6	1e3	.5	.5	NA	NA
[43]	T	.6	.6	1e3	*1*	*.5*	NA	NA
[43, 109]	T	.6	.6	1e3	1	1	NA	NA
[115]	T	.7	.1	1e3	0	0	NA	NA
[109, 115]	T	.7	.6	1e3	0	0	NA	NA
[109]	T	.7	.6	1e3	.5	.5	NA	NA
[109]	T	.7	.6	1e3	1	1	NA	NA
[9]	T	.7	.7	10	NA	NA	high	high
[13, 14, 16, 51]	T	.7	.7	10	NA	NA	NA	NA
[109]	T	.8	.6	1e3	0	0	NA	NA
[109]	T	.8	.6	1e3	.5	.5	NA	NA

Table 9 – *Continued from previous page*

Context		CV		Global	Consistency		Heterogeneity	
Ref.	Use	Task	Mach.	Mean	Task	Mach.	Task	Mach.
[109]	T	.8	.6	1e3	1	1	NA	NA
[125]	T	.8	.8	15	.25	.25	NA	NA
[12]	T	.8	.8	NA	NA	NA	NA	NA
[115]	T	.9	.1	1e3	0	0	NA	NA
[71]	T	.9	.3	1e3	0	0	high	low
[71]	T	.9	.3	1e3	1	.5	high	low
[71]	T	.9	.3	1e3	1	1	high	low
[151, 152]	T	.9	.3	180	1	1	high	low
[43, 109, 115]	T	.9	.6	1e3	0	0	NA	NA
[109]	T	.9	.6	1e3	.5	.5	NA	NA
[109]	T	.9	.6	1e3	1	1	NA	NA
[87]	T	.9	.6	60	NA	NA	NA	NA
[87]	T	.9	.6	600	NA	NA	NA	NA
[86]	T	.9	.6	NA	NA	NA	NA	NA
[150, 151]	T	.9	.9	30	0	0	high	high
[113, 114]	T	.9	.9	100	0	0	high	high
[112, 114]	T	.9	.9	120	0	0	high	high
[71]	T	.9	.9	1e3	0	0	high	high
[71]	T	.9	.9	1e3	1	.5	high	high
[71]	T	.9	.9	1e3	1	1	high	high
[84, 85]	T	.9	.9	3	NA	NA	high	high
[144–146]	T	.9	.9	100	NA	NA	high	high
[3]	T	.9	.9	NA	NA	NA	high	high
[73, 74]	T	.9	.9	20	0	0	NA	NA
[72]	T	.9	.9	30	0	0	NA	NA
[49]	T	1	.25	100	NA	NA	high	low
[109]	T	1	.6	1e3	0	0	NA	NA
[109]	T	1	.6	1e3	.5	.5	NA	NA
[109]	T	1	.6	1e3	1	1	NA	NA
[44, 48–50]	T	1	1	100	NA	NA	high	high
[115]	T	1.1	.1	1e3	0	0	NA	NA
[115]	T	1.1	.2	1e3	0	0	NA	NA
[115]	T	1.1	.3	1e3	0	0	NA	NA
[115]	T	1.1	.4	1e3	0	0	NA	NA
[115]	T	1.1	.5	1e3	0	0	NA	NA
[43, 109, 115]	T	1.1	.6	1e3	0	0	NA	NA
[109]	T	1.1	.6	1e3	.5	.5	NA	NA
[43]	T	1.1	.6	1e3	*1*	*.5*	NA	NA
[43, 109]	T	1.1	.6	1e3	1	1	NA	NA
[11]	T	1.4	.4	NA	0	0	NA	NA
[11]	T	1.8	.4	NA	0	0	NA	NA

B.3 Missing Parameters of Named Instances

The following articles mention some specific instances without providing the exact parameters used to generate them (they may thus be the same as the referenced articles):

- missing values, probably the same as [18]: [15, 27, 45–47, 79, 83, 88, 128, 130–132, 140, 141]
- missing values, probably the same as [17]: [34, 66, 69, 70, 76, 158]
- missing values, probably the same as [26]: [1, 22, 24, 65, 69, 70, 97, 101, 103, 117, 121, 127, 129, 133, 160]
- missing values, probably the same as either [18] or [26]: [21]
- missing values, probably the same as either [17] or [26]: [7, 118, 120, 122–124]

B.4 Discarded Articles

The following references were discarded either because they lack information on the instance generation or because they do not fit our study:

- missing values and unclear generation method: [19, 39, 40, 56, 60, 64, 75, 81, 89–96, 98–100, 102, 104, 126, 135–138, 155, 161, 163, 165, 173]
- no value for the shape parameter and range for the scale parameter: [148, 149]
- only one CV is given: [54, 55, 134]
- mismatched citation: [162]
- mismatched value for high machine heterogeneity: [119]
- use bi or tri-modal distributions with infinite mean: [30]
- range for CV: [4, 5]
- ranges for mean and CV: [106]

C Scheduling Heuristics

In this part, we describe the scheduling algorithms used for the experiments. Part of them are classical algorithms as OLB or Min-Min taken for other research works. We also introduce less classical algorithms as HLPT, an adaptation of LPT to the heterogeneous context, algorithms based on sufferage values for which a sufferage matrix is computed and balance algorithms that try to improve an initial solution by moving the tasks that will not lose too much from changing their mapping.

C.1 Related Work

The OLB algorithm (Opportunistic Load Balancing) takes a list of tasks sorted in random order and allocates them to the first ready machine. This algorithm generates mostly random schedules as they depend of the task list order. On the one hand, it tries to keep the machines busy as it allocates a task to a machine as soon as the former is free. On the other hand, it does not take the $e_{i,j}$ value into account. Thus, it may map a task to its worst machine.

Algorithm 6 Opportunistic Load Balancing (OLB)

Input: T a set of n tasks in an arbitrary order,
M a set of m machines,
E a cost matrix
Output: an allocation function π
1: Init $\forall p \in M, RT[p] \leftarrow 0$ {earliest ready time for each machine}
2: **for all** $t \in T$ **do**
3: $\pi_t \leftarrow p \; s.t. \; RT[p] = \min_k(RT[k])$ {first ready machine}
4: $RT[\pi_t] \leftarrow RT[\pi_t] + e_{t,\pi_t}$
5: **end for**
6: **return** π

Algorithm 7 Minimum Execution Time (MET)

Input: T a set of n tasks in an arbitrary order,
M a set of m machines,
E a matrix of the execution costs of the tasks on the machines
Output: an allocation function π {earliest ready time for each machine}
1: **for all** $t \in T$ **do**
2: $\pi_k \leftarrow p \; s.t. \; e_{t,p} = \min_k(e_{t,k})$ {best machine for the task}
3: **end for**
4: **return** π

The MET algorithm (Minimum Execution Time) maps each task, from a randomly ordered task list, to its best machine, regardless of the machine load. This is a more shorted-viewed algorithm, even more than random, because a single powerful will concentrate all the load.

The MCT algorithm (Minimum Completion Time) maps each task, from a randomly ordered task list, to the machine that will end it the soonest. This is a little smarter that the previous ones as it takes both the load of the machines and the $e_{i,j}$ value into account.

Algorithm 8 Minimum Completion Time (MCT)

Input: T a set of n tasks in an arbitrary order,
M a set of m machines,
E a matrix of the execution costs of the tasks on the machines
Output: an allocation function π
1: Init $\forall p \in M, RT[p] \leftarrow 0$ {earliest ready time for each machine}
2: **for all** $t \in T$ **do**
3: $\pi_t \leftarrow p \; s.t. \; e_{t,p} = \min_k(RT[k] + e_{t,k})$ {min completion time for the task}
4: $RT[\pi_t] \leftarrow RT[\pi_t] + e_{t,\pi_t}$
5: **end for**
6: **return** π

The Min-min algorithm corresponds to the classical EFT algorithm (Earliest Finish Time). Actually, the Min-min algorithm presented in [26] first computes the set M of minimum completion times for each unmapped task, then takes the task with the min completion time, hence the Min-Min name. The version presented here does not compute M but rather finds the task to allocate by computing all the possible completion times and taking the earliest one. So the Min-Min algorithm just searches for the earliest finish task and allocates it to the corresponding machine. Intuitively, this algorithm will not generate good schedules when the

task heterogeneity is high as it first schedules the smallest tasks, leaving the largest ones for the end.

