Bayesian nonparametric model for arbitrary cubic partitioning

Masahiro Nakano MASAHIRO.NAKANO.PR@HCO.NTT.CO.JP
Yasuhiro Fujiwara YASUHIRO.FUJIWARA.KH@HCO.NTT.CO.JP
Akisato Kimura AKISATO.KIMURA.XN@HCO.NTT.CO.JP
Takeshi Yamada TAKESHI.YAMADA.BC@HCO.NTT.CO.JP
Naonori Ueda NAONORI.UEDA.FR@HCO.NTT.CO.JP
NTT Communication Science Laboratories, NTT Corporation

Editors: Vineeth N Balasubramanian and Ivor Tsang

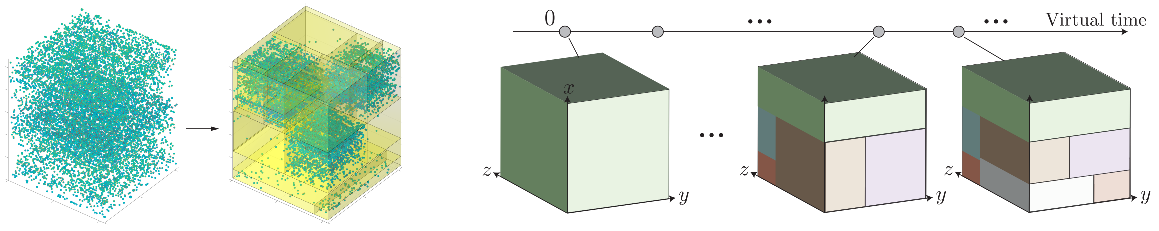
Abstract

In this paper, we propose a continuous-time Markov process for cubic partitioning models of three-dimensional (3D) arrays and its application to Bayesian nonparametric relational data analysis of 3D array data. Relational data analysis is a topic that has been actively studied in the field of Bayesian nonparametrics, and in particular, models for analyzing 3D arrays have attracted much attention in recent years. In particular, the cubic partitioning model is very popular due to its practical usefulness, and various models such as the infinite relational model and the Mondrian process have been proposed. However, these conventional models have the disadvantage that they are limited to a certain class of cubic partitions, and there is a need for a model that can represent a broader class of arbitrary cubic partitions, which has long been an open issue in this field. In this study, we propose a stochastic process that can represent arbitrary cubic partitions of 3D arrays as a continuous-time Markov process. Furthermore, by combining it with the Aldous-Hoover-Kallenberg representation theorem, we construct an infinitely exchangeable 3D relational model and apply it to real data to show its application to relational data analysis. Experiments show that the proposed model improves the prediction performance by expanding the class of representable cubic partitioning.

Keywords: Bayesian nonparametrics, Cubic partitioning, Markov process

1. Introduction

Three-dimensional (3D) array data is a subject that has attracted particular attention in recent years, and advances in its analysis techniques are expected. In particular, the problem of finding hidden clusters in 3D arrays is an important one, and has been used in various applications (Kemp et al., 2006; Roy and Teh, 2009; Roy, 2011; Choi and Wolfe, 2014; Rodriguez and Ghosh, 2009; Shan and Banerjee, 2008; Miller et al., 2009; Ishiguro et al., 2016; Caldas and Kaski, 2008; Airoldi et al., 2013; Fan et al., 2018b, 2016; Lloyd et al., 2012; Ge et al., 2019; Fan et al., 2018a, 2019, 2020b). The most basic clustering technique is called relational data analysis by the stochastic block model (Wasserman and Anderson, 1987; Nowicki and Snijders, 2001; Airoldi et al., 2013), and is formulated as the problem of finding an analysis result for input 3D array data in which the elements of each dimension are rearranged and a cubic partition is looked for, and each block in



(a) Relational data analysis (b) CPP evolution

Figure 1: (a) Relational data analysis through cubic partitioning. For the input array (left), we hope to find the proper permutation of the elements in each dimension and the cubic partitioning (right). The BNP models are able to analyze relational data in a data-driven manner without any foresight or manual tuning of the number or size of blocks. (b) Cubic partitioning process as a Markov process. Our goal is to give this Markov process the ability to represent arbitrary cubic partitioning.

the partition is as homogeneous as possible, as shown in Figure 1 (a). One of the major problems in this type of relational data analysis is that it is difficult to manually adjust the number of partitions and their sizes in advance. In order to overcome such a problem, the Bayesian nonparametric (BNP) methods, which essentially do not require tuning of model complexity, have been studied in relational data analysis using cubic partitioning.

As the historically significant beginning, Kemp et al. (2006) created the BNP model called the infinite relational model (IRM) for cubic partitioning, which consists of the product of three Chinese restaurant processes (CRPs) (Aldous, 1985; Pitman and Yor, 1997). The cubic partitioning class that IRM can represent is called the *regular grid* (Figure 2, left) (Mackisack and Miles, 1996), wherein each of the three dimensions is partitioned by a CRP, and the whole array is partitioned by its direct product. However, the class of cubic partitioning that can be represented by regular grid is very strongly constrained. Subsequently, Roy and Teh (2009) proposed the Mondrian process (MP) for a wider class of cubic partitioning. The class that can be represented by MP is called *hierarchical partitioning* (Figure 2, middle), which represents a class generated by successively selecting a single cube and adding one new partition to it. MP can indeed represent a wide class of cubic partitioning, but there are some partitioning schemes that it cannot represent. In short, the analysis results by IRM and MP could only explore solution candidates within limited classes of cubic partitioning. The class that represents all unconstrained rectangular partitioning is called *arbitrary cubic partitioning* (Figure 2, right), and the corresponding BNP model has been hoped to be constructed as a more flexible model. Unfortunately, there is no known BNP models that can represent arbitrary cubic partitioning. Hence, the construction of such BNP models has been an important problem to be solved.

In this paper, we tackle the construction of a stochastic process that represents arbitrary cubic partitioning, as a Markov process shown in Figure 1 (b). The main difficulty lies in the fact that naive extensions of existing 2D models for arbitrary rectangular partitioning, such as the rectangular tiling process (RTP) (Nakano et al., 2014) and the block-breaking process (BBP) (Nakano et al., 2020), cannot be applied to the 3D case. In the next subsection, we will reveal the reason for the difficulty of 3D extension and our key insights that led to its solution, along with the organization and contributions of this paper.

