MCMC Generation of Cost Matrices for Scheduling Performance Evaluation

Louis-Claude Canon[1], Anthony Dugois[1], Mohamad El Sayah[1], and Pierre-Cyrille Héam[1]

[1]Université Marie et Louis Pasteur, Femto-ST Intistute, CNRS

March 25, 2025

Abstract

In high performance computing, scheduling and allocating tasks to machines has long been a critical challenge, especially when dealing with heterogeneous execution costs. To design efficient algorithms and then assess their performance, many approaches have been proposed, among which simulations, which can be performed on a large variety of environments and application models. However, this technique is known to be sensitive to bias when it relies on random instances with an uncontrolled distribution. In this article, instead of designing a new optimization method, we focus on generating cost matrices to improve the empirical evaluation methodology. In particular, we use methods from the literature to provide formal guarantee on how costs matrices are distributed: we ensure a uniform distribution among the cost matrices with given task and machine heterogeneities. Although the use of randomly generated matrices has often been criticized, this new generation procedure is the first that is proven to prevent biased generation by ensuring a uniform generation with given properties. This method is relevant to assess the performance of scheduling heuristics, in particular when characterizing for which parameter values a given approach performs better than others. When applied to a makespan minimization problem, the methodology reveals when each of three efficient heuristics performs better depending on the instance heterogeneity.

1 Introduction

Empirical assessment is critical to determine the best scheduling heuristics on any parallel platform. However, the performance of any heuristic may be specific to a given parallel computer. In addition to experimentation on real platforms, simulation is an effective tool to quantify the quality of scheduling heuristics. Even though simulations provide weaker evidence, they can be performed on a large variety of environments and application models, resulting in broader conclusions. However, this technique is sensitive to bias when it relies on random instances with an uncontrolled or irrelevant distribution. For instance, in uniformly distributed random graphs, the probability that the diameter is 2 tends exponentially to 1 as the size of the graph tends to infinity [1]. Even though such instances may be sometimes of interest, they prove useless in most practical contexts. We propose a method that generates instances with a known distribution for a set of classical problems where tasks must be scheduled on machines (or processors) with heterogeneous execution costs. This is critical to the empirical validation of many new heuristics like BalSuff [2] for the problem $R||C_{\max}$ and PEFT [3] for $R|\text{prec}|C_{\max}$ in Graham's notation [4].

In this context, an instance consists of a $n \times m$ cost matrix, M, where the element of row i and column j, $M(i, j)$, represents the execution cost of task i on machine j. Like the diameter for graphs, multiple criteria characterize cost matrices. First, the heterogeneity can be determined globally with the variance of all costs, but also relatively to the rows or columns. For instance, the dispersion of the means for each row, which corresponds to the varying costs for each task, impacts the performance of some scheduling heuristics [2].

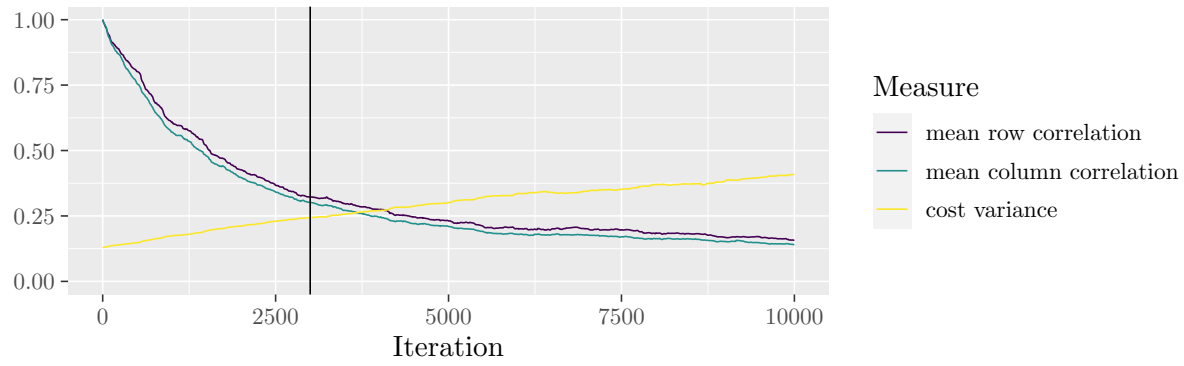

Figure 1: Cost variance and mean row and column correlations at each iteration of the shuffling method [2] when generating a 100×30 cost matrix. The shuffling method arbitrarily stops after 3 000 iterations (represented by the black vertical line).

The correlation between the rows and columns also plays an important role as it corresponds to the machines being either related or specialized (i.e. with some affinity between the tasks and the machines [5]).

Among existing methods, the shuffling one [2] starts by an initial matrix in which rows are proportional to each other (leading to unitary row and column correlations). Then, it proceeds to mix the values in the matrix such as to keep the same sum on each row and column. This ensures that the row and column heterogeneity remains stable, while the correlation decreases. However, this ad hoc approach provides no formal guarantee on the distribution of the instances. In addition, when the number of shuffles increases, the variance of all costs increases, which leads to non-interpretable results (see Figure 1).

While other methods exist, it remains an open problem to ensure a uniform distribution among the instances that have a given task and machine heterogeneity. Our contribution is to control the row and column heterogeneity while limiting the overall variance and ensuring a uniform distribution among the set of possible instances. The approach is based on a Markov Chain Monte Carlo (MCMC) process and relies on contingency tables[1]. More precisely, the proposed random generation process is based on two steps. For a given n (number of tasks), m (number of machines) and N (sum of all task costs):

1. Randomly generate the average cost of each task and the average cycle time of each machine. This random generation is performed uniformly using classical recursive algorithms [6]. To control the heterogeneity, we show how to restrict this uniform random generation to interesting classes of vectors. This step is described in Section 3.

2. Next, use a classical MCMC approach to generate the cost matrices: from an initial matrix, a random walk in the graph of contingency tables is performed. It is known (see for instance [7]) that if the Markov Chain associated with this walk is ergodic and symmetric, then the unique stationary distribution exists and is uniform. Thus, walking enough steps in the graph leads to any state with the same probability. Section 4 provides several symmetric and ergodic Markov Chains for this problem. The main contribution of this section is to extend known results for contingency tables to contingency tables with min/max constraints.

In Section 5, we use our random generation process to evaluate scheduling algorithms. The algorithms are implemented in R and Python and the related code, data and analysis are available online [2]. This article extends our previous initial work [8] mainly with a new generation method for the average costs in Section 3 and various resulting adaptations and improvements.

[1] A contingency table is a positive matrix with the sum of each row (resp. column) displayed in an additional total row (resp. column). They are usually used to show the distribution of two variables.

[2] `https://github.com/lccanon/artifact-mcmc2024`

The article's primary contribution lies in the enhanced generation method detailed in Section 3. Unlike the previous approach, our method enables control over task and machine heterogeneity, key parameters often defining the application context and influencing heuristic performance, as illustrated in Section 5 (see also Table 1). Additionally, the article offers streamlined, comprehensive proofs of the theoretical results and introduces new experiments tailored to the improved generation method, presented in Section 5.

2 Related Work

Two main methods have been used in the literature: RB (range-based) and CVB (Coefficient-of-Variation-Based) [9, 10]. Both methods follow the same principle: n vectors of m values are first generated using a uniform distribution for RB and a gamma distribution for CVB; then, each row is multiplied by a random value using the same distribution for each method. A third optional step consists in sorting each row in a submatrix, which increases the correlation of the cost matrix. However, these methods are difficult to use when generating a matrix with given heterogeneity and low correlation [5, 2].

More recently, two additional methods have been proposed for a better control of the heterogeneity: SB (shuffling-based) and NB (noise-based) [2]. In the first step of SB, one column of size n and one row of size m are generated using a gamma distribution. These two vectors are then multiplied to obtain a $n \times m$ cost matrix with a strong correlation. To reduce it, values are shuffled without changing the sum on any row or column as it is done is Section 4: selecting four elements on two distinct rows and columns (a submatrix of size 2×2); and, removing/adding the maximum quantity to two elements on the same diagonal while adding/removing the same quantity to the last two elements on the other diagonal. While NB shares the same first step, instead of shuffling the elements, it introduces randomness in the matrix by multiplying each element by a random variable with expected value one. When the size of the matrix is large, SB and NB provide some control on the heterogeneity but the distribution of the generated instances is unknown.

Finally, CB (combination-based) and CNB (correlation noise-based) have been proposed to control the correlation [5]. The correlation property, first mentioned with the introduction of RB and CVB, characterizes how each row (or column) relates to each other. With highly correlated matrices, the cost of a task is almost proportional to both its cost and the cycle time of its allocated machine. Inversely, matrices with low correlation are specialized (or unrelated) and each task has an affinity for a distinct machine. This correlation is computed both on the rows and the columns, leading to 2 measures. CB combines correlated matrices with an uncorrelated one to obtain the desired correlation. CNB is a variation of the previous NB (noised-based), which takes as input the target correlation. As for SB and NB, both methods have asymptotic guarantees on the heterogeneity and correlation of generated cost matrices, but no guarantee on how instances are distributed.

The present work relies on contingency tables, which are important data structures used in statistics for displaying the multivariate frequency distribution of variables, introduced in 1904 by K. Pearson [11]. The MCMC (Markov Chain Monte Carlo) approach is the most common way used in the literature for the uniform random generation of contingency tables (see for instance [12, 13]). Mixing time results have been provided for the particular case of $2 \times n$ sized tables in [14] and later using a coupling argument in [15]. In this restricted context, a divide-and-conquer algorithm has also been pointed out [16]. In practice, there are MCMC dedicated packages for most common programming languages: `pymc`[3] for Python, `mcmc`[4] for R, ...