Algorithm 9 Earliest Finish Time or Min-min

Input: T a set of n tasks,
M a set of m machines,
E a matrix of the execution costs of the tasks on the machines
Output: an allocation function π

Init $\forall p \in M, RT[p] \leftarrow 0$ {earliest ready time for each machine}
while $T \neq \emptyset$ **do**
  $eft \leftarrow t_1$ {earliest finish task}
  $efm \leftarrow 1$ {allocation of earliest finish task}
  for all $t \in T$ **do** {find the task with earliest finish time}
    for all $p \in M$ **do**
      if $RT[p] + e_{t,p} < RT[efm] + e_{eft,efm}$ **then**
        $eft \leftarrow t$
        $efm \leftarrow p$
      end if
    end for
    $\pi_{eft} \leftarrow efm$
    $RT[efm] \leftarrow RT[efm] + e_{eft,efm}$
    $T \leftarrow T - \{eft\}$
  end for
end while
return π

The Max-min algorithm is close to the Min-min one except that it chooses the largest task instead of the smallest from the set M. It is thus close to the classical LPT that gives good results in homogeneous platforms.

The Suff algorithm relies on a sufferage value that is computed as the difference between the best and the second best finish time for the task. The algorithm first allocates the task that will suffer the most from not being allocated to its best machine, which is the one that will finish it the earliest.

The genetic algorithm is based on the genetic algorithm described by Siegel in [26]. This is a classical genetic algorithm that represents a schedule as an individual with one chromosome: each gene is a task and its associated value is the machine where the task is mapped. Note that there is no need to define the start time of the tasks as we are only interested by the $C_{\max} = \max_j(\sum_{i=1,\pi(i)=j}^{n} e_{i,j})$, which only depends on the execution duration thanks to the commutativity of sum. The population is composed of 200 chromosomes. The algorithm iterates a thousand times unless the elite chromosome does not change during 150 iterations or all the chromosomes are the same. Elitism is implemented in the algorithm. The best chromosome of each iteration becomes the elite chromosome if it gets a better fitness than the previous elite one. At each iteration the population is filtered using a wheel selection. In wheel selection the probability that a chromosome is maintained in the population is inversely proportional to its fitness. Then a crossover step, with a probably of 0.6, and mutation step, with a probability of 0.4, are applied to generate new chromosomes. Two configurations of the initial population are used. In the first one the initial population is initialised only with random individuals, the tasks are randomly assigned to machines. In the second configuration one of the chromosomes

Algorithm 10 Max-min

Input: T a set of n tasks,
M a set of m machines,
E a matrix of the execution costs of the tasks on the machines
Output: an allocation function π

Init $\forall p \in M, RT[p] \leftarrow 0$ {earliest ready time for each machine}
while $T \neq \emptyset$ **do**
  Init $\forall t \in T, CT[t] \leftarrow \infty$ {earliest completion time for each task}
  for all $t \in T$ **do** {find the task with earliest finish time}
    for all $p \in M$ **do**
      if $RT[p] + e_{t,p} < CT[t]$ **then**
        $CT[t] \leftarrow RT[p] + e_{t,p}$
        $CP[t] \leftarrow p$ {the completing machine for each task}
      end if
    end for
  end for
  $mct \leftarrow t \; s.t. \; CT[t] = \max_k(CT[k])$ {maximum completing task}
  $\pi_{mct} \leftarrow CP[mct]$
  $RT[\pi_{mct}] \leftarrow CT[\pi_{mct}]$
  $T \leftarrow T - \{mct\}$
end while
return π

Algorithm 11 First allocate the task that would suffer most (Suff)

Input: T a set of n tasks,
M a set of m machines,
E a matrix of the execution costs of the tasks on the machines
Output: an allocation function π

Init $\forall p \in M, RT[p] \leftarrow 0$ {earliest ready time for each machine}
while $T \neq \emptyset$ **do**
  $maxSuff \leftarrow 0$
  $maxSuffT \leftarrow t_1$
  $maxSuffM \leftarrow 1$
  for all $t \in T$ **do**
    $finishTime_t \leftarrow sort_k(RT[k] + e_{t,k})$ {sort in non-decreasing order of the sufferage for t}
    if $maxSuff < finishTime_t[2] - finishTime_t[1]$ **then**
      $maxSuff \leftarrow finishTime_t[2] - finishTime_t[1]$
      $maxSuffT \leftarrow t$
      $maxSuffM \leftarrow p \; s.t. RT[p] + e_{t,p} = \; finishTime_t[1]$
    end if
  end for
  $RT[maxSuffM] \leftarrow RT[maxSuffM] + e_{maxSuffT,maxSuffM}$
  $\pi_{maxSuffT} \leftarrow maxSuffM$
  $T \leftarrow T - \{maxSuffT\}$
end while
return π

is based on the mapping generated by the Min-Min algorithm to have a good solution. The algorithm is run 4 times for each configurations.

Algorithm 12 GA

Input: T a set of n tasks,
M a set of m machines,
E a matrix of the execution costs of the tasks on the machines
Output: an allocation function π

Init $nbChrom \leftarrow 200$
$pop \leftarrow InitPop(Rand, MinMin)$ {Initialize population}
while $nbIter < 1000 \wedge lastEliteChange < 150 \wedge noChange = TRUE$ **do**
  $sortedChrom \leftarrow sortByFitness(pop)$
  if $fitness(sortedChrom[1]) < fitness(elit)$ **then**
    $elit \leftarrow sortedChrom[1]$
    $lastEliteChange \leftarrow 1$
  else
    $lastEliteChange \leftarrow lastEliteChange + 1$
  end if
  $pop \leftarrow wheelSelection(pop)$
  for all $indiv \in pop$ **do** {Select individuals for cross-over}
    if $rand(0,1) > crossOverProba$ **then**
      $crossOverPop \leftarrow crossOverPop + indiv$
    end if
  end for
  for all $indiv1, indiv2 \in crossOverPop$ **do** {cross-over}
    $crossOverPlace \leftarrow rand(1,n)$
    $indiv1 \leftarrow crossOver(indiv1, indiv2, crossOverPlace)$
    $indiv2 \leftarrow crossOver(indiv2, indiv1, crossOverPlace)$
  end for
  for all $indiv \in pop$ **do** {Select individuals for mutation}
    if $rand(0,1) > mutationProba$ **then**
      $indiv \leftarrow mutation(indiv)$
    end if
  end for
  $nbIter \leftarrow nbIter + 1$:
  $noChange \leftarrow compareChrom(pop)$
end while
return π

C.2 Other Heuristics

In this part, we present other algorithms used in our simulations. They are not referenced even though they may already be defined in the literature. Since there exists so many papers on heterogeneous scheduling, we did not check them all.

The Heterogeneous Largest Processing Time (HLPT) is an adaptation of the well-known Largest Processing Time where the tasks are sorted in non-increasing order depending of their different processing times. HLPT can be used either with $func$ set to the min or $mean$ function to sort the tasks in non-increasing order. HLPT differs from Max-Min in the way it first sorts the tasks, depending on the chosen function. It then tries to find the best allocation for each

task depending on the machine load, while Max-Min recomputes M set at each iteration and thus sorts the tasks depending on the machine load.

Algorithm 13 Heterogeneous Largest Processing Time (HLPT)

Input: T a set of n tasks,
M a set of m machines,
E a matrix of the execution costs of the tasks on the machines
Output: an allocation function π

```
1: Init ∀p ∈ M, RT[p] ← 0                          {earliest ready time for each machine}
2: T ← {t_1, t_2, ...} such that func_t(e) ≥ func_{t+1}(e)  {sort tasks in non-increasing order using a
   function of their processing costs}
3: for all t ∈ T do                                 {consider each task in given order}
4:    efm ← 1
5:    for all p ∈ M do                              {find the earliest finish time}
6:       if RT[p] + e_{t,p} < RT[efm] + e_{t,efm} then
7:          efm ← p
8:       end if
9:    end for
10:   π_t ← efm
11:   RT[efm] ← RT[efm] + e_{t,efm}
12: end for
13: return π
```

The GreedySuff algorithm relies on the sufferage matrix. In the sufferage matrix each value is the difference between the processing time on the current machine $e_{i,j}$ and the minimum processing time of the task. The higher the value, the more the task will suffer from being mapped on this machine. A null value indicates that the task is processed by the fastest machine for this task.

The GreedySuff algorithm takes the task in the cost matrix order, so it does not take the tasks in a particular order. For each task, it tries to map it on the machine that generates the lower sufferage and does not increase the $C_{\max}$. If the algorithm does not find any machine, then it uses the machine with the earliest finish time, hence the one that increases $C_{\max}$ the least.

The BalSuff algorithm, as the GreedySuff, uses the sufferage matrix. It starts from an initial mapping where the tasks are mapped on their best machine (MET algorithm). Then, the algorithm tries to rearrange the tasks in a way such that the makespan is improved and the chosen task is the one that suffers the least from moving. The algorithm stops when there is no more task on the most loaded machine that could benefit from moving. Note that if two tasks have the same sufferage value on the most loaded machine, the tasks are considered in an arbitrary order

The BalEFT algorithm is a variant of the BalSuff algorithm. After the initial mapping based on the shortest processing time (MET algorithm), this algorithm tries to rearrange tasks using an early finish time criterion: it takes all the tasks of the most loaded machine and tries to find a new allocation such that the global makespan is decreased. The algorithm stops when there is no more task than can decrease the $C_{\max}$ by moving.