1.1. Organization, key insights and contributions of this paper

Our goal is again to construct a stochastic process that can represent arbitrary cubic partitioning. To begin with, Section 2.1 states that in order to achieve this goal, it is sufficient to consider only cubic partitions of $[0,1]\times[0,1]\times[0,1]$, thanks to the Aldous-Hoover-Kallenberg (AHK) representation theorem (Aldous, 1981; Hoover, 1979; Kallenberg, 1992). Next, Section 2.2 briefly review the notion of the Markov process, which is used for the construction of our proposed stochastic process. Then, Section 2.3 clarifies our key insight, which immediately leads to the 2D case of our proposed stochastic process. Very recently, Nakano et al. (2020) used the strategy of using a counting/enumeration algorithm to construct a stochastic process. Specifically, BBP (Nakano et al., 2020) converted the enumeration algorithm (Hong et al., 2000) for Baxter permutations (Baxter, 1964) to a "stochastic" enumeration algorithm in order to construct a stochastic process for rectangular partitioning, noting that there is a one-to-one correspondence between Baxter permutations and a certain class of rectangular partitioning (Hong et al., 2000). However, since such a useful one-to-one correspondence has not been reported to be extendable to the 3D case, the model construction method for BBP could not be naively used to construct a 3D cubic partitioning model. On the other hand, the recent discovery of a new enumeration algorithm (Merino and Mütze, 2021) for a broader class of permutations, called 2-clumped permutations (Reading, 2012) (including Baxter permutations as a subset) with one-to-one correspondence to rectangular partitioning provides us with a new enumeration algorithm option. Section 2.3 provides the detail of this enumeration algorithm for 2D rectangular partitioning. In fact, using this new counting algorithm, Section 3 first constructs a new stochastic process for rectangular partitioning in the 2D case. Unfortunately, as in the case of BBP, we could not continue to naively extend this 2D model to three dimensions, but we succeeded in extending it to three dimensions by gaining new insights, as described below.

- **Stochastic process for arbitrary cubic partitioning (Section 4) -** We found that it is extremely difficult at present to obtain an exact enumeration (i.e., counting) algorithm for 3D cubic partitioning in which every element of a cube partition appears only once. However, the new counting algorithm (Merino and Mütze, 2021) for rectangular partitions also provides important clues for the enumeration of cubic partitions in the 3D case. The naive 3D extension of that algorithm loses the ability to count rectangular partitions exactly (counting the same element only once), however, as long as elements are allowed to appear in duplicate, there is a prospect of regaining enumeration capability with only minor modifications. Owing to this insight, we can devise a new algorithm for enumerating cube partitions by relaxing the counting algorithm and allowing the same element to appear multiple times in duplicate, and transform it into a *stochastic* enumeration algorithm as a Markov process. As a result, we are able to obtain a 3D arbitrary cubic partitioning model. The supplementary material shows the existence of it and that its support is arbitrary rectangular partitions. We will call the proposed model a **C**ubic **P**artitioning **P**rocess (CPP) to distinguish it from other models.

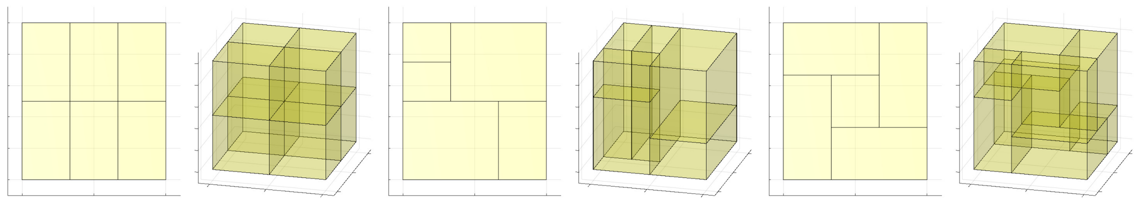


Figure 2: Three classes of rectangular and cubic partitioning. **Left:** Regular grid partitioning. **Middle:** Hierarchical partitioning. **Right:** Arbitrary partitioning. The main goal of this paper is to solve the problem of how to construct a BNP model that can represent arbitrary cubic partitioning.

- **Application to relational data analysis (Section 5) -** By combining CPP with the AHK representation theorem (Aldous, 1981; Hoover, 1979; Kallenberg, 1992), we create an infinitely exchangeable three-dimensional array model, and apply it to relational data analysis. Unlike conventional rectangular partitioning models, there is no restriction on the class of cubic partitioning that can be represented, and as a result, higher predictive performance is expected to be achieved. The supplementary material will provide the details of Bayesian inference algorithms. Our code will be available at `https://github.com/nttcslab/cubic-partitioning-process`.

1.2. Related work

Historically, the origin of the BNP model for array data can be traced back to IRM (Kemp et al., 2006), which allowed clustering of a multi-dimensional array as a product of CRP partitioning applied to each dimension of the array. IRM has the advantage that the model can be easily extended without depending on the dimension of the array, but it also has the disadvantage that the class of partitions that can be represented is limited to a narrow class called regular grid (Figure 2, left). To overcome this disadvantage of IRM, MP (Roy and Teh, 2009; Roy, 2011) and its variant (Lakshminarayanan et al., 2014) was proposed to represent a broader class of rectangular/cubic partitioning called hierarchical partitioning (Figure 2, middle), which could be further extended to be independent of the dimension of the array data (two or even three dimensions). Surprisingly, MP succeeded in preserving self-similarity, and its model construction strategy has had a significant impact on various subsequent studies of BNP model construction. In fact, in later years, some models appeared that relaxed the conditions under which MP allowed only vertical and horizontal cuts, and allowed oblique cuts, including the Ostomachion process (Fan et al., 2016) and the binary space partitioning tree-process/forest (Fan et al., 2019, 2020b) for 2D arrays, and the random tessellation forest (Ge et al., 2019) for 3D arrays. These models are very actively studied topics as BNP partitioning models.

For rectangular/cubic partitioning (Mackisack and Miles, 1996), MP was indeed able to represent a very wide class of partitioning, however, it was still not able to represent arbitrary rectangular/cubic partitioning. From this motivation, the rectangular tiling process (Nakano et al., 2014) was proposed as a model that can represent arbitrary rectangular partitioning (Figure 2, right). However, RTP is limited to the 2D case, and due to the complicated structure of the model, the inference did not work well, making the model difficult to handle in practical use. Very recently, a more sophisticated model, the block-breaking process (BBP) (Nakano et al., 2020), has been proposed, which extends the stick-breaking process (Sethuraman, 1994) for sequence to 2D array, and can represent arbitrary rectangular partitioning. In fact, BBP has a high affinity with Bayesian inference and has become a practical BNP model, but its model construction is limited to the 2D case, and the same model construction method could not be used for 3D arrays. With this motivation, we have attempted to construct a BNP model that can represent arbitrary cubic partitioning.

Although we focused only on the BNP model for rectangular/cubic partitioning in this paper, the BNP models for array data are not limited to the partitioning models. For a more overarching survey, please refer to (Fan et al., 2020a; Orbanz and Roy, 2013). In (Orbanz and Roy, 2013), various BNP models are introduced as examples, along with the basics of BNP theory and fundamental notions such as conditioning, projectivity and exchangeability. In (Fan et al., 2020a), we can learn various models at once in a very comprehensive and bird's eye view, up to the latest results.

2. Preliminaries

2.1. Exchangeable array and Aldous-Hoover-Kallenberg representation

BNP models are almost always called *infinitely exchangeable* models and are required to satisfy two strict conditions: *projectivity* and *exchangeability*. Fortunately, a very easy way to give these requirements to models is known as the AHK representation framework (Aldous, 1981; Hoover, 1979; Kallenberg, 1992). In fact, many BNP models use it to construct a relational model (Roy and Teh, 2009; Roy, 2011; Fan et al., 2016, 2018a,b, 2019; Ge et al., 2019; Lloyd et al., 2012). Originally, it was a representation theorem for *exchangeability*, however, one can also give *projectivity* (a requirement of Kolmogorov's extension theorem for infinite extension of the models) immediately from its construction. We here briefly review the notion of exchangeability and the AHK representation theorem.