More generally, random generation is a natural way for performance evaluation used, for instance, in SAT-solver competitions[5]. In a distributed computing context, it has been used for the random generation of DAG modelling task graph for parallel environments [17, 18].

[3] `https://pypi.python.org/pypi/pymc/`
[4] `https://cran.r-project.org/web/packages/mcmc/index.html`
[5] http://www.satcompetition.org/

3 Contingency vectors initialization

Considering n tasks and m machines, the first step to generate instances is to fix the average cost of each task and the average cycle time of each machine. Since n and m are fixed, instead of generating averages, we generate the sum of the costs/times on each row and column, which is equivalent. Given n, m and N (total cost), the problem becomes to generate randomly (and uniformly) two vectors $\overline{\mu} \in \mathbb{N}^n$ and $\overline{\nu} \in \mathbb{N}^m$ satisfying:

$$\sum_{i=1}^{n} \overline{\mu}(i) = \sum_{j=1}^{m} \overline{\nu}(j) = N, \tag{1}$$

where for any vector $\overline{v} = (v_1, \ldots, v_\ell) \in \mathbb{N}^\ell$, v_i is denoted $\overline{v}(i)$. To ensure a minimum or maximum cost/time, we may also want to control the range of the $\overline{v}(i)$ by imposing that for each i, $\alpha \leq \overline{v}(i) \leq \beta$, where α and β are fixed. Note that choosing $\alpha = 0$ and $\beta = N$ corresponds to the unconstrained case.

To control the heterogeneity of the tasks and machines, we ensure that vectors $\overline{\mu} \in \mathbb{N}^n$ and $\overline{\nu} \in \mathbb{N}^m$ have each a given CV[6]. Even though the description is only given for $\overline{\mu}$, we will use the same algorithm for both vectors. Using Equation (1), one has

$$\mathrm{Var}[\overline{\mu}] = \frac{1}{n} \sum_{i=1}^{n} \overline{\mu}(i)^2 - \left(\frac{N}{n}\right)^2.$$

By setting $S = \sum_{i=1}^{n} \overline{\mu}(i)^2$ and $CV(\overline{\mu}) = \frac{\sqrt{\mathrm{Var}[\overline{\mu}]}}{N/n}$, we further have that

$$CV(\overline{\mu}) = \sqrt{\frac{n}{N^2} S - 1}, \tag{2}$$

or equivalently

$$S = \frac{N^2}{n}(CV(\overline{\mu})^2 + 1). \tag{3}$$

It follows that when N and n are fixed, the value of $CV(\overline{\mu})$ depends only on the value of S.

Before describing the random generation algorithms, we proceed to a simplification to avoid handling the α constraint using a translation. Considering a vector $\overline{\mu}$ such that $\overline{\mu}(i) \geq \alpha$ for each i, we set $\overline{\mu}' = \overline{\mu} - (\alpha, \alpha, \ldots, \alpha)$, $N' = N - n\alpha$ and $S' = \sum_{i=1}^{n} \overline{\mu}'(i)^2$. Since $\mathrm{Var}[\overline{\mu}] = \mathrm{Var}[\overline{\mu}']$ and using (2) and (3), one has

$$\begin{aligned}
S' &= \frac{N'^2}{n}(CV(\overline{\mu}')^2 + 1) \\
&= \frac{N'^2}{n}\left(\frac{\mathrm{Var}[\overline{\mu}']}{(N'/n)^2} + 1\right) \\
&= \frac{N'^2}{n}\left(\frac{\mathrm{Var}[\overline{\mu}]}{(N'/n)^2} + 1\right) \\
&= \frac{N'^2}{n}\left(\frac{(CV(\overline{\mu})N/n)^2}{(N'/n)^2} + 1\right) \\
&= \frac{N'^2}{n}\left(\frac{(CV(\overline{\mu})N)^2}{N'^2} + 1\right) \\
&= \frac{1}{n}\left(CV(\overline{\mu})^2 N^2 + N'^2\right).
\end{aligned}$$

It follows there is a bijection (which is a translation) from the set of vectors $\overline{\mu}$ satisfying $\sum_{i=1}^{n} \overline{\mu}(i) = N$, $CV(\overline{\mu}) = c$ and for each i, $\alpha \leq \overline{\mu}(i) \leq \beta$, and the set of vectors $\overline{\mu}'$ satisfying $\sum_{i=1}^{n} \overline{\mu}'(i) = N - n\alpha$, $CV(\overline{\mu}') = \frac{N}{(N-n\alpha)}c$ and for each i, $0 \leq \overline{\mu}'(i) \leq \beta - \alpha$.

[6] Coefficient-of-Variation, the ratio of standard deviation to the mean, a relative measure of statistical dispersion.

Algorithm 1: Generate sequences with a fixed CV (up to ε)

Input: Integers N, n, α, β; Rational CV, ε

Output: $\overline{\mu} \in \mathbb{N}^n$ such that $\alpha \leq \overline{\mu}(i) \leq \beta$ and $(1-\varepsilon)CV \leq CV(\overline{\mu}) \leq (1+\varepsilon)CV$ and $\sum_i \overline{\mu}(i) = N$ if it is possible, $\perp$ otherwise.

begin

  $N' = N - n\alpha$

  $S_{\min} = \left\lceil \frac{1}{n}((1-\varepsilon)^2 CV^2 N^2 + N'^2) \right\rceil$

  $S_{\max} = \left\lfloor \frac{1}{n}((1+\varepsilon)^2 CV^2 N^2 + N'^2) \right\rfloor$

  for $1 \leq i \leq n$ ***and*** $0 \leq j \leq N'$ ***and*** $0 \leq s \leq S_{\max}$ **do**

    compute $g_{j,i,s}^{\beta-\alpha}$ **using (4) and (5)**

  if $N < n\alpha$ ***or*** $n\beta < N$ ***or*** $\sum_{s=S_{\min}}^{s=S_{\max}} g_{N',n,s}^{\beta-\alpha} = 0$ **then**

    return $\perp$

  pick at random $S \in [S_{\min}, S_{\max}]$ **with** $\mathbb{P}(S = \sigma) = \dfrac{g_{N',n,\sigma}^{\beta-\alpha}}{\sum_{s=S_{\min}}^{s=S_{\max}} g_{N',n,s}^{\beta-\alpha}}$

  $\overline{\mu}$ = Algorithm 2(N', n, S, $\beta - \alpha$, $g_{j,i,s}^{\beta-\alpha}$)

  return $\overline{\mu} + (\alpha, \alpha, \ldots, \alpha)$

Algorithm 2: Generate sequences with a fixed sum of squares

Input: Integers N, n, S, β, $g_{k,i,s}^{\beta}$ for $k \leq N$, $i \leq n, s \leq S$

Output: $\overline{\mu} \in \mathbb{N}^n$ such that $\overline{\mu}(i) \leq \beta$ and $\sum_i \overline{\mu}(i)^2 = S$ and $\sum_i \overline{\mu}(i) = N$ if it is possible, $\perp$ otherwise.

begin

  if $n\beta < N$ ***or*** $g_{N,n,S}^{\beta} = 0$ **then**

    return $\perp$

  $t = 0$

  $s = 0$

  for $i \in [1, \ldots, n-1]$ **do**

    pick at random $\overline{\mu}(i) \in [0, \beta]$ **with** $\mathbb{P}(\overline{\mu}(i) = k) = \dfrac{g_{N-t-k,n-i,S-s-k^2}^{\beta}}{g_{N-t,n-i+1,S-s}^{\beta}}$

    $t = t + \overline{\mu}(i)$

    $s = s + \overline{\mu}(i)^2$

  $\overline{\mu}(n) = N - t$

  return $\overline{\mu}$

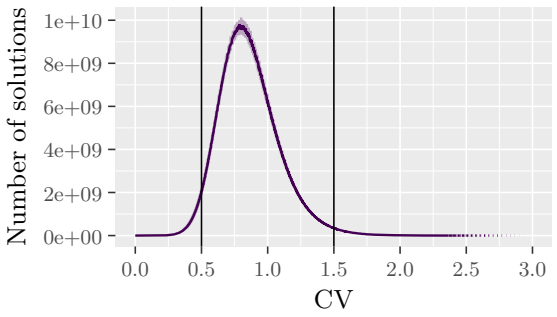

Figure 2: Number of vectors $\overline{\mu}$ for each CV with $n = 10$, $N = 100$, $\alpha = 0$ and $\beta = N$. The smoothed line is obtained by using a rolling median with 15 values (each value is set to the median of the 7 values on the left, the current one and the 7 on the right). The ribbon represents the rolling minimum and maximum with 15 values.

Based on the above results, Algorithm 1 performs the random generation of an integer vector $\overline{\mu}$ satisfying the sum constraint, the α and β constraints, and a CV constraint (up to ε). This is done using a uniform random generator (Algorithm 2) of a vector satisfying the same kind of constraints except on the CV, which is replaced by an equivalent constraint on the sum of squares.