Although not very costly in practice, we were not able to determine the worst-case time complexity of either BalSuff or BalEFT.

Eventually we also have implemented a second version of the GA algorithm, initialized with a random initial population where we added one chromosome for each mapping generated by the other algorithms. This algorithm gives of course always the best result as the elitism procedure

Algorithm 14 Greedily allocate tasks on suffering (GreedySuff)

Input: T a set of n tasks,
M a set of m machines,
E a matrix of the execution costs of the tasks on the machines
Output: an allocation function π

Init $\forall p \in M, RT[p] \leftarrow 0$ {earliest ready time for each machine}
for all $t \in T$, $p \in M$ **do** {generate the sufferage matrix S}
  $S_{t,p} \leftarrow e_{t,p} - \min_k(e_{t,k})$
end for
$C_{\max} \leftarrow 0$
for all $t \in T$ **do** {allocate tasks}
  $SortSuff \leftarrow sort(S_t)$ {sort in non-decreasing order of the sufferage for t}
  $k \leftarrow 1$
  $found \leftarrow false$
  repeat
    $p \leftarrow SortSuff[k]$
    if $RT[p] + e_{t,p} \leq C_{\max}$ **then**
      $\pi_t \leftarrow p$
      $RT[p] \leftarrow RT[p] + e_{t,p}$
      $found \leftarrow true$
    end if
    $k \leftarrow k + 1$
  until $k \geq Card(M)$ **or** $found$
  if $\neg found$ **then** {if no allocation found use EFT}
    $\pi_t \leftarrow \arg\min_i(RT[i] + e_{t,i})$
    $RT[\pi_t] \leftarrow RT[\pi_t] + e_{t,\pi_t}$
  end if
  $C_{\max} \leftarrow \max_i(RT[i])$
end for
return π

Algorithm 15 Sufferage with machine balancing (BalSuff)

Input: T a set of n tasks,
M a set of m machines,
E a matrix of the execution costs of the tasks on the machines
Output: an allocation function π

1: **for all** $t \in T$, $p \in M$ **do** {generate the sufferage matrix S}
2: $S_{t,p} \leftarrow e_{t,p} - \min_k(e_{t,k})$
3: **end for**
4: Init $\forall p \in M, RT[p] \leftarrow 0$ {earliest ready time for each machine}
5: **for all** $t \in T$ **do** {initial allocation based on MET}
6: $\pi_k \leftarrow p \; s.t. \; e_{t,p} = \min_k(e_{t,k})$
7: $RT[\pi_t] \leftarrow RT[\pi_t] + e_{t,\pi_t}$
8: **end for**
9: $C_{\max} \leftarrow \max_k(RT[k])$
10: $newC_{\max} \leftarrow \max_k(RT[k])$
11: $M_{\max} \leftarrow p \; s.t. \; RT[p] = C_{\max}$
12: **repeat** {try to re-arange task allocations}
13: $MinSuff \leftarrow \max_{i,j}(S_{i,j}) + 1$; $mst \leftarrow 0$; $msm \leftarrow 0$
14: **for all** $t \in T$ such that $\pi_t = M_{\max}$ **do** {find the task that suffers less from moving}
15: **for all** $p \in M - \{M_{\max}\}$ **do**
16: **if** $(MinSuff > S_{t,p})$ **and** $(RT[p] + e_{t,p} \leq C_{\max})$ **then**
17: $MinSuff \leftarrow S_{t,p}$; $mst \leftarrow t$; $msm \leftarrow p$
18: **end if**
19: **end for**
20: **end for**
21: **if** $RT[msm] + e_{mst,msm} \leq C_{\max}$ **then** {re-allocation improves $C_{\max}$}
22: $newC_{\max} \leftarrow \max(C_{\max} - e_{mst,msm}, RT[msm] + e_{mst,msm})$
23: $\pi_t \leftarrow msm$ {task is moved}
24: $RT[msm] \leftarrow RT[msm] + e_{mst,msm}$
25: $RT[M_{\max}] \leftarrow RT[M_{\max}] - e_{mst,msm}$
26: **if** $newC_{\max} = RT[msm] + e_{mst,msm}$ **then** {find last finishing machine}
27: $M_{\max} \leftarrow msm$
28: **end if**
29: **end if**
30: $C_{\max} = newC_{\max}$
31: **until** $mst \neq 0$
32: **return** π

Algorithm 16 Earliest Finish Time with machine balancing (BalEFT)

Input: T a set of n tasks,
M a set of m machines,
E a matrix of the execution costs of the tasks on the machines
Output: an allocation function π

for all $t \in T$, $p \in M$ **do** {generate the sufferage matrix S}
  $S_{t,p} \leftarrow e_{t,p} - \min_k(e_{t,k})$
end for
Init $\forall p \in M, RT[p] \leftarrow 0$ {earliest ready time for each machine}
for all $t \in T$ **do** {initial allocation based on MET}
  $\pi_k \leftarrow p \; s.t. \; e_{t,p} = \min_k(e_{t,k})$
  $RT[\pi_t] \leftarrow RT[\pi_t] + e_{t,\pi_t}$
end for
repeat {try to re-arange task allocations}
  $M_{\max} \leftarrow p \; s.t. \; p = \max_k(RT[k])$
  $MinRT \leftarrow RT[M_{\max}]$; $eft \leftarrow 0$; $efm \leftarrow 0$
  for all $t \in T$ such that $\pi_t = M_{\max}$ **do** {find the task that benefit most from moving}
    for all $p \in M - \{M_{\max}\}$ **do**
      $newRT \leftarrow RT[p] + e_{t,p}$
      if $newRT < minRT$ **then**
        $minRT \leftarrow newRT$; $eft \leftarrow t$; $efm \leftarrow p$
      end if
    end for
  end for
  if $RT[efm] + e_{eft,efm} < C_{\max}$ **then** {re-allocation improves $C_{\max}$}
    $newC_{\max} \leftarrow \max(C_{\max} - e_{eft,efm}, RT[efm] + e_{eft,efm})$
    $\pi_{eft} \leftarrow efm$ {task is moved}
    $RT[efm] \leftarrow RT[efm] + e_{eft,efm}$
    $RT[M_{\max}] \leftarrow RT[M_{\max}] - e_{eft,efm}$
    if $newC_{\max} = RT[efm] + e_{eft,efm}$ **then** {find last finishing machine}
      $M_{\max} \leftarrow efm$
    end if
  end if
  $C_{\max} = newC_{\max}$
until $eft \neq 0$
return π

guaranties that we keep the best chromosome (at least the best of the results of the other algorithms).

References

[1] P. Agrawal. *A Novel Heuristic for a Class of Independent Tasks in Computational Grids.* PhD thesis, National Institute of Technology Rourkela, 2013.

[2] A. M. Al-Qawasmeh, A. A. Maciejewski, R. G. Roberts, and H. J. Siegel. Characterizing task-machine affinity in heterogeneous computing environments. In *International Parallel & Distributed Processing Symposium Workshops and Phd Forum (IPDPSW)*, pages 34–44. IEEE, 2011.

[3] A. M. Al-Qawasmeh, A. A. Maciejewski, and H. J. Siegel. Characterizing heterogeneous computing environments using singular value decomposition. In *International Parallel & Distributed Processing Symposium Workshops and Phd Forum (IPDPSW)*, pages 1–9. IEEE, 2010.

[4] A. M. Al-Qawasmeh, A. A. Maciejewski, H. J. Siegel, J. Smith, and J. Potter. Task and Machine Heterogeneities: Higher Momenets Matter. In *Parallel and Distributed Processing Techniques and Applications (PDPTA)*, pages 3–9, 2009.

[5] A. M. Al-Qawasmeh, A. A. Maciejewski, H. Wang, J. Smith, H. J. Siegel, and J. Potter. Statistical measures for quantifying task and machine heterogeneities. *The Journal of Supercomputing*, 57(1):34–50, 2011.

[6] A. M. Al-Qawasmeh, S. Pasricha, A. A. Maciejewski, and H. J. Siegel. Power and Thermal-Aware Workload Allocation in Heterogeneous Data Centers. *Transactions on Computers*, 64(2):477–491, 2013.

[7] F. Alharbi and K. Rabigh. Simple scheduling algorithm with load balancing for grid computing. *Asian Transactions on Computers*, 2(2), 2012.

