Sequential Graph Matching with Sequential Monte Carlo

Seong-Hwan Jun
Department of Statistics
UBC

Samuel W.K. Wong
Department of Statistics
University of Florida

James V. Zidek
Department of Statistics
UBC

Alexandre Bouchard-Côté
Department of Statistics
UBC

Abstract

We develop a novel probabilistic model for graph matchings and develop practical inference methods for supervised and unsupervised learning of the parameters of this model. The framework we develop admits joint inference on the parameters and the matchings. Furthermore, our framework generalizes naturally to K-partite hypergraph matchings or set packing problems. The sequential formulation of the graph matching process naturally leads to sequential Monte Carlo algorithms which can be combined with various parameter inference methods. We apply our method to image matching problems, document ranking, and our own novel quadripartite matching problem arising from the field of computational forestry.

1 Introduction

A matching is a well-known combinatorial structure derived from a graph, and consists of a set of edges such that pairs of edges are required to have no common nodes. Matchings have important applications in machine learning when a feature function is defined on the nodes and edges of the graph. Given such a feature function, the compatibility score of a matching is commonly defined as the dot product of the features and a parameter vector $\theta \in \mathbb{R}^p$. Based on the compatibility score we may optimize or compute expectations over the space of matchings, for example to find the most likely alignment of landmark points between two frames of a video or of two images [1, 2], or of nucleotides in related genomes [3, 4].

Two inter-related technical challenges are associated with such graph matching problems. The first challenge is *parameter estimation* for the matching models, which arises in both supervised and unsupervised settings. In the supervised setting, we are given a set of labelled matchings and the task is to optimize over the set of parameters to minimize a suitably chosen empirical loss function [5]. The unsupervised setting is typically based on a probabilistic approach, where a probability distribution over the space of matchings is defined [6]. One can then optimize over the parameters by marginalizing over matchings or by using an Expectation-Maximization (EM) algorithm. The second challenge consists of *computation of expectations over matchings* $\mathbb{E}[f(M)]$, where M is a matching-valued random variable. This may arise for example in computing the M step in an EM algorithm. Since the simulation of matching-valued random variables is a well-known $\#P$ hard problem, approximate inference methods are needed.

We develop a model and method of inference that addresses these two challenges. Our method is based on viewing a matching as a sequential decision process, where at each step, a node, selected at random or deterministically, picks a match (if any) from among the other nodes. Each decision is parameterized by a locally normalized logistic regression model. We demonstrate that using this local sequential view for *both* the model *and* approximate inference is highly beneficial. In particular, using it as the basis for the model formulation sidesteps the need to estimate an intractable normalization constant. This allows us to approach parameter estimation, the first technical challenge, both in the supervised and unsupervised settings, using standard tools such as Monte Carlo expectation maximization (MC-EM) [7] and stochastic approximation (SA-EM) [8]. This compares favorably to current models based on global Markov random fields, where both maximum likelihood and Bayesian parameter estimation requires specialized methods [9, 10]. While there has been work on using the local sequential decision view for the basis of an MCMC inference method over matchings [1], its application to matching *models* has not been broadly studied. The

Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS) 2017, Fort Lauderdale, Florida, USA. JMLR: W&CP volume 54.

Plackett-Luce model is a special case that has received attention in the literature, where the matching is the basis for constructing a ranking [11, 12, 13]. The local sequential view has comparably been more popular in other types of problems such as hidden Markov models and parse trees [14].

We argue that a natural approach for performing approximate inference over matchings in the local model context is to use Sequential Monte Carlo (SMC) algorithms [15]. This yields a simple algorithm that exploits the sequential structure of the model to explore the matching space efficiently, thus tackling the second challenge. SMC methods have been successfully applied to problems where the latent state space has a combinatorial structure (e.g., [16, 17]) and theoretical grounds for applying SMC methods for combinatorial state spaces have recently been established [17, 18]. Furthermore, we can benefit from parallelization of the computation that comes naturally with SMC methods; as well, methods such as [19, 20] which provide mechanisms for using more particles as needed (automatically) to ensure sufficient exploration of the space of the latent variables.

Our work is motivated by a novel application in computational forestry, where the internal three-dimensional tree branch structure of wood planks is reconstructed by matching the surface knots. This application requires a matching method that can be generalized to quadripartite graph as each of the four long faces of a wood plank represents a partition. This reconstructed 3-D structure is then used for predicting the strength of lumber.

Approximation algorithms on matchings has been approached in the literature using MCMC methods [21, 22] and recently using SMC method by [23]. A variational method has also been proposed [24]. A sequential Monte Carlo sampler for bipartite matching was proposed in [25], but they do not address the problem of parameter estimation and do not account for the *overcounting problem* [18], which occurs in hypergraph matching problems. Such problems are of practical importance, for example in our computational forestry application.

2 Background

2.1 Notation

A hypergraph is a generalization of a graph where each edge (or also called hyperedge) can connect any number of nodes. Matching can then be formulated as a set packing problem, where each edge is a set of nodes. We denote a K-partite hypergraph by $G = (V_1, ..., V_K, E)$. We denote a node in V_k by $v_{k,i}$, $i = 1, 2, \ldots, |V_k|$. An edge $e \in E$ is a set of nodes; for example, an edge between two nodes $v_{1,i}, v_{2,j}$ is denoted by $e = \{v_{1,i}, v_{2,j}\}$. Bipartite matching is a special case of K-partite set packing problem with two partitions and each edge restricted to contain exactly two nodes, one from each partition. A matching is represented by a set of edges and we denote it by M or m (for random variable and realization respectively) with the space of all feasible matchings denoted by $\mathcal{M}$.