Algorithm 2 relies on the following constraints: let n, N and S be positive integers and $G^{\beta}_{N,n,S}$ be the subset of elements $\overline{\mu}$ of $\mathbb{N}^n$ such that $N = \sum_{i=1}^{n} \overline{\mu}(i)$ and $S = \sum_{i=1}^{n} \overline{\mu}(i)^2$ and for all $1 \leq i \leq n$, $0 \leq \overline{\mu}(i) \leq \beta$ (i.e. the set of all possible vectors with values between 0 and β). Algorithm 2 uniformly generates a random vector over $G^{\beta}_{N,n,S}$ by using the pre-computed $g^{\beta}_{k,i,s}$ for $k \leq N$, $i \leq n, s \leq S$, where $g^{\beta}_{N,n,S}$ is the cardinal of $G^{\beta}_{N,n,S}$. By decomposition, one has

$$g^{\beta}_{N,n,S} = \sum_{k=0}^{\min(\beta,N)} g^{\beta}_{N-k,n-1,S-k^2}. \tag{4}$$

Moreover,

$$\begin{aligned} &g^{\beta}_{N,n,S} = 0 \text{ if } n\beta < N \text{ and,} \\ &g^{\beta}_{N,1,N^2} = 1 \text{ if } N \leq \beta \text{ and,} \\ &g^{\beta}_{N,1,S} = 0 \text{ if } N \leq \beta \text{ and } S \neq N^2. \end{aligned} \tag{5}$$

Figure 2 shows the number of vectors $\overline{\mu}$ for each CV. Note there may not always exist a vector $\overline{\mu}$ for any value of S, especially for large values. For instance, all values of S for which there exists at least one vector are even when N is even and odd otherwise (see Proposition 1).

Proposition 1. *For any vector $\overline{\mu}$, the total cost N and the sum of the square of all costs S share the same parity.*

Proof. Recall that $N = \sum_{i=1}^{n} \overline{\mu}(i)$ and $S = \sum_{i=1}^{n} \overline{\mu}(i)^2$. Then $N^2 = S + 2\sum_{i<j} \overline{\mu}(i)\overline{\mu}(j)$. We have both that N and N^2 share the same parity and the second operand of the sum is even, which concludes the proof. □

Therefore, there is at least one value with no possible vector $\overline{\mu}$ for each pair of consecutive S. This motivates using a flexible CV constraint up to ε.

The narrowness of the ribbon indicates that the rolling median is relevant to characterize how the numbers of possible vectors $\overline{\mu}$ are distributed. We observe that with $n = 10$ and $N = 100$, 96.7% of the vectors have a CV between 0.5 and 1.5. Moreover, there are really few possible values for CVs greater than 2 (less than 0.06%). Finally, by construction, the maximum CV for a vector of size $n = 10$ is $\frac{\sqrt{1/n \times N^2 - N^2/n^2}}{N/n} = \sqrt{n-1} = 3$ (one vector value to N, the others to 0).

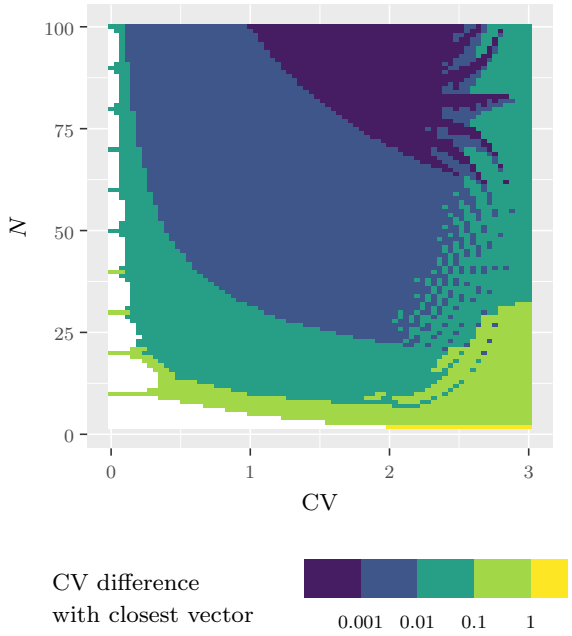

Figure 3: Minimal CV distance to the closest vector with larger CV, the same N, $n = 10$, $\alpha = 0$ and $\beta = N$.

Figure 3 depicts the minimal CV difference between vectors with the same total cost N. For instance, when $N = 50$, vectors with close but distinct CVs will differ from at least 0.001 and at most 0.01 when their CVs are close to 1. For $N = 25$, the minimal CV difference is between 0.01 and 0.1 in the same situation. Hence, the CVs of possible vectors cover a range with a better resolution with $N = 50$ than with $N = 25$.

Therefore, even though the proposed method is likely to produce a vector in most situations, one must be careful to provide a sufficiently large margin ε when the total cost N is low and the target CV is either low or high. This is further corroborated by Proposition 2, which states that the bound on the minimum difference between two CVs decreases as N increases.

Proposition 2. *Let $n \geq 2$, N and S be such that $g^{\beta}_{N,n,S} \neq 0$ and $n\beta > N$. There exists $S' \neq S$ such that $g^{\beta}_{N,n,S'} \neq 0$ and $|CV^2 - CV'^2| \leq 2n/N$ where CV and CV' are the CVs of the vectors with S and S', respectively.*

Proof. The proof relies on the following proposition.

Proposition 3. *Let $n \geq 2$, N and S be such that $g^{\beta}_{N,n,S} \neq 0$ and $n\beta > N$. There exists $S' \neq S$ such that $g^{\beta}_{N,n,S'} \neq 0$ and $|S - S'| \leq 2(\beta - 1)$.*

Proof. Let $\overline{x} = (x_1, \ldots, x_n) \in G^{\beta}_{N,n,S}$. Let $x_i = \max_k x_k$ and $x_j = \min_k x_k$.

Set $\overline{x'} = (x'_1, \ldots, x'_n)$, where $x'_i = x_i - 1$, $x'_j = x_j + 1$ and $x'_k = x_k$ for $k \neq i, j$.

Assume first $x_i > x_j$. Since $x_j < x_i \leq \beta$, then $x_j + 1 \leq \beta$.

Assume now that $x_i = x_j = N/n$. Since $n\beta > N$, then $x_j + 1 \leq \beta$.

In both cases, $\overline{x'} \in G^{\beta}_{N,n,S'}$, with $S' = \sum_{k=1} x'^2_k$. Now $|S - S'| = |x_i^2 + x_j^2 - x'^2_i - x'^2_j| = |x_i^2 - (x_i - 1)^2 + x_j^2 - (x_j + 1)^2| = 2|x_j - x_i - 1| \leq 2(\beta + 1)$. □

The proof of Proposition 2 is done by applying Equation (2) and Proposition 3 with $\beta = N$. □

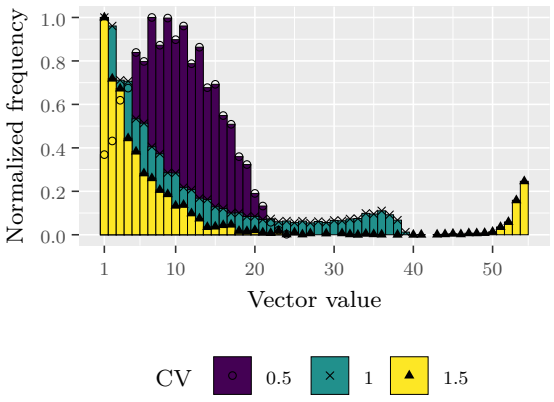

Figure 4: Distribution of values with $n = 10$, $N = 100$, $\varepsilon = 0$, $\alpha = 1$ and $\beta = N$ for 3 vectors.

Finally, Figure 4 depicts the distribution of the vector values when varying the CV for $n = 10$ and $N = 100$. The lower the CV, the narrower the distribution. An extreme case occurs when the variance is zero with all values being equal (assuming n divides N). When the CV increases, the values spread over the valid interval, until 2 modes can be identified. For the largest CV, all values except one take either the min value α or the max value β.

4 Symmetric Ergodic Markov Chains for the Random Generation

We can now generate two random vectors $\overline{\mu}$ and $\overline{\nu}$ containing the sum of each row and column with Algorithm 1. To obtain an actual matrix of costs, we use Markov Chains to generate the corresponding contingency table. Random generation using finite discrete Markov Chains can easily be explained using random walk on finite graphs. Let Ω be the finite set of all possible cost matrices (also called states) with given row and column sums: we want to sample uniformly one of its elements. However, Ω is too large to be built explicitly. The approach consists in building a directed graph whose set of vertices is Ω and whose set of edges represent all the possible transitions between any pair of states. Each edge of the graph is weighted by a probability with a classical normalization: for each vertex, the sum of the probabilities on outgoing edges is equal to 1. One can now consider random walks on this graph. A classical Markov Chain result claims that for some families of probabilistic graphs/Markov Chains, walking long enough in the graph, we have the same probability to be in each state, whatever the starting vertex of the walk [7, Theorem 4.9].

This is the case for symmetric ergodic Markov Chains [7, page 37]. Symmetric means that if there is an edge (x, y) with probability p, then the graph has an edge (y, x) with the same probability. A Markov Chain is ergodic if it is aperiodic (the greatest common divisor of the lengths of loops of the graph is 1) and if the graph is strongly connected. When there is a loop of length 1, the ergodicity issue reduces to the strongly connected problem. In general, the graph is not explicitly built and neighborhood relation is defined by a function, called a random mapping, on each state. For a general reference on finite Markov Chains, see [7].

An illustrative example is depicted on Figure 5. Starting arbitrarily from the central vertex, after one step, we are in any other vertex with probability $\frac{1}{6}$ (and with probability 0 in the central vertex since there is no self-loop on it). After two steps, we are in the central vertex with probability $\frac{1}{6}$ and in any other with probability $\frac{5}{36}$. In this simple example, one can show that after $n+1$ step, the probability to be in the central node is $p_{n+1} = \frac{1}{7}(1 - \left(\frac{-1}{6}\right)^n)$ and is $\frac{1-p_{n+1}}{6}$ for all the other nodes. All probabilities tends to $\frac{1}{7}$ when n grows.

This section is dedicated to building symmetric and ergodic Markov Chains for our problem. In Section 4.1, we define the set Ω that is interesting for cost matrices (3 subsets of interest are also provided). In Section 4.2, Markov Chains are proposed using a dedicated random mapping and are proved to be symmetric and ergodic.