Exchangeability: For any set A and $k \in \mathbb{N}$, $A^{(k)}$ denotes the set of subsets of A of size k. An exchangeable array $X := (X_e)_{e\in\mathbb{N}^{(k)}}$ of random variables satisfies $\mu_X((X_e)_{e\in\mathbb{N}^{(k)}}) = \mu_X((X_{\sigma(e)})_{e\in\mathbb{N}^{(k)}})$, for every permutation π of $\mathbb{N}$, where μ_X is the probability distribution of X. For any exchangeable array $(X_e)_{e\in\mathbb{N}^{(k)}}$, there exists a measurable function $f : [0,1] \times [0,1]^k \times [0,1]^{\binom{k}{2}} \times \ldots, \times [0,1]^{\binom{k}{k-1}} \times [0,1] \to \mathcal{X}$ which is symmetric under the action of the permutation group such that $(X_e)_{e\in\mathbb{N}^{(k)}} = f(U, (U_a)_{a\in e}, (U_a)_{a\in e^{(2)}}, \ldots, (U_a)_{a\in e^{(k-1)}}, U_e)$, where U and $(U_a)_{a\subseteq\mathbb{N},|a|\le k}$ are independently and identically distributed in Uniform($[0,1]$) (Kallenberg, 1989, 1992). The cases $k = 1$ and $k = 2$ correspond to the de Finetti and Aldous-Hoover representation theorems (Aldous, 1981, 1985; Hoover, 1979), respectively.

AHK representation of exchangeable cubic partitioning: In practical terms, a relational model based on AHK representation can be constructed by the following generative model. First, we generate a random partitioning B on $[0,1] \times [0,1] \times [0,1]$ based on some model (e.g., our proposed stochastic process in this paper). Then, for each element of each dimension of the input data $\boldsymbol{Z} = (Z_{l,m,n})_{L\times N\times M}$, we assign a uniform random variable on $[0,1]$, one by one: $U_l^{(1)} \sim$ Uniform($[0,1]$) ($l = 1,\ldots,L$), $U_m^{(2)} \sim$ Uniform($[0,1]$)

$(m = 1, \dots, M)$, and $U_n^{(3)} \sim \text{Uniform}([0,1])$ $(n = 1, \dots, N)$. Finally, each element $Z_{l,m,n}$ belongs to a block of B to which the coordinate $(U_l^{(1)}, U_m^{(2)}, U_n^{(3)})$ belongs. The main advantage of this construction method is that we do not need to consider the cubic partitioning of the input data directly, but only need to consider the model for the virtual partitioning of $[0,1] \times [0,1] \times [0,1]$. Therefore, owing to this AHK representation, we mainly consider a stochastic process on cubic partitioning of $[0,1] \times [0,1] \times [0,1]$ in the following.

2.2. Random cubic partitioning as Markov process

Various strategies such as scaling limits (Bassino et al., 2019a,b; Maazoun, 2019), projective limits (Orbanz, 2011; Nakano et al., 2014, 2020), and time-bounded Markov processes (Roy and Teh, 2009; Fan et al., 2019; Ge et al., 2019) have been used to construct stochastic processes for intermediate random function in the AHK representation. Which method is better is a very important topic from the point of view of BNP theory, but it deviates greatly from the purpose of this paper. Here, without discussing in depth which method is better, we will focus on the method of constructing stochastic processes using Markov processes as the most straightforward way to achieve our goal. Therefore, we will briefly review here MP (Roy and Teh, 2009; Roy, 2011), which is a model with the similar strategy.

We can construct MP (Roy and Teh, 2009; Roy, 2011) as a Markov process based on the following famous recursive procedure. For simplicity, we focus only on MP $\boldsymbol{M} := (M_t; t \geq 0) \sim \text{MP}(\lambda, [a, A] \times [b, B] \times [c, C])$ on a 3-dimensional cube $[a, A] \times [b, B] \times [c, C]$. Starting with an *time limit* λ, MP makes a sequence of cuts that split cubes into sub-cubes. In terms of Markov processes, this cut event corresponds to the occurrence of a jump. Each cut incurs a random elapsed time t. If t exceeds the time limit λ, the process halts and returns the current partitions. Otherwise, MP makes an axis-aligned cut uniformly at random along (a, A), (b, B), and (c, C). The elapsed time t to cut the cube $[a, A] \times [b, B] \times [c, C]$ is distributed exponentially: $(t - 0) \sim \text{Exp}(A - a + B - b + C - c)$. After a cut is made (for example, suppose a cut split (a, A) into (a, x) and (x, A)), the elapsed time t is subtracted from the time limit λ and a new time limit $\lambda' = \lambda - t$ is calculated. Both sub-cubes are drawn independently from $\boldsymbol{M}_< \sim \text{MP}(\lambda', [a, x] \times [b, B] \times [c, C])$ and $\boldsymbol{M}_> \sim \text{MP}(\lambda', [a, A] \times [b, B] \times [c, C])$. The partition on $[a, A] \times [b, B] \times [c, C]$ can be expressed as: $\boldsymbol{M}_< \cup \boldsymbol{M}_>$. As a result, MP has the ability to represent any hierarchical partitioning, and yet it can also vary its own model complexity depending on the time limit of the Markov process.

2.3. Enumeration of rectangular partitioning through 2-clumped permutations

Enumeration algorithms often provide crucial clues to the construction of stochastic processes. Here, we introduce a new enumeration algorithm (Merino and Mütze, 2021) that was discovered very recently for rectangular partitioning in the 2D case. In fact, the reduction of our proposed stochastic process to the 2D case corresponds to the construction of a stochastic enumeration algorithm based on this algorithm.

This enumeration algorithm directly use the idea of *insertion*. See Figure 3. The idea of insertion is to add a new rectangle into the bottom-right corner of the current rectangular partition (Figure 3, left). Given a rectangular partition with n blocks, we first find a set of points that can become the top-left corner of the newly added rectangle. If any rectangle r of the given partition is tangent to the lower boundary of the outer rectangle (Figure 3,

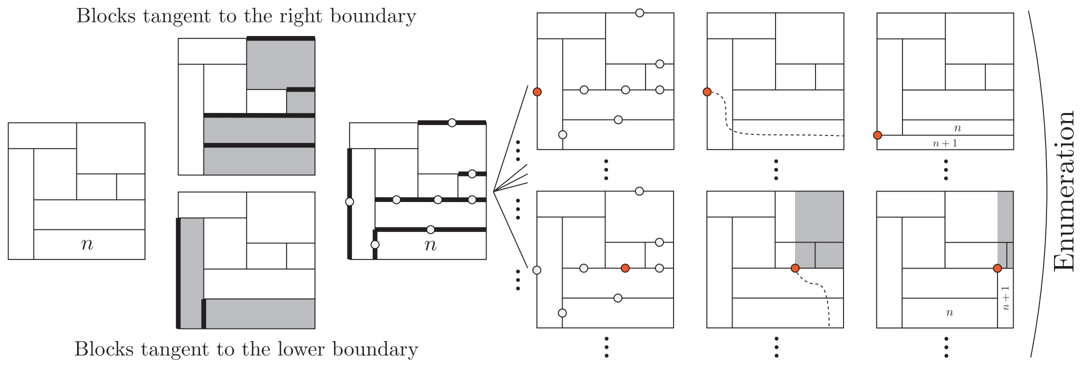

Blocks tangent to the right boundary
Blocks tangent to the lower boundary
n
n
n
n+1
n
n+1
Enumeration