2.2 Probabilistic models

We begin by describing a probabilistic formulation of the matching problem on a bipartite graph, $G = (V_1, V_2, E)$. For ease of exposition, we will assume that $|V_1| = |V_2|$. For bipartite matching, we can think of matching m as a permutation $m : V_1 \to V_2$. A probabilistic formulation of the bipartite matching problem in [6, 1] places a Gibbs measure on the matching:

$$\mathbb{P}(M = m|\theta) = \frac{\exp\langle\phi(m), \theta\rangle}{\sum_{m'} \exp\langle\phi(m'), \theta\rangle}, \tag{1}$$

where $\phi : \mathcal{M} \to \mathbb{R}^p$ is a feature function. However, the summation in the denominator is intractable, ruling out direct optimization over the parameters. Sampling from a posterior distribution of θ with the likelihood defined by Equation 1 is also difficult for the same reason.

The authors of [1] define an MCMC proposal distribution by constructing the bipartite matching iteratively. To do so, they introduce a reference sequence, $\sigma : \{1, ..., |V_1|\} \to \{1, ..., |V_1|\}$, which specifies the order in which the nodes in V_1 are visited. The reference sequence randomizes the order in which the bipartite matching is constructed. We briefly describe their construction here. First, let $v_{1,\sigma(t)}$ be the node visited at the t-th iteration. Let the partial matching up to iteration t be denoted by m_{t-1}, and let $V_{i,t-1}$ be the nodes in V_i that have not been matched up to the t-th iteration (note that m_0 is the empty matching and $V_{i,0} = V_i$). Suppose the node $v_{1,\sigma(t)}$ decides to match with the node $v_{2,j} \in V_{2,t-1}$, resulting in an edge $e_t = \{v_{1,\sigma(t)}, v_{2,j}\}$. The probability of this decision is given by:

$$p(e_t|m_{t-1}, \theta) = \frac{\exp\langle\theta, \phi(m_{t-1} + e_t)\rangle}{\sum_{e'_t} \exp\langle\theta, \phi(m_{t-1} + e'_t)\rangle}, \tag{2}$$

where $e'_t = \{v_{1,\sigma(t)}, v_{2,j'}\}$ for $v_{2,j'} \in V_{2,t-1}$ and we have overloaded the addition operator to denote by $m + e$ an addition of an edge e to the matching m. Unlike in Equation (1), the summation in the denominator can be computed exactly in $O(|V_2|)$.

This local multinomial model induces a Plackett-Luce-type probability distribution on the complete matching, $m = \{e_1, ..., e_{|V_1|}\}$:

$$Q(m|\sigma, \theta) = \prod_{t=1}^{|V_1|} p(e_t|m_{t-1}, \sigma, \theta). \tag{3}$$

The authors of [1] view Equation 3 as an approximation of Equation (1), and develop an MCMC sampler for matchings with proposal distribution given by Q. We take this idea further and propose in Section 3 to replace the distribution in Equation (1) by one that resembles Equation (3). Doing so simplifies parameter estimation as it yields a likelihood model for which pointwise and gradient evaluation can be computed efficiently.

3 A Sequential Decision Model for Matchings

We view the process of matching on a K-partite hypergraph as that of set packing, where each node is placed, sequentially, into a set of "similar" nodes. For full generality, we define the reference sequence to visit every node in the graph, $\sigma : \{1, ..., |V|\} \to \{1, ..., |V|\}$. This contrasts with the bipartite matching problem, where it is sufficient to iterate over the nodes in only one partition. Note that σ is a device that serves to randomize the construction of matching and in full generality, it is viewed as a random variable. Each node $v_{\sigma(t)}$ makes a decision, denoted by $d_{v_{\sigma(t)}}$, among the set of available decisions, denoted by $\mathcal{D}(v_{\sigma(t)}, m_{t-1})$. Here, we use m_{t-1} to denote the partial matching implied by the sequence of decisions, $\{d_{v_{\sigma(1)}}, ..., d_{v_{\sigma(t-1)}}\}$ but we will often omit m_{t-1} for notational simplicity and just write $\mathcal{D}(v_{\sigma(t)})$. The decision $d_{v_{\sigma(t)}}$ consists of potential ways the node $v_{\sigma(t)}$ can be entered into the matching. More precisely, picking $d \in \mathcal{D}(v_{\sigma(t)})$ means that the partial edge $e' = d \cup \{v_{\sigma(t)}\}$ is to be added to the partial matching m_t, i.e. $m_t = m_{t-1} + e' = m_{t-1} \backslash \{d\} \cup \{e'\}$. We use the terminology *partial edge*, since for example an edge with two nodes might be grown at a later iteration $t' > t$ to an edge augmented with a third node.

The decision set is user configurable to suit the problem at hand. For example, in bipartite matching without any restrictions, the decision candidates are $V_{2,t-1}$ (i.e., any nodes in V_2 that have not yet been matched). Depending on the problem at hand, we may also allow for the decision set to include a singleton decision, where the node is placed into a new set by itself. With our formalism, this decision can be modelled with $\mathcal{D}(v_{\sigma(t)})$ containing an empty set.

We illustrate an example of a sequential set packing process in Figure 1. There are four partitions in this example, and the nodes are labelled in the order σ that they are visited (i.e., the first node visited is the blue node labelled 1 and so on). For illustration purposes, assume that σ is given. In this illustration, we perform *pairwise* matching. To be specific, $\mathcal{D}(v_{\sigma(t)})$ contains any node that is in a different partition from $v_{\sigma(t)}$ as a candidate for matching. In our computational forestry experiments, we restrict the decision set to (i) exclude any nodes that are beyond a certain distance away and (ii) restrict each set to contain at most two nodes. These additional constraints are motivated by the application domain.