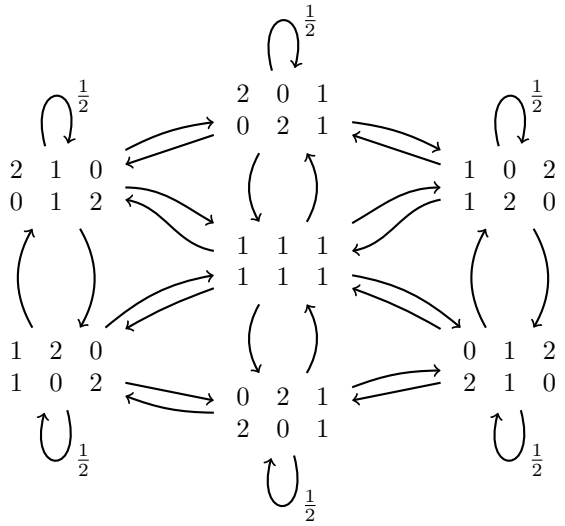

Figure 5: Example of the underlying graph of a Markov Chain when the sum of each row is three and the sum of column is two. Unless otherwise stated, each transition probability is $\frac{1}{6}$.

Finally, in Section 4.3, we improve the Markov Chains to reduce the mixing time (i.e. the number of steps required to be close to the uniform distribution is smaller).

Recall that N, n, m are positive integers and that $\overline{\mu} \in \mathbb{N}^n$ and $\overline{\nu} \in \mathbb{N}^m$ satisfy Equation (1).

4.1 Contingency Tables

In this section, we define the state space of the Markov Chains. We consider contingency tables with fixed sums on rows and columns. We also introduce min/max constraints to control the variance of the costs. We denote by $\Omega^N_{n,m}(\overline{\mu}, \overline{\nu})$ the set of positive $n \times m$ matrices M of integers such that for every $i \in \{1, \ldots, n\}$ and every $j \in \{1, \ldots, m\}$,

$$\sum_{k=1}^{m} M(i,k) = \overline{\mu}(i) \quad \text{and} \quad \sum_{k=1}^{n} M(k,j) = \overline{\nu}(j) \tag{6}$$

For example, the matrix

$$M_{\text{exa}} = \begin{pmatrix} 3 & 2 & 5 \\ 1 & 0 & 10 \end{pmatrix}$$

is in $\Omega_{2,3}(\overline{\mu}_{\text{exa}}, \overline{\nu}_{\text{exa}})$, where $\overline{\mu}_{\text{exa}} = (10, 11)$ and $\overline{\nu}_{\text{exa}} = (4, 2, 15)$.

The first restriction consists in having a global minimal value α and a maximal global value β on the considered matrices. Let α, β be positive integers such that $\alpha \leq \beta$. We denote by $\Omega^N_{n,m}(\overline{\mu}, \overline{\nu})[\alpha, \beta]$ the subset of $\Omega^N_{n,m}(\overline{\mu}, \overline{\nu})$ of matrices M such that for all $1 \leq i \leq n$ and $1 \leq j \leq m$, $\alpha \leq M(i,j) \leq \beta$. For example, $M_{\text{exa}} \in \Omega_{2,3}(\overline{\mu}_{\text{exa}}, \overline{\nu}_{\text{exa}})[0, 12]$. Moreover, according to Equation (6), one has

$$\begin{aligned} \Omega^N_{n,m}(\overline{\mu}, \overline{\nu}) &= \Omega^N_{n,m}(\overline{\mu}, \overline{\nu})[0, N] \\ &= \Omega^N_{n,m}(\overline{\mu}, \overline{\nu})[0, \min(\max_{1 \leq k \leq m} \overline{\mu}(k), \\ &\qquad \max_{1 \leq k \leq n} \overline{\nu}(k))]. \end{aligned} \tag{7}$$

Now we consider min/max constraints on each row and each line. Let $\overline{\alpha}_r, \overline{\beta}_r \in \mathbb{N}^n$ and $\overline{\alpha}_c, \overline{\beta}_c \in \mathbb{N}^m$, such that for all $1 \leq i \leq n$ and $1 \leq j \leq m$, $\overline{\alpha}_r(i) \leq \overline{\beta}_r(i)$ and $\overline{\alpha}_c(j) \leq \overline{\beta}_c(j)$. We denote by $\Omega^N_{n,m}(\overline{\mu}, \overline{\nu})[\overline{\alpha}_r, \overline{\beta}_r, \overline{\alpha}_c, \overline{\beta}_c]$ the subset of $\Omega^N_{n,m}(\overline{\mu}, \overline{\nu})$ of matrices M satisfying: for all $1 \leq i \leq n$ and $1 \leq j \leq m$, $\overline{\alpha}_r(i) \leq M(i,j) \leq \overline{\beta}_r(i)$

and $\overline{\alpha}_c(j) \leq M(i,j) \leq \overline{\beta}_c(j)$. For instance, $M_{\text{exa}} \in \Omega_{2,3}(\overline{\mu}_{\text{exa}}, \overline{\nu}_{\text{exa}})[(2,0),(5,10),(1,0,5),(3,2,10)]$. Using Equation (6), one has for every $\alpha, \beta \in \mathbb{N}$,

$$\Omega^N_{n,m}(\overline{\mu},\overline{\nu})[\alpha,\beta] = \Omega^N_{n,m}(\overline{\mu},\overline{\nu})[(\alpha,\ldots,\alpha),(\beta,\ldots,\beta),\\ (\alpha,\ldots,\alpha),(\beta,\ldots,\beta)]. \tag{8}$$

To finish, the more general constrained case, where min/max are defined for each element of the matrices. Let $A_{\min}$ and $B_{\max}$ be two $n \times m$ matrices of positive integers such that for all $1 \leq i \leq n$ and $1 \leq j \leq m$, $A_{\min}(i,j) \leq B_{\max}(i,j)$. We denote by $\Omega^N_{n,m}(\overline{\mu},\overline{\nu})[A_{\min}, B_{\max}]$ the subset of $\Omega^N_{n,m}(\overline{\mu},\overline{\nu})$ of matrices M such that for all $1 \leq i \leq n$ and $1 \leq j \leq m$, $A_{\min}(i,j) \leq M(i,j) \leq B_{\max}(i,j)$. For instance, one has $M_{\text{exa}} \in \Omega^N_{n,m}(\overline{\mu},\overline{\nu})[A_{\text{exa}}, B_{\text{exa}}]$, with

$$A_{\text{exa}} = \begin{pmatrix} 3 & 2 & 4 \\ 0 & 0 & 5 \end{pmatrix} \quad \text{and} \quad B_{\text{exa}} = \begin{pmatrix} 5 & 4 & 6 \\ 1 & 3 & 12 \end{pmatrix}.$$

For every $\overline{\alpha}_r, \overline{\beta}_r \in \mathbb{N}^n$ and $\overline{\alpha}_c, \overline{\beta}_c \in \mathbb{N}^m$, one has

$$\Omega^N_{n,m}(\overline{\mu},\overline{\nu})[\overline{\alpha}_r,\overline{\beta}_r,\overline{\alpha}_c,\overline{\beta}_c] = \Omega^N_{n,m}(\overline{\mu},\overline{\nu})[A,B], \tag{9}$$

where $A(i,j) = \max\{\overline{\alpha}_r(i), \overline{\alpha}_c(j)\}$ and $B(i,j) = \min\{\overline{\beta}_r(i), \overline{\beta}_c(j)\}$.

4.2 Markov Chains

As explained before, the random generation process is based on symmetric ergodic Markov Chains. This section is dedicated to define such chains on state spaces of the form $\Omega^N_{n,m}(\overline{\mu},\overline{\nu})$, $\Omega^N_{n,m}(\overline{\mu},\overline{\nu})[\alpha,\beta]$, $\Omega^N_{n,m}(\overline{\mu},\overline{\nu})[\overline{\alpha}_c,\overline{\beta}_c,\overline{\alpha}_r,\overline{\beta}_r]$ and $\Omega^N_{n,m}(\overline{\mu},\overline{\nu})[A_{\min}, B_{\max}]$. According to Equations (7), (8) and (9), working on $\Omega^N_{n,m}(\overline{\mu},\overline{\nu})[A_{\min}, B_{\max}]$ is sufficient. To simplify the notation, let us denote by Ω the set $\Omega^N_{n,m}(\overline{\mu},\overline{\nu})[A_{\min}, B_{\max}]$.

For any $1 \leq i_0, i_1 \leq n$ and any $1 \leq j_0, j_1, \leq m$, such that $i_0 \neq i_1$ and $j_0 \neq j_1$, we denote by Δ_{i_0,j_0,i_1,j_1} the $n \times m$ matrix defined by $\Delta(i_0,j_0) = \Delta(i_1,j_1) = 1$, $\Delta(i_0,j_1) = \Delta(i_1,j_0) = -1$, and $\Delta(i,j) = 0$ otherwise. For instance, for $n = 3$ and $m = 4$ one has

$$\Delta_{1,1,2,3} = \begin{pmatrix} 1 & 0 & -1 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

Tuple (i_0, j_0, i_1, j_1) is used as follows to shuffle a cost matrix and to transit from one state to another in the markov chain: Δ_{i_0,i_1,j_0,j_1} is added to the current matrix, which preserves the row and column sums. Formally, let $\mathcal{K} = \{(i_0,j_0,i_1,j_1) \mid i_0 \neq i_1, j_0 \neq j_1, 1 \leq i_0, i_1 \leq n, 1 \leq j_0, j_1 \leq m\}$ be the set of all possible tuples. Let f be the mapping function from $\Omega \times \mathcal{K}$ to Ω defined by $f(M,(i_0,j_0,i_1,j_1)) = M + \Delta_{(i_0,j_0,i_1,j_1)}$ if $M + \Delta_{(i_0,j_0,i_1,j_1)} \in \Omega$ and M otherwise. The mapping is called at each iteration, changing the instance until it is sufficiently shuffled.