[8] S. Ali. *A comparative study of dynamic mapping heuristics for a class of independent tasks onto heterogeneous computing systems.* PhD thesis, Purdue University, 1999.

[9] S. Ali, J.-K. Kim, H. J. Siegel, and A. A. Maciejewski. Static heuristics for robust resource allocation of continuously executing applications. *Journal of Parallel and Distributed Computing*, 68(8):1070–1080, 2008.

[10] S. Ali, J.-K. Kim, H. J. Siegel, A. A. Maciejewski, Y. Yu, S. B. Gundala, S. Gertphol, and V. K. Prasanna. Greedy Heuristics for Resource Allocation in Dynamic Distributed Real-Time Heterogeneous Computing Systems. In *Parallel and Distributed Processing Techniques and Applications (PDPTA)*, pages 519–530, 2002.

[11] S. Ali, J.-K. Kim, Y. Yu, S. B. Gundala, S. Gertphol, H. J. Siegel, A. A. Maciejewski, and V. K. Prasanna. Utilization-Based Heuristics for Statically Mapping Real-Time Applications onto the HiPer-D Heterogeneous Computing System. In *International Parallel & Distributed Processing Symposium (IPDPS)*. IEEE Computer Society, 2002.

[12] S. Ali, J.-K. Kim, Y. Yu, S. B. Gundala, S. Gertphol, H. J. Siegel, A. A. Maciejewski, and V. K. Prasanna. Utilization-based techniques for statically mapping heterogeneous applications onto the HiPer-D heterogeneous computing system. *Parallel and Distributed Computing Practices*, 5(4), 2002.

[13] S. Ali, A. A. Maciejewski, H. J. Siegel, and J.-K. Kim. On the Robustness of Resource Allocation for Parallel and Distributed Computing and Communications. In *Parallel and Distributed Processing Techniques and Applications (PDPTA)*, pages 3–16, 2003.

[14] S. Ali, A. A. Maciejewski, H. J. Siegel, and J.-K. Kim. Measuring the robustness of a resource allocation. *Transactions on Parallel and Distributed Systems*, 15(7):630–641, 2004.

[15] S. Ali, A. A. Maciejewski, H. J. Siegel, and J.-K. Kim. Robust resource allocation for sensor-actuator distributed computing systems. In *International Conference on Parallel Processing (ICPP)*, pages 178–185. IEEE, 2004.

[16] S. Ali, H. J. Siegel, and A. A. Maciejewski. The robustness of resource allocation in parallel and distributed computing systems. In *International Workshop on Parallel and Distributed Computing*, pages 2–10. IEEE, 2004.

[17] S. Ali, H. J. Siegel, M. Maheswaran, and D. Hensgen. Task execution time modeling for heterogeneous computing systems. In *Heterogeneous Computing Workshop (HCW)*, pages 185–199. IEEE, 2000.

[18] S. Ali, H. J. Siegel, M. Maheswaran, D. Hensgen, and S. Ali. Representing task and machine heterogeneities for heterogeneous computing systems. *Tamkang Journal of Science and Engineering*, 3(3):195–208, 2000.

[19] J. Apodaca, B. D. Young, L. D. Briceño, J. Smith, S. Pasricha, A. A. Maciejewski, H. J. Siegel, S. Bahirat, B. Khemka, A. Ramirez, and Y. Zou. Stochastically robust static resource allocation for energy minimization with a makespan constraint in a heterogeneous computing environment. In *ACS/IEEE International Conference on Computer Systems and Applications (AICCSA)*, pages 22–31. IEEE, 2011.

[20] R. K. Armstrong Jr. Investigation of effect of different run-time distributions on SmartNet performance. Technical report, DTIC Document, 1997.

[21] J. Bagherzadeh and M. MadadyarAdeh. An improved ant algorithm for grid scheduling problem. In *CSI Computer Conference (CSICC)*, pages 323–328. IEEE, 2009.

[22] A. K. Bardsiri and S. M. Hashemi. A Comparative Study on Seven Static Mapping Heuristics for Grid Scheduling Problem. *International Journal of Software Engineering and Its Applications*, 6(4):247–256, 2012.

[23] A. K. Bardsiri and M. K. Rafsanjani. Differential evolution algorithms for grid scheduling problem. *International Journal of Physical Sciences*, 6(24):5682–5687, 2011.

[24] A. K. Bardsiri and M. K. Rafsanjani. Scheduling Independent Tasks on Grid Computing Systems by Differential Evolution. *KMITL Science and Technology Journal*, 11(1), 2011.

[25] T. D. Braun, H. J. Siegel, N. Beck, L. L. Bölöni, M. Maheswaran, A. I. Reuther, J. P. Robertson, M. D. Theys, B. Yao, D. Hensgen, and R. F. Freund. A comparison study of static mapping heuristics for a class of meta-tasks on heterogeneous computing systems. In *Heterogeneous Computing Workshop (HCW)*, pages 15–29. IEEE, 1999.

[26] T. D. Braun, H. J. Siegel, N. Beck, L. L. Bölöni, M. Maheswaran, A. I. Reuther, J. P. Robertson, M. D. Theys, B. Yao, D. Hensgen, and R. F. Freund. A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems. *Journal of Parallel and Distributed Computing*, 61(6):810–837, 2001.

[27] T. D. Braun, H. J. Siegel, and A. A. Maciejewski. Heterogeneous computing: Goals, methods, and open problems. In *High Performance Computing (HiPC)*, pages 307–318. Springer, 2001.

[28] T. D. Braun, H. J. Siegel, A. A. Maciejewski, and Y. Hong. Static resource allocation for heterogeneous computing environments with tasks having dependencies, priorities, deadlines, and multiple versions. *Journal of Parallel and Distributed Computing*, 68(11):1504–1516, 2008.

[29] J. M. Breitinger. Optimal size of job pool for initiating a scheduling event. Technical report, DTIC Document, 1999.

[30] L. D. Briceño, B. Khemka, H. J. Siegel, A. A. Maciejewski, C. Groër, G. A. Koenig, G. Okonski, and S. Poole. Time utility functions for modeling and evaluating resource allocations in a heterogeneous computing system. In *International Parallel & Distributed Processing Symposium Workshops and Phd Forum (IPDPSW)*, pages 7–19. IEEE, 2011.

[31] L. D. Briceño, H. J. Siegel, A. A. Maciejewski, M. Oltikar, J. Brateman, J. White, J. Martin, and K. Knapp. Heuristics for robust resource allocation of satellite weather data processing on a heterogeneous parallel system. *Transactions on Parallel and Distributed Systems*, 22(11):1780–1787, 2011.

[32] L. D. Briceño, J. Smith, H. J. Siegel, A. A. Maciejewski, P. Maxwell, R. Wakefield, A. M. Al-Qawasmeh, R. C. Chiang, and J. Li. Robust resource allocation of DAGs in a heterogeneous multicore system. In *International Parallel & Distributed Processing Symposium Workshops and Phd Forum (IPDPSW)*, pages 1–11. IEEE, 2010.

[33] L. D. Briceño, J. Smith, H. J. Siegel, A. A. Maciejewski, P. Maxwell, R. Wakefield, A. M. Al-Qawasmeh, R. C. Chiang, and J. Li. Robust static resource allocation of DAGs in a heterogeneous multicore system. *Journal of Parallel and Distributed Computing*, 73(12):1705–1717, 2013.

[34] M. Canabé and S. Nesmachnow. Parallel implementations of the MinMin heterogeneous computing scheduler in GPU. *CLEI Electronic Journal*, 15(3):8–8, 2012.

[35] L.-C. Canon and E. Jeannot. A comparison of robustness metrics for scheduling dags on heterogeneous systems. In *International Conference on Cluster Computing*, pages 558–567. IEEE, 2007.

[36] L.-C. Canon, E. Jeannot, R. Sakellariou, and W. Zheng. Comparative evaluation of the robustness of dag scheduling heuristics. In *Grid Computing*, pages 73–84. Springer, 2008.

[37] L.-C. Canon and L. Philippe. Code for On the Heterogeneity Bias of Cost Matrices when Assessing Scheduling Algorithms. *http://dx.doi.org/10.6084/m9.figshare.1321295.v3*, Mar. 2015.

[38] H. Casanova, A. Legrand, D. Zagorodnov, and F. Berman. Heuristics for Scheduling Parameter Sweep Applications in Grid Environments. In *Heterogeneous Computing Workshop (HCW)*, pages 349–363. IEEE, 2000.

[39] S. C. Choi and H. Y. Youn. Task mapping algorithm for heterogeneous computing system allowing high throughput and load balancing. In *International Conference on Computational Science (ICCS)*, pages 1000–1003. Springer, 2005.

[40] M. Da Silva and S. Nesmachnow. Heterogeneous Resource Allocation in the OurGrid Middleware: A Greedy Approach. In *High Performance Computing Latinamerica (HPCLATAM)*, 2013.