In the first step of this example, we visit the blue node #1. The decisions available are to form a singleton or to match with red node #2. In step 2, the only decision available for the red node #2 is to select the empty set, this is because the other nodes are beyond the distance span configured into our decision set but also because it is already contained in a set containing two nodes.

We model the process of the node making a decision amongst a list of available decisions by a multinomial logistic model, where the number of categories depends on the number of decisions available for the node. We will denote $e_j \in \mathcal{D}(v_{\sigma(t)})$ for $j = 1, ..., |\mathcal{D}(v_{\sigma(t)})|$ and we denote the edge chosen by node $v_{\sigma(t)}$ by $e_{d_{v_{\sigma(t)}}}$. The conditional probability for $v_{\sigma(t)}$ to be placed into edge e_j given the partial matching, m_{t-1}, is expressed by,

$$p(d_{v_{\sigma(t)}}|m_{t-1}, \sigma, \theta) = \frac{\exp\langle \phi(m_{t-1} + e_{d_{v_{\sigma(t)}}}), \theta \rangle}{\sum_{j'=1}^{|\mathcal{D}(v_{\sigma(t)})|} \exp\langle \phi(m_{t-1} + e_{j'}), \theta \rangle}, \tag{4}$$

where ϕ is a feature vector taking as input a (partial) matching. The likelihood of the complete sequence of decisions is simply:

$$\ell(\theta) = \prod_{t=1}^{|V|} p(d_{v_{\sigma(t)}}|m_{t-1}, \sigma, \theta). \tag{5}$$

We emphasize that the benefit of this model is that we can evaluate Equation (5) and compute its exact gradient efficiently, which permits numerical optimization of the likelihood over the parameters using off-the-shelf convex optimization routines such as L-BFGS [26].

4 Parameter Estimation

4.1 Unsupervised Learning via Monte Carlo Expectation Maximization

We place an isotropic normal prior on the parameters, and focus the discussion here on maximum *a posteriori* (MAP) estimation of $\theta \in \mathbb{R}^p$ using an EM algorithm

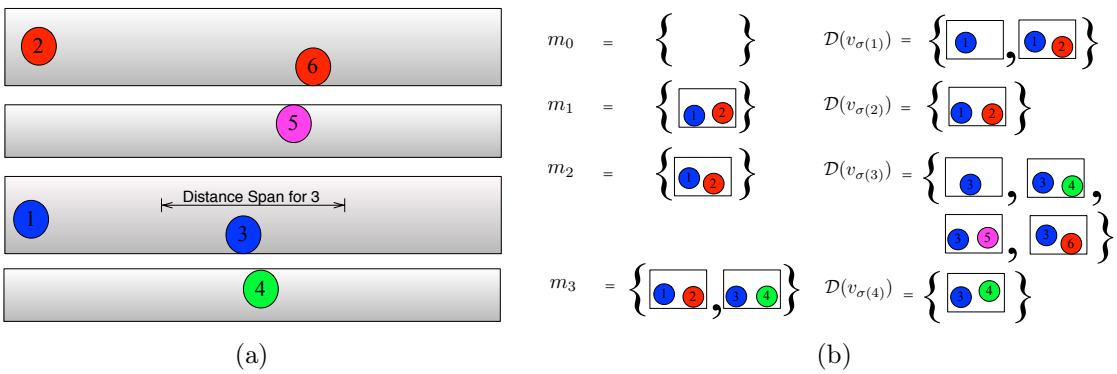


Figure 1: (a) A 4-partite hypergraph representing a piece of lumber. The nodes are labelled in the order σ that they are visited. Nodes outside the 'distance span' cannot reasonably belong to the same tree branch. (b) The decision candidates where only the singleton and doubleton matching are permitted. See text for details.

[27]. Maximum likelihood estimation can be done in a very similar fashion. We also discuss the prospects for a full Bayesian analysis in the conclusion.

The posterior distribution over the parameters given the sequence of decisions denoted by $\boldsymbol{d}_\sigma$ can be expressed as,

$$p(\theta|\boldsymbol{d}_\sigma,\sigma) = \frac{p(\boldsymbol{d}_\sigma,\sigma|\theta)p(\theta)}{p(\boldsymbol{d}_\sigma,\sigma)} = \frac{p(\boldsymbol{d}_\sigma|\sigma,\theta)p(\sigma)p(\theta)}{p(\boldsymbol{d}_\sigma,\sigma)}. \tag{6}$$

Note that the denominator is independent of the parameters. In some problems, such as document ranking, there exists a canonical ordering, making σ a fixed quantity instead of a random one. When no canonical ordering is provided by the context of the problem, we place a uniform distribution over the reference sequence σ, independent of the parameters.