We consider the Markov chain $\mathcal{M}$ defined on Ω by the random mapping $f(\cdot, U_{\mathcal{K}})$, where $U_{\mathcal{K}}$ is a uniform random variable on K.

The following result gives the properties of the markov chain and is an extension of a similar result [12] on $\Omega^N_{n,m}(\overline{\mu},\overline{\nu})$. The difficulty is to prove that the underlying graph is strongly connected since the constraints are hindering the moves.

Theorem 4. *The Markov Chain $\mathcal{M}$ is symmetric and ergodic with $A_{\min}$ and $B_{\max}$ such that for all $1 \leq i \leq n$ and $1 \leq j \leq m$, $A_{\min}(i,j) < B_{\max}(i,j)$.*

The proof of Theorem 4 is based on Lemmas 6 and 7, which are first presented below.

Definition 5. *Let A and B be two elements of Ω. A finite sequence $u_1 = (i_1,j_1), \ldots, u_r = (i_r,j_r)$ of pairs of indices in $\{1,\ldots,n\} \times \{1,\ldots,m\}$ is called a stair sequence for A and B if it satisfies the following properties:*

1. $r \geq 4$,

2. *for any* $k \neq \ell$, *then* $u_k \neq u_\ell$ *(all pairs are distinct),*

3. *for any odd* k *such that* $1 \leq k < r$, *then* $i_k = i_{k+1}$ *and* $A(i_k, j_k) > B(i_k, j_k)$,

4. *for any even* k *such that* $1 \leq k < r$, *then* $j_k = j_{k+1}$ *and* $A(i_k, j_k) < B(i_k, j_k)$,

5. r *is even and* $j_r = j_1$,

Consider, for instance, the matrices

$$A_1 = \begin{pmatrix} 3 & 0 & 0 & 0 & 7 \\ 7 & 4 & 0 & 0 & 0 \\ 0 & 7 & 5 & 0 & 0 \\ 0 & 0 & 7 & 6 & 0 \\ 0 & 0 & 0 & 7 & 5 \end{pmatrix} B_1 = \begin{pmatrix} 2 & 1 & 0 & 0 & 7 \\ 7 & 3 & 1 & 0 & 0 \\ 0 & 7 & 4 & 1 & 0 \\ 0 & 0 & 7 & 5 & 1 \\ 1 & 0 & 0 & 7 & 4 \end{pmatrix}.$$

The sequence $(1,1),(1,2),(2,2),(2,3),(3,3),(3,4),(4,4),(4,5),(5,5),(5,1)$ is a stair sequence for A_1 and B_1.

Lemma 6. *Let* A *and* B *be two distinct elements of* Ω. *There exists a stair sequence for* A *and* B.

Proof. The proof is by construction. Since A and B are distinct, using the constraints on the sums of rows and columns, there exists a pair of indices $u_1 = (i_1, j_1)$ such that $A(i_1, j_1) > B(i_1, j_1)$. Now using the sum constraint on row i_1, there exists $j_2 \neq j_1$ such that $B(i_1, j_2) < A(i_1, j_2)$. Set $u_2 = (i_1, j_2)$. Similarly, using the sum constraint on column j_2, there exists $i_3 \neq i_1$ such that $A(i_3, j_2) > B(i_3, j_2)$. Set $u_3 = (i_3, j_2)$. Similarly, by the constraint on row i_3, there exists $j_4 \neq j_2$ such that $A(i_3, j_4) < B(i_3, j_4)$. Set $u_4 = (i_3, j_4)$. At this step, u_1, u_2, u_3, u_4 are pairwise distinct.

If $j_4 = j_1$, then u_1, u_2, u_3, u_4 is a stair sequence for A and B. Otherwise, by the j_4-column constraint, there exists $i_5 \neq i_3$ such that $B(i_5, j_4) > A(i_5, j_4)$. Now, one can continue the construction until the first step r we get either $i_r = i_s$ or $j_r = j_s$ with $s < r$ (this step exists since the set of possible indexes is finite). Note that we consider the smallest s for which this is case.

- If $i_r = i_s$, $s < r$, the sequence $u_1, u_2, \ldots, u_r$ satisfies Conditions 2., 3. and 4. of Definition 5. Moreover both r and s are odd because the indices of i are always off by construction. The sequence $u_s, \ldots, u_r$ satisfies Conditions 2. to 5. of Definition 5. Since $r > s$ and by construction, $r - s > 4$. If follows that the sequence $u_r, u_{s+1}, \ldots, u_{r-1}$ is a stair sequence for A and B (we shift the sequence to satisfy Condition 3: $i_r = i_{s+1}$ and $A(i_r, j_{r-1}) > B(i_s, j_{s+1})$).
- If $j_r = j_s$, then both r and s are even. The sequence $u_{s+1}, \ldots, u_r$ satisfies Conditions 1. to 5. of Definition 5 and is therefore a stair sequence for A and B.

□

Given two $n \times m$ matrices A and B, the distance from A to B, denoted $d(A, B)$, is defined by:

$$d(A, B) = \sum_{i=1}^{n} \sum_{j=1}^{m} |A(i, j) - B(i, j)|.$$

Lemma 7. *Let* A *and* B *be two distinct elements of* Ω *with* $A_{\min}$ *and* $B_{\max}$ *such that for all* $1 \leq i \leq n$ *and* $1 \leq j \leq m$, $A_{\min}(i, j) < B_{\max}(i, j)$. *There exists* $C \in \Omega$ *such that* $d(C, B) < d(A, B)$ *and tuples* $t_1, \ldots, t_k$ *such that* $C = f(\ldots f(f(A, t_1), t_2) \ldots, t_k)$ *and for every* $\ell \leq k$, $f(\ldots f(f(A, t_1), t_2) \ldots, t_\ell) \in \Omega$.

Proof. By Lemma 6, there exists a stair sequence $u_1, \ldots, u_r$ for A and B. Without loss of generality (using a permutation of rows and columns), we may assume that $u_{2k+1} = (k, k)$ and $u_{2k} = (k, k+1)$ for $k < \frac{r}{2}$, and $u_r = (\frac{r}{2}, 1)$.

To illustrate the proof, we introduce some $\frac{r}{2} \times \frac{r}{2}$ matrix M over $\{+, -, \min, \max\}$, called *difference matrices*, such that: if $M(i,j) = +$, then $A(i,j) > B(i,j)$; if $M(i,j) = -$, then $A(i,j) < B(i,j)$; if $M(i,j) = \min$, then $A(i,j) = A_{\min}(i,j)$; and if $M(i,j) = \max$, then $A(i,j) = B_{\max}(i,j)$. There may be multiple difference matrices for the same pair of matrices A and B.

Considering for instance the matrices A_1 and B_1 defined before, with a global minimum equal to 0 and global maximum equal to 7, a difference matrix is

$$\begin{pmatrix} + & - & \min & \min & \max \\ \max & + & - & \min & \min \\ \min & \max & + & - & \min \\ \min & \min & \max & + & - \\ - & \min & \min & \max & + \end{pmatrix}.$$

Note that there may exist several difference matrices since, for instance, some + might be replaced by a max.

The proof investigates several cases:

Case 0: If $r = 4$, then $k = 1$ and $t_1 = (2, 1, 1, 2)$ works. Indeed, since $B_{\max}(i,j) \geq A(1,1) > B(1,1) \geq A_{\min}(i,j)$, one has $A_{\min}(1,1) \leq A(1,1) - 1 < B_{\max}(1,1)$. Similarly, $A_{\min}(2,1) < A(2,1) + 1 \leq B_{\max}(2,1)$, $A_{\min}(1,2) < A(1,2) + 1 \leq B_{\max}(1,2)$ and $A_{\min}(2,2) \leq A(2,2) - 1 < B_{\max}(2,2)$. It follows that $C = f(A, (2,1,1,2)) \in \Omega$ and $d(C,B) = d(A,B) - 4 < d(A,B)$. In this case, the following matrix is a difference matrix:

$$\begin{pmatrix} + & - \\ - & + \end{pmatrix}.$$

Case 1: If Case 0 does not hold ($r > 4$) and if $A(1, \frac{r}{2}) < B_{\max}(1, \frac{r}{2})$, then, $k = 1$ and $t_1 = (1, \frac{r}{2}, \frac{r}{2}, 1)$ works. One has $d(f(A,t_1), B) = d(A,B) - 4$ if $A(1, \frac{r}{2}) < B(1, \frac{r}{2})$, and $d(f(A,t_1), B) = d(A,B) - 2$ otherwise. In this case, the following matrix is a difference matrix:

$$\begin{pmatrix} + & - & & & & & A(1,\frac{r}{2}) \\ & + & - & & & & \\ & & + & - & & & \\ & & & + & \ddots & & \\ & & & & \ddots & - & \\ & & & & & + & - \\ - & & & & & & + \end{pmatrix}.$$