Figure 3: Enumeration of rectangular partitioning. See the text (Section 2.3) for details.

second from left), we consider all edges that form the left side of r and select one interior point (circles in Figure 3, third from left) from such edges. Similarly, for any rectangle r tangent to the right boundary of the outer rectangle, we consider the set of all edges forming the top edge of r and select one interior point from such edges. Then we focus on one point v (red circles in Figure 3, fourth from left) in the set of such interior points. If v is a vertical insertion point, then the new partition is obtained from the current partition by inserting a new rectangle in the lower right corner. In this case, the new rectangle will be located to the right of v, with all rectangles tangent to the bottom boundary of the outer rectangle exactly above it, and the new rectangle have all rectangles tangent to the vertical wall through v below v exactly to the left. Similarly, if v is a horizontal insertion point, the new partition can be obtained from the current partition by inserting a new rectangle in the lower right corner. Then the new rectangle have all the rectangles that are below v and tangent to the right boundary of the outer rectangle exactly to the left, and the new rectangle will have all the rectangles in the outer rectangle that are to the right of v and tangent to the horizontal wall through v exactly to the top (Figure 3, second and first from right). It is known that this algorithm can enumerate arbitrary rectangular partitioning (See Theorem 8 in (Merino and Mütze, 2021)) due to the one-to-one correspondence between rectangular partitions and two-clumped permutations (Reading, 2012).

3. Markov Process for rectangular partitioning in 2D case

The new enumeration algorithm for 2D rectangular partitioning (described in Section 2.3) provides significant clues to the construction of new stochastic processes. In fact, the reduction of our proposed model to the 2D case corresponds a stochastic enumeration algorithm inspired by this algorithm. Therefore, for simplicity, we will first restrict attention to the case of 2D arrays and describe how to construct a CPP, which is constructed as a Markov process $\boldsymbol{B} := (B_t; t \geq 0)$ in which rectangular partitioning evolve in time over continuous time t. See also Figure 4 for an intuition of our model construction.

Step 1: Occurrence of jump in Markov process - We assume that the current rectangular partition B_t of $[0,1] \times [0,1]$ consists of K blocks $c_1, c_2, \ldots, c_K$. For each block c_k $(k = 1, \ldots, K)$, let $v_k = (v_k^{(1)}, v_k^{(2)})$ be the vertex closest to $(0,0)$ and $w_k = (w_k^{(1)}, w_k^{(2)})$ be the farthest point. See Figure 4. In this figure, the lower right corner of the outer rectangle is shown as $(0,0)$. Therefore, for each block c_k, the lower right vertex corresponds to v_k and

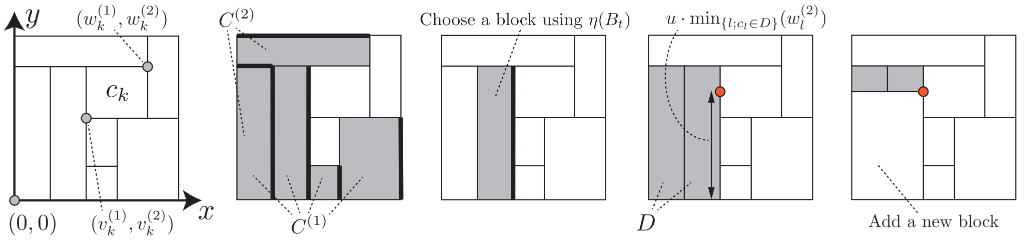


Figure 4: Illustration of 2D CPP. See also the text (Section 3) for details. **Left:** Current CPP B_t indicates a sample of rectangular partitioning. For each block c_k ($k = 1, \ldots, K$), let $v_k = (v_k^{(1)}, v_k^{(2)})$ and $w_k = (w_k^{(1)}, w_k^{(2)})$ be the vertex closest to $(0,0)$ and the farthest point. **Second:** We focus on the blocks tangent to the x- and y-axes, and let the sum of their heights (bold lines) be the intensity $\eta(B_t)$ of the Markov process. **Third and fourth:** We can choose one point (red point) uniformly on the measure $\eta(B_t)$, which corresponds to the upper left corner of the newly inserted block. **Right:** Next rectangular partitioning $B_{t'}$ is obtained.

the upper left vertex corresponds to w_k. Then, for $i = 1, 2, 3$, let $C^{(i)}$ be the set of blocks satisfying $v_k^{(i)} = 0$, that is,

$$C^{(1)} := \left\{ c_k \mid v_k^{(1)} = 0 \right\}, \quad C^{(2)} := \left\{ c_k \mid v_k^{(2)} = 0 \right\}. \tag{1}$$

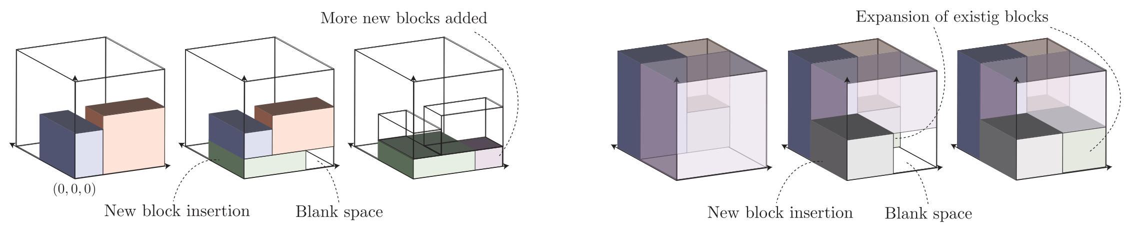
In short, $C^{(1)}$ and $C^{(2)}$ are the sets of blocks tangent to the y-axis, x-axis, respectively. Figure 4 (second from left) shows the blocks belonging to sets $C^{(1)}$ and $C^{(2)}$ in gray. Note that the block containing the origin (i.e., the block c_k such that $v_k^{(1)} = v_k^{(2)} = 0$) is included in both $C^{(1)}$ and $C^{(2)}$. Each block c_k in $C^{(i)}$ ($i = 1, 2$) has an intensity of $(w_k^{(i)} - v_k^{(i)}) = w_k^{(i)}$ (i.e., length of each bold line in Figure 4, second from left). As a result, the Markov process B_t has an intensity $\eta(B_t)$ that is the sum of the intensity of each block:

$$\eta(B_t) = \sum_{i=1}^{2} \sum_{k; c_k \in C^{(i)}} \left(w_k^{(i)} - v_k^{(i)} \right) = \sum_{i=1}^{2} \sum_{k; c_k \in C^{(i)}} w_k^{(i)}. \tag{2}$$

Based on this intensity $\eta(B_t)$, a jump of the Markov process has occurred at time t' ($> t$): $t' - t \sim \text{Exp}\big(\eta(B_t)\big)$. The new partition $B_{t'}$ can be obtained by the following procedure.