The EM algorithm alternates between computing the conditional expectation of the latent variables given the current estimate of the parameters (E step), and maximizing that expectation over the parameters (M step):

$$Q(\theta,\theta^t) = \sum_{\boldsymbol{d}_\sigma,\sigma} p(\boldsymbol{d}_\sigma,\sigma|\theta^t)\log p(\theta|\boldsymbol{d}_\sigma,\sigma), \tag{7}$$

$$\theta^{t+1} = \text{argmax}_\theta\, Q(\theta,\theta^t). \tag{8}$$

Here the summation is difficult to evaluate analytically, and one may approximate it by generating samples from $p(\boldsymbol{d}_\sigma,\sigma|\theta^t)$. Thus, the Monte Carlo E-step approximates the expectation by sampling and taking an average over the sampled values:

$$\tilde{Q}(\theta,\theta^t) = \frac{1}{N}\sum_{n=1}^{N}\log p(\boldsymbol{d}_{\sigma^n}|\sigma^n,\theta) + \log p(\sigma^n) + \log p(\theta) - \log p(\boldsymbol{d}_{\sigma^n},\sigma^n), \tag{9}$$

where $(\boldsymbol{d}_{\sigma^n},\sigma^n) \sim p(\boldsymbol{d}_\sigma,\sigma|\theta^t)$. Note that when we are optimizing Equation (9) over the parameters, the terms $\log p(\sigma^n), \log p(\boldsymbol{d}_{\sigma^n},\sigma^n)$ are independent of θ and hence, need not be evaluated. Also note that if we place an independent Gaussian prior on θ, then we essentially obtain a log-likelihood with l_2 penalty:

$$\tilde{Q}(\theta,\theta^t) = \frac{1}{N}\sum_{n=1}^{N}\log \ell(\theta) - \frac{1}{2}\lambda\|\theta\|^2, \tag{10}$$

where ℓ is defined in Equation (5). As mentioned earlier, computing the gradient of ℓ is simple and efficient, and we use L-BFGS to perform this maximization step.

4.2 Supervised Learning

In the supervised setting, we are given a set of training instances of graph matchings $\{m_i\}$, $i = 1,...,I$. The posterior distribution of the parameters given the training data can be expressed as follows:

$$\begin{aligned} p(\theta|\{m_i\}) &\propto \prod_{i=1}^{I} p(m_i|\theta)p(\theta) = \prod_{i=1}^{I} p(\theta|m_i)p(m_i) \\ &\propto \prod_{i=1}^{I} p(\theta|m_i) = \prod_{i=1}^{I}\sum_{\sigma^i,\boldsymbol{d}_{\sigma^i}} p(\theta,\sigma^i,\boldsymbol{d}_{\sigma^i}|m_i) \\ &= \prod_{i=1}^{I}\sum_{\sigma^i,\boldsymbol{d}_{\sigma^i}} p(\theta|\sigma^i,\boldsymbol{d}_{\sigma^i})p(\sigma^i,\boldsymbol{d}_{\sigma^i}|m_i). \end{aligned} \tag{11}$$

Maximizing over the parameters provides the MAP estimate: $\hat{\theta}_{\text{MAP}} = \text{argmax}_\theta\, p(\theta|\{m_i\})$. One approach to supervised learning is to sample permutations $\sigma^{i,n}$, $n = 1,...,N$, from uniform distribution, then sampling decision sequences $j = 1,...,J$ that lead to m_i for each $\sigma^{i,n}$, i.e., $\boldsymbol{d}^j_{\sigma^{i,n}} \sim p(\boldsymbol{d}|\sigma^{i,n})$ such that the decisions $\boldsymbol{d}^j_{\sigma^{i,n}}$ formulates m_i. We found that when there are large number of training instances, sampling just one decision sequence led to good results as demonstrated in the image matching experiments in Section 7.2.

5 SMC Sampler for Matching

In this section, we develop an SMC sampler to draw samples from $p(\boldsymbol{d}_\sigma, \sigma|\theta)$. This SMC sampler is used for the Monte Carlo E-step of the MC-EM algorithm in the unsupervised setting, and for drawing samples from the posterior predictive distribution of matchings given the parameters estimated in the supervised setting.

5.1 Notation and Background

The SMC samplers method [28] is an important extension to vanilla SMC methods, and allows for drawing samples from an arbitrary state space by designing a sequence of intermediate distributions such that the final distribution coincides with the desired target distribution. One key idea in [28] is that of the backward kernel, which plays an important role when applying SMC to combinatorial problems. In this section, we begin by specifying the notation needed to construct the target and intermediate distributions as well as the backward kernel required to ensure the correctness of the SMC sampler for graph matching.

We remind the reader that the space of interest is the space of matchings, denoted $\mathcal{M}$. We generalize this space and introduce $\mathcal{M}_r, r = 1, ..., R$, the space of partial matching of size r (recall that a matching can be viewed as a set, hence it has a size). Here, r indexes iterations of the SMC algorithm and we will design our SMC algorithm such that $\mathcal{M}_R = \mathcal{M}$. The unnormalized intermediate distributions will be denoted by γ_r. We will use $n = 1, ..., N$ to denote the index of the N SMC particles. We denote by s_r^n and w_r^n the particle n at SMC iteration r and its associated unnormalized weight. The normalized weights are denoted by $\bar{w}_r^n = w_r^n / \sum_{n'} w_r^{n'}$. Recall also that the particles and weights at the last iteration are used to approximate the expectation via $\mathbb{E}[f(M)] \approx \sum_{n=1}^{N} \bar{w}_R^n f(s_R^n)$.

5.2 Poset SMC

A key concept in developing an SMC sampler on a combinatorial state space is the notion of partially ordered set, $(\mathcal{S}, \prec)$ introduced in [17]. The partial order, $\prec$, is a binary relation on the elements of $\mathcal{S}$ that is 1) reflexive, 2) anti-symmetric, and 3) transitive. Note that not all elements of $\mathcal{S}$ are comparable. The partially ordered set induces a Hasse diagram which is an undirected graph $G = (\mathcal{S}, E)$ where the nodes are the elements of the set $\mathcal{S}$ and the edge exists between the nodes $s, s' \in \mathcal{S}$ if s' covers s, which is to say that $s \prec s'$ and there does not exist $s'' \in \mathcal{S}$ such that $s \prec s'' \prec s'$.