[41] C. O. Diaz, M. Guzek, J. E. Pecero, P. Bouvry, and S. U. Khan. Scalable and energy-efficient scheduling techniques for large-scale systems. In *Computer and Information Technology (CIT)*, pages 641–647. IEEE, 2011.

[42] C. O. Diaz, M. Guzek, J. E. Pecero, G. Danoy, P. Bouvry, and S. U. Khan. Energy-aware fast scheduling heuristics in heterogeneous computing systems. In *High Performance Computing and Simulation (HPCS)*, pages 478–484. IEEE, 2011.

[43] C. O. Diaz, J. E. Pecero, and P. Bouvry. Scalable, low complexity, and fast greedy scheduling heuristics for highly heterogeneous distributed computing systems. *The Journal of Supercomputing*, 67(3):837–853, 2014.

[44] D. Ding, S. Luo, and Z. Gao. A matrix scheduling strategy with multi-qos constraints in computational grid. In *Advances in Grid and Pervasive Computing*, pages 59–68. Springer, 2010.

[45] B. Dorronsoro, P. Bouvry, J. A. Canero, A. A. Maciejewski, and H. J. Siegel. Multi-objective robust static mapping of independent tasks on grids. In *Congress on Evolutionary Computation (CEC)*, pages 1–8. IEEE, 2010.

[46] B. Dorronsoro, G. Danoy, P. Bouvry, and A. J. Nebro. Multi-objective cooperative coevolutionary evolutionary algorithms for continuous and combinatorial optimization. In *Intelligent Decision Systems in Large-Scale Distributed Environments*, pages 49–74. Springer, 2011.

[47] B. Dorronsoro, G. Danoy, A. J. Nebro, and P. Bouvry. Achieving super-linear performance in parallel multi-objective evolutionary algorithms by means of cooperative coevolution. *Computers & Operations Research*, 40(6):1552–1563, 2013.

[48] A. Doğan and F. Özgüner. On QoS-based scheduling of a meta-task with multiple QoS demands in heterogeneous computing. In *International Parallel & Distributed Processing Symposium (IPDPS)*, pages 6–pp. IEEE, 2001.

[49] A. Doğan and F. Özgüner. Genetic algorithm based scheduling of meta-tasks with stochastic execution times in heterogeneous computing systemst1. *Cluster Computing*, 7(2):177–190, 2004.

[50] A. Doğan and F. Özgüner. Scheduling of a meta-task with QoS requirements in heterogeneous computing systems. *Journal of Parallel and Distributed Computing*, 66(2):181–196, 2006.

[51] B. Eslamnour and S. Ali. Measuring robustness of computing systems. *Simulation Modelling Practice and Theory*, 17(9):1457–1467, 2009.

[52] R. Friese, T. Brinks, C. Oliver, H. J. Siegel, and A. A. Maciejewski. Analyzing the trade-offs between minimizing makespan and minimizing energy consumption in a heterogeneous resource allocation problem. In *Conference on Advanced Communications and Computation (INFOCOMP)*, pages 81–89, 2012.

[53] R. Friese, B. Khemka, A. A. Maciejewski, H. J. Siegel, G. A. Koenig, S. Powers, M. Hilton, J. Rambharos, G. Okonski, and S. W. Poole. An analysis framework for investigating the trade-offs between system performance and energy consumption in a heterogeneous computing environment. In *International Parallel & Distributed Processing Symposium Workshops and Phd Forum (IPDPSW)*, pages 19–30. IEEE, 2013.

[54] S. K. Garg, R. Buyya, and H. J. Siegel. Scheduling parallel applications on utility grids: time and cost trade-off management. In *Australasian Conference on Computer Science*, pages 151–160. Australian Computer Society, Inc., 2009.

[55] S. K. Garg, R. Buyya, and H. J. Siegel. Time and cost trade-off management for scheduling parallel applications on utility grids. *Future Generation Computer Systems*, 26(8):1344–1355, 2010.

[56] S. Gertphol and V. K. Prasanna. MIP formulation for robust resource allocation in dynamic real-time systems. *Journal of Systems and Software*, 77(1):55–65, 2005.

[57] S. Gertphol, Y. Yu, A. Alhusaini, and V. K. Prasanna. An integer programming approach for static mapping of paths onto heterogeneous real-time systems. In *International Parallel & Distributed Processing Symposium (IPDPS)*, volume 3, pages 30095a–30095a. IEEE Computer Society, 2001.

[58] S. Gertphol, Y. Yu, S. B. Gundala, V. K. Prasanna, S. Ali, J.-K. Kim, A. A. Maciejewski, and H. J. Siegel. A metric and mixed-integer-programming-based approach for resource allocation in dynamic real-time systems. In *International Parallel & Distributed Processing Symposium (IPDPS)*, pages 10–pp. IEEE, 2001.

[59] S. Ghosh and S. G. Henderson. Behavior of the NORTA method for correlated random vector generation as the dimension increases. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 13(3):276–294, 2003.

[60] L. K. Goh, B. Veeravalli, and S. Viswanathan. An energy-aware gradient-based scheduling heuristic for heterogeneous multiprocessor embedded systems. In *High Performance Computing (HiPC)*, pages 331–341. Springer, 2007.

[61] K. S. Golconda, A. Doğan, and F. Özgüner. Static mapping heuristics for tasks with hard deadlines in real-time heterogeneous systems. In *International Symposium on Computer and Information Sciences (ISCIS)*, pages 827–836. Springer, 2004.

[62] R. L. Graham. Bounds on Multiprocessing Timing Anomalies. *Journal of Applied Mathematics*, 17(2):416–429, 1969.

[63] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. R. Kan. Optimization and Approximation in Deterministic Sequencing and Scheduling: a Survey. *Annals of Discrete Mathematics*, 5:287–326, 1979.

[64] T. Hansen, F. M. Ciorba, A. A. Maciejewski, H. J. Siegel, S. Srivastava, and I. Banicescu. Heuristics for Robust Allocation of Resources to Parallel Applications with Uncertain Execution Times in Heterogeneous Systems with Uncertain Availability. In *International Conference of Parallel and Distributed Computing (ICPDC)*, 2014.

[65] S. M. Hashemi and A. K. Bardsiri. An evaluation of heuristic methods for grid scheduling problem. In *Advances in Enterprise Information Systems*, page 381. CRC Press, 2012.

[66] W. Higashino, M. A. Capretz, and M. B. F. Toledo. Evaluation of Particle Swarm Optimization Applied to Grid Scheduling. In *Convergence of Distributed Clouds, Grids and their Management (CDCGM)*, 2014.

[67] J. Hu, M. Li, W. Sun, and Y. Chen. An ant colony optimization for grid task scheduling with multiple QoS dimensions. In *Grid and Cooperative Computing (GCC)*, pages 415–419. IEEE, 2009.

[68] D. Huang, Y. Yuan, L. jun Zhang, and K. qin Zhao. Research on tasks scheduling algorithms for dynamic and uncertain computing grid based on a+ bi connection number of SPA. *Journal of Software*, 4(10):1102–1109, 2009.

[69] S. Iturriaga, S. Nesmachnow, and B. Dorronsoro. A Multithreading Local Search For Multiobjective Energy-Aware Scheduling In Heterogeneous Computing Systems. In *European Conference on Modelling and Simulation (ECMS)*, pages 497–503, 2012.

[70] S. Iturriaga, S. Nesmachnow, B. Dorronsoro, and P. Bouvry. Energy efficient scheduling in heterogeneous systems with a parallel multiobjective local search. *Computing and Informatics*, 32(2):273–294, 2013.

[71] Q. Kang and H. He. A novel discrete particle swarm optimization algorithm for meta-task assignment in heterogeneous computing systems. *Microprocessors and Microsystems*, 35(1):10–17, 2011.

[72] Q. Kang, H. He, and H. Song. Task assignment in heterogeneous computing systems using an effective iterated greedy algorithm. *Journal of Systems and Software*, 84(6):985–992, 2011.

[73] Q. Kang, H. He, H. Song, and R. Deng. Task allocation for maximizing reliability of distributed computing systems using honeybee mating optimization. *Journal of Systems and Software*, 83(11):2165–2174, 2010.

[74] Q. Kang, H. He, and J. Wei. An effective iterated greedy algorithm for reliability-oriented task allocation in distributed computing systems. *Journal of Parallel and Distributed Computing*, 73(8):1106–1115, 2013.

[75] S. Kardani-Moghaddam, R. Entezari-Maleki, and A. Movaghar. A Cost Efficient Two-level Market Model for Task Scheduling Problem in Grid Environment. *Iranian Journal of Science & Technology*, 38(E1):73–90, 2014.