Step 2: Evolution of cubic partitioning - We choose a block c_k with probability $w_k^{(i)}/\eta(B_t)$. Without loss of generality, we assume that c_k belongs to $C^{(1)}$, and is chosen with probability $w_k^{(1)}/\eta(B_t)$. In Figure 4 (third from left), it is assumed that the block adjacent to the one on the lower right is chosen. Then, we draw a uniform random variables $u \sim \text{Uniform}([0,1])$. Next, we will focus on the line $[0, w_k^{(1)}] \times \{0\}$ and consider the set of blocks that are on this line. More specifically, let D be the set of blocks c_j satisfying $w_j^{(1)} \leq w_k^{(1)}$ and $c_j \in C^{(1)}$ (Figure 4, fourth from left):

$$D := \left\{ c_j \in C^{(1)} \mid w_j^{(1)} \leq w_k^{(1)} \right\}. \tag{3}$$



(a) Adding more new blocks (b) Expanding existing blocks

Figure 5: Illustration of why naive 3D extension of CPP is difficult. (a) We imagine a situation where a new green block (middle) is inserted for a part of the current partition (left). If we try to keep the red block as a cube, we end up with a blank area and need to fill it (middle). One strategy is to use more new blocks to fill the blank space. (b) We consider the procedure to generate a cubic partition with five blocks as shown in the right figure from a cubic partition with four blocks as shown in the left figure. In order to generate this cubic partition on the right, it is not possible to just add new blocks to the blank space, we have to use the expansion of existing blocks. As these two examples (a) and (b) show, the difficulty in realizing a 3D CPP lies in the fact that it requires additional procedures to fill in the blank space.

For every block $c_j \in D$, we transform $v_j^{(2)}$ to $u \cdot \min_{\{l; c_l \in D\}}(w_l^{(2)})$ Finally, we add a new block $[0, w_k^{(1)}] \times [0, u \cdot \min_{\{l; c_l \in D\}}(w_l^{(2)})]$, and obtain a new rectangular partitioning $B_{t'}$.

4. Markov Process for cubic partitioning in 3D case

Now that we have constructed the CPP for the 2D case in the previous section, we will construct the CPP for the 3D case, which is our main task. Unfortunately, however, the model construction method described above for the 2D case cannot be straightforwardly extended to the 3D case. First of all, we need to clarify the difficulty of the 3D extension.

Difficulty of 3D extension - Figure 5 (a) provides an illustration. As in the 2D case, we imagine a strategy that maps the evolution of a Markov process to the *insertion* of a cubic block. We are now trying to insert a new cubic block at $(0, 0, 0)$ for a given cubic partition in Figure 5 (a) left. For example, we consider one insertion method, such as the green block in Figure 5 (a) middle. In this case, if we try to keep the existing red block as a cube, we find that there will be a blank space in the lower right region. This blank space is the main reason why the problem is so difficult. In short, the 2D rectangular partitioning is closed under *insertion*, whereas the 3D cubic partitioning is not closed under *insertion*. To sum up, the difficulty of model construction in the 3D case lies in the fact that the same *insertion* as in the 2D case cannot be applied.

Our solution - As mentioned earlier, the *insertion* could not be directly applied to the 3D case, but it can still serve as a very powerful tool for enumerating cubic partitions. This is because most of the rectangular partitions that appear when a cubic partition is cut into planes parallel to the yz-, zx-, and xy-planes can potentially be enumerated by manipulating this *insertion*. The only possible problem is that the *insertion* may result in

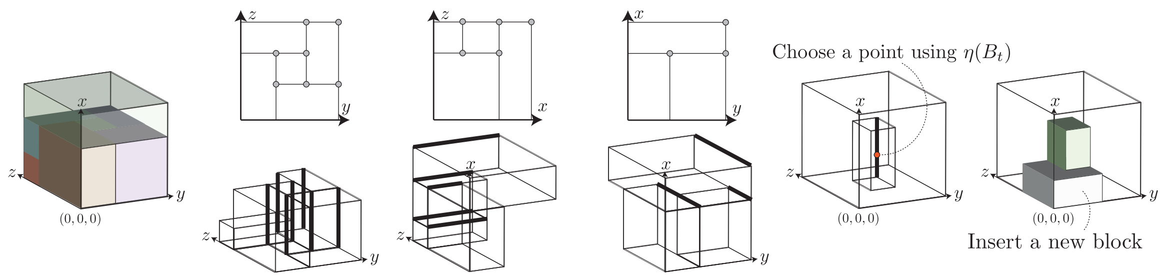


Figure 6: Illustration of **Step 1** for 3D CPP. **Left:** Current cubic partitioning B_t. **Second, third, and fourth:** We focus on the blocks tangent to the yz, zx, and xy-planes. Then, we extract the vertices on the bottom that are not on the x, y, and z axes. The intensity of the Markov process corresponds to the sum of the their heights. (Note that some of the bold lines are duplicated by several blocks.) **Fifth:** According to the intensity $\eta(B_t)$, we choose a block (green). One point on this block is chosen, which corresponds to the vertex farthest from the origin of the newly inserted block. **Right:** The method of inserting a new block (white cube) is continued in the next **Step 2** (Figure 7). See also the text (Section 4.1).

a blank space. Therefore, a possible strategy is to introduce an additional operation to fill this blank space. See Figure 5 (a) again. When we try to add a new block by *insertion*, as shown in Figure 5 (a) middle, there are two possible operations to fill in the blank space. One is to add more new blocks (Figure 5 (a) right), and the other is to expand the existing blocks. It is important to emphasize that both of these two operations are significant, and in order to enumerate all the cubic partitions, one is not enough, but both are necessary. While it is obvious that the former operation is necessary, it is important to note that the latter operation is also essential. Figure 5 (b) shows the most obvious example of the reason for this. In order to represent the cubic partition shown by this figure, the operation of expanding the existing blocks and filling in the blank spaces has become essential. As we can see from the above insights, the operation of filling in the blank space can be seen as the operation of creating a cubic partition again, using the expansion of existing blocks and the addition of new blocks. In other words, the creation of a cubic partition includes the operation of recursively creating a cubic partition.

4.1. Simplified version of 3D CPP

To simplify the discussion, we first describe the general framework of 3D CPP. More specifically, for simplicity, we restrict all fill-in-the-blank operations to just adding new blocks.