The partial order endows the general state space with sequential structure that is needed for SMC. Therefore, a general recipe for defining an SMC sampler for graph matching is to clearly define the state space $\mathcal{S}$ and the initial state s_0, as well as the proposal density, ν^+ to ensure that every state is reachable starting from s_0.

5.3 SMC for Sequential Graph Matching

First, note that the target distribution $p(\boldsymbol{d}_\sigma, \sigma|\theta)$ can be decomposed as follows:

$$p(\boldsymbol{d}_\sigma, \sigma|\theta) = \prod_{r=1}^{|V|} p(d_{v_{\sigma_r(r)}}|m_{r-1}, \sigma_r, \theta) p(\sigma_r|\sigma_{r-1}). \tag{12}$$

This gives us our state space, $\mathcal{S}_r = \mathcal{M}_r \times \Sigma_r$, where $\mathcal{M}_r$ denotes the space of partial matching after r decisions have been made and Σ_r is a set of all possible reference sequences of size r: $\sigma_r \in \Sigma_r$ is a partial map $\sigma_r : \{1, ..., r\} \to \{1, ..., |V|\}$. Equation (12) also provides us with the intermediate un-normalized density: γ_r is simply given by the first r factors in the product in the right hand side of Equation (12). The sampling strategy is as follows. At iteration r, each particle samples a node v_j at random from $V_{\sigma_{r-1}^n}$, and sets $\sigma_r^n(r) = j$ (recall that $V_{\sigma_{r-1}^n}$ denotes the nodes that have not been visited up to iteration r). Alternatively, if the order σ is specified by the structure of the problem, v_j is provided deterministically. Then, we formulate a list of decisions for the sampled node, $\mathcal{D}(v_{\sigma_r^n(r)})$, from which we sample a decision, $d_{v_{\sigma_r^n(r)}}$. A natural proposal density is provided by the model structure,

$$\begin{aligned}\nu^+(s_{r-1}^{a_r^n} \to s_r^n) =& p(d_{v_{\sigma_r^n(r)}}|m_{r-1}, \sigma_r^n) \times \\ & p(\sigma_r^n(r) = j|\sigma_{r-1}^{a_r^n}),\end{aligned}$$

where $p(\sigma_r^n(r) = j|\sigma_{r-1}^{a_r^n}) = 1/|V_{\sigma_{r-1}^n}|$, and the other factor is provided by Equation (4). Note that a_r^n is the index of the parent particle.

5.4 Adjustments for Overcounting

For K-partite hypergraph matching, there may be more than one sequence of decisions that can lead to the same matching. To illustrate, consider Figure 2, where there are four nodes. The gray shaded ellipses denote a state (partial matching) and the white ellipses denote the edges in the matching. The initial state is an empty matching. Consider the pairwise decision model used for Figure 1 with the condition on the size of the edges removed. There are $\binom{4}{2} = 6$ possible states at iteration 1 (note that the empty state is at iteration 0). At iteration 2, we have shown two of the possible states. The state where an edge $\{1, 2, 3\}$ is formed has three different ways of being built whereas the state containing the

edges $\{\{1,2\},\{3,4\}\}$ has two possibilities. To ensure the correctness of the SMC algorithm, we need to account for the different paths that lead to the same state (i.e., the overcounting problem [18]).

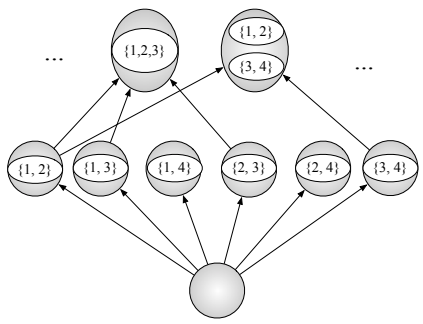


Figure 2: Illustration of the overcounting problem.

When the proposal density satisfies Assumptions (1)-(3) in [17], overcounting cannot occur. This holds for bipartite matching and the decision model corresponding to the set packing view of matching. More details and the proof is provided in the Appendix.

For K-partite matching with $K > 2$, a more general solution is to incorporate an appropriate backward kernel, $\nu^-(s_r^n \to s_{r-1}^{a_r^n})$ as proposed in [18]. The backward kernel amounts to $\mathbf{1}[\nu^+(s_{r-1}^{a_r^n} \to s_r^n) > 0]|\mathcal{Q}(s_r^n)|^{-1}$, where $\mathcal{Q}(s_r^n)$ denotes the set of parent states of s_r^n.

6 Bayes Estimator

We now describe how to construct a Bayes estimator from matchings drawn by the SMC sampler. For an estimator $\hat{M}$ of the matching M we define the loss function

$$L(M,\hat{M}) = \sum_{e\in M} 1[e \notin \hat{M}] + \sum_{e\in \hat{M}} 1[e \notin M], \qquad (13)$$

i.e., the symmetric difference between the two matchings. The Bayes estimator corresponds to the matching that minimizes the expected loss:

$$\hat{M}_{\mathrm{BE}} = \mathrm{argmin}_{\hat{M}}\, \mathbb{E}[L(M,\hat{M})|\theta], \qquad (14)$$

where the expectation is taken with respect to M. Since we cannot evaluate this expectation exactly except for small problems. We approximate it using the samples $\{M^n\}_{n=1}^N$ drawn by SMC:

$$\mathbb{E}[L(M,\hat{M})|\theta] \approx \frac{1}{N}\sum_{n=1}^{N} L(M^n,\hat{M}).$$

So the Bayes estimator satisfies

$$\hat{M}_{\mathrm{BE}} = \mathrm{argmin}_{\hat{M}} \sum_{n=1}^{N}\left(\sum_{e\in M^n} 1[e\notin \hat{M}] + \sum_{e\in\hat{M}} 1[e\notin M^n]\right),$$

i.e. it is the consensus matching, in the sense that it is the *least* different from the other matchings in the Monte Carlo samples. We approximate the consensus matching using a greedy algorithm described in the Appendix.