[76] K. Kaya and B. Uçar. Exact algorithms for a task assignment problem. *Parallel Processing Letters*, 19(03):451–465, 2009.

[77] G. L. Kee. *Design and Performance Evaluation of Energy-Aware DVS-Based Scheduling Strategies for Hard Real-Time Embedded Multiprocessor Systems*. PhD thesis, National University of Singapore, 2012.

[78] S. U. Khan. A Game Theoretical Energy Efficient Resource Allocation Technique for Large Distributed Computing Systems. In *Parallel and Distributed Processing Techniques and Applications (PDPTA)*, pages 48–54. Citeseer, 2009.

[79] S. U. Khan and I. Ahmad. Non-cooperative, semi-cooperative, and cooperative games-based grid resource allocation. In *International Parallel & Distributed Processing Symposium (IPDPS)*, pages 10–pp. IEEE, 2006.

[80] S. U. Khan and C. Ardil. Energy efficient resource allocation in distributed computing systems. In *International Conference on Distributed, High-Performance and Grid Computing*, pages 667–673. Citeseer, 2009.

[81] B. Khargharia, S. Hariri, F. Szidarovszky, M. Houri, H. El-Rewini, S. U. Khan, I. Ahmad, and M. S. Yousif. Autonomic power & performance management for large-scale data centers. In *International Parallel & Distributed Processing Symposium (IPDPS)*, pages 1–8. IEEE, 2007.

[82] B. Khemka, R. Friese, S. Pasricha, A. A. Maciejewski, H. J. Siegel, G. A. Koenig, S. Powers, M. Hilton, R. Rambharos, and S. Poole. Utility maximizing dynamic resource management in an oversubscribed energy-constrained heterogeneous computing system. *Sustainable Computing: Informatics and Systems*, 5:14–30, 2014.

[83] I.-Y. Kim and J.-K. Kim. Enhancing the performance of a distributed mobile computing environment by topology construction. In *Algorithms and Architectures for Parallel Processing*, pages 21–30. Springer, 2012.

[84] J.-K. Kim, S. Shivle, H. J. Siegel, A. A. Maciejewski, T. D. Braun, M. Schneider, S. Tideman, R. Chitta, R. B. Dilmaghani, R. Joshi, A. Kaul, A. Sharma, S. Sripada, P. Vangari, and S. S. Yellampalli. Dynamic mapping in a heterogeneous environment with tasks having priorities and multiple deadlines. In *International Parallel & Distributed Processing Symposium (IPDPS)*. IEEE, 2003.

[85] J.-K. Kim, S. Shivle, H. J. Siegel, A. A. Maciejewski, T. D. Braun, M. Schneider, S. Tideman, R. Chitta, R. B. Dilmaghani, R. Joshi, A. Kaul, A. Sharma, S. Sripada, P. Vangari, and S. S. Yellampalli. Dynamically mapping tasks with priorities and multiple deadlines in a heterogeneous environment. *Journal of Parallel and Distributed Computing*, 67(2):154–169, 2007.

[86] J.-K. Kim, H. J. Siegel, A. A. Maciejewski, and R. Eigenmann. Dynamic mapping in energy constrained heterogeneous computing systems. In *International Parallel & Distributed Processing Symposium (IPDPS)*, pages 64a–64a. IEEE, 2005.

[87] J.-K. Kim, H. J. Siegel, A. A. Maciejewski, and R. Eigenmann. Dynamic resource management in energy constrained heterogeneous computing systems using voltage scaling. *Transactions on Parallel and Distributed Systems*, 19(11):1445–1457, 2008.

[88] S. I. Kim, J. Y. Jun, J.-K. Kim, K.-C. Lee, G. S. Kang, T.-S. Kim, H. K. Moon, H. C. Yoon, H. Kim, and S. H. Lee. Dynamic resource management for a cell-based distributed mobile computing environment. In *Ubiquitous Intelligence and Computing*, pages 174–184. Springer, 2011.

[89] J. Kołodziej and S. U. Khan. Multi-level hierarchic genetic-based scheduling of independent jobs in dynamic heterogeneous grid environment. *Information Sciences*, 214:1–19, 2012.

[90] J. Kołodziej, S. U. Khan, L. Wang, A. Byrski, N. Min-Allah, and S. A. Madani. Hierarchical genetic-based grid scheduling with energy optimization. *Cluster Computing*, 16(3):591–609, 2013.

[91] J. Kołodziej, S. U. Khan, L. Wang, D. Chen, and A. Y. Zomaya. Energy and Security Awareness in Evolutionary-Driven Grid Scheduling. In *Evolutionary Based Solutions for Green Computing*, pages 95–138. Springer, 2013.

[92] J. Kołodziej, S. U. Khan, L. Wang, M. Kisiel-Dorohinicki, S. A. Madani, E. Niewiadomska-Szynkiewicz, A. Y. Zomaya, and C.-Z. Xu. Security, energy, and performance-aware resource allocation mechanisms for computational grids. *Future Generation Computer Systems*, 31:77–92, 2014.

[93] J. Kołodziej, S. U. Khan, L. Wang, and A. Y. Zomaya. Energy efficient genetic-based schedulers in computational grids. *Concurrency and Computation: Practice and Experience*, 2012.

[94] J. Kołodziej, S. U. Khan, and F. Xhafa. Genetic algorithms for energy-aware scheduling in computational grids. In *P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC)*, pages 17–24. IEEE, 2011.

[95] J. Kołodziej, M. Szmajduch, S. U. Khan, L. Wang, and D. Chen. Genetic-Based Solutions For Independent Batch Scheduling In Data Grids. In *European Conference on Modelling and Simulation (ECMS)*, pages 504–510, 2013.

[96] J. Kołodziej, M. Szmajduch, T. Maqsood, S. A. Madani, N. Min-Allah, and S. U. Khan. Energy-Aware Grid Scheduling of Independent Tasks and Highly Distributed Data. In *Frontiers of Information Technology (FIT)*, pages 211–216. IEEE, 2013.

[97] J. Kołodziej and F. Xhafa. Enhancing the genetic-based scheduling in computational grids by a structured hierarchical population. *Future Generation Computer Systems*, 27(8):1035–1046, 2011.

[98] J. Kołodziej and F. Xhafa. Meeting security and user behavior requirements in Grid scheduling. *Simulation Modelling Practice and Theory*, 19(1):213–226, 2011.

[99] J. Kołodziej and F. Xhafa. Modern approaches to modeling user requirements on resource and task allocation in hierarchical computational grids. *International Journal of Applied Mathematics and Computer Science*, 21(2):243–257, 2011.

[100] J. Kołodziej and F. Xhafa. Integration of task abortion and security requirements in GA-based meta-heuristics for independent batch grid scheduling. *Computers & Mathematics with Applications*, 63(2):350–364, 2012.

[101] J. Kołodziej, F. Xhafa, and M. Bogdański. A Stackelberg Game for Modelling Asymmetric Users' Behavior in Grid Scheduling. In *International Conference on Computer Modelling and Simulation (UKSim)*, pages 497–502. IEEE, 2010.

[102] J. Kołodziej, F. Xhafa, and M. Bogdański. Secure and task abortion aware ga-based hybrid metaheuristics for grid scheduling. In *Parallel Problem Solving from Nature (PPSN)*, pages 526–535. Springer, 2010.

[103] J. Kołodziej, F. Xhafa, and Ł. Kolanko. Hierarchic genetic scheduler of independent jobs in computational grid environment. In *European Conference on Modelling and Simulation (ECMS)*, pages 9–12, 2009.

[104] D. Kumar and B. Sahoo. Energy Efficient Heuristic Resource Allocation for Cloud Computing. *Artificial Intelligent Systems and Machine Learning*, 6(1):32–38, 2014.

[105] J. Y.-T. Leung, editor. *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*. Chapman & Hall/CCR, 2004.

[106] J. Li, Z. Ming, M. Qiu, G. Quan, X. Qin, and T. Chen. Resource allocation robustness in multi-core embedded systems with inaccurate information. *Journal of Systems Architecture*, 57(9):840–849, 2011.

[107] P. Lindberg, J. Leingang, D. Lysaker, K. Bilal, S. U. Khan, P. Bouvry, N. Ghani, N. Min-Allah, and J. Li. Comparison and analysis of greedy energy-efficient scheduling algorithms for computational grids. *Energy-Efficient Distributed Computing Systems*, pages 189–214, 2011.

[108] P. Lindberg, J. Leingang, D. Lysaker, S. U. Khan, and J. Li. Comparison and analysis of eight scheduling heuristics for the optimization of energy consumption and makespan in large-scale distributed systems. *The Journal of Supercomputing*, 59(1):323–360, 2012.