Case 2: If Cases 0 to 1 do not hold, $r > 4$ and $A(1, \frac{r}{2}) = B_{\max}(1, \frac{r}{2})$. Thus, $i_0 = \max\{i \mid 1 \leq i \leq \frac{r}{2} - 2 \text{ and } A(i, \frac{r}{2}) > A_{\min}(i, \frac{r}{2})\}$ exists (it is at least the case for $i = 1$ because $A(1, \frac{r}{2}) = B_{\max}(1, \frac{r}{2}) > A_{\min}(1, \frac{r}{2})$). In this case, $t_1 = (i_0, i_0 + 1, i_0 + 1, \frac{r}{2}), t_2 = (i_0 + 1, i_0 + 2, i_0 + 2, \frac{r}{2}), \ldots, t_{\frac{r}{2} - 1 - i_0} = (\frac{r}{2} - 2, \frac{r}{2} - 1, \frac{r}{2} - 1, \frac{r}{2})$ works because $A(i, \frac{r}{2}) = A_{\min}(i, \frac{r}{2}) < B_{\max}(i, \frac{r}{2})$ for $i_0 < i \leq \frac{r}{2} - 2$. With $C = f(\ldots f(f(A,t_1), t_2) \ldots, t_{\frac{r}{2} - 1 - i_0})$, one has $d(C,B) = d(A,B) - 2 \times (\frac{r}{2} - i_0 - 1) - 2$ if $A(i_0, \frac{r}{2}) > B(i_0, \frac{r}{2})$, and $d(C,B) = d(A,B) - 2 \times (\frac{r}{2} - i_0 - 1)$ otherwise. Moreover, for every $\ell \leq \frac{r}{2} - 2 - i_0$,

$f(\dots f(f(A,t_1),t_2)\dots,t_\ell) \in \Omega$. In this case, the following matrix is a difference matrix:

$$\begin{pmatrix} + & - & & & & & \max \\ & + & - & & & & \\ & & + & - & & & A(i_0,\frac{r}{2}) \\ & & & + & \ddots & & \\ & & & & \ddots & - & \\ & & & & & + & - \\ - & & & & & & + \end{pmatrix}.$$

□

One can now prove Theorem 4.

Proof. If $A = f(B,(i_0,j_0,i_1,j_1))$, then $B = f(A,(i_1,j_1,i_0,j_0))$, proving that the Markov Chain is symmetric.

Let $A_0 \in \Omega$. We define the sequence $(A_k)_{k\geq 0}$ by $A_{k+1} = f(A_k,(1,1,2,2))$. The sequence $A_k(1,2)$ is decreasing and positive. Therefore, one can define the smallest index k_0 such that $A_{k_0}(1,2) = A_{k_0+1}(1,2)$. By construction, one also has $A_{k_0} = A_{k_0+1}$. It follows that the Markov Chain is aperiodic.

Since d is a distance, irreducibility is a direct consequence of Lemma 7. □

Consider the two matrices A_1 and B_1 defined previously with $B_{\max}$ containing only the value 7. Case 4 of the proof can be applied. One has $t_1 = (1,2,2,5)$ and

$$f(A_1,t_1) = A_2 = \begin{pmatrix} 3 & 1 & 0 & 0 & 6 \\ 7 & 3 & 0 & 0 & 1 \\ 0 & 7 & 5 & 0 & 0 \\ 0 & 0 & 7 & 6 & 0 \\ 0 & 0 & 0 & 7 & 5 \end{pmatrix}.$$

Next, $t_2 = (2,3,3,5)$ and

$$f(A_2,t_2) = A_3 = \begin{pmatrix} 3 & 1 & 0 & 0 & 6 \\ 7 & 3 & 1 & 0 & 0 \\ 0 & 7 & 4 & 0 & 1 \\ 0 & 0 & 7 & 6 & 0 \\ 0 & 0 & 0 & 7 & 5 \end{pmatrix}.$$

We have $t_3 = (3,4,4,5)$ and

$$f(A_3,t_3) = A_4 = \begin{pmatrix} 3 & 1 & 0 & 0 & 6 \\ 7 & 3 & 1 & 0 & 0 \\ 0 & 7 & 4 & 1 & 0 \\ 0 & 0 & 7 & 5 & 1 \\ 0 & 0 & 0 & 7 & 5 \end{pmatrix}.$$

Finally, $f(A_4,(5,1,1,5)) = B_1$ (Case 0): there is a path from A_1 to B_1 and, since the chain is symmetric, from B_1 to A_1.

4.3 Rapidly Mixing Chains

The chain $\mathcal{M}$ can be classically modified to mix faster: once an element of $\mathcal{K}$ is picked up, rather than changing each element by $+1$ or -1, each one is modified by $+a$ or $-a$, where a is picked uniformly to respect the constraints of the matrix. This approach, used for instance in [15], allows moving faster, particularly for large N's.

Moving in $\Omega^N_{n,m}(\overline{\mu},\overline{\nu})$, from matrix M, while (i_0, j_0, i_1, j_1) has been picked in $\mathcal{K}$, a is uniformly chosen such that $a \leq \min\{M(i_0, j_1), M(i_1, j_0)\}$ to keep positive elements in the matrix. It can be generalized for constrained Markov Chains. For instance, in $\Omega^N_{n,m}(\overline{\mu},\overline{\nu})[\alpha,\beta]$, one has

$$a \leq \min\{\alpha - M(i_0, j_0), \alpha - M(i_1, j_1), \\ M(i_0, j_1) - \beta, M(i_1, j_0) - \beta\}.$$

This approach is used in the experiments described in Section 5. Before covering them, we can determine the time complexities of the overall proposed solution to generate multiple instances.

4.4 Time Complexity

Generating the vectors involves a pre-computed table, the $g^{\beta}_{N,n,S}$, that must be computed only once for all instances. It requires $O(\varepsilon^2 N^4)$ steps where N is the sum of the vector values and ε the allowed margin for the CV constraint. Even though this step should only be performed once, it limits the sum of elements in the vector N to a few hundreds in practice. In the experiments, we multiply each obtained value by 10 to overcome this limitation by choosing a lower N. Since the Markov Chain will shuffle the values, the final costs will not be significantly impacted by this approach.

Then, for each instance, two vectors must be computed, which takes $O(\varepsilon^2 N^2 + nN)$ steps. Converting the two vectors into an initial matrix for the Markov Chain takes $O(nm)$ steps. Then, each iteration takes a constant time because only four elements of the matrix are changed each time. Therefore, it takes $O(P)$ operations where P is the number of iterations.

Overall, the time complexity for generating the table with pre-computed values is $O(\varepsilon^2 N^4)$ and the time complexity for then generating each cost matrix is $O(\varepsilon^2 N^2 + (n+m)N + nm + P)$.

5 Performance Evaluation of Scheduling Algorithms

This section studies the effect of the constraints on the matrix properties (Section 5.1) and on the performance of some scheduling heuristics from the literature (Section 5.2).

We use matrices of size 20×10 with non-zero cost by using $\alpha \geq m$ for $\overline{\mu}$, $\alpha \geq n$ for $\overline{\nu}$ and a matrix $A_{\min}$ containing only ones. Even though this size might appear small, there are often tasks and machines of similar type. This size corresponds to a platform with 20 categories of task and 10 categories of machines. We rely on previous estimations [8] for the convergence time of the Markov Chain depending on the size of the cost matrix in the absence of constraints on the vectors (α and β) and on the matrix ($A_{\min}$ and $B_{\max}$). We assume that the convergence time does not strongly depend on the constraints. Moreover, we inflate the number of iterations, i.e. to 50 000, for safety, starting from the proportional matrix[7].

5.1 Constraints Effect on Cost Matrix Properties

Figure 6 show how the CV set for $\overline{\mu}$ and $\overline{\nu}$ influences the matrix properties. On the left of the plot, only $\overline{\nu}$ is constrained, in the center only $\overline{\mu}$ and in the right, both $\overline{\mu}$ and $\overline{\nu}$. Each row is dedicated to a property from the CV to the column correlation, with the inclusion of the $\overline{\mu}$ and $\overline{\nu}$ CV, which also serve as the parameters.

The heterogeneity of a cost matrix can be defined in two ways [2]: using either the CV of $\overline{\mu}$ and $\overline{\nu}$, or using the mean of the row and column CVs (third and fourth row). For instance, a cost matrix has a low heterogeneity on the task costs either if the CV of $\overline{\mu}$ or the mean of the column CVs is low. Although constraining $\overline{\mu}$ and $\overline{\nu}$ (by setting low CV) controls perfectly the former kind of heterogeneity, the latter only changes marginally. Moreover, considering the former kind again (the CV of $\overline{\mu}$ and $\overline{\nu}$), we see a relative independence between the row and column heterogeneity: controlling one has no effect on the other.

Figure 7 shows the impact of constraining both $\overline{\mu}$ and $\overline{\nu}$ with distinct values. As expected, the cost CV increases as both parameters increase. Moreover, the mean column (resp. row) CV increases as $CV_{\overline{\mu}}$ (resp.

[7] In a proportional matrix, rows (and columns) are proportional to each other.