7 Experiments

7.1 Document Ranking

We begin by presenting results on a supervised learning experiment where the order is provided by the structure of the problem. With our model, supervised learning in such context reduces to simple logistic regression training, which can be performed by convex optimization using the exact gradient. In contrast, globally normalized methods such as [6] require approximate inference even in the supervised setting. These supervised learning experiments allow us to focus on the performance of the model; evaluation of the SMC algorithm is deferred to the subsequent subsections.

We apply the sequential decision model to the document ranking task. For this problem, we are given a set of documents for a given query, and our goal is to rank the documents by relevance. As described in [6], the document ranking task can be formulated as a bipartite matching task where the task is to match the rank to the documents with the goal of matching higher rank with relevant documents and lower rank with irrelevant documents. A common approach to this problem is to use supervised learning methods where we are given a dataset of queries and corresponding documents for each query, where each document is labelled with a relevance value. In fact, our sequential decision model is related to the listwise approach proposed in [12] when applied to the document ranking task, with the difference that [12] uses neural networks and a custom-defined loss function to estimate the model parameters whereas we optimize the log-likelihood. We demonstrate the effectiveness of our method on the LETOR 3.0 benchmark [29]. We performed experiments on the OHSUMED and TD2003 datasets. In each of these two datasets, we have a query and documents pair, which we denote by $\{(q_i,\{x_{ij},y_{ij}\}_{j=1}^{J_i})\}_{i=1}^{I}$, where I is the total number of queries and J_i is the total number of documents corresponding to query q_i. Here, x_{ij} denotes the features for document j retrieved for query q_i and y_{ij} denotes the relevance label. For OHSUMED, $y_{ij} \in \{0,1,2\}$ and $y_{ij} \in \{0,1\}$ for TD2003.

To perform training, we adopt the mechanism used in [6] where each query-document pair $(q_i,\{x_{ij},y_{ij}\})$ is broken up to yield B training instances, each containing L documents. Each training instance is obtained by first choosing a document from each relevance class

and then randomly sampling the rest of the documents without replacement. The number of training instances is then $I \times B$. In [6], the experiments were carried out with $L \in \{3, 4, 5\}$, partly due to the problem of intractable summation arising for a larger value of L. One advantage of our method is that we can experiment with larger values of L, as intractable summation is not a problem under our framework.

The metric we use for measuring the performance of ranking on the test dataset is the standard NDCG@k metric [6]. The performance of the sequential decision model for $k = 1, ..., 10$ is shown in Figure 3 (a) and (b) (the dotted red line). Our method attains a level of performance that is competitive to the top methods, and in some cases strictly better than all the other methods.

7.2 Image Feature Matching

We now turn to the image matching experiment. This experiment demonstrates a case where there is no natural order σ provided by the problem. For training, we sampled a single sequence of decision $\sigma^i, \boldsymbol{d}_{\sigma^i}$ for each training instance $i = 1, ..., I$. The SMC algorithm comes in at the prediction stage after having trained the parameters. In this case σ is sampled jointly with the decisions by our SMC algorithm.

We are given a pair of images containing 30 landmark points. These landmark points correspond to the nodes in the bipartite graph to be matched. We test our method on the CMU House dataset, which was used for evaluating a supervised graph matching algorithm in [5]. There are 111 frames in the video, where each frame is an image still with a slight modification from the previous frame. Each landmark point u is associated with a shape context feature, $f(u) \in \mathbb{R}_+^{60}$. For any proposed matching (u, v), the feature function is defined as: $\phi_p(u, v) = |f_p(u) - f_p(v)|$, $p = 1, ..., 60$.

We split the data exactly as in [5] for ease of comparison. For each $l \in \{25, 15, 10, 5, 3, 2\}$, the first scenario takes the training set to be the images that are divisible by l with the remaining as testing set, while the second scenario takes the training set to be the images that are not divisible by l and the testing set to be the images that are divisible by l. Thus, a total of 12 scenarios are tested.

In [5], Delaunay triangulation was performed on each image yielding a graph structure $G_i = (V_i, E_i)$ for each image $i = 1, ..., 111$. We incorporate this graph structure to extract edge features (pairwise affinities) as follows. Suppose we are matching two images G_i and $G_{i'}$. Let m_{t-1} be the partial matching and let $e = \{v_{i,j}, v_{i',k}\}$ be the proposed matching. For each $e_{j'k'} = \{v_{i,j'}, v_{i',k'}\} \in m_{t-1}$, we compute,

$$c_{j'k'} = \mathbf{1}[\{v_{i,j}, v_{i,j'}\} \in E_1, \{v_{i',k}, v_{i',k'}\} \notin E_2] + \mathbf{1}[\{v_{i,j}, v_{i,j'}\} \notin E_1, \{v_{i',k}, v_{i',k'}\} \in E_2].$$