[109] P. Luo, K. Lü, and Z. Shi. 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, 2007.

[110] M. Maheswaran, S. Ali, H. J. Siegel, D. Hensgen, and R. F. Freund. Dynamic mapping of a class of independent tasks onto heterogeneous computing systems. *Journal of Parallel and Distributed Computing*, 59(2):107–131, 1999.

[111] M. Maheswaran, S. Ali, H. J. Siegel, D. Hensgen, and R. F. Freund. Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems. In *Heterogeneous Computing Workshop (HCW)*, pages 30–44. IEEE, 1999.

[112] A. M. Mehta, J. Smith, H. J. Siegel, A. A. Maciejewski, A. Jayaseelan, and B. Ye. Dynamic Resource Allocation Heuristics for Maximizing Robustness with an Overall Makespan Constraint in an Uncertain Environment. In *Parallel and Distributed Processing Techniques and Applications (PDPTA)*, pages 24–30, 2006.

[113] A. M. Mehta, J. Smith, H. J. Siegel, A. A. Maciejewski, A. Jayaseelan, and B. Ye. Dynamic resource management heuristics for minimizing makespan while maintaining an acceptable level of robustness in an uncertain environment. In *International Conference on Parallel and Distributed Systems (ICPADS)*. IEEE, 2006.

[114] A. M. Mehta, J. Smith, H. J. Siegel, A. A. Maciejewski, A. Jayaseelan, and B. Ye. Dynamic resource allocation heuristics that manage tradeoff between makespan and robustness. *The Journal of Supercomputing*, 42(1):33–58, 2007.

[115] E. U. Munir, J.-Z. Li, S.-F. Shi, Z.-N. Zou, and Q. Rasool. A new heuristic for task scheduling in heterogeneous computing environment. *Journal of Zhejiang University Science*, 9(12):1715–1723, 2008.

[116] E. U. Munir, J.-Z. Li, S.-F. Shi, Z.-N. Zou, and D.-H. Yang. MaxStd: A task scheduling heuristic for heterogeneous computing environment. *Information Technology Journal*, 7(4):679–683, 2008.

[117] S. Nejatzadeh, M. Karamipour, and M. Eskandari. A New Heuristic Approach for Scheduling Independent Tasks on Grid Computing Systems. *International Journal of Grid & Distributed Computing*, 6(4), 2013.

[118] S. Nesmachnow. Parallel multiobjective evolutionary algorithms for batch scheduling in heterogeneous computing and grid systems. *Computational Optimization and Applications*, 55(2):515–544, 2013.

[119] S. Nesmachnow, E. Alba, and H. Cancela. Scheduling in heterogeneous computing and grid environments using a parallel CHC evolutionary algorithm. *Computational Intelligence*, 28(2):131–155, 2012.

[120] S. Nesmachnow and M. Canabé. GPU implementations of scheduling heuristics for heterogeneous computing environments. In *XVII Congreso Argentino de Ciencias de la Computación*, 2011.

[121] S. Nesmachnow, H. Cancela, and E. Alba. Heterogeneous computing scheduling with evolutionary algorithms. *Soft Computing*, 15(4):685–701, 2010.

[122] S. Nesmachnow, H. Cancela, and E. Alba. A parallel micro evolutionary algorithm for heterogeneous computing and grid scheduling. *Applied Soft Computing*, 12(2):626–639, 2012.

[123] S. Nesmachnow, B. Dorronsoro, J. E. Pecero, and P. Bouvry. Energy-aware scheduling on multicore heterogeneous grid computing systems. *Journal of Grid Computing*, 11(4):653–680, 2013.

[124] S. Nesmachnow and S. Iturriaga. Multiobjective grid scheduling using a domain decomposition based parallel micro evolutionary algorithm. *International Journal of Grid and Utility Computing*, 4(1):70–84, 2013.

[125] M. Oltikar, J. Brateman, J. White, J. Martin, K. Knapp, A. A. Maciejewski, and H. J. Siegel. Robust resource allocation in weather data processing systems. In *International Conference on Parallel Processing Workshops (ICPP Workshops)*. IEEE, 2006.

[126] M. A. Oxley, S. Pasricha, H. J. Siegel, and A. A. Maciejewski. Energy and Deadline Constrained Robust Stochastic Static Resource Allocation. In *Parallel Processing and Applied Mathematics*, pages 761–771. Springer, 2014.

[127] F. Pinel, G. Danoy, and P. Bouvry. Evolutionary algorithm parameter tuning with sensitivity analysis. In *Security and Intelligent Information Systems*, pages 204–216. Springer, 2012.

[128] F. Pinel, B. Dorronsoro, and P. Bouvry. A new parallel asynchronous cellular genetic algorithm for scheduling in grids. In *International Parallel & Distributed Processing Symposium Workshops and Phd Forum (IPDPSW)*, pages 1–8. IEEE, 2010.

[129] F. Pinel, B. Dorronsoro, and P. Bouvry. Solving very large instances of the scheduling of independent tasks problem on the GPU. *Journal of Parallel and Distributed Computing*, 73(1):101–110, 2013.

[130] F. Pinel, B. Dorronsoro, J. E. Pecero, P. Bouvry, and S. U. Khan. A two-phase heuristic for the energy-efficient scheduling of independent tasks on computational grids. *Cluster Computing*, 16(3):421–433, 2013.

[131] F. Pinel, J. E. Pecero, P. Bouvry, and S. U. Khan. A two-phase heuristic for the scheduling of independent tasks on computational grids. In *High Performance Computing and Simulation (HPCS)*, pages 471–477, 2011.

[132] F. Pinel, J. E. Pecero, S. U. Khan, and P. Bouvry. Energy-efficient scheduling on milliclusters with performance constraints. In *International Conference on Green Computing and Communications*, pages 44–49. IEEE Computer Society, 2011.

[133] M. K. Rafsanjani and A. K. Bardsiri. A New Heuristic Approach for Scheduling Independent Tasks on Heterogeneous Computing Systems. In *International Conference on Machine Learning and Computing (ICMLC)*, 2011.

[134] Rajni and I. Chana. Bacterial foraging based hyper-heuristic for resource scheduling in grid computing. *Future Generation Computer Systems*, 29(3):751–762, 2013.

[135] B. S. Rao. Resource Allocation in Physically Distributed System using Non-Cooperative Game Theory. Master's thesis, National Institute of Technology Rourkela, 2013.

[136] B. Sahoo, S. K. Jena, and S. Mahapatra. Simulated Annealing based heuristic approach for dynamic Load balancing Problem on Heterogeneous Distributed Computing System. *Artificial Intelligent Systems and Machine Learning*, 5(3):116–124, 2013.

[137] B. Sahoo, D. Kumar, and S. K. Jena. Analysing the impact of heterogeneity with greedy resource allocation algorithms for dynamic load balancing in heterogeneous distributed computing system. *International Journal of Computer Applications*, 62(19):25–34, 2013.

[138] R. Sakellariou and H. Zhao. A hybrid heuristic for DAG scheduling on heterogeneous systems. In *International Parallel & Distributed Processing Symposium (IPDPS)*, page 111. IEEE, 2004.

[139] A. Saltelli, K. Chan, and E. M. Scott. *Sensitivity analysis*. Wiley New York, 2009.

[140] S. Sarathambekai and K. Umamaheswari. Comparison among four Modified Discrete Particle Swarm Optimization for Task Scheduling in Heterogeneous Computing Systems. *International Journal of Soft Computing and Engineering*, 3(2):371–378, 2013.

[141] A. Sewaiwar and U. Sharma. Grid scheduling: Comparative study of MACO & TABU search. *An international journal of advanced computer technology*, 3(6):825–830, 2014.

[142] Z. Shi. *Scheduling tasks with precedence constraints on heterogeneous distributed computing systems*. PhD thesis, University of Tennessee, 2006.

[143] Z. Shi, E. Jeannot, and J. J. Dongarra. Robust task scheduling in non-deterministic heterogeneous computing systems. In *International Conference on Cluster Computing*, pages 1–10. IEEE, 2006.

[144] S. Shivle, R. Castain, H. J. Siegel, A. A. Maciejewski, T. Banka, K. Chindam, S. Dussinger, P. Pichumani, P. Satyasekaran, W. Saylor, D. Sendek, J. Sousa, J. Sridharan, P. V. Sugavanam, and J. Velazco. Static mapping of subtasks in a heterogeneous ad hoc grid environment. In *International Parallel & Distributed Processing Symposium (IPDPS)*, page 110. IEEE, 2004.

[145] S. Shivle, H. J. Siegel, A. A. Maciejewski, T. Banka, K. Chindam, S. Dussinger, A. Kutruff, P. Penumarthy, P. Pichumani, P. Satyasekaran, D. Sendek, J. Sousa, J. Sridharan, P. V. Sugavanam, and J. Velazco. Mapping of subtasks with multiple versions in a heterogeneous ad hoc grid environment. In *International Workshop on Parallel and Distributed Computing*, pages 380–387. IEEE, 2004.