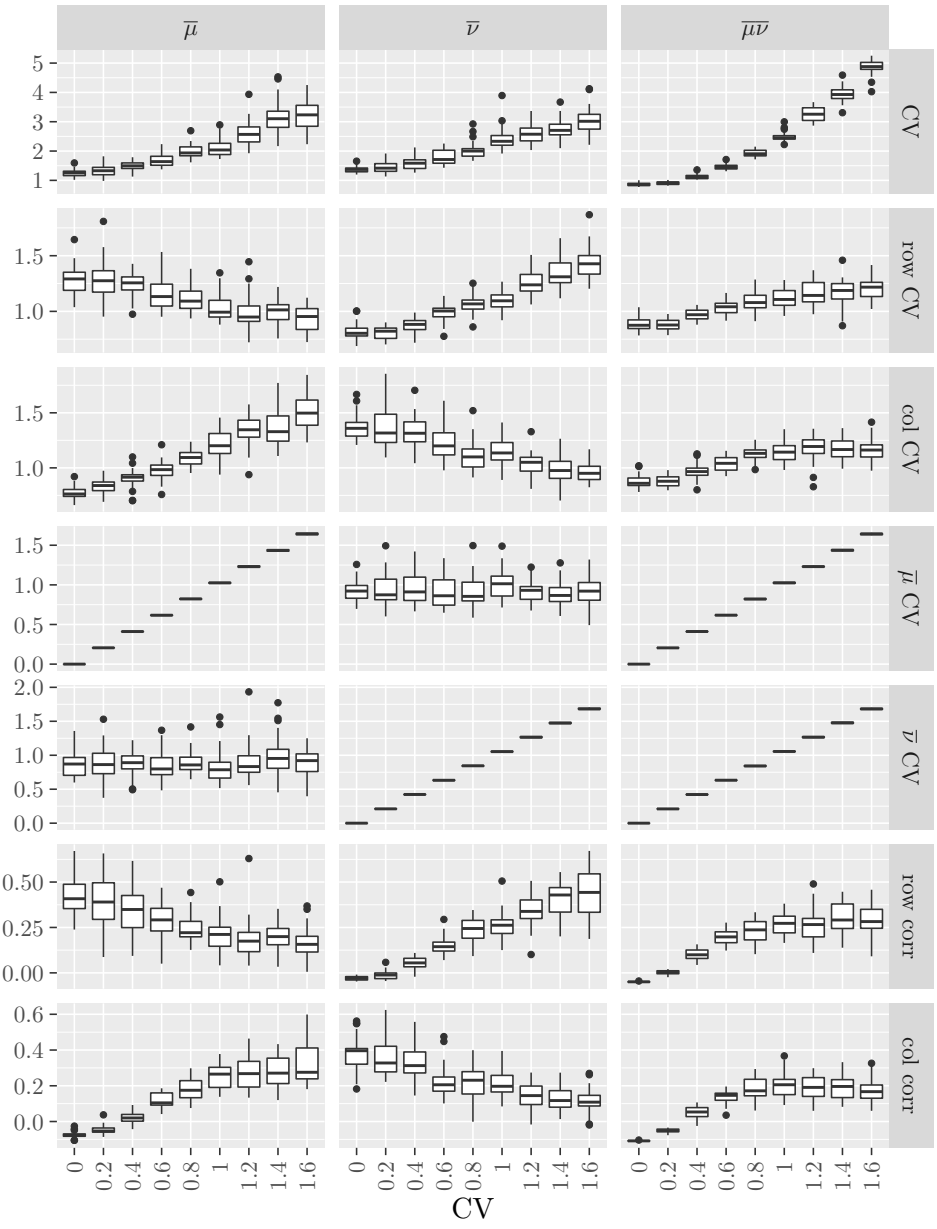

Figure 6: Matrix properties after 50 000 iterations with a proportional 20×10 matrix. The base vectors ($\overline{\mu}$ and $\overline{\nu}$) of the matrix are generated with a CV set for either one or both of them, and with $N = 400$, $\varepsilon = 0.01$, $\alpha = 1$ for the row sums ($\overline{\mu}$) and $\alpha = 2$ for the column sums ($\overline{\nu}$). The base vectors are multiplied by 10 thereafter (giving $N = 4\,000$ for instance). Each matrix contains non-zero costs. Each boxplot corresponds to 30 matrices, each based on distinct row and column sums.

$CV_{\overline{\nu}}$) increases, which indicates an association between both kinds of heterogeneity. Note that when increasing either of the heterogeneity parameters ($\overline{\mu}$ or $\overline{\nu}$ but not both), row or column correlations tend to increase.

5.2 Constraints Effect on Scheduling Algorithms

Generating random matrices with parameterized constraints allows the assessment of existing scheduling algorithms in different contexts. In this section, we focus on the impact of cost matrix properties on the performance of three heuristics for the problem denoted $R||C_{\max}$. This problem consists in assigning a set of independent tasks to machines such that the *makespan* (i.e. maximum completion time on any machine) is minimized. The cost of any task on any machine is provided by the cost matrix and the completion time on any machine is the sum of the costs of all tasks assigned to it.

The heuristics we consider constitute a diversified selection in terms of principle and cost among the numerous heuristics that have been proposed for this problem:

- *BalSuff*, an efficient heuristic [2] that balances the load by migrating iteratively each task.
- *HLPT*, Heterogeneous-Longest-Processing-Time, iteratively assigns the longest task to the machine with minimum completion time in $O(nm + n\log(n))$ steps. This is a natural extension of LPT [19] and variant of HEFT [20] in which the considered cost for each task is its minimal one.
- *EFT*, Earliest-Finish-Time, (or MinMin) is a classic principle, which iteratively assigns each task by selecting the task that finishes the earliest on any machine. Its time complexity is $O(n^2m)$.

To limit the heterogeneity according to both definitions proposed in [2], we constraint the matrix with $A_{\min}$ and $B_{\max}$ to limit how much the cost matrix can deviate from an ideal fractional proportional matrix. The constraint on the matrix is performed with a parameter $\lambda \in (0, 1)$. With this constraint, the matrix is similar to a matrix M with $M(i, j) = \frac{\overline{\mu}(i)\times\overline{\nu}(j)}{N}$ (a proportional fractional matrix) when this parameter is one. Note that when $\lambda = 1$, $A_{\min} = B_{\max}$ and Theorem 4 does not guarantee the convergence of the MCMC. This is however not an issue because there is a single possible cost matrix in each of these cases.

We selected three scenarios that represent some extremes in terms of parameters: $CV = 1.5$ and $\lambda = 0$ for a large cost heterogeneity; $CV = 0$ and $\lambda = 0.75$ for a matrix with low cost heterogeneity that is close to the proportional one; a proportional matrix ($\lambda = 1$) with no constraint on the heterogeneity. Figure 8 depicts the results: for each scenario and matrix, the makespan for each heuristic was divided by the best one among the three. Table 1 provides additional statistics. All heuristics exhibit different behaviors depending on the scenario. BalSuff outperforms its competitors except in the last case, where it is outperformed by HLPT. Finally, EFT performs poorly except when $CV = 0$ and $\lambda = 0.75$. In this case, tasks are similar and it relates to the problem $Q|p_i = 1|C_{\max}$. These instances, for which the row correlation is high and column correlation is low, have been shown to be the easiest for EFT [5].

CV	λ	Heuristic	Min	Max	Median	Mean	Std. dev.	Coeff. of var.
0	0.75	BalSuff	1.00	1.09	1.00	1.01	0.03	0.03
0	0.75	HLPT	1.00	1.42	1.21	1.22	0.13	0.11
0	0.75	EFT	1.00	1.25	1.06	1.09	0.09	0.08
1.5	0	BalSuff	1.00	1.33	1.00	1.02	0.07	0.07
1.5	0	HLPT	1.00	1.40	1.25	1.19	0.16	0.13
1.5	0	EFT	1.00	1.80	1.33	1.25	0.20	0.16
NA	1	BalSuff	1.00	1.36	1.07	1.10	0.11	0.10
NA	1	HLPT	1.00	1.38	1.00	1.06	0.10	0.09
NA	1	EFT	1.00	1.83	1.36	1.37	0.21	0.15

Table 1: Descriptive statistics for each heuristic.