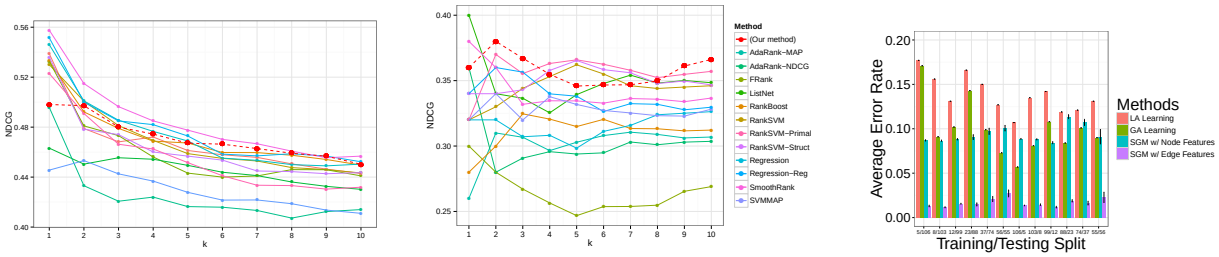
Then, set $\phi_{61}(m_{t-1} + e) = -\sum_{j'k'} c_{j'k'}$. The results are shown in Figure 3 (c). Note that our method based on the shape context feature outperforms the linear assignment learning methodology used in [5], which uses only the shape context features; likewise, our method with the edge feature outperforms graduated assignment learning, which uses the edge features.

7.3 Knot Matching

Our set packing methodology was motivated by a novel application in computational forestry. An active area of research in this field is the development of a computer vision-based, automatic strength assessment system for lumber (see for example [30]). Using high-resolution images of the four wood surfaces (see Figure 7.1), the first step is to detect and localize *knots*, which are remnants of tree branches and known to be one of the most important types of strength-reducing characteristic of lumber. This step is carried out with standard tools from image processing. The second step is to determine which knot faces appearing on different surfaces come from the same tree branch, which we call *knot matching*. The methodology in this paper was developed for this purpose. Finally, the knot information will be input into a model for strength prediction, such as the Bayesian statistical framework in [31]. The probabilistic approach to matching is especially appealing in this context since uncertainty in the matching represented by SMC samples can be easily propagated into Bayesian strength prediction models.

To formulate the knot matching problem, we represent each surface as a partition: $V_1, \ldots, V_4$. Hypothetically, a matching can contain up to 4 nodes but the vast majority of the matching are doubletons (matching with two nodes) with few 3-matchings. We have not observed 3-matchings in our dataset so we focus on 2-matchings for the experiments, which corresponds to formulating the decision set to allow for singletons and doubletons. Furthermore, the decision set can be reduced to exclude any knot faces that are beyond a certain distance (refer to the distance span in Figure 1 (a)). This decision model corresponds to the decision model used for illustrating Figure 1.

A natural sequence σ is given by the problem structure in the sense that we can start from either end of the board and proceed sequentially through the knot faces as they appear. In our experiments, we did not observe the direction to be having a noticeable impact on the performance. Note that this sequence combined with



(a) (b) (c)

Figure 3: The performance on (a) OHSUMED, (b) TD2003, (c) image matching.

Figure 4: The four sides of a board. The dark, circular objects are the knots to be matched.

Table 1: Results for the quadripartite matching problem from computational forestry.

		Unsupervised		Supervised		
Board	Num. Nodes	Consensus	MAP	Learning	No learning	Time (s)
1	32	1.00	1.00	1.00	0.38	0.086
2	50	0.92	0.92	0.92	0.12	0.147
3	34	0.81	0.79	1.00	0.29	0.081
4	24	1.00	1.00	1.00	0.42	0.039
5	36	1.00	1.00	1.00	0.33	0.084

the "locality" of the knots help create fast mixing of the SMC sampler – decisions made earlier in the sequence do not have much effect on future decisions. This allows us to achieve accurate results using only 100 particles for prediction as well as at each iteration of MC-EM.

We have manually labelled 5 pieces of lumber for evaluation only. The parameters were trained using MC-EM, which was executed for 100 iterations. Upon termination, we found the consensus matching and compared against the ground truth. For comparison, we have also carried out supervised learning of the parameters via leave-one-out procedure (i.e., held one board out for testing, and trained on the rest). We used the consensus matching as our prediction in the supervised setting whereas in the unsupervised setting we have compared the consensus matching as well as the maximum *a posteriori* matching found in the samples. We have also provided the timing results for prediction in the supervised setting. The fast execution time allows for implementation of our method for real time prediction of strength. The results are summarized in Table 1.

8 Conclusion

We have presented a method for learning a graph matching on a hypergraph by modelling the matching as a sequence of local decisions. The sequential decision model allows for incorporating the compatibility score in a similar fashion to the Gibbs measure whilst addressing the parameter inference problem via simple algorithms based on MC-EM combined with sequential Monte Carlo samplers. We have focussed on MAP parameter estimation, but our model is also amenable to full Bayesian analysis. While globally normalized models require doubly-intractable methods for full Bayesian analysis [10], our model can be easily combined to implement particle MCMC (PMCMC) methods [32]. Compatibility with PMCMC also opens the door to more advanced SMC methods (for example, [33]). Finally, the code and the data is made available at `https://github.com/junseonghwan/sgmwsmc`.

Acknowledgments

We thank FPInnovations for providing the data for the knot matching application. This work was supported by FPInnovations and a CRD grant from the Natural Sciences and Engineering Research Council of Canada.

References

[1] M. Volkovs and R. Zemel. Efficient Sampling for Bipartite Matching Problems. *Advances in Neural Information Processing Systems*, 25:1322–1330, 2012.

[2] I. L. Dryden and K. V. Mardia. *Statistical shape analysis*, volume 4. J. Wiley Chichester, 1998.