Step 1: Occurrence of jump in Markov process - See also Figure 6. We assume that the current cubic partitioning B_t of $[0,1]\times[0,1]\times[0,1]$ consists of K blocks $c_1, c_2, \ldots, c_K$. For each block c_k ($k = 1, \ldots, K$), let $v_k = (v_k^{(1)}, v_k^{(2)}, v_k^{(3)})$ be the vertex closest to $(0,0,0)$ and $w_k = (w_k^{(1)}, w_k^{(2)}, w_k^{(3)})$ be the farthest point. Then, for $i = 1,2,3$, let $C^{(i)}$ be the set of blocks satisfying $v_k^{(i)} = 0$, that is,

$$C^{(1)} := \left\{c_k \mid v_k^{(1)} = 0\right\}, \quad C^{(2)} := \left\{c_k \mid v_k^{(2)} = 0\right\}, \quad C^{(3)} := \left\{c_k \mid v_k^{(3)} = 0\right\}. \tag{4}$$

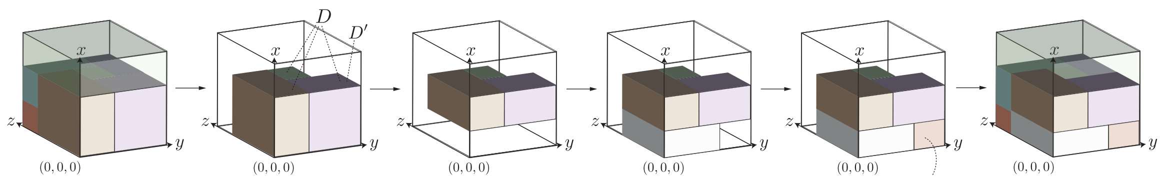


Figure 7: Illustration of **Step 2** and **Step 3** for 3D CPP. **Left:** Suppose now that the green cube c_k in the middle is the focus of attention by **step 1** (See also Figure 6). **From Second to fourth:** The blocks inside and on the boundary of the area covered by the region connecting w_k and $(0, 0, 0)$ will be shaved off to make room for a new block (white). **Fifth:** In order to keep the whole to be a cubic partition, we insert new additional blocks (pink) further down D' (**Step 3**). **Right:** Finally, the next cubic partitioning $B_{t'}$ is obtained. See also the text (Section 4.1).

In short, $C^{(1)}$, $C^{(2)}$, and $C^{(3)}$ are the sets of blocks tangent to the yz-plane, zx-plane, and xy-plane, respectively. For every block $c_j \in C^{(1)}$, we extract the vertices on the bottom (yz-plane) that are not on the y and z axes (Figure 6, second from left (top row)). For each extracted vertex, we then calculate the height of the vertex as the intensity of the Markov process (Figure 6, second from left (bottom row)). Note that some of the blocks also have an intensity of $4w_j^{(1)}$, while others have an intensity of $2w_j^{(1)}$. The same operation is performed for $C^{(2)}$ and $C^{(3)}$ to obtain the intensity $\eta(B_t)$ of the Markov process B_t. To put it simply, the sum of the lengths of the bold lines[1] in Figure 6 (from second to fourth) is the intensity $\eta(B_t)$ of the Markov process. Based on this intensity $\eta(B_t)$, we assume that a jump of the Markov process has occurred at time t' ($> t$). The new cubic partitioning $B_{t'}$ can be obtained by the following procedure.

Step 2: Evolution of cubic partitioning - According to the intensity ratio assigned to each block, we choose a block c_k. Without loss of generality, we assume that the chosen block c_k belongs to the set $C^{(1)}$. Next, we will focus on the plane $\{0\} \times [0, w_k^{(2)}] \times [0, w_k^{(3)}]$ and consider the two sets of blocks:

$$D \quad := \quad \left\{c_j \in C^{(1)} \mid v_j^{(2)} \le w_k^{(2)} \wedge v_j^{(3)} \le w_k^{(3)}\right\}, \tag{5}$$

$$D' \quad := \quad \left\{c_j \mid c_j \in D \wedge \left(w_j^{(2)} > w_k^{(2)} \vee w_j^{(3)} > w_k^{(3)}\right)\right\}. \tag{6}$$

Intuitively, D represents the set of blocks on the plane $\{0\} \times [0, w_k^{(2)}] \times [0, w_k^{(3)}]$, D' represents the set of blocks involving the boundary. For every $c_j \in D$, we transform $v_j^{(1)}$ to $u \cdot \min_{\{l; c_l \in D\}}(w_l^{(1)})$, where $u \sim \text{Uniform}([0, 1])$. Then, we add a new block $[0, u \cdot w_k^{(1)}] \times [0, w_k^{(2)}] \times [0, w_k^{(3)}]$. Finally, we need a few additional blocks to maintain the consistency of the overall partitioning based on the following procedure.

Step 3: Filling blank spaces - This procedure is specific to the 3D case, which was not necessary in the 2D case. For every $c_j \in D'$ (without loss of generality, we can here assume that $w_j^{(2)} > w_k^{(2)} \wedge w_j^{(3)} \le w_k^{(3)}$), we add a new block $[0, u_1 w_k^{(1)}] \times [w_k^{(2)}, w_j^{(2)}] \times [v_j^{(3)}, w_j^{(3)}]$. Finally, we can obtain the new cubic partitioning $B_{t'}$.

1. Note that some of the bold lines are duplicated by several blocks.

This simplified CPP described above certainly has the ability to represent cubic partitioning beyond hierarchical partitioning. However, it is not able to represent all cubic partitions. The reason for this is that we only used a simple operation to fill the blank space generated in **step 2** with new blocks in **step 3**. To overcome this issue, in the next section, we show how to recursively fill the blank space using CPP with a slight modification.

4.2. Nested CPP

We make a few modifications so that CPP can represent all cubic partitions. Specifically, we modify the CPP so that it can be called recursively. In order to be able to call CPP recursively, we need to consider a Markov process that takes into account the expansion of existing neighboring blocks. Now consider the CPP for $[p, P] \times [q, Q] \times [r, R]$. Let E_t be the set of blocks that touch the outside of $[p, P] \times [q, Q] \times [r, R]$ at $x = P$, $y = Q$, $z = R$ at time t. The CPP $(B_t; t \geq 0) \sim \mathrm{CPP}(\lambda, E_t, [p, P] \times [q, Q] \times [r, R])$ can be realized by adding the following modifications to the simplified version described above:

Step 1: We add virtual blocks derived from the set E when generating the set of $C^{(1)}$, $C^{(2)}$, and $C^{(3)}$. For every block c_j to which E belongs, perform the following steps. Without loss of generality, we assume that c_j is tangent to $x = A$. Then we add a virtual block $[p, P] \times [v_j^{(2)}, w_j^{(2)}] \times [v_j^{(3)}, w_j^{(3)}]$.

Step 2: Owing to the virtual blocks in **Step 1**, we may now have a block that can be merged with the block in E (e.g. the generated block $[p'(\geq p), P] \times [v_j^{(2)}, w_j^{(2)}] \times [v_j^{(3)}, w_j^{(3)}]$, which may be merged with c_j). At the moment when such a block is created for the first time, we decide whether or not to merge them from the Bernoulli distribution.

Step 3: If there are blank spaces in **step 3**, we call CPPs on them recursively.