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

[147] H. J. Siegel and S. Ali. Techniques for mapping tasks to machines in heterogeneous computing systems. *Journal of Systems Architecture*, 46(8):627–639, 2000.

[148] J. Smith, J. Apodaca, A. A. Maciejewski, and H. J. Siegel. Batch Mode Stochastic-Based Robust Dynamic Resource Allocation in a Heterogeneous Computing System. In *Parallel and Distributed Processing Techniques and Applications (PDPTA)*, pages 263–269, 2010.

[149] J. Smith, E. K. Chong, A. A. Maciejewski, and H. J. Siegel. Stochastic-based robust dynamic resource allocation in a heterogeneous computing system. In *International Conference on Parallel Processing (ICPP)*, pages 188–195. IEEE, 2009.

[150] P. V. Sugavanam, H. J. Siegel, A. A. Maciejewski, S. A. Ali, M. Al-Otaibi, M. Aydin, K. Guru, A. Horiuchi, Y. G. Krishnamurthy, P. Lee, A. M. Mehta, M. Oltikar, R. Pichel, A. Pippin, M. Raskey, V. Shestak, and J. Zhang. Processor allocation for tasks that is robust against errors in computation time estimates. In *International Parallel & Distributed Processing Symposium (IPDPS)*. IEEE, 2005.

[151] P. V. Sugavanam, H. J. Siegel, A. A. Maciejewski, M. Oltikar, A. M. Mehta, R. Pichel, A. Horiuchi, V. Shestak, M. Al-Otaibi, Y. G. Krishnamurthy, S. A. Ali, J. Zhang, M. Aydin, P. Lee, K. Guru, M. Raskey, and A. Pippin. Robust static allocation of resources for independent tasks under makespan and dollar cost constraints. *Journal of Parallel and Distributed Computing*, 67(4):400–416, 2007.

[152] P. V. Sugavanam, H. J. Siegel, A. A. Maciejewski, J. Zhang, V. Shestak, M. Raskey, A. Pippin, R. Pichel, M. Oltikar, A. M. Mehta, P. Lee, Y. G. Krishnamurthy, A. Horiuchi, K. Guru, M. Aydin, M. Al-Otaibi, and S. A. Ali. Robust processor allocation for independent tasks when dollar cost for processors is a constraint. In *International Conference on Cluster Computing*, pages 1–10. IEEE, 2005.

[153] M. D. Theys, T. D. Braun, H. J. Siegel, A. A. Maciejewski, and Y. Kwok. Mapping tasks onto distributed heterogeneous computing systems using a genetic algorithm approach. In *Solutions to Parallel and Distributed Computing Problems: Lessons from Biological Sciences*, pages 135–178. John Wiley & Sons, New York, NY, 2001.

[154] H. Topcuoglu, S. Hariri, and M.-y. Wu. Performance-effective and low-complexity task scheduling for heterogeneous computing. *IEEE transactions on parallel and distributed systems*, 13(3):260–274, 2002.

[155] B. Uçar, C. Aykanat, K. Kaya, and M. Ikinci. Task assignment in heterogeneous computing systems. *Journal of Parallel and Distributed Computing*, 66(1):32–46, 2006.

[156] B. M. Virlet. Scheduling of Stream-Based Real-Time Applications for Heterogeneous Systems. Master's thesis, University of Illinois at Urbana-Champaign, 2010.

[157] B. M. Virlet, X. Zhou, J.-P. Giacalone, B. Kuhn, M. J. Garzaran, and D. Padua. Scheduling of Stream-Based Real-Time Applications for Heterogeneous Systems. *ACM SIGPLAN notices*, 46(5):1–10, 2011.

[158] J. Wang, B. Gong, H. Liu, S. Li, and J. Yi. Heterogeneous Computing and Grid Scheduling with Hierarchically Parallel Evolutionary Algorithms. *Journal of Computational Information Systems*, 10(8):3291–3298, 2014.

[159] R. Wisnesky. Evaluating Scheduling Algorithms on Distributed Computational Grids. In *High Performance Distributed Computing (HPDC)*. Citeseer, 2002.

[160] M.-Y. Wu and W. Shu. A high-performance mapping algorithm for heterogeneous computing systems. In *International Parallel & Distributed Processing Symposium (IPDPS)*. IEEE, 2001.

[161] F. Xhafa and A. Abraham. Meta-heuristics for grid scheduling problems. In *Metaheuristics for Scheduling in Distributed Computing Environments*, pages 1–37. Springer, 2008.

[162] F. Xhafa, J. Carretero, B. Dorronsoro, and E. Alba. A tabu search algorithm for scheduling independent jobs in computational grids. *Computing and Informatics*, 28:1001–1014, 2009.

[163] F. Xhafa and J. Kołodziej. Game-theoretic, Market and Meta-Heuristics Approaches for Modelling Scheduling and Resource Allocation in Grid Systems. In *P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC)*, pages 235–242. IEEE, 2010.

[164] J. Xiao, Y. Zhang, S. Chen, and H. Yu. An Application-Level Scheduling with Task Bundling Approach for Many-Task Computing in Heterogeneous Environments. In *Network and Parallel Computing*, pages 1–13. Springer, 2012.

[165] J. Xu, A. Y. Lam, and V. O. Li. Chemical reaction optimization for task scheduling in grid computing. *Transactions on Parallel and Distributed Systems*, 22(10):1624–1631, 2011.

[166] B. D. Young, J. Apodaca, L. D. Briceño, J. Smith, S. Pasricha, A. A. Maciejewski, H. J. Siegel, B. Khemka, S. Bahirat, A. Ramirez, and Y. Zou. Energy-constrained dynamic resource allocation in a heterogeneous computing environment. In *International Conference on Parallel Processing Workshops (ICPPW)*, pages 298–307. IEEE, 2011.

[167] B. D. Young, J. Apodaca, L. D. Briceño, J. Smith, S. Pasricha, A. A. Maciejewski, H. J. Siegel, B. Khemka, S. Bahirat, A. Ramirez, and Y. Zou. Deadline and energy constrained dynamic resource allocation in a heterogeneous computing environment. *The Journal of Supercomputing*, 63(2):326–347, 2013.

[168] B. D. Young, S. Pasricha, A. A. Maciejewski, H. J. Siegel, and J. T. Smith. Heterogeneous makespan and energy-constrained DAG scheduling. In *Workshop on Energy efficient high performance parallel and distributed computing*, pages 3–12. ACM, 2013.

[169] Y. Yu and V. K. Prasanna. Resource allocation for independent real-time tasks in heterogeneous systems for energy minimization. *Journal of Information Science and Engineering*, 19(3):433–449, 2003.

[170] A. Zarrabi and K. Samsudin. Task scheduling on computational Grids using Gravitational Search Algorithm. *Cluster Computing*, pages 1–11, 2013.

[171] W. Zheng and R. Sakellariou. A Monte-Carlo approach for full-ahead stochastic DAG Scheduling. In *International Parallel & Distributed Processing Symposium Workshops and Phd Forum (IPDPSW)*, pages 99–112. IEEE, 2012.

[172] W. Zheng and R. Sakellariou. Stochastic DAG scheduling using a Monte Carlo approach. *Journal of Parallel and Distributed Computing*, 73(12):1673–1689, 2013.

[173] P. Zhou and W. Zheng. An Efficient Biobjective Heuristic for Scheduling Workflows on Heterogeneous DVS-Enabled Processors. *Journal of Applied Mathematics*, 2014, 2014.

FEMTO-ST INSTITUTE, headquarters
15B Avenue des Montboucons - F-25030 Besançon Cedex France
Tel: (33 3) 63 08 24 00 – e-mail: contact@femto-st.fr

FEMTO-ST — AS2M: TEMIS, 24 rue Alain Savary, F-25000 Besançon France
FEMTO-ST — DISC: UFR Sciences - Route de Gray - F-25030 Besançon cedex France
FEMTO-ST — ENERGIE: Parc Technologique, 2 Av. Jean Moulin, Rue des entrepreneurs, F-90000 Belfort France
FEMTO-ST — MEC'APPLI: 24, chemin de l'épitaphe - F-25000 Besançon France
FEMTO-ST — MN2S: 15B Avenue des Montboucons - F-25030 Besançon cedex France
FEMTO-ST — OPTIQUE: 15B Avenue des Montboucons - F-25030 Besançon cedex France
FEMTO-ST — TEMPS-FREQUENCE: 26, Chemin de l'Epitaphe - F-25030 Besançon cedex France

http://www.femto-st.fr