[3] I. Holmes and G. M. Rubin. Pairwise RNA structure comparison with stochastic context-free grammars. In *Proceedings of the Pac. Symp. Biocomputing*, volume 7, pages 163–174, 2001.

[4] G. Lunter, A. J. Drummond, I. Miklós, and J. Hein. Statistical alignment: Recent progress, new applications, and challenges. In *Statistical methods in molecular evolution*, pages 375–405. Springer, 2005.

[5] T. S. Caetano, J. J. McAuley, L. Cheng, Q. V. Le, and A. J. Smola. Learning graph matching. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(6):1048–1058, 2009.

[6] J. Petterson, J. Yu, J. J. McAuley, and T. S. Caetano. Exponential family graph matching and ranking. In *Advances in Neural Information Processing Systems*, pages 1455–1463, 2009.

[7] G. C. Wei and M. A. Tanner. A Monte Carlo implementation of the EM algorithm and the poor man's data augmentation algorithms. *Journal of the American statistical Association*, 85(411):699–704, 1990.

[8] B. Delyon, M. Lavielle, and E. Moulines. Convergence of a stochastic approximation version of the EM algorithm. *Annals of statistics*, pages 94–128, 1999.

[9] N. A. Smith and J. Eisner. Contrastive estimation: Training log-linear models on unlabeled data. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 354–362. Association for Computational Linguistics, 2005.

[10] J. Møller, A. N. Pettitt, R. Reeves, and K. K. Berthelsen. An efficient Markov chain Monte Carlo method for distributions with intractable normalising constants. *Biometrika*, 93(2):451–458, 2006.

[11] R. L. Plackett. The analysis of permutations. *Applied Statistics*, pages 193–202, 1975.

[12] Z. Cao, T. Qin, T. Y. Liu, M. F. Tsai, and H. Li. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*, pages 129–136. ACM, 2007.

[13] F. Caron, Y. W. Teh, and T. B. Murphy. Bayesian nonparametric Plackett-Luce models for the analysis of preferences for college degree programmes. *The Annals of Applied Statistics*, 8(2):1145–1181, 2014.

[14] T. Berg-Kirkpatrick, A. Bouchard-Côté, J. DeNero, and D. Klein. Painless unsupervised learning with features. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL10)*, volume 8, pages 582–590, 2010.

[15] A. Doucet and A. M. Johansen. A tutorial on particle filtering and smoothing: Fifteen years later. *Handbook of Nonlinear Filtering*, 12:656–704, 2009.

[16] Y. W. Teh, H. Daumé III, and D. M. Roy. Bayesian agglomerative clustering with coalescents. In *Advances in Neural Information Processing Systems*, volume 20, 2008.

[17] A. Bouchard-Côté, A. Sankararaman, and M. I. Jordan. Phylogenetic inference via sequential Monte Carlo. *Systematic Biology*, 61(4):579–593, 2012.

[18] L. Wang, A. Bouchard-Côté, and A. Doucet. Bayesian phylogenetic inference using the combinatorial sequential Monte Carlo method. *Journal of the American Statistical Association*, 110:1362–1374, 2015.

[19] S-H. Jun and A. Bouchard-Côté. Memory (and time) efficient sequential Monte Carlo. In *International Conference on Machine Learning (ICML)*, volume 31, pages 514–522, 2014.

[20] B. Paige, F. Wood, A. Doucet, and Y. W. Teh. Asynchronous anytime sequential Monte Carlo. In *Advances in Neural Information Processing Systems*, pages 3410–3418, 2014.

[21] M. Jerrum and A. Sinclair. Approximating the permanent. *SIAM journal on computing*, 18(6):1149–1178, 1989.

[22] M. Jerrum, A. Sinclair, and E. Vigoda. A polynomial-time approximation algorithm for the permanent of a matrix with nonnegative entries. *Journal of the ACM (JACM)*, 51(4):671–697, 2004.

[23] J. Wang and A. Jasra. Monte Carlo algorithms for computing α-permanents. *Statistics and Computing*, 26(1-2):231–248, 2016.

[24] A. Bouchard-Côté and M. I. Jordan. Variational inference over combinatorial spaces. In *Advances in Neural Information Processing Systems 23 (NIPS)*, volume 23, pages 280–288, 2010.

[25] Y. Suh, M. Cho, and K. M. Lee. Graph matching via sequential Monte Carlo. In *European Conference on Computer Vision*, pages 624–637. Springer, 2012.

[26] D. C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical programming*, 45:503–528, 1989.

[27] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the royal statistical society. Series B (methodological)*, 39:1–38, 1977.

[28] P. Del Moral, A. Doucet, and A. Jasra. Sequential Monte Carlo samplers. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(3):411–436, 2006.

[29] T-Y. Liu, J. Xu, T. Qin, W. Xiong, and H. Li. Letor: Benchmark dataset for research on learning to rank for information retrieval. In *Proceedings of SIGIR 2007 workshop on learning to rank for information retrieval*, pages 3–10, 2007.

[30] R. Hietaniemi, M. B. López, J. Hannuksela, and O. Silvén. A real-time imaging system for lumber strength prediction. *Forest Products Journal*, 64(3):126–133, 2014.

[31] S. W. K. Wong, C. Lum, L. Wu, and J. V. Zidek. Quantifying uncertainty in lumber grading and strength prediction: a Bayesian approach. *Technometrics*, 58:236–243, 2015.

[32] C. Andrieu, A. Doucet, and R. Holenstein. Particle Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(3):269–342, 2010.

[33] F. Lindsten, M. I. Jordan, and T. B. Schön. Particle Gibbs with ancestor sampling. *The Journal of Machine Learning Research*, 15(1):2145–2184, 2014.