In summary, we start with $\mathrm{CPP}(\lambda, \{\}, [0, 1] \times [0, 1] \times [0, 1])$ (where $\{\}$ indicates the empty set), and when a blank space is created in $[p, P] \times [q, Q] \times [r, R]$ at time t, we call an additional Markov process $\mathrm{CPP}(\lambda, E_{t'}, [p, P] \times [q, Q] \times [r, R])$. Due to the additivity of Poisson processes, these multiple time-inhomogeneous Markov Processes can be integrated into a single Markov Process.

5. Application to 3D relational data analysis

Relational model - The CPP-based relational model is applied to the input observation array $\boldsymbol{Z} = (Z_{l,m,n})_{L \times N \times M}$, consisting of categorical elements, i.e., $Z_{l,m,n} \in \{1, 2, \ldots, D\}$ ($D \in \mathbb{N}$). The generative model can be constructed as follows. First, we draw a random cubic partitioning of $[0, 1] \times [0, 1] \times [0, 1]$ from CPP: $\boldsymbol{B} = (B_t; t \geq 0) \sim \mathrm{CPP}(\lambda, [0, 1] \times [0, 1] \times [0, 1])$, where λ is non-negative tunable real parameter. Then, we use B_λ (that is, a sample of cubic partitioning after the last jump in the Markov chain occurs) as cubic partitioning. Next, inspired by the AHK representation theorem (Aldous, 1981; Hoover, 1979; Kallenberg, 1992), we draw i.i.d. uniform random variables on $[0, 1]$: $U_l^{(1)} \sim \mathrm{Uniform}([0, 1])$ ($l = 1, \ldots, L$), $U_m^{(2)} \sim \mathrm{Uniform}([0, 1])$ ($m = 1, \ldots, M$), and $U_n^{(3)} \sim \mathrm{Uniform}([0, 1])$ ($n = 1, \ldots, N$). Each block of B_λ has a latent Dirichlet random variable $\vartheta_k \sim \mathrm{Dirichlet}(\boldsymbol{\alpha}_0)$ ($k = 1, 2, \ldots$), where $\boldsymbol{\alpha}_0$ is a D-dimensional non-negative hyper parameter. Each element $Z_{l,m,n}$ is drawn from the multinomial distribution with the parameter $\vartheta_{\mathbf{k}(l,m,n)}$, where $\mathbf{k}(l, m, n)$ indicates the block index to which the coordinate $(U_l^{(1)}, U_m^{(2)}, U_n^{(3)})$ belongs. For Bayesian inference algorithms,

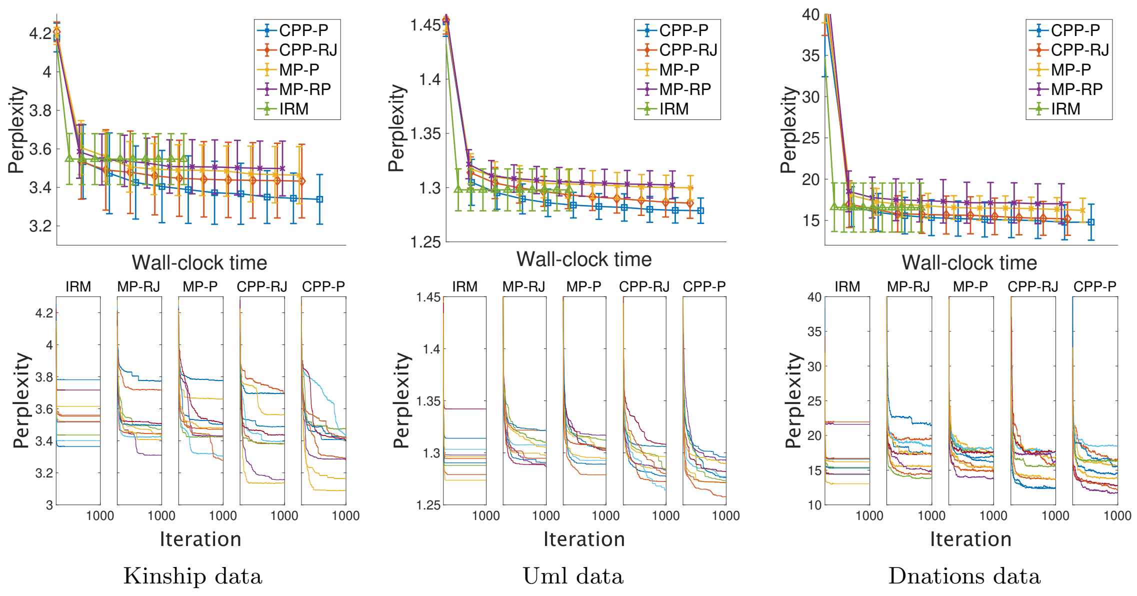


Figure 8: Experimental results of perplexity comparison. Each column corresponds to each real world data. **Top:** Relationship between test perplexity (mean$\pm$std) evolution and wall-clock time. **Bottom:** Relationship between test perplexity evolution and 1000 MCMC iterations for 10 trials.

we employ the reversible jump MCMC (RJMCMC) method (referred to as CPP-RJ) (Wang et al., 2011) and (2) the particle MCMC (PMCMC) method (referred to as CPP-P) (Fan et al., 2018a, 2019; Ge et al., 2019). See the supplementary material for details.

Datasets - We used three benchmark data sets for our evaluation: (1) **Kinship -** $104 \times 104 \times 26$ binary array (Denham, 1973). Data on kinship systems in Australian tribes, famous among anthropologists for the complex relational structure of their kinship systems. (2) **Uml -** $135 \times 135 \times 49$ binary array (McCray et al., 2001). A semantic network for biomedical ontology. (3) **Dnations -** $14 \times 14 \times 56$ binary array (Rummel, 1999). Political data set including 14 countries and binary predicates representing interactions between countries. We compare CPP-P and CPP-RJ with the existing BNP models for cubic partitioning, IRM (Kemp et al., 2006), MP (Roy and Teh, 2009) with RJMCMC (MP-RP) and PMCMC (MP-P). For evaluation, we held out 20% cells of the input data for testing, and each model was trained by the MCMC using the remaining 80% of the cells. We evaluated the models using perplexity as a criterion: $\text{perp}(\hat{Z}) = \exp(-(\log p(\hat{Z}))/N)$, where N is the number of non-missing cells in the partitioned matrix $\hat{Z}$.

Experimental result - We ran 10 trials of analysis for each method on each data set. Figure 8 summarizes the test perplexity comparison results. We recall that IRM and MP are limited in the class of cubic partitions they can represent. On the other hand, CPP can represent arbitrary cubic partitions, so in principle, CPP is expected to perform better than IRM and MP. Indeed, it can be seen that CP-P and CPP-RJ shows better prediction performance. As for the comparison of inference algorithms, it can be seen that PMCMC is more likely to find better local optima than RJMCMC. In summary, we have confirmed that CPP often finds better clustering results and that the inference algorithm works well.

6. Conclusion

We propose a stochastic process that can represent arbitrary cubic partitions in order to realize relational data analysis with a model that has more flexible representation capability. Through the prediction problem on 3D relational data, it is confirmed that the proposed method has better prediction performance than the existing models that can only represent a limited class of rectangular partitions.