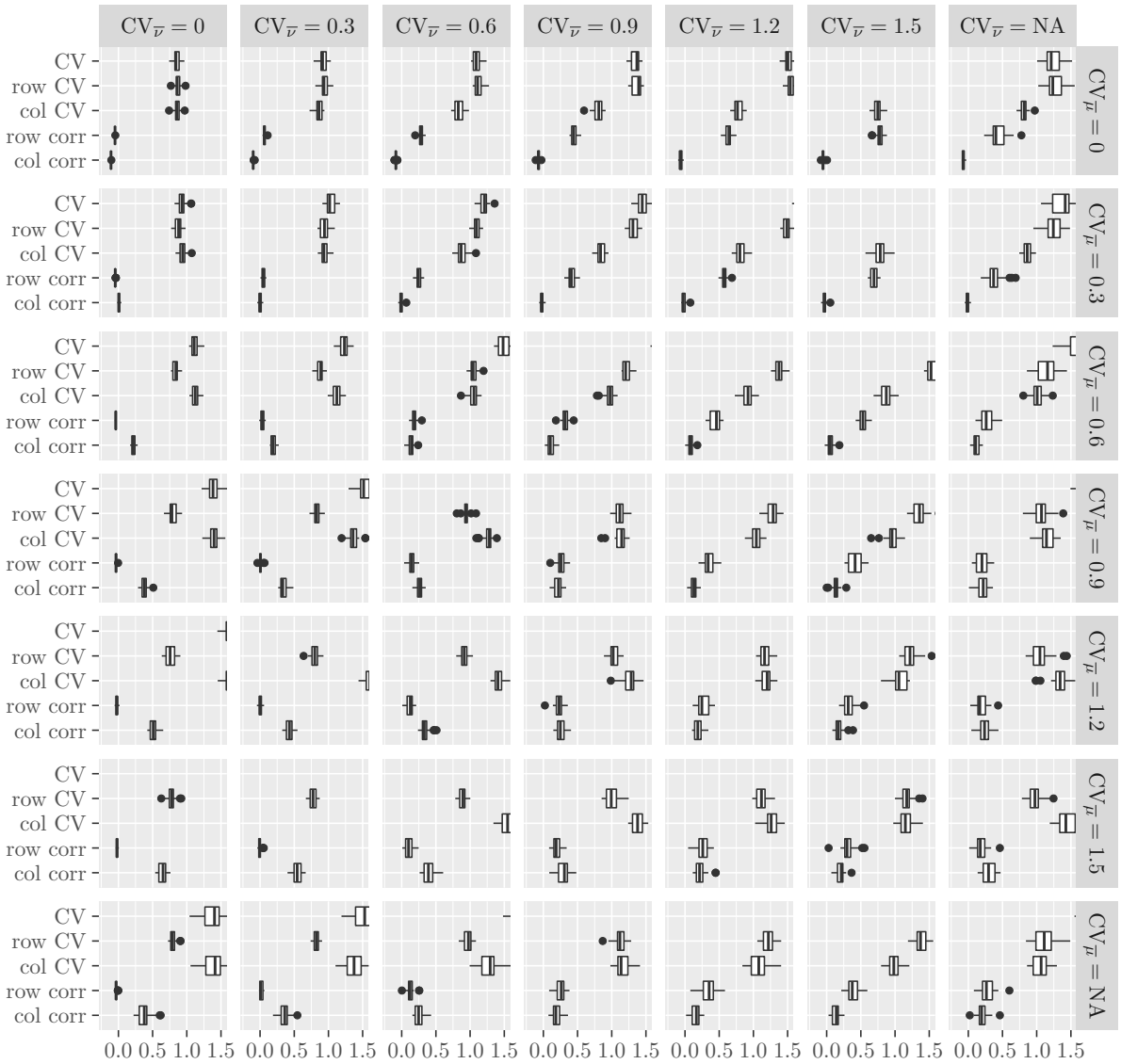

Figure 7: Matrix properties after 50 000 iterations starting with a proportional 20×10 matrix with different constraints on $\overline{\mu}$ and $\overline{\nu}$, and with $N = 20 \times n \times m = 4\,000$. Each matrix contains non-zero costs. Each boxplot corresponds to 30 matrices, each based on distinct row and column sums.

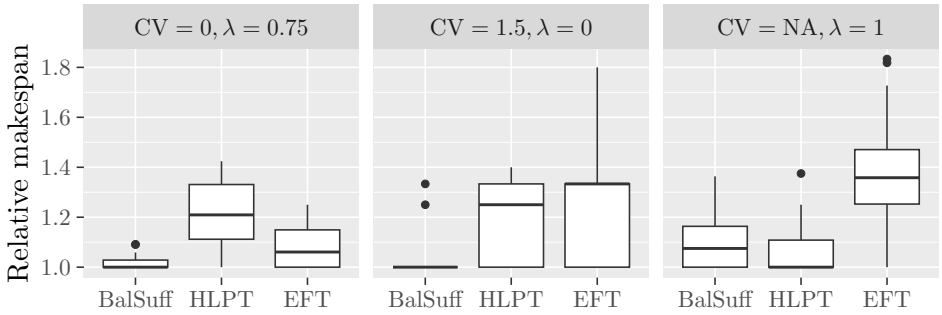

Figure 8: Ratios of makespan to the best among BalSuff, HLPT and EFT (lower is better). $\overline{\mu}$ and $\overline{\nu}$ are constrained with the same parameter CV. The constraint on the matrix is parameterized by a coefficient λ such that $A_{\min} = \lfloor \lambda M \rfloor$ and $B_{\max} = \lceil M/\lambda \rceil$ with $M(i,j) = \frac{\overline{\mu}(i) \times \overline{\nu}(j)}{N}$. The cost matrices were generated as in Figure 7. Each boxplot corresponds to 100 cost matrices, each based on distinct row and column sums.

6 Conclusion

Random instance generation allows broader experimental campaigns but can be hindered by bias in the absence of guarantee on the distribution of the instances. This work focuses on the generation of cost matrices, which can be used in a wide range of scheduling problems to assess the performance of any proposed solution. We propose a Markov Chain Monte Carlo approach to draw random cost matrices from a uniform distribution: at each iteration, some costs in the matrix are shuffled such that the sum of the costs on each row and column remains unchanged. By proving its ergodicity and symmetry, we ensure that its stationary distribution is uniform over the set of feasible instances. Moreover, the result holds when restricting the set of feasible instances to limit their heterogeneity. Finally, experiments were consistent with previous studies in the literature.

As a future direction, this work would benefit from experimental data on the actual characteristics of existing cost matrices in various contexts (from different applications and computing platforms). We could then generate matrices with heterogeneity values focused around relevant ones instead of covering a more widespread range. Having such practical data would constitute a significant contribution but involving work.

Another more formal future direction would be to apply the current methodology on the generation of other types of instances such as task graphs.

References

[1] R. Fagin, Probabilities on finite models, J. Symb. Log. 41 (1) (1976) 50–58.

[2] L.-C. Canon, L. Philippe, On the heterogeneity bias of cost matrices for assessing scheduling algorithms, IEEE Transactions on Parallel and Distributed Systems 28 (6) (2017) 1675–1688.

[3] H. Arabnejad, J. G. Barbosa, List scheduling algorithm for heterogeneous systems by an optimistic cost table, IEEE Transactions on Parallel and Distributed Systems 25 (3) (2014) 682–694.

[4] R. L. Graham, E. L. Lawler, J. K. Lenstra, A. H. G. R. Kan, Optimization and approximation in deterministic sequencing and scheduling: a survey, Annals of Discrete Mathematics 5 (1979) 287–326.

[5] L.-C. Canon, P.-C. Héam, L. Philippe, Controlling the correlation of cost matrices to assess scheduling algorithm performance on heterogeneous platforms, Concurrency and Computation: Practice and Experience 29 (15) (2017).

[6] P. Flajolet, P. Zimmermann, B. V. Cutsem, A calculus for the random generation of labelled combinatorial structures, Theor. Comput. Sci. 132 (2) (1994) 1–35.

[7] D. A. Levin, Y. Peres, E. L. Wilmer, Markov chains and mixing times, American Mathematical Society, 2006.

[8] L.-C. Canon, M. El Sayah, P.-C. Héam, A markov chain monte carlo approach to cost matrix generation for scheduling performance evaluation, in: 2018 International Conference on High Performance Computing & Simulation (HPCS), 2018, pp. 460–467.

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

[10] S. Ali, H. J. Siegel, M. Maheswaran, D. Hensgen, Task execution time modeling for heterogeneous computing systems, in: Heterogeneous Computing Workshop (HCW), IEEE, 2000, pp. 185–199.

[11] K. Pearson, On the theory of contengency and its relation to association and normal correlation, Drapers' Company Reserach Memoirs (1904).

[12] P. Diaconis, L. S. Coste, Random walk on contingency tables with mixed row and column sums, Tech. rep., Harvard University, Department of Mathematics (1995).

[13] M. del Carmen Pardo, On testing indenpendence in multidimensional contingency tables with stratified random sampling, Inf. Sci. 78 (1-2) (1994) 101–118.

[14] D. Hernek, Random generation of 2×n contingency tables, Random Struct. Algorithms 13 (1) (1998) 71–79.

[15] M. E. Dyer, C. S. Greenhill, Polynomial-time counting and sampling of two-rowed contingency tables, Theor. Comput. Sci. 246 (1-2) (2000) 265–278.

[16] S. DeSalvo, J. Y. Zhao, Random sampling of contingency tables via probabilistic divide-and-conquer, CoRR (2015). arXiv:1507.00070.

[17] D. I. G. Amalarethinam, P. Muthulakshmi, Dagitizer – a tool to generate directed acyclic graph through randomizer to model scheduling in grid computing, in: Advances in Computer Science, Engineering & Applications, 2012, pp. 969–978.

[18] D. Cordeiro, G. Mounié, S. Perarnau, D. Trystram, J. Vincent, F. Wagner, Random graph generation for scheduling simulations, in: International Conference on Simulation Tools and Techniques, SIMUTools, ICST/ACM, 2010, p. 60.

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

[20] H. Topcuoglu, S. Hariri, M.-y. Wu, Performance-effective and low-complexity task scheduling for heterogeneous computing, IEEE Transactions on Parallel and Distributed Systems 13 (3) (2002) 260–274.

A Notation

Table 2 provides a list of the most used notations in this report.

Symbol	Definition
n	Number of rows (tasks)
m	Number of columns (machines)
$M(i,j)$	Element on the ith row and jth column of matrix M
N	Sum of elements in a matrix ($\sum_{i,j} M(i,j)$)
$\overline{\mu}$	Vector of size n. $\frac{\overline{\mu}(i)}{m}$ is the mean cost of the ith task.
$\overline{\nu}$	Vector of size m. $\frac{\overline{\nu}(j)}{n}$ is the mean cost on the jth machine.
$\mathrm{Var}[\overline{\mu}]$	Variance of vector $\overline{\mu}$.
S	Sum of element squares in a vector.
$CV(\overline{\mu})$	Coefficient-of-Variation of vector $\overline{\mu}$.
$H_{N,n}^{\alpha,\beta}$	Elements $\overline{v} \in \mathbb{N}^n$ s.t. $\alpha \leq \overline{v}(i) \leq \beta$ and $\sum_{i=1}^{n} \overline{v}(i) = N$.
$h_{N,n}^{\alpha,\beta}$	Cardinal of $H_{N,n}^{\alpha,\beta}$.
$d(A,B)$	Distance between matrices A and B.
$\Omega_{n,m}^{N}(\overline{\mu},\overline{\nu})$	Set of contingency tables of sum N and sums of rows and columns $\overline{\mu}$ and $\overline{\nu}$.
α, β	Scalar constraints on minimal/maximal values for generated matrices.
$\overline{\alpha}$, $\overline{\beta}$	Vector constraints on minimal/maximal values for generated matrices.
$A_{\min}$, $B_{\max}$	Matrix constraints on minimal/maximal values for generated matrices.
$\Omega_{n,m}^{N}(\overline{\mu},\overline{\nu})[...]$	Subset of $\Omega_{n,m}^{N}(\overline{\mu},\overline{\nu})$ min/max-constrained by [...].
$f(\cdot,\cdot)$	Random mapping for the Markov Chains.

Table 2: List of notations.