References

E. M. Airoldi, T. B. Costa, and S. H. Chan. Stochastic block model approximation of a graphon: Theory andconsistent estimation. In *Advances in Neural Information Processing Systems*, 2013.

D. J. Aldous. Representations for partially exchangeable arrays of random variables. *Journal of Multivariate Analysis*, 11:581–598, 1981.

D. J. Aldous. Exchangeability and related topics. *École d'Été St Flour, Lecture Notes in Mathematics*, 1117:1–198, 1985.

F. Bassino, M. Bouvel, V. Feray, L. Gerin, M. Maazoun, and A. Pierrot. Universal limits of substitution-closed permutation classes. *Journal of the European Mathematical Society*, 22(11):3565–3639, 2019a.

Frédérique Bassino, Mathilde Bouvel, Valentin Féray, Lucas Gerin, Mickaël Maazoun, and Adeline Pierrot. Scaling limits of permutation classes with a finite specification: a dichotomy. *arXiv:1903.07522*, 2019b.

G. Baxter. On fixed points of the composite of commuting functions. *Proceedings of American Mathematical Society*, 15:851–855, 1964.

J. Caldas and S. Kaski. Bayesian biclustering with the plaid model. In *2008 IEEE Workshop on Machine Learning for Signal Processing*, pages 291–296, 2008.

D. S. Choi and P. J. Wolfe. Co-clustering separately exchangeable network data. *Annals of Statistics*, 42:29–63, 2014.

W. Denham. *The detection of patterns in Alyawarra nonverbal behavior.* University of Washington, 1973.

X. Fan, B. Li, and S. A. Sisson. The binary space partitioning-tree process. In *International Conference on Artificial Intelligence and Statistics*, pages 1859–1867, 2018a.

Xuhui Fan, Bin Li, Yi Wang, Yang Wang, and Fang Chen. The Ostomachion Process. In *AAAI Conference on Artificial Intelligence*, pages 1547–1553, 2016.

Xuhui Fan, Bin Li, and Scott Sisson. Rectangular bounding process. In *Advances in Neural Information Processing Systems*, pages 7631–7641, 2018b.

Xuhui Fan, Bin Li, and Scott Anthony Sisson. Binary space partitioning forests. *arXiv:1903.09348*, 2019.

Xuhui Fan, Bin Li, Ling Luo, and Scott A. Sisson. Bayesian nonparametric space partitions: A survey. *arXiv:2002.11394*, 2020a.

Xuhui Fan, Bin Li, and Scott A. Sisson. Online binary space partitioning forests. *arXiv:2003.00269*, 2020b.

Shufei Ge, Shijia Wang, Yee Whye Teh, Liangliang Wang, and Lloyd Elliott. Random tessellation forests. In *Advances in Neural Information Processing Systems 32*, pages 9575–9585. 2019.

X. Hong, G. Huang, Y. Cai, J. Gu, S. Dong, C.K. Cheng, and J. Gu. Corner block list: an effective and efficient topological representation of non-slicing floorplan. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, 2000.

D. N. Hoover. Relations on probability spaces and arrays of random variables. Technical report, Institute of Advanced Study, Princeton, 1979.

Katsuhiko Ishiguro, Issei Sato, Masahiro Nakano, Akisato Kimura, and Naonori Ueda. Infinite plaid models for infinite bi-clustering. pages 1701–1708, 2016.

O. Kallenberg. On the representation theorem for exchangeable arrays. *Journal of Multivariate Analysis*, 30(1):137–154, 1989.

O. Kallenberg. Symmetries on random arrays and set-indexed processes. *Journal of Theoretical Probability*, 5(4):727–765, 1992.

C. Kemp, J. B. Tenenbaum, T. L. Griffiths, T. Yamada, and N. Ueda. Learning systems of concepts with an infinite relational model. In *AAAI Conference on Artificial Intelligence*, pages 381–388, 2006.

Balaji Lakshminarayanan, Daniel Roy, and Yee Whye Teh. Mondrian forests: Efficient online random forests. In *Advances in Neural Information Processing Systems*, 06 2014.

J. Lloyd, P. Orbanz, Z. Ghahramani, and D. M. Roy. Random function priors for exchangeable arrays with applications to graphs and relational data. In *Advances in Neural Information Processing Systems*, 2012.

M. Maazoun. On the brownian separable permuton. *Combinatorics, Probability and Computing*, 29(2):241–266, 2019.

M. S. Mackisack and R. E. Miles. Homogeneous rectangular tessellation. *Advances on Applied Probability*, 28:993, 1996.

A. T. McCray, A. Burgun, and O. Bodenreider. Aggregating umls semantic types for reducing conceptual complexity. *Medinfo*, 10(1):216–220, 2001.

Arturo Merino and Torsten Mütze. Combinatorial generation via permutation languages. iii. rectangulations. *arXiv:2103.09333*, 2021.

Kurt Miller, Michael I. Jordan, and Thomas L. Griffiths. Nonparametric latent feature models for link prediction. pages 1276–1284, 2009.

M. Nakano, A. Kimura, T. Yamada, and N. Ueda. Baxter permutation process. In *Advances in Neural Information Processing Systems*, 2020.

Masahiro Nakano, Katsuhiko Ishiguro, Akisato Kimura, Takeshi Yamada, and Naonori Ueda. Rectangular tiling process. In *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 361–369, 2014.

K. Nowicki and T. A. B. Snijders. Estimation and prediction for stochastic block structures. *Journal of the American Statistical Association*, 96:1077–1087, 2001.

P. Orbanz. Conjugate projective limits. *arXiv:1012.0363*, 2011.

P. Orbanz and D. M. Roy. Bayesian models of graphs, arrays and other exchangeable random structures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37:437–461, 2013.

J. Pitman and M. Yor. The two-parameter poisson-dirichlet distribution derived from a stable subordinator. *IBM Journal of Research and Development*, 2(25):855–900, 1997.

N. Reading. Generic rectangulations. *European Journal of Combinatorics*, 33(4):610–623, 2012.

A. Rodriguez and K. Ghosh. Nested partition models. Technical report, JackBaskin School of Engineering, 2009.

D. M. Roy. *Computability, inference and modeling in probabilistic programming*. PhD thesis, Massachusetts Institute of Technology, 2011.

D. M. Roy and Y. W. Teh. The Mondrian process. In *Advances in Neural Information Processing Systems*, 2009.

R. J. Rummel. *Dimensionality of Nations project: attributes of nations and behavior of nation dyads*. ICPSR data file, 1999.

J. Sethuraman. A constructive definition of Dirichlet priors. *Statistica Sinica*, 4:639–650, 1994.

H. Shan and A. Banerjee. Bayesian co-clustering. In *IEEE International Conference on Data Mining*, pages 530–539, 2008.

P. Wang, K. B. Laskey, C. Domeniconi, and M. I. Jordan. Nonparametric bayesian co-clustering ensembles. In *SIAM International conference on Data Mining*, pages 331–342, 2011.

S. Wasserman and C. Anderson. Stochastic a posteriori block models: Construction and assessment. *IBM Journal of Research and Development*, 9(1):1–36, 1987.